Adaptive Hybrid Compression for Wireless Sensor Networks

Azad Ali, TU Darmstadt
Mohammadreza Mahmudimanesh, TU Darmstadt
Abdelmajid Khelil, TU Darmstadt
Neeraj Suri, TU Darmstadt

Wireless Sensor Networks (WSN) are often deployed to sample the desired environmental attributes and deliver the acquired samples to a central station, termed as the sink, for processing as needed by the application. Many applications stipulate high granularity and data accuracy that results in high data volumes. However, sensor nodes are battery powered and sending the requested large amounts of data rapidly depletes their energy. Fortunately, environmental attributes (e.g., temperature, pressure) often exhibit spatial and temporal correlations. Moreover, a large class of applications such as scientific analysis and simulations tolerate high latency for sensor data collection. Hence, we exploit the spatio-temporal correlation of sensor readings while benefiting from possible data delivery latency tolerance to minimize the amount of data to be transported to the sink. Accordingly, we develop a fully distributed adaptive hybrid compression scheme that exploits both spatial and temporal data redundancies and fuses both temporal and spatial compression for maximal data compression with accuracy guarantees. We present two main contributions: (i) *an adaptive modeling technique* that allows frugal and maximized *temporal compression* on resource-constraint sensor nodes by exploiting the data collection latency, and (ii) *a novel model-based hierarchical clustering technique* that allows for maximized *spatial compression* resulting into a hybrid compression scheme. Compared to the existing spatio-temporal compression approaches, our approach is fully decentralized and the proposed clustering scheme is based on sensor data models rather than instantaneous sensor data values, which allows merging nearby nodes with similar models into large clusters over a longer period of time rather than specific time instances. The analysis for computation and message overheads, the analysis for theoretical compressibility, and simulations using real world data demonstrate that our proposed scheme can provide significant communication/energy savings without sacrificing the accuracy of collected data.

Categories and Subject Descriptors: C.2.2 [**Wireless Sensor Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Model-based Clustering, Spatio-temporal Compression, Delay-tolerant Networks, Data Analysis.

1. THE PROBLEM AND THE APPROACH

In Wireless Sensor Network (WSN) deployments, battery powered sensor nodes are often distributed over the area of interest for varied applications involving unattended environmental monitoring and supervisory functions. The sensor nodes regularly update a central station, termed as the sink, with the sampled environmental data for subsequent processing and analysis. The (multihop) data delivery, from the nodes to the sink, requires exchange of several messages thus depleting the batteries of involved sensor nodes, and reducing the residual network lifetime. Hence, the dual target of continuous collection of data while prolonging the network lifetime is a challenging problem.

Fortunately, WSNs naturally generate highly redundant spatial samples due to (a) the redundant sensor node deployment for connectivity and failure tolerance, and (b) the correlated values of the environmental attribute over larger spatial areas. This is also substantiated by our observation of available real world data [Madden 2003], where we noticed that the trends/patterns are similar for a large number of nodes in a given physical neighborhood. Fig. 1 depicts the sensor readings of two pairs of nodes, from [Madden 2003], over four days time period. The nodes in each pair are 1-hop neighbors and both pairs of nodes are separated by several hops. The sampled attribute values also exhibit temporal correlation [Tulone and Madden 2006] that can be

exploited by constructing the models of the data and reporting the model parameters instead of raw sample values.

Scoping the Problem: Applications requiring continuous data collection utilize the data for two prominent use cases of (a) live/real-time decision making, such as surveillance, or (b) offline/delay-tolerant processing such as modeling, analysis [Tolle et al. 2005] and inference [Ali et al. 2009]. The focus of our current work is on delay-tolerant data collection. Various WSNs deployed for scientific monitoring [Werner-Allen et al. 2005; Akyildiz et al. 2005; Tolle et al. 2005] continuously harvest data for modeling, analysis and simulations. They generally tolerate a certain data collection latency. Hence, for such applications real-time data acquisition does not have precedence over the quality and quantity of the data. This *delay-tolerance* in reporting the data to the sink opens up a fundamental design flexibility in data collection to significantly improve WSN energy efficiency. We propose an adaptive hybrid compression scheme that explicitly exploits the delay-tolerance in data collection to compress the data both in time and space, which gives us considerable benefits over the real-time schemes that require frequent updates to be synchronized with the sink.

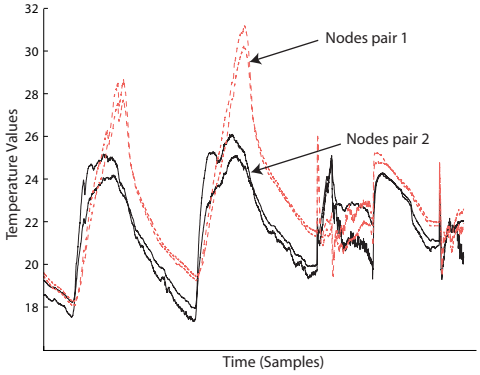


Fig. 1: Phenomenon Distribution

Compression Efforts: A number of approaches such as aggregation, temporal compression [Jindal and Psounis 2006; Zhao et al. 2002] and spatial compression [Deshpande et al. 2004; Zhang et al. 2006] have been proposed to reduce the data volume to be transported to the sink. However, only a few composite spatio-temporal approaches, such as [Gedik et al. 2007; Tulone and Madden 2006; Wang et al. 2012; Yoon and Shahabi 2007; Ali et al. 2011], exist. These approaches exploit both spatial and temporal correlation in data, thus reaching higher data traffic reduction. However, the existing approaches are (a) partially centralized limiting local decision making for adaptability, (b) tailored to specific attribute dynamics, providing none or limited self-adaptability, or (c) limited in their scope of exploiting either spatial or temporal correlation reducing compressibility. [Ali et al. 2011] is the only work to consider delay-tolerance in spatio-temporal data compression. However, it is limited due to the use of static models and limited scope for data compression.

The Basis for our Proposed Approach: We exploit temporal redundancies using simple, computationally inexpensive models considering the limited computation resources of the sensor nodes. The simple models can naturally only approximate a limited number of samples. Hence, in order to exploit the delay-tolerance in data collection, the nodes consecutively construct a batch of simple models to increase the number of approximated sample values. However, our scheme is optimized such that only a small number of nodes need to construct these batches of models to approximate the entire WSN. The spatial redundancy is exploited using a two-level hierarchical clustering. In contrast to existing literature [Jiang et al. 2011; Pham et al. 2010; Gedik et al. 2007], our proposed clustering is based on models instead of raw sample values. A model-based clustering allows us to evaluate the correlations across the nodes over a longer period of time in comparison to the sample value based clustering which determines correlation only at a given instance of time. Hence, the existing approaches require continuously maintaining large monolithic clusters, incurring additional en-

ergy overheads. Using our hierarchical clustering, we initially form 1-hop clusters by grouping nearby nodes with high correlation. The small 1-hop clusters require minimal maintenance. The model batches are constructed on a subset of the 1-hop clusters. The models constructed by these clusters are used to approximate the surrounding clusters based on the user defined error tolerance. The 1-hop clusters approximated by a few batches of models are merged to form the second clustering hierarchy. The second clustering hierarchy depicts the dynamic correlations between the 1-hop clusters and is reconstructed in each iteration. Following this scheme, only a few model batches approximate the entire network sample values over a long duration of time (depending on the delay-tolerance limits). The model batches are sent to the sink, which uses these models to regenerate the sample values of each sensor node in the network.

Contributions: The contributions of the current work are as follows

— We propose a fully decentralized scheme that does not require any intervention from the sink apart from a few initial parameters for setup.
— The proposed adaptive modeling compression scheme can approximate a wide variety of environmental attributes within the user defined accuracy requirements by dynamically adapting to tune the parameters (Section 4.3).
— We propose a novel technique of hierarchical clustering based on adaptive models for spatial compression. The clustering adapts to the distribution of the environmental attribute over the sensor network to maximize the spatial compression (Section 4.4).
— We propose a technique to exploit the delay-tolerance in data collection in various WSN applications to enhance the compression both in time and space.
— We present compressibility analysis with theoretical upper and lower bounds for compression (Section 5.2) and a detailed message complexity analysis (Section 6).

Building on our preliminary delay-aware compression ASTC approach [Ali et al. 2011], we now present a comprehensive framework applicable to a broad range of applications and with significantly enhanced bounds on data compressibility and message overhead efficiency. The paper is structured as follows. Following a discussion on related work in Section 2, we present the system model and design requirements in Section 3. The proposed hybrid compression scheme is presented in Section 4. Subsequently, Section 5 details the efficiency and compressibility analysis of the proposed scheme and Section 6 evaluates the message and the computation costs. Section 7 details the analysis and evaluation results.

2. RELATED WORK

This article focuses on continuous data collection rather than event-based WSN design. Our primary objective is to spatio-temporally compress the data for continuous data collection with sensor-node level granularity while exploiting delay-tolerance in data collection. While there is extensive research in WSN for data compression [Nakamura et al. 2007], e.g., simple aggregation [Fasolo et al. 2007], suppression [Zhou et al. 2008], filtering, TinyDB [Madden et al. 2005] and Cougar [Gehrke and Madden 2004], to name a few, however, there exists very limited work in spatio-temporal compression. We summarize some of the important research work regarding spatial and temporal compression and discuss in detail the existing spatio-temporal techniques.

Spatial Compression: The objective of spatial sampling is to collect interested attribute snapshots. The key idea behind spatial compression is to constrain neighboring sensor nodes with similar senor readings from transmitting redundant data. [Mahmudimanesh et al. 2010; Jindal and Psounis 2006; Solis and Obraczka 2005; Zhao et al. 2002] are a few spatial compression techniques relying on compressive sensing, spatially correlated models, filtering and aggregation. The main weakness of all pure spatial compression approaches is their focus on the spatial redundancy while neglect-

ing the temporal redundancy. Temporal resolution depends on the implemented snapshot update mechanism. Pure spatial compression techniques naively address temporal redundancy by creating a new snapshot or sending an incremental update.

Temporal Compression: The driver of temporal compression is to exploit temporal correlation present in the attribute values. The key idea in these approaches is to let every sensor node create a prediction model for its sensor readings and send the model to the sink. The sensor node should send an updated model only if the model is not valid due to changes in the signal dynamics. The approaches in [Mini et al. 2005] and [Zhang et al. 2006] construct models based on Markov-chains and time series, respectively. These temporal compression schemes mainly focus on the temporal redundancy. Some use limited forms of spatial compression (e.g., by constructing 1-hop clusters and allowing only cluster heads keep consistent models with the sink).

Spatio-Temporal Compression: There exists very limited research addressing spatio-temporal compression in WSN. In [Vuran et al. 2004], the authors developed a theoretical framework to model the spatial and temporal correlations in WSN. This framework enables the development of efficient medium access and reliable event transport protocols, which exploit these advantageous intrinsic features of the WSN paradigm. However, this work does not focus on continuous data collection as needed for the targeted applications. [Jiang et al. 2011; Pham et al. 2010; Gedik et al. 2007] proposes value-based clustering mechanisms that form large monolithic clusters based on instantaneous values or aggregates of the values. Hence, they need to continuously collect data from the member nodes to check for consistency and maintain these clusters by breaking up and merging them. This is expensive computationally and for energy consumption. Our alternative proposed approach uses hierarchical clustering based on models instead of instantaneous values, i.e., (1) 1-hop cluster rarely requiring maintenance, and (2) cliques that are not maintained rather reconstructed at a minimal message cost. [Tulone and Madden 2006] propose real-time data collection through repeated model construction and synchronizing them with the sink. [Tulone and Madden 2006] is limited both in spatial and temporal compressibility. [Wang et al. 2010] and [Wang et al. 2012] relaxes the spatial constraints of [Tulone and Madden 2006] to form larger than 1-hop clusters. Due to their focus on real-time updates they bounded to approximate only a limited amount of data in time, consequently limiting the temporal compressibility. On the other hand the focus of our proposed scheme are the applications that are delay-tolerant concerning the data collection, hence we can more efficiently compress the data both in space and time due to relaxed time constraints. [Villas et al. 2011] proposes values based scheme to exploit temporal and spatial correlation and explicitly focuses on the real-time reporting of the values. Our approach proposes a model based scheme to exploit the temporal and spatial correlation, extending the compressibility both in space and time. [Min and Chung 2010] uses Kalman filter for modeling within 1-hop clusters, which incurs heavy computation cost on a sensor node. [Baek et al. 2004] proposes to reduce energy consumption through hierarchical aggregation. Both [Min and Chung 2010] and [Baek et al. 2004] require the node location information, which could be prohibitively expensive. Our proposed scheme works independent of the location information and uses simple models that are easily computable on a sensor node. [Liu et al. 2007] and [Gupta et al. 2008] use centralized heuristics for cluster formation and nodes correlation determination. [Liu et al. 2007] assumes all sensor nodes to be within communication range of the sink, or, the deployment of dedicated nodes, to act as cluster heads, to be able to communicate directly with the sink. Such assumptions limit the applicability of the approach [Liu et al. 2007] to a generic WSN. In contrast, our proposed scheme assumes a generic WSN deployment with no additional requirement on communication range or of any dedicated sensor nodes. Moreover, it is completely distributed and does not require

any global information other than a few static parameters initially required from the sink to setup the proposed distributed scheme as per the user requirements. [Yoon and Shahabi 2007] has two modes of operation, i.e., interactive mode is limited to spatial compression only and streaming mode performs both spatial and temporal compression. [Wang and Deshpande 2008] and streaming mode of [Yoon and Shahabi 2007] both construct Probability Density Function (PDF) for modeling the attribute. AHC and probabilistic modelling have fundamentally different assumptions about the statistical process of attributes to be monitored. AHC assumes the statistical process to be dynamic and that the process statistics can change at any time. [Wang and Deshpande 2008] assumes the process statistics to remain the same and needs manual setup for appropriate functioning. In probabilistic modelling once the PDF has been constructed, it is assumed the PDF will remain the same for the rest of monitoring period and can be used to predict the future samples. Accordingly, correctness of such schemes can not be guaranteed. A PDF constructed in a certain time window can not be guaranteed to be valid after attribute dynamics change limiting the use of such schemes in long term continuous monitoring. They also require expensive long training time (e.g., 15 days for [Wang and Deshpande 2008]). We do not assume the system dynamics to be known in advance and dynamically construct the model based on the changes in WSN attributes. Therefore, our technique not only tracks any changing attribute dynamic but also adjusts in case of unexpected changes in the dynamics. In our prior work [Ali et al. 2011], we proposed spatio-temporal compression using models and clustering. In this article, we improve over [Ali et al. 2011] with flexibility to dynamically select a range of models instead of one fixed model, dynamic temporal compression scope and relaxed time synchronization as compared to strict time synchronization.

