A Survey of Adaptive Sampling and Filtering Algorithms for the Internet of Things

Dimitrios Giouroukis[1] Alexander Dadiani[1] Jonas Traub[1] Steffen Zeuch[1,2] Volker Markl[1,2]
[1]Technische Universität Berlin [2]DFKI GmbH
firstname.lastname@tu-berlin.de firstname.lastname@dfki.de

ABSTRACT

The Internet of Things (IoT) represents one of the fastest emerging trends in the area of information and communication technology. The main challenge in the IoT is the timely gathering of data streams from potentially millions of sensors. In particular, those sensors are widely distributed, constantly in transit, highly heterogeneous, and unreliable. To gather data in such a dynamic environment efficiently, two techniques have emerged over the last decade: *adaptive sampling* and *adaptive filtering*. These techniques dynamically reconfigure rates and filter thresholds to trade-off data quality against resource utilization.

In this paper, we survey representative, state-of-the-art algorithms to address scalability challenges in real-time and distributed sensor systems. To this end, we cover publications from top peer-reviewed venues for a period larger than 12 years. For each algorithm, we point out *advantages*, *disadvantages*, *assumptions*, and *limitations*. Furthermore, we outline current research challenges, future research directions, and aim to support readers in their decision process when designing extremely distributed sensor systems.

CCS CONCEPTS

• **Information systems → Sensor networks**.

KEYWORDS

Sensor Networks, Adaptive Sampling, Adaptive Filtering

ACM Reference Format:
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 *The 14th ACM International Conference on Distributed and Event-based Systems (DEBS '20), July 13–17, 2020, Virtual Event, QC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3401025.3403777

1 INTRODUCTION

The Internet of Things (IoT) will create environments with millions of heterogeneous sensor nodes that provide data in real time [41]. Timely acquisition of data from such a highly distributed sensor deployment poses complex challenges for data management systems.

DEBS '20, July 13–17, 2020, Virtual Event, QC, Canada

ACM ISBN 978-1-4503-8028-7/20/07…$15.00
https://doi.org/10.1145/3401025.3403777

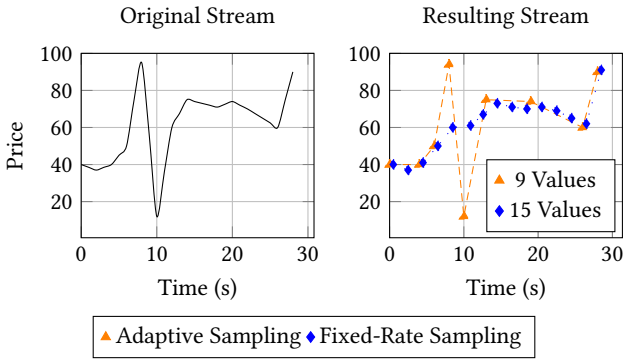

Figure 1: Adaptive Sampling shows *fewer reads* and *fewer transmissions* compared to Fixed Rate Sampling, which lead to *better data quality*.

The main research challenges of sensors networks are: *(i) heterogeneity of resources*, *(ii) widely distributed communication networks*, and *(iii) nodes with diverse sets of capabilities* [17, 48, 49].

In this paper, we examine how current research tackles those challenges in highly distributed sensor environments. To this end, we focus on two classes of algorithms that enable data management in very large IoT environments. In particular, we present *adaptive sampling* and *filtering algorithms* that run on the data sources, i.e., on sensor nodes. With these algorithms, we are able to address many important scalability challenges when one gathers streams of sensor data from the IoT. Adaptive sampling enables the system to decide *when*, *where*, and *how often* to read [28] (i.e., sample) a value from a sensor. In contrast, adaptive filtering allows the system to decide *which* values to transmit to an event-processing engine for further analysis. Adaptive sampling and filtering enable data management systems to scale up the number of sensors as well as reduce the volume of network traffic without harming the precision of the result.

The literature on adaptive sampling and filtering is fragmented across many different conferences, journals, and research communities. For this survey, we reviewed 91 publications from 64 distinct journals and conferences. We first acquired publications based on keyword searches and then extended our review to related work for each publication. We selected 19 techniques, which we discuss in detail in this survey. These techniques are representative for different sets of algorithms that we have identified in our broad literature review. To highlight the unique characteristics of adaptive sampling and filtering, we provide a summary of the core ideas in the following subsections.

1.1 Adaptive Sampling

Adaptive sampling changes the sampling rates on a sensor node such that *(i)* sensors observing an interesting event provide detailed

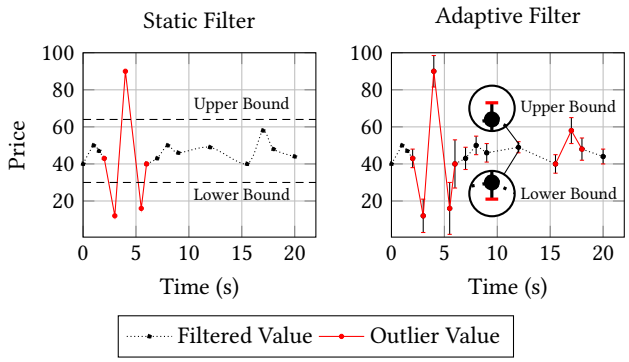

Figure 2: Adaptive Filtering captures the context of *change* compared to a Static Filter, which leads to *better data quality*.

data (high sampling rate) and *(ii)* sensors that do not observe interesting events reduce sampling rates to not overload the receiver. Ideally, at any time, a subset of sensors dynamically switches to a higher sampling rate while the majority of sensors provides data at lower rates. Thus, a highly adaptive sampling approach will enable future IoT deployments with millions of sensor nodes.

An example for adaptive sampling is shown in Figure 1. The red signal on the left represents a physical phenomenon, such as air pressure, temperature, or an acceleration. A sensor observes the phenomenon while sampling (i.e., read the current value) at a fixed or adaptive sampling rate. The sensor node transmits the observed values together with sensor readings to a central analysis engine. The analysis engine reconstructs the physical phenomenon. Essentially, the goal of adaptation is to compress the data stream based on interpolation instead of transmission of values, ideally with minimal loss of information.

In Figure 1, blue diamonds mark sensor readings performed with a fixed rate. In contrast, orange triangles mark sensor readings performed with an adaptive rate. In this example, adaptive sampling has three advantages compared to fixed rate sampling. First, it performs fewer sensor reads, which leads to energy savings on the sensor nodes. Second, it transmits fewer values to the central analysis engine, which saves network traffic. Third, the reconstructed phenomenon of the adaptive technique is closer to the original.

1.2 Adaptive Filtering

Adaptive filtering techniques focus on finding a threshold that helps deciding whether a system should transmit a sensor value. In particular, if a sensor value is similar to previous values or evolves predictably, a node can avoid data transmission and save network traffic. In contrast, if a value changes unexpectedly and does not follow a prediction, a node needs to transmit an update to maintain the precision of the reconstructed signal on the receiver side. Since the behavior of a signal may change frequently, static a-priori filtering would lead to sub-optimal decisions. Thus, filter thresholds and rules for value filtering must adapt over time on the observed values.

Figure 2 depicts an example of adaptive filtering. This example compares two different filtering techniques (left and right). Both techniques transmit the values for the intervals marked in red for a stream of price changes. However, on the left, the technique transmits changes that exceed the predefined threshold only, ignoring later changes. In contrast, the right side captures changes depending on the magnitude of change between values. A technique with fixed thresholds may underestimate the importance of changes. By adapting the threshold, a filtering technique reduces network traffic and maintains data quality by leveraging context from the data.

1.3 Paper Outline

In the remainder of this paper, we present how adaptive sampling and filtering techniques enable highly distributed sensor deployment. First, § 2 introduces the methodology for selecting papers, defines terms that are used throughout the paper, and discusses selection criteria and evaluation metrics. The section concludes with an introduction to our taxonomy of algorithms in detail. After that, we present our selected adaptive sampling techniques in § 3 and adaptive filtering techniques in § 4. We further examine algorithms that combine adaptive sampling or adaptive filtering with other techniques in § 5. Finally, we conclude in § 6.

2 METHODOLOGY

In this section we present our research methodology. We provide a description of the surveyed literature in § 2.1. Next, we introduce the common terminology of the literature in § 2.2. Then, we list the selection criteria for the algorithms in § 2.3 and conclude with a discussion on the evaluation metrics in § 2.4.