Another important class of compression methods for WSNs is based on *transform compression*. In transform compression, a linear transform is applied on the sensed signal that produces a more compressible version of the data and consequently reduces the amount of data that needs to be transmitted to the sink [Duarte et al. 2012]. Both transform compression and model-driven classes of data compression methods for WSNs take advantage of the compressibility of the sensed signal to reduce the amount of in-network transmissions, though using different approaches. In this paper, we study transform compression as an important related method for WSN data compression and compare our work with a representative implementation of transform compression. Distributed Transform Compression (DTC) methods (also known as Distributed Transform Coding techniques) are based on the fact that the raw data recorded from natural phenomena such as the data sensed by a WSN are compressible under certain linear transforms [Duarte et al. 2012]. After transforming and compressing the raw data, the compressed data are sent to the sink. The sink then applies the inverse transformation to acquire the original data.

In summary, the existing hybrid approaches require the signal and its statistical properties (dynamics) to be known a priori, require location information, are partially/completely centralized or use instantaneous values instead of models for clustering. The common factor among all contemporary schemes is that they target real-time/immediate data collection. None of existing works, other than [Ali et al. 2011], exploits the inherent delay-tolerance of many applications, and thus lose the efficiency enhancement potential. In contrast, our approach is adaptive, does not require location information, is fully decentralized, uses simple easily computable models and exploits the delay-tolerance to maximize the data collection.

3. PRELIMINARIES

3.1. System Model

We consider a conventional WSN system model composed of N static sensor nodes $\{S_1, S_2, \ldots, S_N\}$ and a static sink. We assume that the clocks of nodes are synchronized, e.g., using [Faizulkhakov 2007]. Sensor nodes sample the environment attribute simultaneously and periodically every τ time units. This synchronized sampling allows nodes to run duty cycling for maximized energy efficiency, which is out of scope of this article. The sink is powerful enough to store large amount of data. Sensor nodes are battery powered and possess limited storage and processing capabilities. The WSN deployment follows an arbitrary node distribution with varying spatial node densities as per connectivity, coverage, fault-tolerance and sensing requirements. We assume the availability of a reliable end-to-end data transport service, such as [Shaikh et al. 2010], to transport messages from sensor nodes to the sink, and acknowledgement schemes to ensure message delivery between neighboring nodes.

3.2. Design Requirements

Our objective is to maximize the spatio-temporal data compression ratio with accuracy guarantees for continuous delay-tolerant data collection (i.e., monitoring of phenomena or environmental attributes). In order to meet these objectives, we require that the proposed scheme must facilitate the reconstruction of the signal (sample values) of each sensor node from the collected compressed data on the sink, within the application-driven error bound, with a minimal bandwidth and energy overhead. The proposed scheme should adapt to evolving attribute dynamics, and be independent of the network properties such as node distribution, topology and routing protocols. The data compression models should be efficiently computable on resource-limited sensor nodes.

4. THE PROPOSED ADAPTIVE SPATIO-TEMPORAL COMPRESSION

We propose a decentralized adaptive hybrid compression technique that exploits application delay-tolerance. As observed in Fig. 1, nodes in close proximity of 1-hop distance generally exhibit persistently high correlations in sample values. However, correlation between the nodes farther away from each other, i.e., between 1-hop clusters, are generally non-persistent and dynamic when observed over a long time window. The asymmetry in correlations between the nodes in close proximity and the nodes farther away makes the spatio-temporal compression a challenging problem. The existing literature addresses this asymmetry by limiting the modeling to 1-hop clusters [Tulone and Madden 2006], assuming the attribute dynamics to be constant [Chu et al. 2006], requiring assistance from the sink [Liu et al. 2007; Gupta et al. 2008] or by clustering based on the instantaneous values rather than models [Jiang et al. 2011; Pham et al. 2010; Gedik et al. 2007].

We propose to exploit the spatial redundancy in two stages, i.e., (1) Proactive formation of 1-hop clusters that are usually highly correlated, (2) Merge 1-hop clusters to larger regions/clusters. Our spatial compression approach differs from contemporary approaches such as [Gedik et al. 2007; Yoon and Shahabi 2007]. These form large monolithic clusters making them vulnerable to frequent reconfiguration that incurs heavy maintenance overhead. We exploit temporal redundancy by constructing simple models on 1-hop clusters.

4.1. A Guide through the Proposed Adaptive Hybrid Compression (AHC) Scheme

Given the nature of WSN deployment redundancies and sensed attributes correlations, the proposed scheme performs spatio-temporal compression in three stages:

— Stage 1: 1-hop clusters formation
— Stage 2: Temporal modeling on clusters
— Stage 3: Merging 1-hop clusters

In Stage 1, neighborhoods of sensor nodes with correlated sensor readings form small 1-hop clusters based on a short history of the attribute values to exploit strong local correlation. Depending on the deployed sensor density the 1-hop cluster may consist of up to a dozen nodes.

In Stage 2, we exploit the temporal correlations by constructing the models on a small number of clusters, referred to as *master clusters* (depicted in Fig. 2 bearing crown). Each constructed model is initially limited to the respective master cluster and approximates the sampled values of all member sensor nodes of the master cluster.

In Stage 3, we propose mechanisms to utilize the models constructed on the master clusters to also approximate the sampled values of nodes in surrounding 1-hop clusters. The master cluster sends the model to its neighboring clusters. The cluster members fit the received model to their sampled values and accordingly either accept the model or reject it. The clusters accepting the model merge to form a correlated region (a larger cluster) and further propagate the model to the 1-hop clusters on the border of this region.

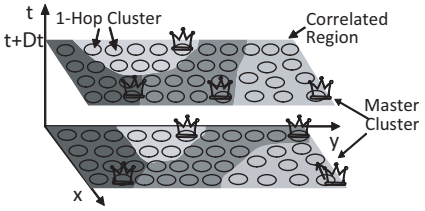


Fig. 2: Clusters, Master Clusters and Correlated Regions

Following this scheme, only a small set of the models constructed on master clusters can approximate the entire network both in space and time. The resulting spatial compression is a two level hierarchical clustering. The first and second hierarchy levels are formed during Stages 1 and 3. Stage 1 is executed only once, while Stages 2 and 3 are repeated to continuously model the sampled value and adapt to the changing dynamics. In the following, we detail these three stages.

4.2. Stage 1: 1-Hop Cluster Formation

In contrast to existing approaches, we initially form small 1-hop clusters instead of large monolithic clusters to exploit the strong local spatial correlations. The 1-hop clusters are subsequently (Stage 3) merged to form larger clusters to model the dynamic correlations between the 1-hop clusters. Constructing and maintaining large monolithic clusters, in comparison to smaller 1-hop clusters, would incur heavy maintenance costs as we explain further in Section 4.4. As cluster formation is a very well studied topic in WSN [Abbasi and Younis 2007], we only briefly describe the formation of 1-hop clusters.

Notation	Description
S_i	i^{th} Sensor node
S_{CH}	Cluster head
C_i	i^{th} Cluster
r	Cluster members count
C_N	Neighboring clusters list

Table I: Cluster Notations

1-hop clusters are formed based on the similarity of short history of the attribute values that are transmitted by the nodes that candidate to be the cluster head. All the nodes in the network are initially candidates for cluster heads. The sink issues the task of sampling the environment and sending the compressed sampled values back to the sink. Nodes wait for a random time t_{1h}. On the expiry of t_{1h} a node assumes the role of a cluster head and issues a "join request" along with a set of its sampled values to the 1-hop neighbors. Meanwhile, nodes receiving the join request can not issue the

join requests. If more than one node issue the join request within each others 1-hop neighborhood, the node with lower id withdraws its request. Additionally, nodes may still receive join requests from other nodes as a given node might be 1-hop member of many nodes that are not 1-hop neighbors of each other.

Nodes receiving the join request compare the received values with their own sampled value. Nodes join the cluster for which the difference between their sampled values and received values is within the accuracy requirement set by the application/user. Occasionally, the requests sent by the candidate cluster might not be received by its 1-hop cluster members due to collisions or hidden terminal problem [Tobagi and Kleinrock 1975]. Nodes unable to join a cluster, either because they did not receive the request or the error was too high, initiate their own cluster formation requests.

The later requesting nodes can be claimed by the already existing cluster heads if the cluster head finds the values of the requesting node to be within the given threshold to compensate for the lost request message. Accordingly, we define a 1-hop cluster (C_i) to consist of one cluster head node (S_{CH}) and 'r' member nodes (S_i) such that they are 1-hop away from the cluster head, i.e., $C_i = \{S_{CH}, S_1, S_2, \ldots, S_r\}, \wedge \forall S_i \in C_i \sqcup hopdist(S_{CH}, S_i) = 1$ (where $hopdist()$ returns the hop distance between two nodes). Members of one cluster might be able to listen to more than one cluster head but exclusively belong to one cluster, i.e., $C_i \cap C_j = \emptyset$, $i \neq j$.

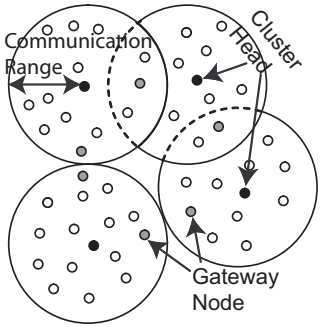


Fig. 3: 1-Hop Clusters

Each 1-hop cluster head discovers immediate 1-hop neighboring clusters (C_N) around it. In the discovery process, the cluster heads exchange the cluster ids (same as cluster head id) and hop distances of cluster head from the sink. They also identify the nodes that will be used to communicate between the two neighboring cluster heads referred to a gateway nodes as depicted in Fig. 3. The 1-hop cluster formation process is performed only once. We do not explicitly refresh 1-hop clusters in order to maintain the correlations between the nodes within these clusters. Instead, we design Stages 2 and 3 in an adaptive manner such that 1-hop cluster rearrangement takes place dynamically in response to the changes in the physical phenomenon, as detailed further in Section 4.4.5.

4.3. Stage 2: Temporal Modeling in 1-Hop Clusters

We now elaborate a temporal compression scheme to model the sampled values of a sensor node. We will use the developed temporal compression approach in Section 4.3.8 to model the sampled values of nodes in 1-hop clusters initially and extend the modeling beyond 1-hop clusters in Section 4.4. The scheme has been developed such that the model construction is carried out only by master clusters (selection of master clusters is explained in Section 4.4.1), while the rest of the clusters use the models constructed by the master clusters to approximate their sampled values.

Based on the requirements from Section 3.2, we require the models to be (a) computationally inexpensive to estimate, (b) require low memory, and (c) must be adaptive in order to fit varied attribute dynamics. Based on these requirements, we develop the temporal compression scheme.

4.3.1. Piece-Wise Modeling. The use of computationally expensive models to approximate generic patterns, typically observed in sampled attributes, is often not achievable on a sensor node due to (a) the limited nodes resources and (b) the unknown attribute dynamics. We formalize the temporal compression by representing the sampled data

Notation	Description
$v(t)$	Sample value at time t
$\hat{v}(t)$	Approximated values at time t
$V(t)$ or V	Sampled values queue for training models (training queue)
$\hat{V}(t)$ or $\hat{V}$	Estimated values
T	Training queue length
p	Model order of AR models
ϕ	Model parameters
Φ_i	i^{th} Model
W_Φ	Approximation window of a model
Ψ	Model cache (set of models)
W_Ψ	Approximation window of a model cache
$\hat{V}_\Psi$	Set of estimated samples by a model cache
$m_\#$	Number of models in Ψ
ϵ	Error threshold
$O_{\#_{max}}$	Maximum allowed number of outliers per model per node

Table II: Modeling Notations

of a sensor node as an infinite uni-variate time-series. In order to model the sampled values on a sensor node, we can decompose the sampled values (time series) into finite duration segments, that we term as training queue and denote as $V(t)$, which are used to estimate model parameters

$$V(t) = (v(t), v(t-1), v(t-2), \ldots, v(t-T+1)) \tag{1}$$

where $v(t)$ is data sampled by a sensor node at time t and T is the length of the training queue (Section 4.3.3). Each segment can be modeled as a linear and a random component.

4.3.2. Random Component Estimation. A prominent design goal is to support a wide range of applications. Consequently it needs to be designed without any specific dependencies on the the statistical process of the sampled attribute values. Hence, we do not make any assumptions for the underlying process other than requiring the process to be weakly stationary, which is generally true for physical processes [Ljung 1998]. This allows to model the random component as a linear difference equation of Auto-regressive Moving Average (ARMA) model. As Moving Average (MA) component of the ARMA model is relatively expensive to compute, hence, to reduce the computation complexity we limit the estimation to an Auto-regressive (AR) model, denoted as follows:

$$x(t) = \phi_1 x(t-1) + \cdots + \phi_p x(t-p) + w(t) = \sum_{i=1}^{p} \phi_i x(t-i) + w(t) \tag{2}$$

where $x(t)$ is the random component of the signal, $w(t)$ is white noise series with mean zero and variance σ^2 ($WN(0, \sigma^2)$), ϕ are model coefficients, p is the model order with $p \in \mathbb{N}$. We denote the model constructed using Eq. (2) as $\Phi(x)$ (or Φ in short).