2.1 Literature Review

We surveyed publications that contain keywords related to sensor networks, sampling, filtering, adaptive monitoring, energy expenditure in sensor networks, clustering, data acquisition, and in-network processing for sensor networks. After the initial results, we studied their citations and related work and kept publications that relate to the topics of *adaptive sampling* and *adaptive filtering* and are of algorithmic nature. We excluded any publications that are unrelated to our core topics and do not specify an algorithm.

In order to analyze and compare the selected algorithms, we surveyed 91 different papers. The papers originate from proceedings of 23 unique academic conferences, 41 unique academic journals, and one book. Our survey includes 53 papers from journals, 35 papers from conferences, and one book entry. We also include two tech reports in our survey. The topics of the surveyed papers range among clustering algorithms, time synchronization, duty cycling, topology control, in-network data aggregation, data compression, and routing algorithms. Our goal is to put algorithms from different domains and communities together under one survey and help readers put the algorithms into perspective.

For every paper, we extracted a specific set of properties, i.e., the algorithm *implementation details*, descriptions of *strengths and limitations*, and *message and network overhead* estimations. We summarize our findings in Table 1 at the end of this survey.

2.2 Terminology

The literature related to *sampling* and *filtering* from sensors defines many different terms such as *data collection* [24, 44], *data sampling* [21, 40], *data gathering* [25, 26, 45], and *data sensing* [30, 33]. Some publications use these terms as synonyms while other publications use different terms to differentiate concepts. In the following, we introduce definitions of terms used throughout the paper:

Sensor Node: A sensor node is a device that provides a processing unit, a network interface, and at least one sensor [1, 20]. The sensor captures values from at least one continuous signal.

Sensor: A sensor is a device that creates information from environmental events. It converts a physical phenomenon to a stream of values and transmits them to other devices.

Sampling Rate: A sampling rate, or the *sampling frequency*, is the number of samples per time unit taken from a continuous signal to create a discrete signal.

Sampling Period: The sampling period is the time between two consecutive samples. For example, if the sampling rate is 10Hz (10 values per second), then the sampling period is 0.1 sec.

Sensing and Sampling: The definitions of *sampling* and *sensing* are inconsistent in the literature. Zhao et al. [51] use sampling in the context of a sink assigning sampling tasks to sensor nodes, which consists of a time window and a sampling period. Trihinas et al. [42] do not differentiate between sampling and sensing. Aquino et al. [4], define *sensing* as the process of measuring a physical phenomenon with sensor units and *sampling* as software that takes samples from the sensor unit data stream. Deshpande et al. [13] argue that a sensor only provides *samples* of an observed, continuous phenomenon. In this paper, we do not distinguish between *sampling* and *sensing*. We use the definition of Trihinas et al. [42], i.e., one samples a sensor when she *acquires* a value from that sensor.

Filtering: The process of suppressing values from sensor samples that do not exceed a certain threshold.

Adaptivity: For sampling, adaptive refers to the change of the frequency of gathering samples from sensors. For filtering, being adaptive refers to changing the filtering threshold. In both cases, adaptivity allows an algorithm to *react* to changes.

2.3 Selection Criteria

One common goal of the surveyed algorithms is optimal resource usage. Energy, memory, network overhead, or computational power are examples of resources commonly found in the bibliography. Another common goal of the examined algorithms is optimal sensor node lifetime through energy expenditure reduction. Sampling the sensor nodes and disseminating the observed values are the main culprits of energy expenditure in sensor networks. A common assumption in sensor research is that sensor nodes are heavily constricted in terms of resources. Given these common goals and assumptions, we select the algorithms for our taxonomy based on the following criteria:

Optimization Focus: The algorithms that we examine are focusing on resource expenditure optimization on sensor nodes (e.g., energy usage or network consumption).

Sensor Centered: The algorithms that we examine propose solutions applicable to sensor nodes. Thus, algorithm benefits do not exploit other levels (e.g., task offloading on a cloud server) and always start from the level of sensor nodes.

Explicit Assumptions: The examined algorithms propose solutions that are tested on various settings. The selected papers explain in detail their assumptions and their implementation.

Clear Evaluation Scheme: The examined papers present their findings in a separate evaluation section in their text. A clear evaluation section makes the paper easy to follow and gives a concise explanation of the improvement.

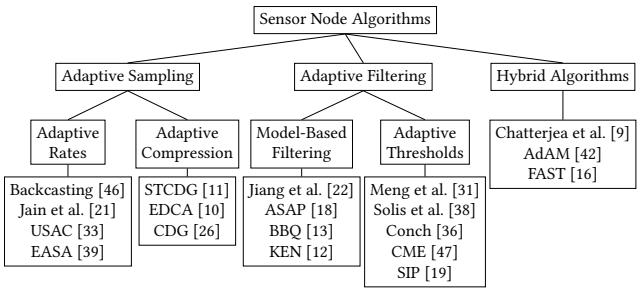

Figure 3: Taxonomy of the Selected Algorithms

2.4 Evaluation Metrics

To provide a better overview of the selected algorithms, we summarize our findings in Table 1. We use a common set of metrics for the discussion of all algorithms:

Assumptions: The selected algorithms assume specific network characteristics that are tailored to the observed phenomenon. We classify these characteristics as the *basic assumptions* of an algorithm, since the assumptions are necessary for the correct execution of the algorithms.

Advantages: We present the advantages of selected algorithms, as defined in their respective papers. We also include unique algorithm traits.

Limitations: Limitations refer to design caveats, e.g., an algorithm is unsuitable for large-scale deployment or is unsuitable for a specific task.

Message Overhead: We include message overhead as a property in our study since it varies between network types. Overhead, e.g., control messages, may lead to message delays. We define different classes of message overhead, specifically *Low*, *Medium*, and *High*. In the Low class, sensor nodes work autonomously, i.e., the sinks do not send control messages. In the Medium class, sensor nodes operate autonomously with minimal sink communication. In the High class, there is frequent exchange of control messages.

2.5 Taxonomy

Figure 3 depicts our taxonomy in its entirety. The first major class of algorithms is *Adaptive Sampling*. The *Adaptive Sampling* category is divided further into two sub-classes, *Adaptive Rates* and *Adaptive Compression*. § 3 covers the Adaptive Sampling class and contains algorithms that manipulate the sampling rates of sensors. The second major class, *Adaptive Filtering*, consists of two sub-classes, i.e., *Adaptive Thresholds* and *Model-Based* filtering. § 4 covers the Adaptive Filtering class. The section contains algorithms that focus on suppressing sampled data by utilizing spatial or temporal thresholds over sampled values. The final class, *Hybrid Algorithms*, contains techniques that combine either adaptive sampling or filtering with another class. § 5 covers the Hybrid Algorithms class. In our listing, we address each algorithm by its assigned name from the relevant paper, otherwise we reference the relevant authors.

3 ADAPTIVE SAMPLING

The category of *Sampling* algorithms incorporates algorithms that focus on the manipulation of the sampling rate of sensor nodes.

In Figure 1, a fixed rate technique and an adaptive technique both sample the same stream while sending results to a visualization dashboard. The example highlights the benefits of adaptive sampling, i.e., *fewer sensor reads*, *fewer transmissions*, and *higher precision* of the reconstructed phenomenon in the dashboard. The surveyed algorithms within the Sampling category aim for the same benefits but utilize different approaches. Therefore we further split the Sampling category in two sub-categories: *adaptive rates* and *compressive sampling*. Both sub-categories target the sampling rates of sensor nodes in order to reduce resource consumption. However, adaptive sampling algorithms react to changes in the behavior of an observed phenomenon through manipulation of the sensing rate. In contrast, compressive sampling utilizes techniques that sample signals below a certain rate [8] and later reconstruct them with high accuracy. An exemplary sampling rate is the Nyquist rate, well known in the field of signal processing for being the minimum acceptable sampling rate for artifact-free results.

3.1 Adaptive Rates

Algorithms in the *adaptive rates* subcategory deal with the dynamic adjustment of the sampling rate, based on the current metric evolution, in order to reduce energy consumption and back-pressure when there are stable phases in a data stream [42]. For the remainder of the subsection, we will go through the selected algorithms that focus on the dynamic adjustment of the sampling rate.

3.1.1 Backcasting. The key idea of *Backcasting* [46] is that a monitored environment exhibits correlations in the domains of time and space. This property may be exploited in order to reduce the number of required sensor nodes. Essentially, a subset of the sensors transmits values to a fusion center, where it aggregates information.

Assumptions: Willet et al. [46] assume that the sensor network is deployed in a uniformly distributed fashion. Initially, a subset of the sensor nodes is chosen by the sink to provide an initial estimate of the sensed phenomenon with a Recursive Dyadic Partition (RDP). An RDP is a tree where leaves are cells to a dyadic partition of the domain [37]. In the case of Backcasting, the domain is the set of sensor nodes. Subsets of nodes form multiple clusters of nodes, with a cluster head assigned to every cluster. Cluster heads route the data from the sensor nodes to the sink and vice versa. The sink receives the sensed data and activates additional nodes to improve the quality of the sampling by reducing the Mean Squared Error.

Advantages and Limitations: The increase of the number of nodes leads to additional message overhead since control messages are routed through the network. The authors evaluate their approach and conclude that a Wireless Sensor Network (WSN) with 10k sensors and enough energy to operate continuously for a year, would run for 10 years with Backcasting. The authors point out that the network lifetime can be improved if mechanisms for cycling the position of cluster head were present. Although Backcasting does not manipulate the sampling rates of individual sensor nodes, it adaptively changes the sensing tasks of nodes based on the dynamics of the observed phenomenon.

3.1.2 Algorithm of Jain et al. The main contribution of the algorithm of Jain et al. [21] is changing the sampling rate of individual sensor nodes in a stream sensor network based on the *importance* of the observed event. An important event may be a camera observing non-standard driving behavior of a car, e.g., driving in an erratic course or a temperature spike in a data center.

Assumptions: At the sensor node level, Kalman filters predict sensed values. The algorithm compares the predicted values with the actual sensed values. The values are stored locally, in a sliding window context. An estimation error is computed over the sliding window. The estimation error indicates the changing dynamics of an observed phenomenon. After each sample, a sensor node adjusts its sampling rate. If the desired sampling rate lies in the range predicted by the Kalman filter, the sensor node can change its sampling rate autonomously. Otherwise, the sensor node has to request a new sampling rate from the sink. The sink keeps a metric of available communication resources, which is updated after a request to change a sampling rate is accepted. The requests are then stored in a queue. The approval of a sampling rate request is a linear optimization problem.

Advantages and Limitations: The additional communication with the sink induces a high message overhead. Jain et al. [21] evaluate their approach on synthetic spatiotemporal data. The major metrics are the mean fractional estimation error η, the proportion of messages that the source and the sink exchange, and the number of values sensed by the source nodes `m`. The adaptive sampling technique outperforms the alternative uniform sampling algorithm in the majority of the tests. Their tests included the use of different numbers of sensor nodes as well as sliding window sizes. The authors indicate that the algorithm does not apply to multi-hop sensor networks where sensor nodes are required to have direct connections to a sink. Furthermore, both the sliding window and the sampling rate interval parameters are manually tuned. The authors state on their future work section that the message overhead is high. Currently, this makes the algorithm less suitable for WSNs as the communication overhead consumes a major amount of energy [34].

3.1.3 Utility-based Sensing And Communication protocol (USAC). The algorithm of Padhy et al. [33] is an adaptive sampling scheme for WSNs. It is designed for a sensor network that monitors glaciers. *USAC* uses a linear regression model on each sensor node. The node timely captures phase shifts of observed phenomena.

Assumptions: The model predicts values locally, at the sensor node. The values are checked against the actually sensed readings. If a predicted value lies in the user-defined confidence interval then the sensor node reduces its sampling rate by a multiplicative factor α until the sampling rate reaches f_{min}. The f_{min} parameter is user-defined. If the value is not within the interval bounds, then the sensor node raises its sampling rate to f_{max} to capture changes in the dynamicity of the observed phenomenon. No additional communication happens with the sink.

Advantages and Limitations: Padhy et al. [33] compare USAC and the older protocol of the glacier monitoring application *GLACSWEB*. In the GLACSWEB protocol, sensor nodes send their data directly to a sink. GLACSWEB is energy inefficient as the power required to transmit data from one node to another is proportional to the square of the distance between the nodes. Additionally, GLACSWEB has a static sampling rate that induces unnecessary sampling.

Testing is conducted in a simulated environment with historical data from the application. The authors experiment with different network topologies, number of sensor nodes in a network, and the

number of changes in the dynamicity of the data. The experiments show that USAC outperforms its baseline GLACSWEB, by 470%, when distributing sensor nodes randomly around a central base station. The main metric is the value of data gained over the amount of energy consumed, for every test case.

3.1.4 Energy Aware Adaptive Sampling Algorithm (EASA). The algorithm of Srbinovski et al. [39] is an adaptive sampling algorithm for energy-hungry sensors for perpetually operating RSNs, which builds upon the Adaptive Sampling Algorithm (ASA) [2]. The ASA algorithm leverages the Nyquist Theorem for finding the optimal (minimum) sampling rate F_N:

$$F_N > F_{max} \cdot 2$$

Assumptions: Srbinovski et al. [39] claim that the sensor unit is the main energy consumer [6], contrary to the assumption that communication consumes the most energy in sensor nodes [35].

To estimate the maximum rate in the power spectrum F_{max}, a fast Fourier transform is used on the first W samples of the process. The algorithm runs on the sink due to computational complexity; thus some message overhead is present. *EASA* expands ASA with energy awareness, i.e., adjusting the optimal sampling rate of a sensor node based on current and critical battery level and rate of energy saved per time unit. When the battery level drops below a user-defined critical threshold $\mathtt{m}$, the sampling rate deviates from the optimal ASA value.

Advantages and Limitations: The EASA-induced threshold may lead to sensors that do not capture the signal fully since the Nyquist theorem may be violated. The authors argue that a potential loss in data quality is the cost of staying continuously in the network. The authors test EASA extensively on data from two deployments. EASA is tested with different saving rates against ASA and on the second deployment against a fixed rate. A simulated environment hosts all tests. In both deployments, ASA and EASA have the same sampling rate when the battery level is above m. ASA may deliver data at higher quality since the sampling rate is always optimal. EASA on the other hand stabilizes the energy levels of the sensor node after 36 days of operation at 60% with a m of 1/3 in the first deployment. With ASA the energy levels stay at 20%. In the case of the second deployment, ASA is outperformed in energy consumption by EASA. The fixed sampling rate deployment depleted its energy after 13 days.

3.2 Discussion of Adaptive Rate Algorithms

In § 3.1, we detailed the algorithms that focus on adjusting the sampling rates of sensors and thus increase or decrease the amount of messages over the network. In Table 1, the average message overhead of the category is *medium*, according to our classification, since network consumption depends on algorithm assumptions. Backcasting and USAC have low message overhead while EASA and the algorithm of Jain et al. [21] exhibit high message overhead. In general, the algorithms provide high quality data streams since they capture sudden changes in the phenomenon at run time. The number of sensed events immediately impacts the effectiveness of the algorithms, since they reduce network traffic significantly when events are few or non-existent. If the stream contains a lot of consecutive events, the algorithms adapt and provide a high quality representation of the stream but at the cost of computation and network resources, which results in small spikes of energy expenditure.

3.3 Adaptive Compression

Adaptive compression enables the reconstruction of a discrete signal from a set of randomly chosen values from a vector that is computed by a linear transform on the original discrete signal [14, 29]. The main principles of adaptive compression are sparsity and incoherence [8]. A sparse signal contains a large number of zero-valued elements [15]. Incoherence is the point in time when a sample of a sparse signal has an extremely dense representation in a domain. For the remainder of the subsection, we will go through the selected algorithms that focus on sampled signal compression.

3.3.1 Compressive Data Gathering (CDG). The goal of the algorithm of Luo et al. [26] is to decrease energy expenditure and distribute energy consumption evenly across all sensor nodes, in order to maximize the lifetime of the network in large scale sensor networks.

Assumptions: *CDG* aims to reduce data traffic in a network. By compressing data readings at each hop, M messages arrive at a sink from N sensor nodes, where $M << N$. The sink broadcasts a random seed to the network that is the basis for local seed creation in each sensor node. Afterwards, no other message overhead is present. Sensor nodes use the local seed to generate a random coefficient ϕ_i, which they transmit together with their sensor reading d_i to their parent node. Parent nodes receive readings from their children nodes and sum up the input with their own d and ϕ. Therefore, every sensor node transmits only one value. This ensures the sink receives M weighted sums y. Each sum consists of a matrix of the random coefficients, Φ and the following sensor readings $\mathbf{d}$:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix} = \begin{pmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{M1} & \phi_{M2} & \cdots & \phi_{MN} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix}$$

The random matrix Φ is not transmitted since the sink may compute the matrix if it knows the identifier of the sensor nodes. The original sensor readings $\mathbf{d}$ can be reconstructed at the sink level by solving the following equation:

$$\min_{x \in R^N} ||x||_{l_1} \quad s.t. \;\; y = \Phi \mathbf{d}, \;\; \mathbf{d} = \Psi x$$

Ψ is the domain in which $\mathbf{d}$ can be represented by $K << N$ coefficients and x is the vector of coefficients.