4.3.3. Model Construction. In order to construct the model, a node maintains a training queue $V = \{v_1, v_2, ..., v_T\}$ of T sampled values. The AR model parameters are tuned by (a) estimating the fitting error ($e(\Phi)$), and (b) minimizing the estimation error. The AR

model fitting error $e(\Phi)$ is minimized in mean-square error sense as given by Eq. (3)

$$\frac{\partial}{\partial \phi_k} e(\Phi)^2 = \frac{\partial}{\partial \Phi} \left(\sum_{i=p+1}^{T} \left(x(i) - \sum_{j=1}^{p} \phi_j x(i-j) \right)^2 \right) = 0 \tag{3}$$

The sample values as approximated by the model are given by

$$\hat{v}(t) = \mu + \sum_{i=1}^{p} \phi_i (v(t-i) - \mu) \tag{4}$$

where $\hat{v}(t)$ is the estimated sample value and μ is the series mean. We denote $\hat{V}$ as the approximated values set, where $\hat{V} = \{\hat{v}_1, \hat{v}_2 ... \hat{v}_{W_\Phi}\}$, W_Φ is the number of estimated values referred to as approximation window (explained further in Section 4.3.6).

4.3.4. Piece-Wise Adaptive Modeling (AM) through Outlier Detection. As we assume the statistical process of sampled values to be unknown, the natural precondition is to have adaptability of the model according to the changing statistical dynamics. The adaptability can be achieved by providing feedback to the model to adapt to the new dynamics. This is expensive due to continuous updates transmitted to the sink with new model parameters elevating the message cost. Hence, we use an adaptive update algorithm based on outliers rather than the model parameters.

Outlier Detection: The sample values that cannot be approximated by the model can be tolerated and classified as outliers. If α is the maximum tolerable estimation error then the estimated value must lie between $[v - \alpha, v + \alpha]$. If an estimated value does not lie between the bounds, the node replaces the value with the original sampled value and classifies it as an outlier value. The outlier values should be reported to the sink separately for accurate signal regeneration. Nodes can determine an estimated value to be an outlier using Eq. (5):

$$\hat{v}(t) = \begin{cases} v(t), if\ |\hat{v}(t) - v(t)| > \alpha; \\ \hat{v}(t), otherwise. \end{cases} \tag{5}$$

The 1-hop cluster head gathers and reports the outliers of its cluster members to the sink to maintain accuracy within α. Fig. 4 shows the block diagram for adaptive modeling based on Eq. (5) (z^-1 represents the time delay).

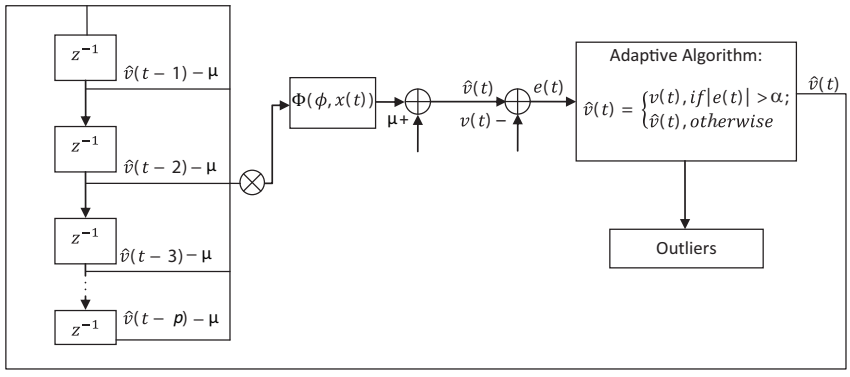


Fig. 4: Adaptive Modeling

The error upper bound and error probability associated with the estimated values can be given by Lemma 4.1 (the detailed proof can be found in [Tulone and Madden 2006]).

LEMMA 4.1. *Let $\alpha = \nu\sigma$, where ν is an application specified real constant larger than 1, the actual sampled value $v_i(t)$ is contained in $[\hat{v}(t) - \alpha, \hat{v}(t) + \alpha]$ with error probability of at most $1/\nu^2$.*

Model Invalidation and Outliers Upper Bound: A model becomes invalid if it can no longer approximate the sampled values within the defined accuracy bounds. Dissatisfying the accuracy bounds results into outliers that increase the message cost. We consider a model to be valid until it results into a maximum number of outliers ($O_{\#max}$). The nodes approximate the sampled values based on the constructed model using Eq. (4) and outliers are counted using Eq. (5). If the generated outliers for a given model are more than $O_{\#max}$ the model becomes invalid and a new model must be constructed. By adapting this scheme, we control the maximum number of the generated outliers, satisfy the accuracy requirement and avoid the unnecessary reconstruction of the model.

We have discussed the standard modeling technique, how to make it adaptive and proposed to use simple AR models to meet the minimal computation and memory requirements. However, we do not know what order of AR model will best approximate the sample values. Hence, in the next section we discuss the selection of the model that approximates the sample values with minimal error, generating the least outliers, while still satisfying the resource constraints.

4.3.5. Dynamic Adaptive Modeling (DAM) through Dynamic Model Order Selection. We have developed an adaptive modeling scheme, that the sensor nodes use to dynamically select an appropriate model order that best approximates the sampled values and reduces the message overhead. It also avoids unnecessary higher order computation.

Sensor nodes have limited computational capabilities, and higher order models (e.g., AR(4) and higher) are typically computationally expensive. Hence, we allow the nodes to choose the model from AR(1) to AR(3). A node initially constructs the AR(1) and AR(2) models as described in Section 4.3.2 and counts outliers for each model (Alg. 1, L. 1-6). Both models have low computation cost as AR(1) solves a linear equation and AR(2) solves two linear equations. The outliers for lower order model (AR(1)) are compared with those for higher order model (AR(2)) (Alg. 1, L. 7). If a higher order model does not outperform a lower order model in terms of outliers, i.e., it does not have lower number of outliers, then it is discarded and no further search is carried out (Alg. 1, L. 10-11). However, if the higher order model outperforms a lower order model then the higher order model is considered and the lower order model is discarded (Alg. 1, L. 8-9). Similarly, the next higher order model is evaluated and compared until no further improvement is observed. Consequently, the nodes dynamically select a model order that optimally

Algorithm 1 Dynamic Model Order Selection

```
1:  modelOrder ← 1;
2:  Φ_Current ← train(V, modelOrder);
3:  Outliers_Current ← AM(Φ_Current, x(t));
4:  for modelOrder ← 2 to maxOrder do
5:     Φ_New ← train(x(t), modelOrder)
6:     Outliers_New ← AM(Φ_Current, V);
7:     if Outliers_New < Outliers_Current then
8:        Φ_Current ← Φ_New;
9:        Outliers_Current ← Outliers_New;
10:    else
11:       break; ;
12:    end if
13: end for
```

matches the approximated sampled data. The resultant scheme can adaptively model the attribute samples and dynamically selects its order, and is termed as Dynamic Adaptive Modeling (DAM). Conceptually, our scheme is neither bound to the maximum AR(3) model nor to the AR model type. If sensor nodes would possess better computation resources, then the choice for the model order can be adaptively increased or other model types can be used exclusively or in addition to AR models.

With our adaptive scheme, the number of samples approximated by a certain model is also dynamic and hence the compression scope of each model may be different at different time in order to adapt to the changing signal dynamics. In the next section, we detail the adaptability process to approximate the varying number of samples.

4.3.6. Dynamic Approximation Window. A model, in general, can approximate only a limited number of sampled values within the accuracy bounds. We refer to the number of samples that a model can approximate as the approximation window and define it as:

DEFINITION 1. *Approximation window (W_Φ) for a model is the number of samples that a model (Φ) can approximate within the required accuracy bounds while allowing $O_{\#max}$ outliers.*

The approximation windows depends on accuracy bounds, allowed outliers, model order and the attribute dynamics. Increasing the tolerated accuracy bounds, maximum allowed outliers and the model order usually increases the approximation window.

Whereas, the increasing non-linearity in the signal generally decreases the approximation window because of the use of linear models, the accuracy bound is fixed based on the user requirements. The maximum allowed number of outliers is fixed by design to limit the message cost. The model order is dynamically selected for optimal compression (Section 4.3.5). Hence, given the accuracy requirements and outliers bounds the attribute dynamics determine the approximation window for a certain model.

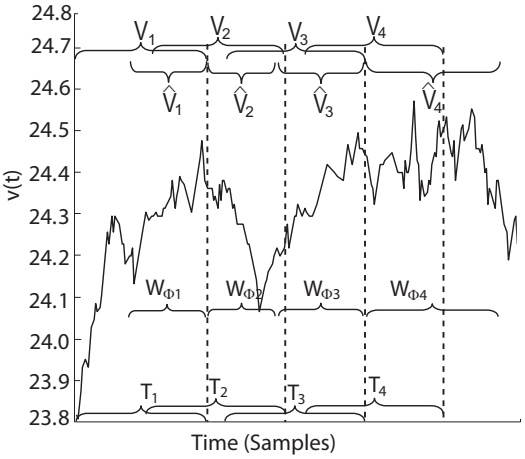


Fig. 5: Dynamic Approximation Window

The constructed model is used to approximate the values that it was trained with, i.e., the training queue (V). The approximated values set is denoted by $\hat{V}$ ($\hat{V} = \{\hat{v}(t), \hat{v}(t-1)...\hat{v}(t-W_\Phi+1)\}$). Due to the dynamic nature, the number of samples for an approximation window may vary. In Fig. 5, we illustrate the idea of dynamic approximation window. The tailing sample values are estimated using the model (Φ) constructed from the training sampled values. The number of estimated values (model approximation windows) is denoted as W_Φ. W_Φ can vary in number of estimated values from approximated sample short of the training length to samples beyond it depending on the underlying process dynamics, error threshold and maximum allowed outliers.

Fig. 5 depicts various cases, e.g., $\hat{V}_1$ estimates samples to the length of training vector $\hat{V}_1$, $\hat{V}_2$ runs short of V_2 and $\hat{V}_4$ runs pass the training length of V_4. In the case of $\hat{V} \subsetneq V$, where the estimated values in $\hat{V}$ run short of the training samples length, the values

not estimated in a training set are passed to the next training set. For example, in Fig. 5 the samples of V_2 not estimated by $\hat{V}_2$ are passed to V_3 to train the next model. Occasionally, a model may estimate all the sample values in the training set (e.g., $\hat{V}_4$) and still the number of outliers is less than $O_{\#max}$. In this case, the model keeps on approximating the next values (i.e., predicting) until the number of outliers are less than $O_{\#max}$.

We now develop a mechanism to construct a batch of models so that instead of sending individual models, a batch of models may be sent to the sink in just one message.

4.3.7. Maximal Temporal Compression and Model Caching. We assume that data delivery can be delayed as specified by the delay-tolerance for the application. We denote the application delay-tolerance in terms of samples that can be collected and used to construct the model cache and then report to the sink. The delay is represented as the total number of samples approximated by the model cache, given as W_Ψ. The tolerated latency, and hence the length of $\hat{V}_\Psi$, are generally many orders longer than the approximation window of a simple AR model. In general the nodes will require more than one model to approximate the entire length of the sampled data.

The master cluster heads construct a consecutive batch of models referred to as the *model cache*. A sensor node collects training data (V) with T samples that are used to train the model as described in Section. 4.3.3. The constructed model is used to approximate the samples ($\hat{V}$). Subsequent samples are used to construct a new training queue. The new training queue is again used in the same manner to construct the next model. This process keeps on repeating until W_Ψ samples have been approximated. We define the set of $m_\#$ models, that are used to approximate W_Ψ samples, as a model cache denoted by Ψ and defined as $\Psi = \{\Phi_1, \Phi_2, \ldots \Phi_{m\#}\}$. The construction of a model cache has also been illustrated in Fig. 6. Due to delay-tolerance and presence of the model cache, we do not repeatedly need to construct a model, send it to the sink and send a new model once the previous model is invalid. Such a scheme would require repeated transmission of the model parameters to the sink. By constructing the model caches the master clusters refrain from reporting each model and avoid such repeated re-transmissions, decreasing the message cost considerably. Consequently, we increase the temporal compression window, handle non-linearities, decrease message cost, save energy and still use simple computationally inexpensive models. Next, in 4.2 we show that the samples estimated by a model cache error lie within user defined error threshold.

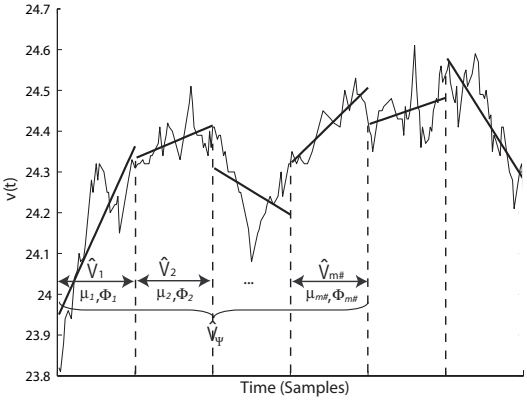


Fig. 6: Model Cache

LEMMA 4.2. *For a model cache Ψ, the actual sampled valued $v(t)$ belonging to the approximated model cache value $\hat{v}_\Psi(t)$ ($\wedge \hat{v}_\Psi(t) \in \hat{V}_\Psi(t)$) is contained in $[\hat{v}_\Psi(t) - \alpha\ ,\ \hat{v}_\Psi(t) + \alpha]$.*

PROOF. A model cache is set of $m_\#$ AR models, i.e., $\Psi = \{\Phi_1, \Phi_2, \ldots \Phi_{m\#}\}$. The sampled values for each model $\Phi_i \in \Psi$ are contained in $[\hat{v}_{\Phi_i}(t) - \alpha, \hat{v}_{\Phi_i}(t) + \alpha]$ (where

$\hat{v}_{\Phi_i}$ is the approximated sampled value using model Φ_i) due to Lemma 4.1. Hence, the sampled values approximated by model cache Ψ_i are contained in $[\hat{v}_\Psi(t) - \alpha\ ,\ \hat{v}_\Psi(t) + \alpha]$. □

The temporal compression scheme developed so far can be used by any sensor node to dynamically determine a model cache that best approximates its sampled values. However, we have developed our proposed scheme such that only a few nodes in the network belonging to the master cluster actually execute the proposed scheme to construct the models. Next, we describe how a cluster head of a master cluster uses the developed technique to construct a model. The selection of the master clusters and how other clusters can be approximated using the model cache constructed on master cluster are discussed later in Stage 3.