Advantages and Limitations: Luo et al. [26] evaluated CDG against centralized exact (all nodes transmit as soon as they acquire a reading) with the ns-2 simulator [5]. They evaluate the reconstruction capability on two real-life data sets. For the simulated environment, the authors create two synthetic sensor networks. The first network is a network with a chain topology, with 1k sensor nodes placed 10 meters apart from each other and with sinks located at each extremity of the chain. The second network setup is a grid-like routing tree with 1089 sensor nodes and a sink node in the center of the network. The authors vary the signal input intervals and observe the package loss and output interval change. CDG outperforms centralized exact in both topologies, with input intervals 5x and 2.3x smaller than the centralized exact in the chain and grid

topologies, respectively. CDG achieved a packet loss of near zero in both topologies, while the centralized exact measures package loss rates of 5% and 20% in the chain and grid topologies, respectively.

The real-life datasets consist of measurements collected in the Pacific Ocean by a single moving device and a sensor network in a data center. The authors argue that the Pacific Ocean dataset has the same properties as data collected by a sensor network. The authors found the Pacific Ocean data to be sparse in the wavelet domain. CDG reconstructs the initial 1k data points from 40 data points with > 98% precision. The data points are the 40 highest coefficients in the wavelet domain of the data. The authors did not find a sparse representation of the datacenter dataset as the data exhibit little spatial correlation. Instead, they opted for data set reorganization by sorting the temperature values in ascending order at moment t_0. The result is sparse in the wavelet domain, and the original signal is reconstructable.

3.3.2 Efficient Data Collection Approach (EDCA). The algorithm of Cheng et al. [10] builds upon the low-rank feature of a matrix, where related work has proven that a matrix formed from spatially and temporally correlated data is approximately low-rank and is recoverable with only a subset of that data [7, 43].

Assumptions: In *EDCA*, sensor nodes sample at a fixed rate and send values to a central sink in a multi-hop scheme. The sink sends only the sampling rate to the nodes and no other control messages. However, Cheng et al. [10] point out that missing values, i.e., no values in some time slots, are recoverable with low error probability. The focus of the algorithm lies in the recoverability of low-rank matrices. The authors use the nuclear norm to solve the following rank minimization problem:

$$minimize\ rank(X),\ s.t.\ A(\cdot)=B$$

where X is the matrix arriving at the sink and $A(\cdot)$ is an operator that represents the incompleteness of the matrix. Since the problem is NP-hard, the authors use a heuristic and shift the problem to a convex optimization problem:

$$minimize\ ||\ A(LR^T)-B\ ||_F^2+||\ L\ ||_F^2\ +||\ R\ ||_F^2$$

where X is deconstructed with SVD into $X = U\Sigma V^T$ and $L = U\Sigma^{1/2}$; $R=V\Sigma^{1/2}$. The solution is taken from Zhang et al. [50].

Advantages and Limitations: The authors use a publicly available dataset of temperature measurements from sensor nodes from the Intel Berkeley Research lab [27] and a synthetic dataset. The authors test the accuracy of EDCA with different sampling ratios for the 54 sensor nodes. The authors observe only small recovery errors. With a sampling ratio of 0.2, i.e. every fifth sensed value is actually transmitted, the standard deviation σ is $<0.15C$. With the synthetic data set, the authors compared EDCA with centralized exact, where the sink receives all sensed data points, on the metric of network lifetime. The authors define the lifetime of a sensor network as the time duration of the first sensor node which runs out of power. The lifetime ratio is defined as $(1/M_{max})/(1/M_0)$, where M_{max} is the maximum power wasted on every EDCA simulation and M_0 is the power wasted by the centralized exact technique. The authors find that the lifetime ratio is 5 when the sampling ratio is 0.2, compared to the centralized exact.

3.3.3 Spatio-Temporal Compressive Data Collection (STCDG). The algorithm of Cheng et al. [11] is an expansion of EDCA [10]. *STCDG* exploits the low-rank feature and the short-term stability of data gathered through a sensor network, similar to EDCA.

Assumptions: Cheng et al. [11] argue that, unlike CDG [26], STCDG is more flexible and does not need customization for a specific sensor network. With CDG the domain in which the data is sparse has to be known in advance. Additionally, CDG utilizes only the sparsity of the data, which requires a full dataset reorder if the data is not sparse. The authors of STCDG deploy a sensor network in a residential building. They use the data to analyze the short-term stability and low-rank of spatially and temporally correlated data. The authors find that the data have a good low-rank approximation and short-term stability. Short-term stability is defined as the difference between adjacent gaps of sensor readings, with gaps being the time between two adjacent sensor readings. The difference is defined as:

$$dif(n,t)=X(n,t+1)+X(n,t-1)-2X(n,t)$$

where $1<=n<=N$ *and* $2<=t<=T-1$; thus, if $dif(n,t)$ is small, then sensor readings at node `n` around timeslot `t` are stable.

The authors expand their optimization problem formulation to include a tuning parameter ζ, in order to express the trade-off between fitting the algorithm to the data and achieving low-ranking. The notion of short term stability is added to the optimization problem, as the difference for all data points in the original matrix X, $||(LR^T)S^T||_F^2$ the optimization problem is:

$$minimize\ ||(LR^T)\cdot Q-B||_F^2+\zeta(||L||_F^2+||R||_F^2)+\eta||(LR^T)S^T||_F^2$$

Another novelty of STCDG is the handling of empty columns. Empty columns may appear at the sink when the sampling ratio is low or the packet loss rate is high. The authors point out that such cases can lead to very high number of recovery errors. Therefore, the algorithm ignores empty columns first and recovers the original matrix. The short-term stability features populate the empty columns. Additionally, abnormal values sensed by sensor nodes are transmitted independently of the static sampling ratio.

Advantages and Limitations: The authors test STCDG against EDCA and CDG on three different datasets in terms of recovery error, power consumption, and network capacity. The tests used the Intel Berkeley Research lab [27] sensor data, a synthetic trace generated with the ns-2 simulator [5], and the residential building data gathered by the authors. Specifically, the authors subdivided the laboratory and residential building data sets into singular sensed phenomena, namely light and temperature. Normalized Mean Absolute Error (NMAE) was used to measure the recovery performance of the algorithms. The authors found that the algorithms have critical sampling ratios, which if surpassed, lead to a high NMAE. STCDG performs better than the other techniques, having low NMAE (sub 0.1) with low sampling ratios (0.1 - 0.2). CDG is the lowest performer on all metrics. The authors state that the sparsity feature is not always present in real life data sets. The performance of all algorithms drops on datasets with low temporal/spatial correlation and few sensor nodes (24 and 54 sensor nodes in the residential building and laboratory datasets, respectively). The authors argue that CDG is outperformed at high sampling ratios by a centralized

exact scheme, which sends all sensor readings at every time slot to the sink. STCDG and EDCA exhibit similar energy consumption.

3.4 Discussion on Adaptive Compression

In § 3.3, we present algorithms that focus on compressing the results of sensor sampling while retaining data quality, instead of performing filtering. In Table 1 the average message overhead of the category is *low*. We categorize all adaptive compression algorithms in the low category, since this is the main goal of compressing the original signal. In general, the algorithms are suitable for large-scale networks and they are resilient to packet loss. They are suitable for large networks since the benefit of reduced network usage scales well to the number of nodes while helping reduce energy expenditure from the relayed messages. Resiliency to packet loss is relevant to the number of erroneous messages saved by transmitting only a subset of the messages.

4 ADAPTIVE FILTERING

The category of *Filtering* contains algorithms that primarily focus on the reduction of the volume of data sent through the network after they are sampled. The category includes only value-filtering algorithms.

In Figure 2, two data streams are shown to exhibit the same pattern under different context. Sampling algorithms should be able to adjust their thresholds at run time in order to capture the importance of the sensed phenomenon and filter the second pattern, since it is common for that stream. The example highlights the benefits of adaptive filtering, i.e., *fewer transmissions* and *higher precision* in preserving context. The surveyed algorithms of the Filtering category aim for these benefits but with different approaches. For this reason, we further split the Filtering category in sub-categories.

The algorithms are classified further in two sub-categories, *model-based filtering* and *adaptive thresholds*. Model-based filtering algorithms use probabilistic models to suppress sensor readings, if the readings can be predicted with acceptable accuracy at another sensor node or sink. The Adaptive thresholds subcategory includes techniques that suppress sensed values based on temporal and spatial correlations between sensor nodes in the network.