4.3.8. Master Cluster Model Construction. The fundamental basis of our compression scheme is that only one node (or a set of a few nodes) should construct a model that carefully approximates the largest number of nodes in its surrounding clusters, i.e., within the defined accuracy bounds. Accordingly, we construct the models on master clusters that are used to approximate the sampled values of the nodes not only in the master cluster but also in clusters surrounding the master clusters. The selection of master clusters is discussed in Section 4.4.1. Once, the master cluster is selected the model cache is constructed as described earlier.

For a model cache constructed on cluster head to approximate the sampled values of other sensor nodes, we need to define a measure for the similarity of the sampled values between two sensor nodes.

DEFINITION 2. *If the approximated sampled values from a sensor s_i and s_j are given by $\hat{v}_i(t)$ and $\hat{v}_j(t)$ respectively, then approximation loss for approximating sample values of sensor node s_j using model cache (Ψ_i) of sensor node s_i can be given by $L_{ij}(t) = |\hat{v}_i(t) - \hat{v}_j(t)|$.*

In order to meet the user's accuracy requirements, we define two attribute values to be similar only if their approximation loss is bounded by β. Accordingly, if a sensor node s_i and s_j satisfy the approximation loss bound, i.e., $L_{ij}(t) \leq \beta$, then the approximation error between the two nodes can be given by Lemma 4.3.

LEMMA 4.3. *The maximum approximation error for sensor node s_i using model cache Ψ_j from nodes s_j while satisfying the approximation loss bound, is at the most $\alpha + L_{ij} = \alpha + \beta$, with the error probability less than $1/\nu^2$.*

PROOF. Suppose sensor nodes s_i constructs a model cache Ψ_i and sensor node s_j uses Ψ_i to estimate its sampled values such that s_i and s_j satisfy the approximation loss bound at any given time t for the length of the model cache. The approximation error between the two sensors can be given by $|v_i(t) - \hat{v}_j(t)| = |v_i(t) - \hat{v}_i(t) + \hat{v}_i(t) - \hat{v}_j(t)| \leq |v_i(t) - \hat{v}_i(t)| + |\hat{v}_i(t) - \hat{v}_j(t)| = |v_i(t) - \hat{v}_i(t)| + L_{ij}(t)$. Using lemma 4.1, $|v_i(t) - \hat{v}_i(t)|$ can be at the most α with he probability $1/\nu^2$, additionally $L_{ij}(t)$ has an upper bound of β. Hence, the approximation error can be at the most $\alpha + \beta$ with a probability of $1/\nu^2$. □

In order to simplify the application of the developed scheme, we define a single parameter for error threshold, ϵ, instead of using two separate parameters α and β. Accordingly, we will use ϵ to define the error threshold for the sensor nodes constructing their own model caches and the sensor nodes that use the model caches from other sensor nodes (e.g., master cluster heads). Where necessary, the original error threshold parameters can be used to maintain the necessary level of granularity.

The model cache constructed on the master cluster is used to approximate a larger number of nodes. Hence, the constructed model cache should produce minimal approx-

imation error. However, the sampled data may naturally contain spurious and noisy values. The sampled values are used to train the models and the noisy values may introduce modeling errors. Hence, the model cache constructed on a particular node, e.g., cluster head, may not necessarily be the model cache with minimal approximation error. In order to build the minimal error model cache, the cluster head along with a few randomly selected member nodes build the model caches and the best model cache amongst them is selected. Accordingly, the master cluster head broadcasts a request to its member nodes to construct the model cache. A few randomly selected cluster members build the model cache using DAM (Section 4.3.5) and send it to the cluster head. The cluster head accumulates all the model caches and broadcasts them to the member nodes. Each member node (including cluster head) fits each model cache to its sampled values using AM scheme (Section 4.3.4) and reports back outliers for each model. The cluster head selects the model cache that generates the least number of total outliers for the cluster member nodes. This criteria allows us to select the model cache that best approximates the sampled values of all sensor nodes in the cluster but also results into minimal message cost as we have to spend fewer messages to send the outliers to the sink.

The search for least error model cache incurs some additional cost in terms of computations and message exchange. However, this single model cache is used to approximate large number of nodes in the surrounding clusters. Hence, this additional cost is compensated with the saving of large number of outliers that would otherwise incur large message penalties.

4.4. Stage 3: Merging 1-Hop Clusters Based on Master Cluster Model Cache

In Stage 1, we exploited limited but strong spatial correlations by forming 1-hop clusters. In Stage 2, we constructed the model caches to exploit the temporal correlation on the master clusters, achieving spatio-temporal compression. Now, we detail our scheme to extend the spatio-temporal compression beyond the master clusters. Various existing techniques use monolithic clustering based on instantaneous values [Abbasi and Younis 2007; Gedik et al. 2007; Yoon and Shahabi 2007] to exploit the spatial correlation. They generally are very costly as they incur continuous cluster maintenance cost. We use models instead of instantaneous values to form the clusters. Large monolithic clusters based on models would require tracking of all sensor nodes that fit the entire model cache length. Moreover, the attribute dynamics may often change beyond the 1-hop cluster range. Hence, active tracking of the sensor nodes to form and maintain large clusters can require continuous message exchanges resulting in high energy consumption. Therefore, instead of forming large monolithic clusters we use a two level hierarchical clustering. The first level is 1-hop clusters (Stage 1) where (typically) the nodes are highly correlated. The second hierarchical clustering level is constructed by merging the 1-hop clusters that can be approximated by the same model cache to form larger clusters referred to as model cache cliques or simply cliques (Def. 4). The schemes described next (including the schemes in Stage 1 and 2) are executed on sensor nodes without assistance from the sink highlighting the fully decentralized nature of our approach. Before describing clique formation, we discuss how the master clusters are selected and how a model cache constructed on the master cluster can be used to approximate the sampled values of the nodes in surrounding clusters .

4.4.1. Master Cluster Head Selection. A master cluster merges with its surrounding clusters based on its model cache to form cliques or regions. We are interested in determining the smallest possible number of regions in the network or smallest set of model caches that best approximates the largest network parts in order to achieve maximal spatio-temporal compression, i.e., transport maximum data in least number of

messages to the sink. We formulate the problem of defining the minimum number of spatio-temporally correlated regions in terms of cliques in a network graph.

DEFINITION 3. *Given a sensor network consisting of a set of sensor nodes S, the topology of the sensor network can be modeled as an undirected graph $G(D, E)$, where D is the set of vertices and E is the set of edges. An edge (S_i, S_j) is included in the edge set E if Nodes S_i and S_j can communicate directly with each other. The network sub-graph induced by a subset S_M, of set S, is the sub-graph of G involving only the vertices and nodes in S_M.*

Following the formation of 1-hop clusters, we get a virtual network consisting of a set of cluster heads S_{CH} that are modeled as an undirected graph $G_C(D_C, E_C)$, where D_C is the set of vertices and E_C is the set of edges based on cluster heads. (C_i, C_j) is included in the edge set E_C if Cluster Heads C_i and C_j can communicate (through a gateway node $S_G \in D$) with each other.

DEFINITION 4. *We define a clique as a network sub-graph Q_C, a subset of the cluster heads of set D_C, such that a model cache Ψ approximates the samples values within the given error bounds for all the member nodes of each cluster belonging to cluster heads in Q_C.*

Finding the master clusters that can construct the model caches that forms the largest coverage clique (approximates largest number of clusters) is NP complete [Abello et al. 1999]. Hence, we utilize a heuristic based on the requirement of minimizing the message overhead. As the information flow direction is from the network to the sink, we propose the farthest cluster to be selected as the master cluster to initiate the clique formation and expand the clique in the direction of the sink by sending its model cache to the neighboring clusters. Looking at the alternate possibility if the clusters nearer to the sink initiates the clique formation, the clique might expand in the direction away from the sink. We will have to spend additional messages to transport the accumulated information back to the sink. Hence, the heuristic biased to expand in the direction of the sink generally reduces the message cost to transport the information to the sink. Each cluster head knows its hop distance and the neighboring cluster heads hop distance from the sink as explained in Section 4.2. The cluster heads use this information to figure out the farthest cluster and hence the master cluster locally. In case of cluster heads having the same number of hops, the cluster head with higher id has the precedence. Next, we describe how a model cache constructed on master cluster (Section 4.3.7) can be used to approximate the sampled values of the nodes in the surrounding 1-hop clusters.

4.4.2. Model Cache Acceptance by 1-Hop Clusters. The master cluster constructs the model cache as described in Section 4.3 and sends it to the neighboring clusters. A model cache received by a neighboring cluster must approximate the values of its member nodes within the defined error bounds (Section 4.3.4). Because of similar attribute distribution and redundant deployment a model constructed in a master cluster can approximate the sensor nodes in the surrounding clusters.

To evaluate whether the received model cache approximates the sampled values, each member node approximates its samples values as:

$$\hat{v}(t) = \mu_{local} + \sum_{i=1}^{p} \phi_i.(v(t-i) - \mu_{local}) \tag{6}$$

where $v(t)$ are the sample values of the node, μ_{local} is the sample mean of the local cluster head and model parameters are as received from the master cluster. Each cluster

uses μ_{local} as we observed μ changes very quickly from one cluster to the other, while model parameters remain similar.

The nodes in the surrounding clusters approximate their sampled values using received model parameters as given in Eq. (6), calculate the error $\hat{v}(t) - v(t)$ and count the outliers. As discussed in Section 4.3.4, we allow a certain number of outliers for a model to be accepted by a sensor node. Total count of the outliers for each model Φ in the model cache Ψ should be less than the maximum number of allowed outliers ($O_{\#max}$) for model acceptance (Section 4.3.4). Hence, for each model in the model cache to be accepted, the node counts the outliers for each model as:

$$count\,|v_j(t) - \hat{v}_j(t) > \epsilon| \leq O_{\#max}, \hat{v_j}(t) \in \hat{V}_i, i = 1...m_{\#} \tag{7}$$

If all models in the cache satisfy the criteria in Eq. (7) the model cache is accepted by the node.

In Alg. 2, we describe the model cache acceptance by the cluster head and model cache evaluation by the sensor nodes in a neighboring cluster. When a neighboring cluster head receives the model cache from the master cluster, it broadcasts the model cache with local means, calculated from cluster receiving cluster head's sample values, to its cluster members (Alg. 2, L. 1-4). The cluster head starts a timer to wait for the responses to be collected (Alg. 2, L. 3). Each cluster member prepares a vote, using Eq. (7), by counting outliers to either reject or accept the model cache (Alg. 2, L. 7). Each member node must accept all the models in the model cache to accept it for the entire 1-hop cluster. In order to reduce the message cost we use the negative acknowledgement scheme. Accordingly, the cluster head is not notified of the acceptance of the model cache (implicit vote), rather when the timer expires on the cluster head it assumes the model cache to be accepted by its member nodes. The negative votes are, however, explicitly reported along with the outliers to be reported to the sink (Alg. 2, L. 9). Withholding report for positive votes saves messages because most of the nodes in the 1-hop cluster usually accept the model cache. The member nodes send outliers to the cluster head instead of sending them to the sink (Alg. 2, L. 10). The cluster heads report the outliers efficiently to the sink by concatenating multiple outliers in one message.

Algorithm 2 Model Cache Acceptance

1: MSG.type='Ψfit';
2: MSG.$\Psi \leftarrow \Psi$;
3: timer.start();
4: broadcast(MSG)
5: function msgReceive(MSG)
6: **if** MSG.type=='Ψfit' **then**
7: vote $\leftarrow$ (solve Eq. 7 for Φ, $\Phi \in \Psi$)?reject:accept
8: **if** $vote == reject$ **then**
9: unicast(vote,outliers[],C_H);
10: **else if** count($outliers > 0$) **then**
11: unicast(outliers[],C_H);
12: **end if**
13: **end if**
14: add Ψ to Ψ_List;
15: function timer.expired()
16: Ψ.accepted $\leftarrow$ (Ψ_List total accepts$> \%k, k \in C$)
17: broadcast(Ψ.acceptance);
18: **if** Ψ.accepted **then**
19: remove member that rejected Ψ & add the new requesting member
20: **end if**

The cluster head counts the votes for model cache when the timer expires (Alg. 2, L. 15-16). A model cache is accepted as a model cache for the cluster if the cluster head received acceptance from at least $k\%$ member nodes (Alg. 2, L. 16).

We discussed the construction of model cache on a master cluster and how a model constructed on a master cluster can be used to approximate the sample values of the nodes in other clusters. However, it is required by the clusters to be synchronized

in time so that other clusters can use the model cache of the master cluster to approximate nodes sample values. If the clusters are not synchronized, the sample values might have been collected at different time instances (or time segments). Consequently, master cluster model cache may not approximate the other cluster sample values as the model construction time segment and the time segment of the samples to be approximated might be entirely different.

4.4.3. Self-adaptive Clique Formation. So far, we have described the formation of the 1-hop clusters, the construction of the model cache on the master clusters and how a model cache constructed on the master cluster can be used to approximate the values of sensor nodes in the surrounding clusters. We now describe the clique formation based on the model caches of the master clusters. For clique formation the master cluster sends its model cache to its immediate neighboring 1-hop clusters. Each neighboring cluster evaluates the model cache for acceptance. If the neighboring cluster head accepts the model cache, it joins the clique and sends the model cache to its neighboring clusters (except the neighbor that sent the model cache) and so forth. Hence, the clique formation is essentially a controlled flooding of model cache over the cluster heads around the master cluster. The flooding stops once the model cache is not accepted anymore by the surrounding cluster heads or it reaches the boundary of the network. We now detail this scheme.

Alg. 3 describes the expansion of the clique and the functionality of the clusters that constitute the clique and send the model caches to their neighboring clusters to further expand the clique. Initially, only master cluster constitutes the clique as it initiates the clique. Alg. 4 describes the functionality of a neighboring cluster receiving the clique 'join' request to merge into the clique.

Model Cache Dispersion and Clique Expansion: A master cluster initiates the clique formation by constructing the model cache. Fig. 7 shows the format of the *JOIN message* payload that is used by the cluster heads to propagate the list of constituting clusters and the model cache. The master cluster head adds its cluster id, the model parameters and the sample mean values for each model in the model cache to the message payload. It sends the JOIN message to all the neighboring clusters (C_N) through the gateway sensor nodes (S_G) (Alg. 3, L. 1). Each neighboring cluster head executes the 1-hop cluster model acceptance according to Alg. 2 (Alg. 4, L. 4). If the model cache is accepted, the cluster joins the clique by appending its cluster id and the sample mean values to the message payload. The cluster joining the clique considers itself on the boundary of the clique and executes Alg. 3 to further propagate the JOIN message to its neighboring clusters, hence expand the clique (Alg. 4, L. 8-9). The neighboring clusters receiving JOIN message always notify the requesting cluster whether it is joining the clique (Alg. 4, L. 11). Each cluster head maintains a local record of its neighboring cluster with respect to their status regarding the clique. The receiving cluster heads update their local record

Algorithm 3 Clique Expansion

1: 1HSend($JOIN$, $C_i \wedge \forall C_i \epsilon C_N$)
2: function receive(MSG)
3: **if** MSG.type==’$RESP$’ **then**
4: clique.add(MSG.accepted,MSG.C_{HR}); R#++;
5: **if** R#==$C_{N\#}$ **then**
6: **if** $\forall$ $clique.acctance$ **then**
7: $NBC.clique \leftarrow$ clique;
8: 1HSend(NBC, $C_i \wedge \forall C_i \in C_N$)
9: **end if**
10: **end if**
11: **else if** MSG.type==’NBC’ **then**
12: clique.$C_{Hi}.boundary \leftarrow' false' \wedge C_{Hi} == MSG.C_H$;
13: clique.remove($(NBC.clique)$;
14: **end if**

regarding the neighboring cluster status from the clique JOIN messages during each message exchange (Alg. 4, L. 7). Once the responses from all the neighboring cluster C_N are received by the requesting clusters, it uses its local record to check if all C_N around it belong to the same clique. If all the surrounding clusters belong to the same clique, it implies that this cluster is not on the clique boundary. It notifies all cluster heads in C_N that it is not on boundary anymore through a "Not Boundary Cluster" message (NBC) and transfers its local list of clusters in the clique to the neighboring clusters (Alg. 3, L. 6-9). Each cluster head knows the status of each neighboring cluster whether it has joined the clique and whether it is on the clique boundary by continuously maintaining the local record and forwarding it to its neighbors.

Model Cache Flood Control: The clique expansion continues until the new clusters return positive join responses. A cluster head then knows that the clique expansion has stopped locally if it received the responses from all the neighboring clusters and one or more responses are negative. This cluster now stays on the clique boundary. As the join message is flooded from the master cluster to the final boundary of the clique, each boundary cluster possesses the partial list of clusters forwarded during clique expansion by the clusters from the clique body. The border is traversed to accumulate the clusters list. The border traversal is initiated by a cluster possessing special token termed as the Boundary Traversal Token (BTT). The master cluster initially assigns itself the BTT. The BTT possessing cluster forwards it to its neighboring cluster with least hops from sink if the BTT possessing cluster is no longer on the boundary. The boundary cluster possessing BTT transfers its partial list of the clique constituting cluster to its neighboring cluster on the boundary. The neighboring cluster merges its partial list with the received list and repeats the process until all the boundary clusters are traversed. The last cluster on the edge of the boundary forwards the aggregated list to the sink.

The clusters beyond the clique boundary follow the normal procedure of determining the farthest cluster to initiate and form a new clique. The cluster adjacent to the boundary of the clique exclude their neighbors that are already part of another clique in determining the farthest cluster. Using our proposed scheme the nodes dynamically group to create a region that is spatially and temporally correlated for a given attribute for the modeled time duration. Hence, we have one model cache to be reported to the sink that represents the behavior of the spatio-temporally correlated region. During the clique formation each node in each cluster takes part in model cache acceptance, hence, the data regenerated from the model caches on the sink are accurate to the level of single node. The particular discrepancies are corrected through the outliers sent by the cluster heads for their respective members. We do not assume any particular distribution of the sensor nodes. Therefore, there can be multiple 1-hop clusters in different parts of the network that may have the maximum number of hops in the local neighborhood. We also set an upper bound on the time (T_{max}) that a cluster head can wait for the larger hop number cluster (clusters farther from sink) to initiate the following round of clique formation. On the expiration of the wait time period the cluster head initiates clique formation.

Algorithm 4 Clique Joining by Neighboring Cluster

```
1: function receive(MSG)
2: if MSG.type ← 'join' then
3:    RESP.accept ← 'false';
4:    Call Alg. 2 to evaluate MSG.Ψ
5:    if MSG.Ψ.accepted then
6:       RESP.accept ← 'true';
7:       clique.add(MSG.C_RequestCH);
8:       me.Boundary ← true;
9:       Execute Alg. 3
10:   end if
11:   1HSend(RESP, C_HR)
12: end if
```

Hence, there can be multiple instances of clique formation executing in parallel. Two or more cliques formations execute mutually exclusively, i.e., a growing clique stops at the boundary of the other growing clique. We put this restriction because the two cliques are growing based on two different model caches.

4.4.4. Iterations and Dynamic Adaptability. The second level of cluster hierarchy or the clique is a temporary entity to determine the correlated region based on the model cache and is not strictly a larger cluster in conventional sense. It does not have a cluster head and we do not maintain it. Maintaining such a large cluster body may be very costly, because it is built using the model cache, which requires the clusters to agree for a longer duration of time. Hence, cliques are reconstructed rather than being maintained. We show in Section 6.1 that message overhead to construct a clique is very low. The reconstruction of the cliques allows to continuously adapt to the changing dynamics. The 1-hop cluster that were part of one clique may be part of another clique or even make their own clique in the next iteration of clique formation depending on the dynamics of the phenomenon. After reporting the model caches, the cluster heads wait for enough data to be collected to construct the next model cache as explained in Section 4.3.7. The process of the model cache construction and the clique formation is repeated and the sink receives accurate continuous data.

4.4.5. 1-Hop Cluster Dynamic Rearrangement. The 1-hop cluster members usually stay correlated. However, due to changes in the physical phenomenon, the correlations even at the 1-hop cluster level may change and the clusters require rearrangement. We do not implement an explicit mechanism to detect such changes as it would require further message overhead. We determine such changes in the node correlations in Section 4.4.2 when $k\%$ of nodes agree with the model cache and the model cache is accepted by the cluster head due to majority vote. The cluster head evicts the remaining $r - k\%$ nodes that send the negative votes (reject the model cache). Using overlapping nature of the clusters the evicted sensor nodes wait and snoop the model caches sent by the rest of surrounding clusters. An evicted sensor node checks the model cache as if it were part of this cluster and updates the cluster head accordingly. If the model cache is accepted by the cluster head, the evicted sensor node joins the cluster. If the evicted node can not join any surrounding cluster it forms a new cluster and becomes the cluster head as described in Section 4.2. Such rare condition arises due to a new phenomenon developing in local region. The sensor nodes in this region (around the evicted sensor node) will start leaving their current cluster (as the evicted sensor node did) and will eventually form a cluster with the evicted sensor node. Consequently, the sensor nodes rearrange the 1-hop cluster to self-adapt to the environmental changes.

5. EFFICIENCY AND COMPRESSIBILITY ANALYSIS

In this section, we carry out the cost analysis of our proposed scheme in terms of number of messages required for compression and transport of the information. We discuss the efficiency of our proposed scheme in terms of theoretical compressibility that we can achieve. Finally, we also discuss the computational overhead incurred by the proposed scheme.

5.1. Message Payload

The analysis and structure of the message payload that carries the compressed data is very important, because it directly impacts the message cost. The general structure of the model cache message (Msg_{Ψ}) payload is as depicted in Fig. 7. The complete information about the model cache and the clusters taking part in the formation of the clique are contained in the message payload. The payload consists of three parts: 1) The model type of each model in the model cache, 2) all the model parameters of

each model in the model cache, and 3) the cluster ids (C_{id}) and the local means of each cluster.

We denote the bytes required to denote a certain parameter by a "B" in subscript, e.g., $model\ type_B$ denotes bytes required for model type, or ϕ_B denotes bytes required by a model parameter. If there are $m_\#$ number of models in the model cache and p denotes the model order then the total size of the message payload can be given as:

$$payload_B \ = \ model\ type_B + \sum_{\Phi \in \Psi} \phi_B \times \Phi_p + \sum_{C \in Q} (C_{id_B} + \mu_{C_B} \times m_\#)$$

The three parts of the equation correspond to the three parts of the message payload.

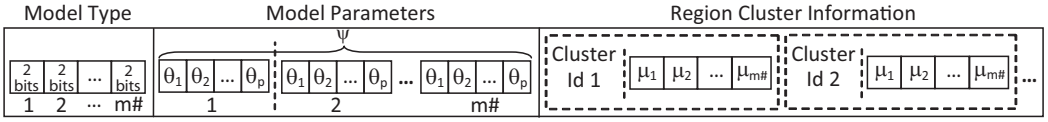


Fig. 7: Message Payload Format

The size of payload in comparison to the amount of data that is compressed using our proposed scheme is fairly small. For typical values of 3 models per model cache and 2 bytes to store each model parameter, the bytes required for modeling will be $2bytes + 2bytes \times 3(parameters) \times 3(models) = 20$ bytes. If we reserve 2 byte for cluster id and 2 bytes for mean value, then we require 8 bytes ($2\ bytes\ for\ cluster\ id + 3\ models \times 2\ bytes\ for\ mean\ per\ model$) for each new participating cluster. Hence, for the given case, we typically require 28 bytes to store the complete information for one cluster and 8 bytes for each additional cluster in the clique.

Notation	Description
r	Cluster members count
μ_C	Mean value for Cluster C
Q	Clique
v_B	Bytes per sample value
$S_{\#C_i}$	Sensors in cluster C_i
Msg_{Req}	Model cache request
Msg_Ψ	Model cache message
Msg_{Out}	Outliers message
$S_{\#rand}$	Random number of nodes
$S_{\#G}$	Number of gateway nodes

Table III: Efficiency and Compressibility Analysis Notations

TinyOS [Levis et al. 2004] (an established OS for WSN) has a default payload size of 28 bytes. However, radios on various popular platforms, such as telos [Polastre et al. 2005], support message lengths of up to 128 bytes. Hence, we can easily transport the model cache with multiple cluster information in a single message by changing the default payload size. In contrast to TinyOS, Contiki [Dunkels et al. 2004] (another popular platform) does not have such limitations.

5.2. Maximum Theoretical Compressibility and Efficiency

In order to keep the nodes synchronized, we limit the number of samples that a node can compress using a single model cache to $\hat{V}_\Psi$. Hence, each node in each cluster in the clique compresses $\hat{V}_\Psi$ samples using the model cache. If $S_{\#C_i}$ denotes the number of nodes in cluster C_i accepting the model cache (minimum is $k\%$, otherwise model cache is rejected), then the maximum theoretical compression that our proposed scheme can achieve is given by Eq. (8):

$$Total_{bytes} = \sum_{C_i \in Q} \hat{V}_{\Psi_B} \times (S_{\#C_i}) \tag{8}$$

Eq. (8) shows that the maximum theoretical compression in a given cluster is equal to the number of bytes representing the sample values (max: $\hat{V}_\Psi$) of nodes accepting the model cache. To determine the total achieved compression for the clique, this factor should be summed for all the clique member clusters. Assuming a conservative value for $\hat{V}_\Psi = 75$, average cluster size of 7 members and 2 bytes per sample, the total data that we compress is $2 \times 75 \times 7 = 1050\ bytes$ for one cluster. Interestingly, we require only 28 bytes to represent the complete data in the cluster as shown in Section 5.1. Consequently, we would achieve a compression ratio of more than 37 times for the assumed parameter values. For each next cluster, we require only 8 bytes in message payload to compress an additional 1050 bytes of data. In terms of message we require only one message. We present the analytic details of message costs and elaborate it further in Section 6.1.

The actually achieved compression, however, falls slightly short of the theoretical maximum compressibility. The reduction in compressibility is due to the outlier values that cannot be approximated by the models to satisfy the accuracy requirements imposed by the user. However, the decrement is limited. The worst case compression that we can really achieve can be given by the following equation

$$Total_{bytes} = \sum_{C_i \in Q} (V_{\Psi_B} \times (S_{\#C_i}) - v_B \times O_{\#max} \times m_\#) \tag{9}$$

Eq. (9) gives the compression when we assume each node in the cluster has outliers equal to the maximum number of outliers allowed. If we assume maximum allowed outliers value to be 5, the achieved compression decrements by $2\ bytes \times 5 \times 3 \times 7 = 210\ bytes$. Still we are able to compress 840 bytes after fulfilling the accuracy requirements for the given parameters.