4.1 Model-Based Filtering

We categorize filtering algorithms as *model-based* if they use a prediction model to filter sensor data at the sink level or even stop data transmission altogether. For the remainder of this subsection, we expand on the selected algorithms.

4.1.1 Barbie-Q (BBQ). The algorithm of Deshpande et al. [13] comprises a query processing engine that answers user queries to a sensor network until some error threshold is met. The system utilizes a prediction model where sensed values are transmitted through the network only if the prediction is not considered as accurate enough.

Assumptions: *BBQ* employs a model-based scheme in order to answer user queries. The authors argue that while easing the communication load on a network, statistical models may identify faulty sensors and fill missing values in the network by extrapolation of missing data. The authors use a time-varying multivariate Gaussian model in BBQ, however they point out that their framework is model agnostic.

In BBQ, model construction uses historical data; therefore initial values need to be collected to initialize the model. Afterwards, users may query the network, e.g. ask for the temperature sensed by a group of sensors, with an error margin and a confidence interval. BBQ answers the query while minimizing the number of sensor nodes asked. Based on the underlying model, the system builds an observation plan, which specifies *how* and in *which* order the sensor nodes are queried. The reason behind this is that some attributes, like temperature and voltage, are highly correlated. The authors show that the correlation can be leveraged by the model, as voltage may be cheaper to sample than temperature. Additionally, spatial and temporal correlations are also leveraged in order to reduce the number of sensors queried.

Advantages and Limitations: The authors evaluate BBQ and compare it with TinyDB [28] and *Approximate-Caching* on two datasets collected from sensors (11 sensors and 54 sensors each). The concept of Approximate-Caching is introduced in the paper and is inspired by work by Olston et al. [32]. TinyDB is a sensor network query system in which the readings of sensor nodes are routed through the network and aggregated en-route to the sink. With Approximate-Caching, sensor nodes send their sensed values to a sink only if they differ from the previous values by user-defined margin ϵ. In comparison, TinyDB reports the exact values for a query while Approximate-Caching is tweakable with error bounds. The acquisition costs of TinyDB are constant while BBQ outperforms the other algorithms on high error margins ($\epsilon = 1$). Acquisition costs are an order-of-magnitude lower than those of Approximate-Caching. As the authors point out, BBQ is not suitable for anomaly detection since it requires constant sensor sampling.

4.1.2 KEN. The algorithm of Chu et al. [12] targets queries where *every* sensor node in the network has to report values. No data reduction takes place and such queries are thus expensive, with regards to energy consumption and communication.

Assumptions: *KEN* employs a prediction model that synchronizes at sink and sensor node levels. Users query the sensor network with sampling rate and error interval arguments, e.g., values from all sensor nodes every f seconds with an error margin of $\pm\epsilon$. A sensor node collects values and checks if the sink can predict the accumulated values correctly within $\pm\epsilon$. If the values lie within the interval, the sensor node suppresses its reading. If the values do not lie within the interval, the sensor node pushes its values down the network to the sink. The models at the sink node use the new values to *re-synchronize* with the model at the sensor nodes. In a multi-hop sensor network, additional compression based on spatial correlation may take place between different hops. In order to enable compression based on spatial correlation, Chu et al. [12] propose a novel clustering scheme. In this scheme, there are multiple clusters of sensor nodes with a single cluster head that communicates directly with the sink.

Advantages and Limitations: The scheme reduces communication overhead, as sensor readings do not have to be stored centrally in the network. Each cluster head maintains a synchronized prediction model with its child nodes. Additionally, the sink maintains prediction models synchronized with the cluster heads. To find such clustering groups, KEN uses a heuristic approach where sensor nodes are assigned to a cluster head from a performance

indicator for the model. The cluster formation algorithm runs at the sink, which induces communication overhead if clusters have to re-organize. The authors evaluate their approach on two real-world datasets with a small, fixed amount of sensors (one with 11 and the other with 49 sensors). KEN is tested with the clustering technique enabled and with an average model, where predictions utilize the average of all sensed values. No clusters are built or re-organized. The authors conclude that KEN performs better with the clustering scheme, when the communication costs to the sink are higher than communicating with a neighbor. KEN is not able to handle the presence of outliers, since it requires re-construction of the model.

4.1.3 Adaptive Sampling Approach (ASAP). *ASAP* [18] expands on KEN [12] and addresses its major issues. ASAP deals with cases where the observed phenomenon changes unpredictably over the course of time, which is not covered in detail with KEN.

Assumptions: In cases where the observed phenomenon is unpredictable, the prediction model has to adapt, which introduces high communication overhead if the model is constructed centrally at the sink. Additionally, the authors of KEN did not take into account the added energy expenditure of cluster heads, compared to regular sensor nodes. ASAP addresses these issues in its implementation and organizes sensor nodes into clusters. Clustering is done periodically, i.e. every τ_c seconds. This rotates the cluster heads and prevents premature power outage of a sensor node. The formation of clusters and the election of cluster heads is done inside the network, without the mediation of the sink. Each sensor node computes the probability of becoming a cluster head, based on a user set percentage of sensor nodes becoming cluster heads (f_c) and the relative energy level from neighboring sensor nodes. Sensor nodes that did not become cluster heads choose a cluster head by an attraction factor. The attraction factor is a combination of the hop distance to the cluster head and a weighted data similarity with the readings of the cluster head and the sensor node, α. After cluster construction, cluster heads divide their clusters into sub-clusters, of size β, based on correlations between every sensor node in the cluster. Sensed values of all cluster nodes are collected in the cluster head. This is repeated every τ_f seconds to update correlations and adapt to the changing dynamics in an observed phenomenon. The cluster head selects the fraction σ of sensor nodes to act as samplers in a sub-cluster, based on the remaining energy of each sensor node. Only the sampler nodes and the cluster head sense the environment (with a sampling rate of τ_d) and communicate their sensed values to the sink. Additionally, for each sub-cluster, cluster heads compute a data mean vector and a covariance matrix, which they send to the sink as model parameters. The sink receives the sensed values from the sampler nodes and predicts the values for the rest of the sensor nodes.

Advantages and Limitations: Message communication cost, network performance, energy consumption and data quality are the main metrics of the tests. Centralized exact and extreme variations of ASAP are used for comparison. The variations are two-fold, one local and one centralized approach. In the local approach predictions happen at the cluster heads and predicted values are sent to the sink. In the central approach all predictions are carried out at the sink level and values of all sensor nodes for updating the model are also sent to the sink. The authors find that ASAP outperforms other algorithms in the number of messages sent per second as well as the per-sensor-node energy consumption, while alternating σ. The authors study the trade-off between the prediction error and network lifetime and observe high lifetime improvements (90% longer lifetime in comparison to centralized exact), if user-defined absolute error thresholds are close to 1.

ASAP is best suited for environmental monitoring as anomaly detection is not feasible due to only a subset of sensors being active at a time. Users should allow some prediction error in order to utilize ASAP effectively. In the paper, the authors suggest an alternative configuration that minimizes overhead.

4.1.4 Algorithm of Jiang et al. The algorithm of Jiang et al. [22] is based on the computational overhead of prediction schemes, which outweights energy savings of predicting a value at a sink, instead of sending the value through the network. The algorithm utilizes clustering and duty cycling for saving energy.

Assumptions: A sensor network is divided into clusters. Sensor nodes that are not cluster heads, are either asleep, awake, or sensing the environment. An *asleep* node is inactive (e.g., powered off) while an *awake* node is waiting for commands. A *sensing* sensor means that the device converts a phenomenon to values. Sensor nodes hold a history of predicted or sensed data points, while cluster heads have a history of the values from all sensor nodes in their cluster. Based on the historical data, an auto-regressive model can be trained to predict data locally at sensor nodes and cluster heads. Cluster heads issue prediction bans to sensor nodes if a local prediction is less energy-efficient than communicating the values to the cluster head. The authors want to prevent sensor nodes from computing a prediction if the prediction is not accurate enough since the sensed data still has to be sent. In these cases, the power used for the prediction is wasted. If no ban is issued and the predicted value lies in the user-specified error bound, the sensor node updates its local historical data with the value but does not transmit it. If a ban is issued, the affected sensor node sends sampled data to the cluster head and updates the local historical data with the sampled data. The authors point out that for applications with data loss, acknowledgment messages can be sent from sensor nodes to cluster heads.

Advantages and Limitations: The authors evaluate their algorithm on a synthetic dataset by varying the ratio of transmission energy consumption and prediction energy consumption. They compare their algorithm with and without the prediction ban feature and conclude that additional energy savings may be achieved when prediction costs are higher than communication costs. The authors claim that the algorithm does not focus on cluster creation and other algorithms, e.g., ASAP, are more suitable for such a task.

4.1.5 Spanish Inquisition Protocol (SIP). Under *SIP* [19], a sensor node transmits a value only if the receiver node *does not expect it*. Each node transmits values *only* when the observed stream contains values that the receiving node could not predict.

Assumptions: SIP is based on the core notion of a Dual Prediction Scheme (DPS) algorithm. DPS algorithms expect the same phenomenon model to be present in every pair of *source - sink* nodes. SIP transmits a state vector *estimate* instead of every observed reading. The basic trade-off of SIP lies between the accuracy of recorded values and number of transmitted packets, with lower accuracy thresholds leading to lower number of transmissions. For the estimation of the state to be transmitted, SIP utilizes a number

of methods, namely Kalman Filters, Normalized Least Mean Squares (NLMS), or Exponentially Weighted Moving Average (EWMA). The current sampling rate, acceptable error threshold, as well as the data itself affect the reduction of future transmissions.

Advantages and Limitations: SIP aims to reduce the overall energy consumption of sensor nodes, throug the reduction of the number of transmitted packets. SIP provides an accurate depiction of the original data stream with just less than 5% of the original samples. For the Intel Lab data set [27], SIP transmitted packets 10 times less than other schemes. These advantages are affected when different versions of networking stacks are used on the sensor nodes [17]. SIP needs careful planning with regards to the underlying networking stack since the gained tranmission benefits range from 10-fold to just 21%, in total.

4.2 Discussion on Model-Based Filtering

In § 4.1, we gave details on algorithms that focus on the suppression of the results of sensor sampling through the use of various model prediction schemes. As seen in Table 1, average message overhead of the category, based on our classification, is *low*, since message suppression is the focus. The only exception is KEN [12], where further communication is needed when constructing the models in unpredictable phenomena. The examined algorithms suit networks with slow or no topology changes, since the filtering process depends on the spatiotemporal characteristics of the node. Model re-construction contributes to this intricacy as well, since edge nodes may not be capable to aggregate and perform adequate training of the model. In general, the algorithms offer a large reduction in energy expenditure and network transmissions.

4.3 Adaptive Thresholds

Algorithms fall into the category of *adaptive thresholds* if, and only if, they only deal with the reconfiguration of a threshold for their filter function, during execution. In the remainder of the subsection, we expand on the four selected algorithms.

4.3.1 Algorithm of Meng et al. The algorithm of Meng et al. [31] focuses on in-network message suppression, based on Contour Mapping. The approach exploits the spatial and temporal correlation in the messages.

Assumptions: The authors describe Contour Mapping as a technique where data points in a diagram are connected based on similarities. Step size controls the factor similarity. The authors use this technique to construct Contour Maps in sensor networks, where sensor nodes with similar readings do not have to transmit data. Sensor nodes suppress readings locally while sinks interpolate suppressed readings. For that reason, sensor nodes sample their sensors every τ seconds, where τ is between τ_{min} and τ_{max}. The sampling period is relative to the magnitude of the sensed value. An initial randomness factor is necessary since some sensor nodes may report their readings first while neighboring sensor nodes listen for those readings. A sensor node that overhears the values of its neighbors decides if it will transmit its readings, based on the distance of its data to the average δ of the neighbor data. Additionally, a sensor node may compare sampled values to m previous sampled values and based on the same threshold δ, it suppresses or transmits its readings. Sensor nodes consume some energy while overhearing messages sent by neighbor nodes. The sinks know the geographical location of the sensor nodes as well as which sensor nodes (silent nodes) suppressed their readings. Therefore, the missing values of the silent nodes are interpolated by the average of the readings of nearby sensor nodes. Additional smoothing may be applied on the data by using the values of neighbors that are more than one hop away from a silent node, as weighted averages.

Advantages and Limitations: The authors test the accuracy and energy consumption of the algorithm in a simulated environment with 528 sensor nodes, from which only 92 sent readings to the sink. The maximum deviation error of values, i.e., the distance between the value assigned to a silent node by the sink and the actual value, did not exceed 10. With the introduction of 20% network loss, 17 readings out of 92 were dropped on the way to the sink. In that case, the maximum deviation error was 20. Energy savings were compared against a centralized exact scheme while transmitting data, listening for neighbors data, and receiving data. The algorithm of Meng at al. [31] achieves $\approx$ 88% better energy savings over centralized exact in each category.

4.3.2 Algorithm of Solis et al. The algorithm of Solis et al. [38] leverages contour mapping to generate maps of environmental phenomena. Similar to the algorithm of Meng et al. [31], the algorithm suppresses the reporting of readings at sensor nodes which are in the same isocluster, with no isoline between them.

Assumptions: The space between isolines depends on the application. Less space equals to higher data resolution since potential isolines exist between sensor nodes. This comes at the expense of energy as more readings are communicated. The area between isolines is user configured and propagates to every sensor node. An isoline moves if a sensor node senses a reading that differs from a previous one. The sensor node broadcasts the change in isolines to its neighbors. Readings are scheduled from the farthest leaf nodes (in the case of a tree topology) to the sink. This enables additional energy conservation as sensor nodes go into a sleep mode after sampling. Solis et al. [38] indicate that further temporal suppression is applicable as sensor nodes only report a single reading, if that reading changed the isocluster.

Advantages and Limitations: The authors evaluate their approach in a simulated environment against other approaches, e.g., an aggregation scheme, where data is aggregated at parent nodes into groups and the averages of the groups are sent in the network. The other schemes are centralized exact approaches, with and without additional temporal suppression. The authors find that their approach outperforms the other algorithms in energy saving as well as accuracy, when constructing contour maps from the readings.

4.3.3 Conch. The algorithm of Silbersten et al. [36] focuses on the reduction of reporting messages of a monitoring system. *Conch* leverages the differences of consecutive sensor node values, e.g., when two sensor node readings differ significantly. When one reading may be inferred from another, no values are transmitted.

Assumptions: In Conch, a sensor node has a set of sensor nodes from which it receives updates (updaters) and a set of sensor nodes to which it updates to (reporters). A sensor node broadcasts its sensed values to the reporters, if its new sensed value differs from the old sensed value by some margin. When a sensor node receives a value from an updater node, it computes the difference to its own

sensed value. If no value is received, the sensor node assumes no changes. The sink monitors all differences between sensor nodes (called *edges*), so initially all reporters send their edges to the sink. Some sensor nodes are monitored directly. The directly monitored sensor nodes route their sensed values and their edges to the sink directly. At every time step, which is derived from a pre-configured sampling rate, the sink receives updated edges from reporter nodes and sensed values from directly monitored sensor nodes. This enables the sink to compute the value of every sensor node at any time step, if every sensor node is reachable from a directly monitored sensor node and the corresponding edges.

Advantages and Limitations: Conch is a monitoring algorithm; routing schemes are used to route a report from a sensor node to the sink efficiently. The authors [36] consider Conch as flexible since they present ways to build *Conch plans*, i.e., monitoring topologies, to focus on minimizing energy consumption or increasing sensor node and message failure resilience. The authors tested Conch in different simulated environments against multiple algorithms and on SNs with different sensor node densities, predictable increases of the observed value, and outlier detection. Conch outperforms all other schemes in terms of energy consumption at every test and only the algorithm of Meng et al. [31] exhibits similar energy consumption with increasing sensor density.

4.3.4 CME. The algorithm of Xu et al. [47] leverages contour mapping and utilizes binary classification and clustering to detect contours and reduce the total number of transmissions from sensor nodes. In contrast to Solis et al. [38], CME employs node clustering to create contour maps.

Assumptions: In CME, a contour is a curve that connects points with equal feature values. The range between two contours is application specific. *Contour nodes* describe sensor nodes that have a reading that belongs to a different contour range than the reading of neighbor sensor nodes. All nodes participate in contour node identification in a cluster, by broadcasting their sampled values. Only contour nodes communicate their readings to the cluster head. CME employs Support Vector Machines (SVM) at the cluster head to classify nodes membership to one side of the contour. The cluster head computes the contour segment of its cluster and then it sends it to the sink, where the contour map of the network may be constructed. Xu et al. [47] point out that in a multi-hop routing scheme, additional aggregation may take place on routing cluster heads.

Advantages and Limitations: The authors evaluate CME in a simulated event detection scenario with 2500 sensor nodes. They compared CME against a centralized exact implementation and Solis et al. [38]. The authors test CME on energy consumption and contour map accuracy, while varying contour steps and network sizes. CME outperforms every algorithm in every test and exhibits similar accuracy with lower energy expenditure.