6. MESSAGE AND COMPUTATION COST FOR COMPRESSION

We now detail the cost of message exchange to achieve the data compression.

6.1. Message Overhead

In Section 5.2, we discussed the maximum theoretical and practically achieved compression in terms of the number of bytes reduced using our scheme. However, for WSN the notion of number of messages is the true measure of compression to be achieved. Hence, in WSN reduction in number of bytes is only an indirect measure and may be eclipsed or become completely irrelevant if the compression is achieved at the cost of high number of messages exchanged. Hence, we have systematically broken down each stage of the proposed scheme to account for the message cost incurred to achieve the spatio-temporally compression and transportation of information to the sink.

6.1.1. Master Cluster Model Cache Construction. Model cache Ψ is constructed on a few master clusters. The cluster head broadcasts the request (Msg_{REQ}) to the cluster members to send their model caches (Msg_Ψ). A few random members ($S_{\#rand}$) send their model caches. The cluster head broadcasts back all the collected model caches to the members. Finally the cluster members respond back with the number of outliers for each model cache (Msg_{OUT}). If the master cluster has r members then the total number of messages exchanged in a master cluster to construct a model cache can be estimated by the following equations:

$$Msgs = Msg_{REQ} + Msg_\Psi \times S_{\#rand} + Msg_\Psi + Msg_{OUT} \times r \tag{10}$$

The cluster head and the nodes constructing the model cache in the master cluster transmit two messages and the rest of the nodes in the cluster transmit just one mes-

sage to construct a model cache. Hence, in worst case a node in the master cluster needs to transmit two messages to compress the data and construct the model cache.

6.1.2. Intra-Cluster Agreement. The clusters other than the master cluster use the model cache constructed by the master cluster to estimate the sample values of the nodes in the cluster. To evaluate whether the model cache estimates the sample values of the sensor nodes, the cluster head broadcasts the model cache. The sensor nodes evaluate the models and respond with the outlier values to the cluster head. The count of messages for this stage can be given by the following equation:

$$Msgs \; = \; Msg_{\Psi} + Msg_{OUT}$$

Accordingly, in the clusters other than master cluster the worst case count of messages to decide a model cache is one for any node in the cluster.

6.1.3. Model Cache Dispersion. Once the master cluster has constructed a model cache or a normal cluster has accepted a model cache, it disperses the model cache to its neighboring cluster through the gateway nodes. If S_G denotes a gateway node and $S_{\#G}$ is the count of gateway nodes then the total number of messages required by each cluster to disperse the model cache is given by the following equation:

$$Msgs \; = \; Msg_{\Psi} + Msg_{\Psi} \times S_{\#G}$$

This equation represents the worst case situation when the cluster are non-overlapping (which is not typically the case as in Fig. 3) and all gateway nodes transmit without optimization, like avoid transmitting to the neighboring clusters that already are part of clique.

6.1.4. Joining Clique. If the model cache is accepted by the cluster head, it joins the clique. The joining cluster head sends a message to the requesting cluster head to report for joining the clique through the gateway node, accounting to one message for cluster and the corresponding gateway node.

6.1.5. Clique Border Expansion. During the model cache dispersion the clique expands and the border of the clique extends further. If a cluster head finds out, after receiving the responses from the neighboring cluster, that it no longer is on the border of the clique it sends this confirmation to the neighboring clusters so that it may no longer be considered on the border of the clique.

$$Msgs \; = \; Msg_{\Psi} + Msg_{\Psi} \times S_{\#G}$$

The clique border expansion accounts for one message broadcast from the cluster head and further transmission to neighboring cluster by the gateway node.

In summary the number of transmissions range from one to four messages depending on their role and whether they are in the master cluster or the non-master cluster. In the worst case and with very large clique formation, more than one packets may be required to contain the complete clique information. The above calculations assume the roles of the nodes, from functional point of view, to be mutually exclusive. However, in reality they may not necessarily be exclusive, e.g., a gateway node may also be the same node as the node participating in model cache construction.

6.2. Computation Overhead

The computation cost arises mainly during the model learning phase where the model parameters are evaluated (Section 4.3.3). During the training phase the intent is to avoid unnecessary model construction as proposed in Section 4.3.5. Initially only the first and second order models are constructed. The third order model (or any higher order model in general) is estimated, and used only if an improvement is observed with

increasing order. The parameter estimation consists of solving $AX = B$ in p unknowns and accordingly computing the matrices A and B. For a third order model, which represents the worst case situation, we need to carry out $12(T - 3)$ sum and production operations and additionally the cost to the solve the linear system of equations. As we compute a higher order model only if it is needed, hence generally we need to carry out even fewer computations. Additionally, the linear set of equations resulting due to higher order contains terms already computing for the lower order system. Hence, computing a higher order model is not same as the computation of each term in the linear system of equation rather only a few new terms in the higher order system. In addition to the optimization mechanisms that we have in place to reduce the computations as much as possible, we also limit the model construction only to a small sub-set of the clusters, i.e., the master clusters. Other clusters only use the estimated model parameters to determine the model's estimation accuracy, which is a fairly inexpensive operation. The nodes use Eq. (6) for this purpose, which comprises of 3 multiplication and 4 addition operations to estimate a sample value for a third order model.

7. EXPERIMENTS AND DISCUSSION

In this section, we evaluate the proposed spatio-temporal compression technique.

7.1. Simulation Settings

In order to carry out comprehensive simulations for the evaluation, we used publicly available real-world data set [Madden 2003], containing traces for temperature, humidity, light and voltage. The network simulations are performed in TOSSIM. The signal reconstruction at the sink is conducted using MATLAB.

For extensive evaluation of the proposed scheme the simulations were performed on temperature and humidity data as they are not monotonic and continuously change during day and night and hence provides a good opportunity to test the adaptability to the changing dynamics.

The considered network consists of 52 nodes. We selected the AR models which are fairly inexpensive to evaluate. The order of the models is dynamically selected as described in Sec. 4.3.5. The model training length has been fixed to $T = 75$. [Tulone and Madden 2006] shows that long training windows do not necessarily improve either accuracy or efficiency. We also conducted initial studies to determine optimal value (range) for T and came to the same conclusion as [Tulone and Madden 2006] that accuracy and efficiency almost stagnates beyond $T = 75$, while the cost to train the model keeps increasing. Hence, in our simulations we deliberately did not vary this parameter and used the specific value of T=75.

AHC has been thoroughly evaluated with a wide array of various parameter values. The maximum allowed outliers ($O_{\#max}$) is set to the values of 15, 30 and 45 outliers per model cache per node. Model cache approximation window size W_Ψ is assigned values of 60, 75, 90, 105 and 120 samples. The desired error threshold ϵ is simulated for the values of 0.01, 0.05 and 0.1. Comprehensive simulations have been carried out to evaluate and analyze the impact and role of each parameter. As we discussed in Section 5.1, we require a larger payload size than the default size of 28 bytes in TinyOS. Hence, we have set the message payload size to 90 bytes. [Haas and Wilke 2011] shows that increasing the payload size to some extent does not increase the energy consumption, which we exploit here.

7.1.1. Comparison with the State of the Art Techniques. In order to put the performance of the proposed scheme in perspective, we compare the performance of AHC to that of our prior delay-tolerant spatio-temporal data collection scheme, i.e., ASTC [Ali et al. 2011], that of the time-series based real-time data spatio-temporal collection scheme

PAQ [Tulone and Madden 2006] and that of another class of in-network compression methods, namely *transform compression* that is based on signal compression. This comparison gives us an even more comprehensive study of the performance of AHC alongside other data compression methods that are based on a different paradigm. The dataset that is being used by our simulation shows high compressibility of the temporal data under Discrete Cosine Transform (DCT) [Ahmed et al. 1974]. We evaluate the performance of representative Distributed Transform Coding or Distributed Transform Compression (DTC) method that uses DCT as its compressive basis [Duarte et al. 2012]. The sensor nodes first compute the DCT of the temporal data and select the largest DCT coefficients. The magnitude and location of the most significant DCT coefficients are then communicated to the sink. The sink reconstructs the original data by applying the inverse DCT transform on the received data while setting the value of non-significant coefficients to zero. We set the accuracy requirements for ASTC, AHC, PAQ and DTC the same and compare the amount of in-network transmissions required by each method to fulfill the accuracy prerequisite. For DTC, we observed that a high enough accuracy is attainable by looking for the most significant DCT coefficients in the first 20 values of the DCT transform. Accordingly, we limit the size of the DCT transform in order to avoid too much memory requirements as the memory of a sensor node is quite limited.

7.1.2. Design Parameters and Trade-offs. Various parameters were introduced while developing AHC. Each parameter influences the performance achieved by AHC. In Section 4.3.5 we have already discussed the effect of certain parameters, such as model order, on the performance of AHC. Accordingly, an automated mechanism was developed to optimally select the best values for these parameters. However, we have to explicitly define other parameters, such as error threshold (ϵ) and approximation window (W_Φ), user accuracy requirements and delay-tolerance level. Hence, before discussing the results, we describe here the implications of various parameters and how they can affect compressibility, efficiency and accuracy.

User defined error threshold (ϵ) defines the maximum error desired in the reproduced data at the sink. It plays a key role in the degree of compression that we can achieve both temporally and spatially. In terms of temporal compressibility, stringent accuracy requirement (lower values of ϵ) can increase the number of resultant outliers. It can decrease the number of sample values that can be approximated by a given model, as a model is valid only for a limited number of outliers ($O_{\#max}$).

In terms of temporal compression, smaller values of $O_{\#max}$ will invalidate a model quickly and we may require more models per model cache to approximate the same number of samples. For spatial compression fewer neighboring clusters may accept a model cache from a master cluster for fewer number of outliers, forming smaller cliques and generating more model caches. Increasing the allowed outliers may increase the temporal compressibility and form larger clique (hence fewer model caches). But larger $O_{\#max}$ also means transporting larger number of raw sample values, which may in turn increase the message cost again.

In terms of temporal compression the larger the value of W_Ψ, less number of model caches are required to transport the data. However, in terms of spatial compression, increasing W_Ψ may negatively impact the spatial compressibility, because larger W_Ψ requires the clusters to agree for a larger number of samples (hence longer duration of time). Hence, larger W_Ψ may result in smaller cliques and may consequently require more model caches to be constructed. Increasing W_Ψ also means increasing the latency/delay in reporting the samples to the sink. Hence, the upper bound for W_Ψ is already defined by the tolerated latency in collecting the sampled values.

The discussion about the parameters and their impact on the degree of compression provides the basic understanding of their behavior while discussing them independently. However, they do influence each other. For example, ϵ influences the resultant number of outliers or $O_{\#max}$ influences the number of a models in a model cache. We will further discuss these parameters and their inter-dependencies in the results section next.

7.2. Experimental Performance Evaluation

Our performance evaluation is based on two key metrics: Accuracy of collected data and message efficiency. Accuracy measures how closely the sampled data is approximated after being approximated through AHC and reproduced on the sink. Whereas, message efficiency measures the number of messages required during the whole operation of AHC. Less message efficiency (more number of messages) would imply less energy efficient, as more energy would be consumed with more messages. Similarly, more message efficiency (fewer number of messages) would imply more energy efficient.

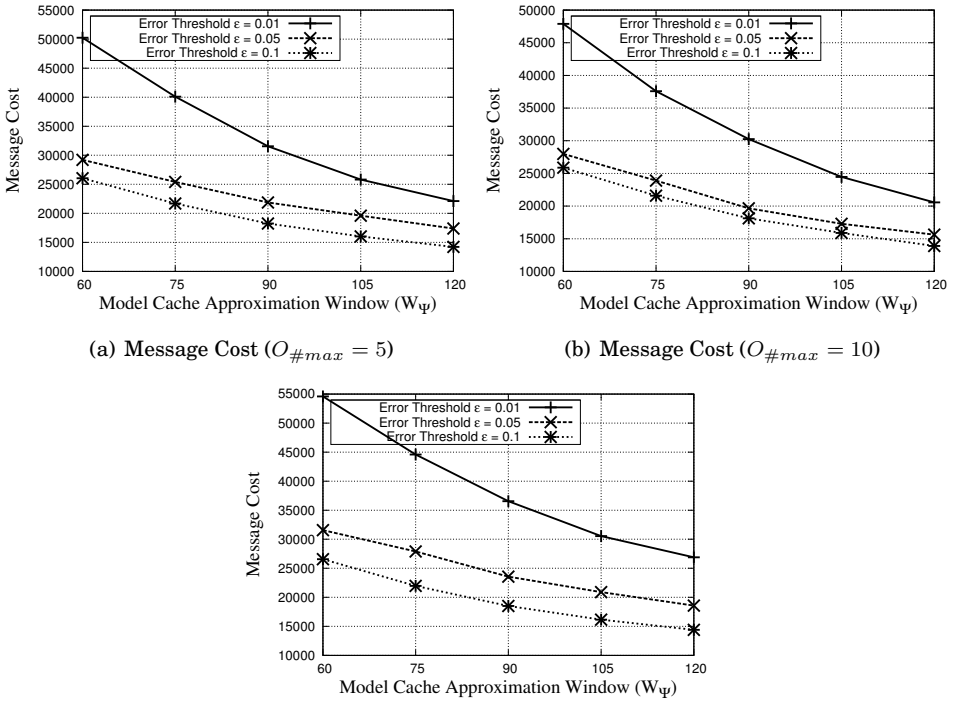
7.2.1. Efficiency. We define efficiency in terms of message overhead, i.e., the total message transmissions that account for all messages transmitted during all three stages of AHC. Message overhead comprises of intra-cluster and inter cluster message exchange for model caches construction and verification, and then reporting of the model caches and outliers to the sink.