4.4 Discussion on Adaptive Thresholds

In § 4.3, we detailed our selected algorithms that focus on the suppression of the results of sensor sampling through the reconfiguration of filtering thresholds. In Table 1 the average message overhead of the category is *medium*, based on our own classification. The examined algorithms suit networks with slow or no topology changes, due to the process being dependent on the spatiotemporal correlation of the produced signals. Adaptation of the thresholds plays a large role in the volume of the generated traffic and in the quality of the resulting data stream. In general, the algorithms offer flexibility between energy efficiency and number of network transmissions, with the risk of over/under shooting the threshold selection significantly. Finally, they require prior knowledge of the topology. We expect the algorithms of Adaptive Thresholds to have lower adaptability to the sampled stream compared to the category of Model-Based Filtering. In contract, we expect the algorithms to allow for better energy expenditure on the sensor nodes since they are not as expensive computationally.

5 HYBRID ALGORITHMS

In this section, we present a set of algorithms that combine techniques; thus considered as *hybrid*. The algorithms in the subcategory combine sampling or filtering with other approaches.

5.1 Algorithm Descriptions

For the remainder of the section, we will go through the selected algorithms that we consider as hybrid approaches.

5.1.1 AdAM. By incorporating an adaptive sampling and an adaptive filtering algorithm in the same algorithm, *AdAM* [42] provides a monitoring framework for the IoT.

Assumptions: While AdAM runs on sensor nodes, a data stream M reduces the number of sampling periods when the metric stream does not fluctuate and vice versa. A sampling period T_i is computed by estimating the metric stream evolution. Trihinas et al. [42] use a Probabilistic Exponential Weighted Moving Average (PEWMA) to produce an estimated metric stream M'. PEWMA is a variation of Exponential Weighter Moving Average (EWMA) that provides a *one-step-ahead estimation*.

Advantages and Limitations: The authors state that PEWMA is more robust against abrupt transient changes in the metric evolution than EWMA. When M' differs from M by a user-specified imprecision value γ, then T_i increases to a maximum sampling period T_{max}. Otherwise, T_i decreases to a minimal sampling period T_{min}. Sampled data points v_i are temporally suppressed if they lie in a user-specified interval $v_i \in [v_{i-1}-R_i, v_{i-1}+R_i]$, where R_i is the adaptable filter range. With AdAM, an adaptive filtering range is computed at a sensor node in every time step i. Fano Factor is used to measure the variability of the data stream at a current timestep. If the variance of the data stream increases, then the Fano Factor increases as well. The Fano Factor is compared against γ. R_{i+1} is shortened if the Fano Factor is greater than γ and widened if the Fano Factor is less than γ.

5.1.2 FAST. The goal of *FAST* [16] is enabling private and continuous streams of aggregate information for data mining purposes. It uses *sampling* in order to extract selected values in time series. It utilizes *filtering* in order to predict dynamically non-sampled values as well as corrections for the sampled values.

Assumptions: FAST includes an adaptive sampling component, based on a *Proportional-Integral-Derivative (PID)* controller [3] to adapt the sampling interval T.

A PID controller achieves a desired result (e.g., desired speed) by continuously adjusting a configuration (e.g., acceleration). The

Algorithm		Assumptions	Advantages	Limitations	Message Overhead
Adaptive Rates	*Backcasting* [46]	Spatial and temporal correlation of signal	Random sensor deployment	Death of cluster heads	High
	Jain et al. [21]	Continuous sensor data streams	Important data receives more bandwidth	Only single-hop SN	High
	USAC [33]	Fixed number of sensors	Captures sudden changes	Static confidence interval	Low
	EASA [39]	Sensing consumes more energy than communicating	Adaptive to energy levels	Complex computations	High
Adaptive Compression	*CDG* [26]	Knowing where observed signal is k-sparse	Minimal packet loss	Unsuitable for small SN	Low
	EDCA [10]	Matrix of collected values exhibits low-rank features	Robust against packet loss	Empty columns reduce recovery accuracy	Low
	STCDG [11]	Low-rank and short term stability of matrix	Adaptable since SN-type is independent	Unsuitable for small SN	Low
Adaptive Thresholds	*Meng et al.* [31]	Events are sensed by more than one sensors	Suppression is computationally simple	Small difference sensor reads are not leveraged	Medium
	Solis et al. [38]	Spatial correlation of sensor readings	Simple computation at sensor nodes	No contour map calculation given	Medium
	Conch [36]	Spatial and temporal correlation of signals	Trade-off network robustness for energy	Sensor nodes must know network topology	Low
	CME [47]	Stationary sensor nodes	Enables scalability	Death of cluster heads	Medium
Model-Based Filtering	*BBQ* [13]	Slow topology changes	Exploit correlation between sensor and voltage levels	Unsuitable for anomaly detection	Low
	KEN [12]	Slow topology changes	Suitable for anomaly detection	Prediction models constructed on sink-level	Medium
	ASAP [18]	Forgo data quality for less energy expenditure	In-network construction of models and clusters	Unsuitable for anomaly detection	Low
	Jiang et al. [22]	Forgo data quality for less energy expenditure	Prevents energy costs and inaccurate predictions	Unsuitable for anomaly detection	Low
	SIP [19]	Transmit only values that the sink does not expect	Large reduction of transmissions, energy	Homogeneous networking stack	Low
Hybrid Algorithms	*AdAM* [42]	Hybrid of adaptive sampling and filtering	Easily tunable for performance	Trade-off accuracy for efficiency	Low
	FAST [16]	Execution of PID controllers and Kalman Filters is possible	Anticipate changes in sampled values	Domain knowledge needed for tuning filters	Low
	Chatterjea et al. [9]	Sampling rate adapts to the predictions of the model	Event detection with high probability of success	Unsuitable for time-critical applications	Low

Table 1: Table of presented Algorithms

controller uses an error (e.g., the difference between desired and current speed) to calculate three terms: the *proportional (P)*, *integral (I)*, and *derivative (D)* term.

P is proportional to the current error. The larger the error, the larger the configuration change. *I* integrates over past values of the error. Thus, the longer an error persists, the larger the configuration change. *D* operates based on the change in the error. The smaller the change, the more dampening occurs to prevent overshooting.

FAST uses a *Kalmann Filter Prediction* procedure [23] to compute a prediction (x_{i-1}) of the metric evolution of M. After reading a new value, a *correction* mechanism updates x_{i-1} to x_i. The error between x_{i-1} and x_i is calculated as follows:

$$E_i = |x_{i-1} - x_i| \ / \ x_i$$

The error E_i is the input of the PID-Controller used by FAST:

$$\Delta_T = C_p E_i + \frac{C_i}{k} \sum_{j=i-k}^{i} E_j + C_d \frac{E_i - E_{i-1}}{T_i}$$

k is the number of previous error values that is considered in the integral term. C_p, C_i, and C_d specify the weight of the proportional, integral, and derivative term. The result (Δ_T) and two pre-configured interval adjustment parameters (θ and ξ) allow for computing T_{i+1}:

$$T_{i+1} = T_i + \theta(1 - e^{\frac{\Delta_T - \xi}{\xi}})$$

Advantages and Limitations: FAST combines PID controllers with Kalman Filter Prediction. FAST anticipates the changes in the sampled values and reduces the gravity of perturbation errors. Such errors may be introduced by any differential privacy mechanism. FAST combines any noisy observations with a Kalman predicted value. The resulting value is used as feedback to the system itself, for further correcting future predictions as well as adjusting sampling.

The Kalman filter needs noisy input data to accurately predict values. Fan et al. [16] state that domain expert knowledge is needed in order to further tune the filter threshold `R`. `R` is the Gaussian measurement noise used to approximate the Laplace perturbation noise and highly depends on the nature of the data.

5.1.3 Algorithm of Chatterjea et al. The algorithm of Chatterjea et al. [9] leverages model-based prediction and adaptive sampling to reduce energy expenditure in energy-hungry sensor networks. A sensor node is energy-hungry when computation tasks consume more energy than communication tasks.

Assumptions: Every sensor node fills a buffer r with a user-specified length with sampled values at a user-specified sampling period. Afterwards, a time series model predicts an amount from a sample at each consecutive period. If the prediction falls within a user-specified error margin δ, a counter is incremented by one, specifying the number of next sampling periods to skip. The maximum value of the counter is based on the number of neighbors that can detect an event, for a specific sensor node. Successful predictions are stored in the buffer and if predictions do not satisfy the error margin, the buffer stores the sampled values and the counter is reset. Additionally, the sink stores the prediction models for every sensor node. Initially all sensor nodes send their model parameters to the sink and keep a copy of the model locally. If a sensor node detects an inaccurate prediction at the sink level, it sends an updated sink model to the sink.