Total Message Overhead. Fig. 8 shows the total message cost for intra and inter-cluster communication and model caches and outliers transport to the sink. Fig. 8 (a), (b) and (c), show message cost for maximum allowed outliers ($O_{\#max}$) of 5, 10 and 15 per model per node respectively. Each figure depicts the variation in message cost for increasing approximation window (W_Ψ) for different error thresholds (ϵ). From Fig 8, we make the following notable observations:

(1) For stringent accuracy requirement, i.e., $\epsilon = 0.01$, the message overhead is highest and the message cost decreases rapidly with decreasing ϵ, i.e., to $\epsilon = 0.05$ and $\epsilon = 0.1$, resulting in fewer outliers and more clusters accepting the model caches forming larger cliques resulting in fewer messages to be transported.
(2) The message overhead decreases with increasing approximation window because, as explained in Section 7.1.2, the increasing approximation window requires fewer rounds to complete the complete length of data to be compressed and transported. It results in lesser number of model caches to be constructed, saving the intra-inter cluster communication costs and model cache transportation costs.
(3) Message overhead drops when $O_{\#max}$ is increased to 10 (Fig. 8(b)) but the total message overhead increases when $O_{\#max}$ is increased further to 15 in Fig. 8(c). It happens due to drop in inter- and intra-cluster messages and increase in outliers message overhead.

Message Overhead for Inter- and Intra-Cluster Communication. : We now dig deeper in the total message transportation costs results and explore further why we observe various trends while discussing Fig. 9 and 10, depicting the total message cost excluding the outliers cost and only outliers cost respectively. In Fig. 9 (a), (b) and (c), similar to Fig. 8, we depict the inter- and intra-cluster message exchange for various maximum allowed outliers (excluding the outliers cost). The interesting results to be observed here are:

— We observe a considerable drop in the inter- and intra-cluster message overhead when the error threshold is relaxed from $\epsilon = 0.01$ to $\epsilon = 0.05$ in Fig. 9 (a). The re-



(a) Message Cost ($O_{\#max} = 5$)

(b) Message Cost ($O_{\#max} = 10$)

(c) Message Cost ($O_{\#max} = 15$))

Fig. 8: Total Message Cost for Inter- and Intra-Cluster Communication, Cache Construction and Caches and Outliers Transportation of Temperature Data to the Sink

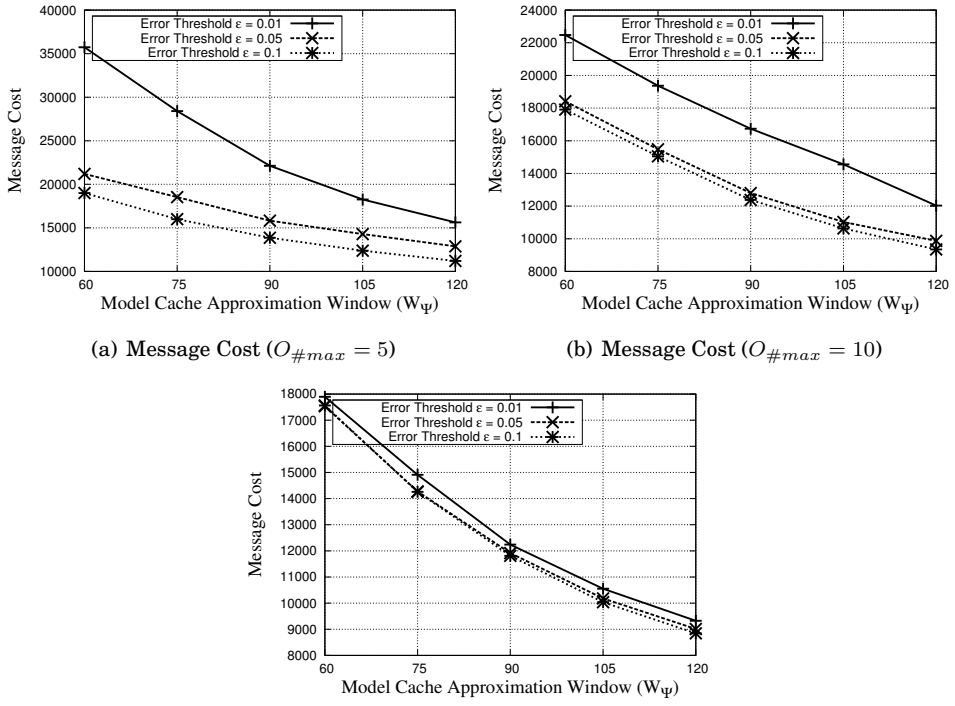
laxed error threshold results in larger differences between the approximated value and the sensed values to be tolerated. Hence, we have larger cliques, fewer model cache rejections, fewer new model cache constructions, fewer model caches to be transported to the sink and consequently fewer overall messages to be exchanged.

— Reduction in message overhead is observed with increasing $O_{\#max}$. More outliers relax the requirements for the acceptance of a model cache by a neighboring cluster and allow it to accept a model cache in spite of more outliers. Hence, we also observe a significant drop in the message overhead for $\epsilon = 0.01$ when $O_{\#max}$ is increased from 15 to 30 as depicted in Fig. 9 (a) and (b). However, because of relaxed error threshold that results in wider model cache acceptance by the neighboring clusters in general, we do not observe a similarly significant drop in Fig. 9 (c).

Message Overhead for Reporting Outliers. Fig. 10 depicts only the message cost associated with the outliers for various $O_{\#max}$ and ϵ. The notable observations are:

— Similar to the cost of model cache messages, we observe very small changes in message cost for relaxed error threshold when maximum outliers are increased.

— However, contrary to the model cache, we see a significant increase in message overhead as we increase the maximum outliers for $\epsilon = 0.01$. The outliers are expensive to transport not only because they are raw sample values but also because they carry extra overhead of value indices and the value owner id. Whereas, for $\epsilon = 0.05$ and $\epsilon = 0.1$ the message cost remains almost similar because relaxed accuracy re-



(a) Message Cost ($O_{\#max} = 5$)

(b) Message Cost ($O_{\#max} = 10$)

(c) Message Cost ($O_{\#max} = 15$))

Fig. 9: Message Cost for Inter- and Intra-Cluster Communication, Cache Construction and Transportation of Temperature Data to the Sink for Various $O_{\#max}$

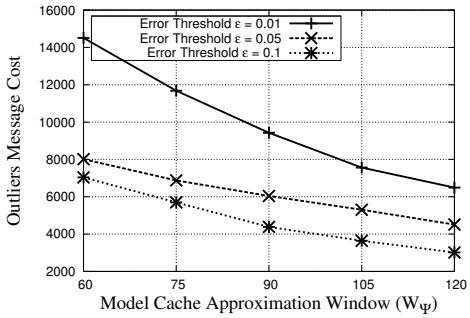
quirements do not dictate the data to be more accurate and does not produce any significantly greater number of outlier messages.

— Though we observe a consistent increase in message overhead with increasing $O_{\#max}$, we initially observe a reduction in the total message overhead in Fig. 8, because the reduction in the inter- and intra-cluster message cost is greater than the increase in the outliers message overhead. However, when we increase $O_{\#max}$ even further, the increase in the outliers message cost exceeds the reduction in the inter- and intra-cluster message cost and the total message starts to increase again.

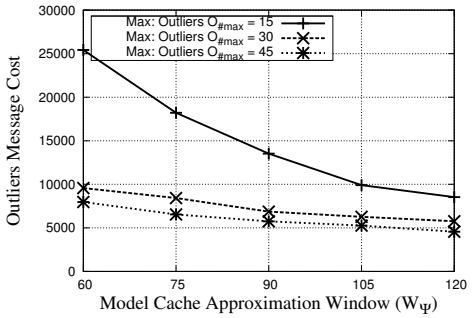
In Fig. 11, we compare the message cost of AHC with ASTC with extended approximation window W_Ψ from 35 to 120 samples and error threshold from ϵ from 0.01 to 0.2 for AHC. Whereas, ASTC has been simulated at $\epsilon = 0.2$. We can see that AHC consistently performs betters than ASTC by a large margin not only for the same error threshold (i.e., $\epsilon = 0.2$) but even for lower error threshold thanks to adaptive modeling (Section 4.3.4) and dynamic approximation window (Section 4.3.6) that considerably reduce the inter- and intra-message cost and yield considerably fewer outliers.

7.3. Comparison to Related Work

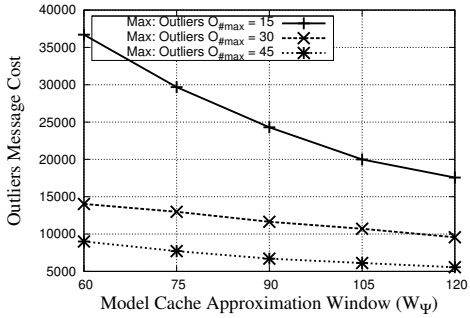
Efficiency. Fig. 12 depicts the message overhead comparison between AHC, ASTC, PAQ and DTC. We observe that AHC clearly outperforms the other techniques in terms of message cost, which was the design goal for AHC. The large gap between PAQ and AHC is due to the design choices, as PAQ targets realtime monitoring applications,



(a) Message Cost ($O_{\#max} = 5$)



(b) Message Cost ($O_{\#max} = 10$)



(c) Message Cost ($O_{\#max} = 15$))

Fig. 10: Outliers Message Cost for Temperature Data Set to the Sink for Various $O_{\#max}$

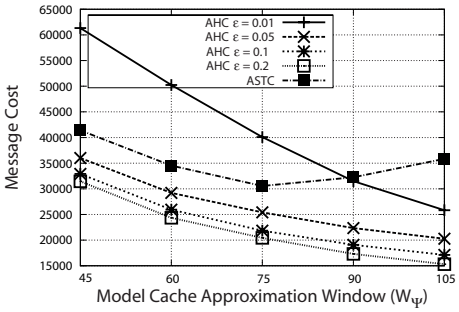


Fig. 11: Impact of ϵ on Message Cost on ASTC and AHC

hence in contrast to AHC it can not exploit the temporal redundancies. Whereas, ASTC similar to AHC not only exploits the spatial and temporal redundancies but also exploits the delay-tolerance. Hence, it also successively collects the data samples, compresses them and then the compressed models are sent to the sink collectively. However, adaptive model and dynamic prediction window of AHC give it an edge over ASTC that reduce the message cost further. Consequently, we can conclude that if the application can tolerate delays in data collection then AHC could be a better choice, as its design takes into consideration the application delay tolerance. Hence, for delay

tolerant applications AHC can compress the data even further and reduce the message cost.

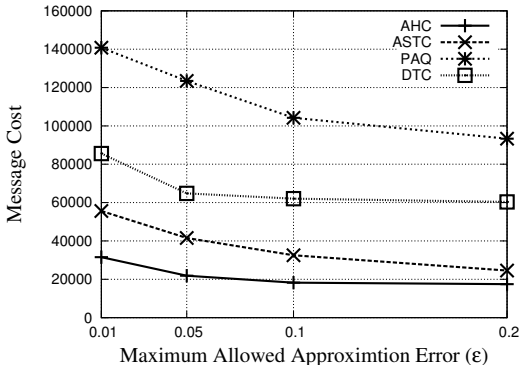


Fig. 12: Message Cost Comparison Between Various Compression Scheme for Different Error Thresholds ϵ

Accuracy. In Fig. 13, we depict the mean error of all sensor nodes in the network for the entire length of monitored data for AHC, ASTC and PAQ. The figure depicts for AHC the attained error versus the maximum error thresholds for various prediction windows. The error thresholds set in the simulations mean very high accuracy requirement. The error threshold of $\epsilon = 0.01$, $\epsilon = 0.05$ and $\epsilon = 0.1$ represent maximum error of 0.044%, 0.22% and 0.43% of the mean sample value. In Fig. 13, we observe an increase in the attained error but it always remains well below the defined maximum threshold. We can observe that all three schemes easily attain the user accuracy requirement. However, the message cost incurred to attain the desired accuracy level shows the efficiency of the scheme. We can put the results from Fig. 13 in perspective by looking at the message cost (Fig. 12) incurred by each scheme to achieve the accuracy level. By considering both figures, we observe that AHC is manifolds more efficient then ASTC and PAQ in terms of message cost in order to meet the same user accuracy requirement. Hence, AHC is able to meet the user requirement with fewer number of messages that effectively reduces the energy consumption and can prolong the life of the individual nodes and the network overall.

Note: We cannot include DTC in our comparison w.r.t. accuracy, since there is no equivalent parameter for error threshold ϵ in DTC. Nevertheless, Fig. 12 still gives a good comparison of the discussed compression methods. In Fig. 12, both axes of the diagram correspond the parameters that exist in all classes of the compression schemes mentioned here. Our implementation of DTC involves transmitting the largest DCT coefficients of the transformed temporal data. The number of transmitted coefficients is determined before data gathering such that a certain level of accuracy is achieved when the DCT coefficients are sent to the sink. Determining the outliers for a given error threshold ϵ requires inverse DCT computation at the sensor nodes which is beyond the hardware capabilities of these nodes. Therefore, it is not possible to compare DTC with ASTC, AHC and PAQ based on the error threshold ϵ as such an implementation of DTC that finds outliers depending on a given ϵ is not realistic.