Advantages and Limitations: The algorithm of Chatterjea et al. [9] detects events with high success probability but the detection

may be untimely and introduce latency, from the occurrence of the event itself until it is reported. The authors point out that their algorithm is not suitable for time-critical applications.

5.2 Discussion on Hybrid Algorithms

In § 5.1, we elaborated on the selected algorithms that combine techniques under a single implementation. The essence of the works of Adam [42], FAST [16], and Chatterjea et al. [9] is that multiple techniques may be used to reduce sampling in periods of low variability of an observed signal and further reduce communication load by suppressing already sampled values or predicting them at the sink level. For this reason, all algorithms are under the *low* class of message overhead in Table 1. The algorithms in the category expose a way to tune between efficiency and accuracy. Compared to other categories, they allow a certain level of configuration, where the total behavior changes, depending on user input. One interesting topic of research is to further tune the required configuration, based on the observed phenomenon and any trade-off thresholds.

6 CONCLUSION

In this paper we present a catalogue of dataprive sampling and adaptive filtering algorithms for sensor data. We show state-of-the-art algorithms that address scalability challenges in real-time, extremely distributed sensor networks. The algorithms are categorized into *adaptive sampling*, *compressive sampling*, *model based filtering schemes*, and *adaptive filtering*. We additionally show *hybrid algorithms*, which combine techniques in a single approach. We summarize our findings in a compact taxonomy and sum up the evaluation of the algorithms in a table. With our work, we aim to help researchers that work in the intersection of sensor networks and distributed stream processing.

Acknowledgements: This work has been supported by the European Commision as FogGuru (765452), the German Ministry for Education and Research as BIFOLD (01IS18025A, 01IS18037A), and the German Federal Ministry for Economic Affairs and Energy as Project ExDra (01MD19002B).

REFERENCES

[1] I. F. Akyildiz, W. Su, et al. Wireless sensor networks: a survey. Computer Networks, 2002.
[2] C. Alippi, G. Anastasi, et al. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In MASS. 2007.
[3] K. H. Ang, G. Chong, et al. PID control system analysis, design, and technology. Transactions on Control Systems Technology, 2005.
[4] A. L. Aquino, O. S. Junior, et al. Musa: multivariate sampling algorithmfor wireless sensor networks. Transactions on Computers, 2014.
[5] S. Bajaj, L. Breslau, et al. Improving simulation for network research. Tech. rep., 1999.
[6] D. Boyle, B. Srbinovski, et al. Energy analysis of industrial sensors in novel wireless shm systems. In SENSORS. 2012.
[7] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. Foundations of Computational Mathematics, 2009.
[8] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. Signal Processing Magazine, 2008.
[9] S. Chatterjea and P. Havinga. An adaptive and autonomous sensor sampling frequency control scheme for energy-efficient data acquisition in wireless sensor networks. In DCOSS. 2008.
[10] J. Cheng, H. Jiang, et al. Efficient data collection with sampling in wsns: making use of matrix completion techniques. In GLOBECOM. 2010.
[11] J. Cheng, Q. Ye, et al. Stcdg: An efficient data gathering algorithm based on matrix completion for wireless sensor networks. Transactions on Wireless Communications, 2013.
[12] D. Chu, A. Deshpande, et al. Approximate data collection in sensor networks using probabilistic models. In ICDE. 2006.
[13] A. Deshpande, C. Guestrin, et al. Model-driven data acquisition in sensor networks. In VLDB. 2004.
[14] D. L. Donoho. Compressed sensing. Transactions on Information Theory, 2006.
[15] A. M. Elzanati, M. F. Abdelkader, et al. A collaborative approach for compressive spectrum sensing. In Handbook of Research on Software-Defined and Cognitive Radio Technologies for Dynamic Spectrum Management. 2015.
[16] L. Fan, L. Xiong, et al. Fast: differentially private real-time aggregate monitor with filtering and adaptive sampling. In SIGMOD. 2013.
[17] E. I. Gaura, J. Brusey, et al. Edge mining the internet of things. Sensors, 2013.
[18] B. Gedik, L. Liu, et al. Asap: An adaptive sampling approach to data collection in sensor networks. TPDS, 2007.
[19] D. Goldsmith and J. Brusey. The spanish inquisition protocol—model based transmission reduction for wireless sensor networks. In SENSORS. 2010.
[20] M. Healy, T. Newe, et al. Wireless sensor node hardware: A review. In SENSORS. 2008.
[21] A. Jain and E. Y. Chang. Adaptive sampling for sensor networks. In VLDB. 2004.
[22] H. Jiang, S. Jin, et al. Prediction or not? an energy-efficient framework for clustering-based data collection in wireless sensor networks. TPDS, 2011.
[23] R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.
[24] D. Laiymani and A. Makhoul. Adaptive data collection approach for periodic sensor networks. In IWCMC. 2013.
[25] A. Lazerson, D. Keren, et al. Lightweight monitoring of distributed streams. TODS, 2018.
[26] C. Luo, F. Wu, et al. Compressive data gathering for large-scale wireless sensor networks. In MobiCom. 2009.
[27] S. Madden. Intel lab data. https://web.archive.org/web/20200416023957/http://db.csail.mit.edu/labdata/labdata.html, 2004. [Online; accessed June 2, 2020].
[28] S. R. Madden, M. J. Franklin, et al. Tinydb: an acquisitional query processing system for sensor networks. TODS, 2005.
[29] M. Mahmudimanesh, A. Khelil, et al. Reordering for better compressibility: Efficient spatial sampling in wireless sensor networks. In SUTC. 2010.
[30] —. Balanced spatio-temporal compressive sensing for multi-hop wireless sensor networks. In MASS. 2012.
[31] X. Meng, L. Li, et al. Event contour: An efficient and robust mechanism for tasks in sensor networks. Tech. rep., 2004.
[32] C. Olston and J. Widom. Best-effort cache synchronization with source cooperation. In SIGMOD. 2002.
[33] P. Padhy, R. K. Dash, et al. A utility-based sensing and communication model for a glacial sensor network. In AAMAS. 2006.
[34] V. Raghunathan, C. Schurgers, et al. Energy-aware wireless microsensor networks. Signal Processing Magazine, 2002.
[35] S. Santini and K. Romer. An adaptive strategy for quality-based data reduction in wireless sensor networks. In INSS. 2006.
[36] A. Silberstein, R. Braynard, et al. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In SIGMOD. 2006.
[37] A. Singh, R. Nowak, et al. Active learning for adaptive mobile sensing networks. In IPSN. 2006.
[38] I. Solis and K. Obraczka. Efficient continuous mapping in sensor networks using isolines. In MobiQuitous. 2005.
[39] B. Srbinovski, M. Magno, et al. An energy aware adaptive sampling algorithm for energy harvesting wsn with energy hungry sensors. Sensors, 2016.
[40] P. Szczytowski, A. Khelil, et al. Asample: Adaptive spatial sampling in wireless sensor networks. In SUTC. 2010.
[41] J. Traub, S. Breß, et al. Optimized on-demand data streaming from sensor nodes. In SoCC. 2017.
[42] D. Trihinas, G. Pallis, et al. Adam: An adaptive monitoring framework for sampling and filtering on iot devices. In Big Data. 2015.
[43] M. C. Vuran, Ö. B. Akan, et al. Spatio-temporal correlation: theory and applications for wireless sensor networks. Computer Networks, 2004.
[44] C. Wang, H. Ma, et al. Adaptive approximate data collection for wireless sensor networks. Transactions on Parallel and Distributed Systems, 2012.
[45] J. Wang, S. Tang, et al. Data gathering in wireless sensor networks through intelligent compressive sensing. In INFOCOM. 2012.
[46] R. Willett, A. Martin, et al. Backcasting: adaptive sampling for sensor networks. In IPSN. 2004.
[47] Y. Xu, W.-C. Lee, et al. Cme: A contour mapping engine in wireless sensor networks. In ICDCS. 2008.
[48] Y. Yao, Q. Cao, et al. Edal: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks. TON, 2015.
[49] S. Zeuch, A. Chaudhary, et al. The nebulastream platform for data and application management in the internet of things. In CIDR. 2020.
[50] Y. Zhang, M. Roughan, et al. Spatio-temporal compressive sensing and internet traffic matrices. In SIGCOMM CCR. 2009.
[51] Y. Zhao, D. Guo, et al. Cats: Cooperative allocation of tasks and scheduling of sampling intervals for maximizing data sharing in wsns. TOSN, 2016.