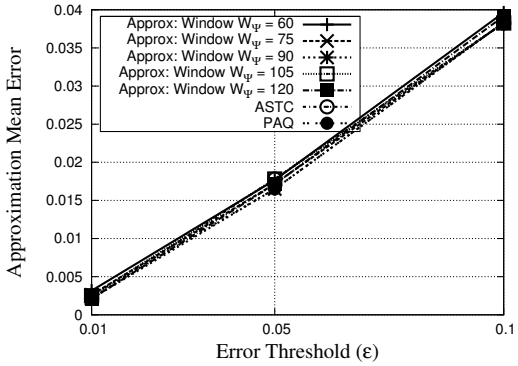


Fig. 13: Mean Approximation Error on Sink for Temperature Data set

7.4. Versatility - Phenomena Independence

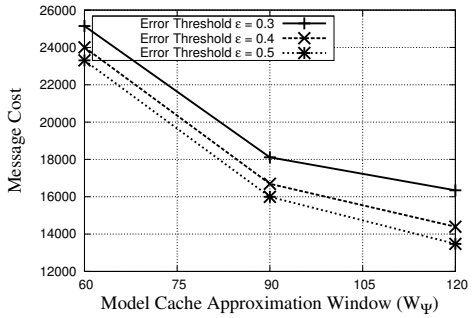
In order to evaluate the versatility of the proposed scheme, we also carried out extensive simulations (however, to save space we show here only limited results), similar to temperature, for the humidity data that is also available in the real-world data set [Madden 2003].

Efficiency. Similar to temperature data, Fig. 14 depicts the message cost results for humidity. Fig. 14 (a) shows the total message cost incurred in inter-intra cluster communication, model cache and outliers transportation. Fig. 14 (b) and (c) further break down the message cost. Fig. 14 (c) shows only the message cost related to transporting the outliers to the sink, where as Fig. 14 (b) depicts the rest of the message cost related to inter-intra cluster message exchange and model caches transportation. The trends observed in Fig. 14 are similar to the ones discussed related to the temperature data, hence we do not discuss them again here. However, these results signify the versatility, consistency and adaptability of the proposed scheme to various data types.

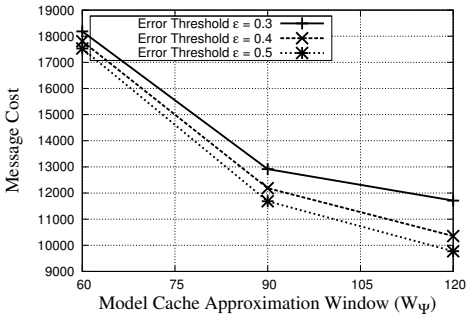
Accuracy. Fig. 15 depicts the mean approximation error for the humidity data set reproduced from its model caches on the sink. Similar to temperature data set we can observe that the proposed scheme not only meets the accuracy requirement but easily exceeds the desired accuracy level. We can further observe that the reproduced data error remains well below the error threshold even for one standard deviation around the mean.

8. CONCLUSION AND FUTURE WORK

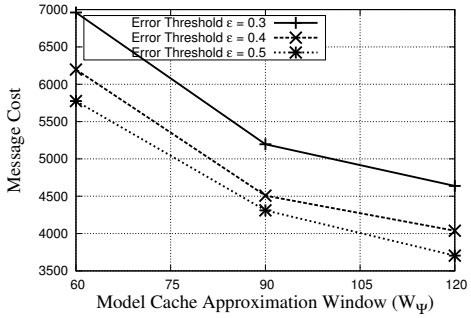
We have developed AHC, a fully distributed spatio-temporal data compression technique for accurate continuous sensor data collection in WSN. AHC dynamically self-adapts to approximate the monitored attribute both in space and time. In order to achieve the adaptability, AHC proposes (a) automated mechanisms to determine optimal models to best approximate the observed attribute, (b) dynamic hierarchical clustering based on models, and (c) automated mechanisms to adjust the compression scopes both in time and space. The use of 'simple models batches' instead of complex monolithic models was the key idea to design a technique that adapts to the dynamic changes of the monitored attribute. This key design choice has allowed for the first time to delay data collection rounds as tolerated by the application, thus, maximizing compression ratio and significantly reducing the message cost. In our experiments, we were able to reconstruct the signal from spatio-temporally compressed data on the sink with granularity of a single node and mean error less than 0.04%. In order to



(a) Total Message Cost



(b) Inter-intra Message Cost



(c) Outliers Message Cost

Fig. 14: Message Cost for Compressing and Transporting Humidity Data to the Sink

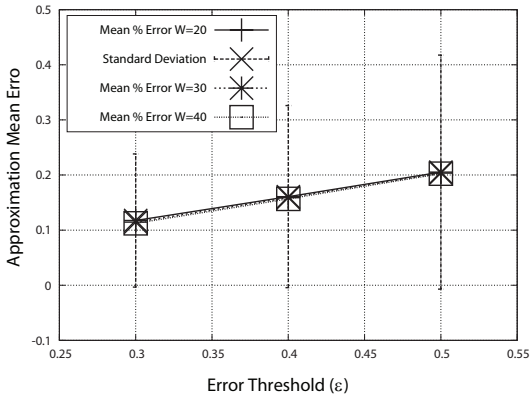


Fig. 15: Mean Approximation Error on Sink for Humidity Data Set

put the performance of AHC in perspective, we compare it to the relevant state of the art work PAQ, ASTC and DTC. The simulation results demonstrate that AHC easily outperforms the rest for delay-tolerant applications that it specifically targets.

AHC is our first step to efficiently transport large volumes of wireless sensing data with accuracy guarantees in extremely computation/energy constrained devices and networks. Despite this effort, additional work to further extend the network lifetime

is needed. For instance integrating a duty cycling technique or intelligent selection of subset of nodes while maintaining the spatial resolution to save energy without sacrificing the required data accuracy. In addition, we plan to enhance (a) the resilience of AHC through tolerating node crashes and spurious data, and (b) the AHC functionality through support of multi-variate compression.

REFERENCES

ABBASI, A. A. AND YOUNIS, M. 2007. A survey on clustering algorithms for wireless sensor networks. *Computer Communication 30*, 2826–2841.

ABELLO, J., PARDALOS, P., AND RESENDE, M. G. C. 1999. On maximum clique problems in very large graphs. *External memory algorithms*, 119–130.

AHMED, N., NATARAJAN, T., AND RAO, K. R. 1974. Discrete cosine transform. *IEEE Trans. on Computers 100,* 1, 90–93.

AKYILDIZ, I. F., POMPILI, D., AND MELODIA, T. 2005. Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks 3*, 257–279.

ALI, A., KHELIL, A., SHAIKH, F. K., AND SURI, N. 2009. MPM: map based predictive monitoring for wireless sensor networks. In *3rd Int. Conf. on Autonomic Computing and Comm. Sys.* 79–95.

ALI, A., KHELIL, A., SZCZYTOWSKI, P., AND SURI, N. 2011. An adaptive and composite spatio-temporal data compression approach for wireless sensor networks. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*. 67–76.

BAEK, S. J., DE VECIANA, G., AND SU, X. 2004. Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation. *IEEE Journal on Selected Areas in Communications 22*, 1130–1140.

CHU, D., DESHPANDE, A., HELLERSTEIN, J. M., AND HONG, W. 2006. Approximate data collection in sensor networks using probabilistic models. In *Proc. of the 22nd Int. Conf. on Data Engineering*. 48–.

DESHPANDE, A., GUESTRIN, C., MADDEN, S. R., HELLERSTEIN, J. M., AND HONG, W. 2004. Model-driven data acquisition in sensor networks. *Proc. of the 13th int. conf. on Very large data bases 30*, 588–599.

DUARTE, M. F., SHEN, G., ORTEGA, A., AND BARANIUK, R. G. 2012. Signal compression in wireless sensor networks. *Phil. Tran. of the Royal Society A: Mathematical, Phy. and Eng. Sciences 370,* 1958, 118–135.

DUNKELS, A., GRONVALL, B., AND VOIGT, T. 2004. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks.*

FAIZULKHAKOV, Y. R. 2007. Time synchronization methods for wireless sensor networks: A survey. *Programming and Computing Software 33*, 214–226.

FASOLO, E., ROSSI, M., WIDMER, J., AND ZORZI, M. 2007. In-network aggregation techniques for wireless sensor networks: a survey. *IEEE Wireless Communications 14,* 2, 70–87.

GEDIK, B., LIU, L., AND YU, P. S. 2007. Asap: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems 18,* 12, 1766–1783.

GEHRKE, J. AND MADDEN, S. 2004. Query processing in sensor networks. *IEEE Perv. Comp. 3*, 46–55.

GUPTA, H., NAVDA, V., DAS, S., AND CHOWDHARY, V. 2008. Efficient gathering of correlated data in sensor networks. *ACM Transaction on Sensor Networks 4*, 4:1–4:31.

HAAS, C. AND WILKE, J. 2011. Energy evaluations in wireless sensor networks: a reality check. In *Proc. of the 14th ACM int. conf. on Modeling, analysis and simulation of wireless and mobile systems*. 27–30.

JIANG, H., JIN, S., AND WANG, C. 2011. Prediction or not? an energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Tran. on Parallel and Dist. Sys. 22*, 1064–1071.

JINDAL, A. AND PSOUNIS, K. 2006. Modeling spatially correlated data in sensor networks. *ACM Transaction on Sensor Networks 2*, 466–499.

LEVIS, P., MADDEN, S., POLASTRE, J., SZEWCZYK, R., WOO, A., GAY, D., HILL, J., WELSH, M., BREWER, E., AND CULLER, D. 2004. Tinyos: An operating system for sensor networks. In *in Ambient Intelligence*.

LIU, C., WU, K., AND PEI, J. 2007. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Tran. on Parallel and Dist. Sytems 18*, 1010–1023.

LJUNG, L. 1998. *System Identification: Theory for the User (2nd Edition)*. Prentice Hall PTR.

MADDEN, S. 2003. Intel lab data. `http://db.cscail.mit.edu/labdata/labdata.html`.

MADDEN, S. R., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. 2005. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transaction on Database Systems 30*, 122–173.

MAHMUDIMANESH, M., KHELIL, A., AND SURI, N. 2010. Reordering for better compressibility: Efficient spatial sampling in wireless sensor networks. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*. 50–57.

MIN, J.-K. AND CHUNG, C.-W. 2010. Edges: Efficient data gathering in sensor networks using temporal and spatial correlations. *Journal of Systems and Software 83*, 271–282.

MINI, R. A. F., VAL MACHADO, M. D., LOUREIRO, A. A. F., AND NATH, B. 2005. Prediction-based energy map for wireless sensor networks. *Ad Hoc Netw. 3*, 235–253.

NAKAMURA, E. F., LOUREIRO, A. A. F., AND FRERY, A. C. 2007. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computer Surveys 39*.

PHAM, N. D., LE, T. D., PARK, K., AND CHOO, H. 2010. Sccs: Spatiotemporal clustering and compressing schemes for efficient data collection applications in wsns. *Int. Journal of Comm. Sys. 23,* 11, 1311–1333.

POLASTRE, J., SZEWCZYK, R., AND CULLER, D. 2005. Telos: enabling ultra-low power wireless research. In *Proc. of the 4th int. symposium on Information processing in sensor networks*. IPSN '05. IEEE Press.

SHAIKH, F. K., KHELIL, A., AYARI, B., SZCZYTOWSKI, P., AND SURI, N. 2010. Generic information transport for wireless sensor networks. In *Proceedings of the 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*. SUTC '10. 27–34.

SOLIS, I. AND OBRACZKA, K. 2005. Isolines: energy-efficient mapping in sensor networks. In *10th IEEE Symposium on Computers and Communications*. 379 – 385.

TOBAGI, F. AND KLEINROCK, L. 1975. Packet Switching in Radio Channels: Part II–The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution. *Communications, IEEE Transactions on [legacy, pre - 1988] 23,* 12, 1417–1433.

TOLLE, G., POLASTRE, J., SZEWCZYK, R., CULLER, D., TURNER, N., TU, K., BURGESS, S., DAWSON, T., BUONADONNA, P., GAY, D., AND HONG, W. 2005. A macroscope in the redwoods. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*. SenSys '05. 51–63.

TULONE, D. AND MADDEN, S. 2006. Paq: Time series forecasting for approximate query answering in sensor networks. In *European Conference on Wireless Sensor Networks*. 21–37.

VILLAS, L. A., BOUKERCHE, A., GUIDONI, D., ARAUJO, R. B., AND LOUREIRO, A. A. F. 2011. An energy-aware spatial correlation mechanism to perform efficient data collection in wsns. In *Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks*. LCN '11. 882–889.

VURAN, M. C., AKAN, O. B., AND AKYILDIZ, I. F. 2004. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks: The International Journal of Computer and Telecommunications Networking 45,* 3, 245–259.

WANG, C., MA, H., HE, Y., AND XIONG, S. 2010. Approximate data collection for wireless sensor networks. In *Proc. of the 2010 IEEE 16th Int. Conf. on Parallel and Distributed Systems*. ICPADS '10. IEEE Computer Society, Washington, DC, USA, 164–171.

WANG, C., MA, H., HE, Y., AND XIONG, S. 2012. Adaptive approximate data collection for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 23,* 6, 1004–1016.

WANG, L. AND DESHPANDE, A. 2008. Predictive modeling-based data collection in wireless sensor networks. In *Proceedings of the 5th European conference on Wireless sensor networks*. EWSN'08. 34–51.

WERNER-ALLEN, G., JOHNSON, J., RUIZ, M., LEES, J., AND WELSH, M. 2005. Monitoring volcanic eruptions with a wireless sensor network. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*. IEEE, 108–120.

YOON, S. AND SHAHABI, C. 2007. The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Transaction on Sensor Networks 3*.

ZHANG, C., LI, M., AND WU, M.-Y. 2006. Model-aided data collecting for wireless sensor networks. In *Proc. of the Second international conference on High Performance Computing and Comm.* HPCC'06. 692–699.

ZHAO, Y. J., GOVINDAN, R., AND ESTRIN, D. 2002. Residual Energy Scans for Monitoring Wireless Sensor Networks. In *IEEE Wireless Communications and Networking Conference (WCNC 02)*.

ZHOU, X., XUE, G., QIAN, C., AND LI, M. 2008. Efficient data suppression for wireless sensor networks. In *Proceedings of 14th IEEE International Conference on Parallel and Distributed Systems*. 599–606.