An Adaptive and Composite Spatio-Temporal Data Compression Approach for Wireless Sensor Networks[*]

Azad Ali, Abdelmajid Khelil, Piotr Szczytowski and Neeraj Suri
Department of CS, TU Darmstadt, Germany
{azad, khelil, szczytowski, suri}@deeds.informatik.tu-darmstadt.de

ABSTRACT

Wireless Sensor Networks (WSN) are often deployed to sample the desired environmental attributes and deliver the acquired samples to the sink for processing, analysis or simulations as per the application needs. Many applications stipulate high granularity and data accuracy that results in high data volumes. Sensor nodes are battery powered and sending the requested large amount of data rapidly depletes their energy. Fortunately, the environmental attributes (e.g., temperature, pressure) often exhibit spatial and temporal correlations. Moreover, a large class of applications such as scientific measurement and forensics tolerate high latencies for sensor data collection. Accordingly, we develop a fully distributed adaptive technique for spatial and temporal in-network data compression with accuracy guarantees. We exploit the spatio-temporal correlation of sensor readings while benefiting from possible data delivery latency tolerance to further minimize the amount of data to be transported to the sink. Using real data, we demonstrate that our proposed scheme can provide significant communication/energy savings without sacrificing the accuracy of collected data. In our simulations, we achieved data compression of up to 95% on the raw data requiring around 5% of the original data to be transported to the sink.

Categories and Subject Descriptors

C.2.1 [**Communication Networks**]: Network Architecture and Design —*network communications, wireless communication*; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

General Terms

Algorithms, Design, Measurement, Performance

Keywords

spatial and temporal correlations, hierarchical clustering, energy efficiency, modeling, approximation, accuracy

[*]Research supported in part by HEC and DFG GRK 1362 (TUD GKMM).

1. THE PROBLEM AND THE APPROACH

In WSN deployments, sensor nodes are often distributed over the monitoring area for unattended environmental monitoring or for supervisory functions. The typical WSN functionality being (i) local event detection and reporting it to the sink, and (ii) continuous data collection by sampling the environment and sending the samples to the sink. In this paper, we deal with continuous data collection. Applications utilize continuously collected data for (a) real-time decision making, such as surveillance, or (b) delay-tolerant processing such as modeling, analysis [17] and inference [3]. In this work, we develop adaptive modeling algorithms that exploit the delay-tolerance of the data collection to maximize data compression. As examples, various scientific applications, such as, volcano monitoring [21] or eco-systems [17] [5], require detailed ambient data with high spatio-temporal sampling resolution for fine-granular understanding of the physical processes. For such applications, a WSN is essentially a spatio-temporal sampling system. High spatial resolution is reachable if we sample at a granular node level. Similarly, high temporal resolution is achieved by transmitting the samples repeatedly. Consequently, large data volumes need to be sent to the sink. Sending a message is a costly energy consuming operation and fetching more data from the network results into more message transmissions and higher energy consumption. Sensor nodes have limited computational capabilities and are powered by finite energy sources. Accordingly, highly accurate continuous data collection is a challenging problem in WSN given their energy, communication and computational constraints.

Fortunately, WSNs exhibit naturally high redundancy in spatial sampling due to the redundant sensor node deployment for connectivity and failure tolerance. Often, the sampled attributes are also temporally compressible due to temporal correlation [18]. Redundant deployment and temporal correlation allow a significant reduction of communication overhead through spatio-temporal compression. Various WSNs deployed for scientific monitoring [21], [5], [17] 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 *latency-tolerance* in reporting the data to the sink opens up fundamental design flexibility in data collection to significantly improve WSN energy efficiency. We propose a scheme to exploit the redundancies present in WSN along with the data collection delay-tolerance to maximize spatio-temporal data compression.

Our Contributions: A number of varied attempts such as aggregation, temporal compression [10] [8] [24] and spatial compression [6] [18] [23] have been proposed to reduce data volume to be transported to the sink. However, only a few composite spatio-temporal approaches, such as [18] [22] 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 no self-adaptability or (c) limited in their scope of exploiting either spatial or temporal correlation reducing compressibility.

Based on this background, we summarize our contributions as follows:

- A fully decentralized Adaptive and Composite Spatio-Temporal Compression (ASTC) technique.
- First exploitation of latency tolerance in reporting for maximal spatio-temporal compression.
- A novel model-based hierarchical clustering that adapts to attribute dynamics for spatio-temporal compression.
- Mechanisms to guarantee attribute value accuracy to be within defined bound and ensure the sample granularity to be on node level.

The paper is structured as follows. We present the system model and design requirements in Section 2. The proposed composite ASTC compression is presented in Section 3. Section 4 details the analysis and evaluation of ASTC with related work discussed in 5.

2. PRELIMINARIES

2.1 System Model

We consider a WSN composed of N static sensor nodes $S = \{s_1, s_2, \ldots, s_N\}$ and a static sink. Each sensor node periodically samples the environment attribute values every T time units. The nodes are battery powered with limited processing and storage capabilities. The WSN deployment consists of arbitrary node distributions with varying spatial node densities as per coverage requirements. We consider the clocks of nodes to be synchronized [7]. A reliable end-to-end data transport service, such as [15], to transport messages from sensor nodes to the sink and acknowledgement schemes to ensure inter-node message delivery are utilized.

Notation	Description
C_i	ith Cluster
S_i	ith Sensor node
V	Sample Value Queue
ϕ	Model Parameters
Φ_i	ith Model
Ψ	Model Cache
$m_{\#}$	Models in Ψ
ϵ	Error Threshold
$O_{\#max}$	Maximum Outliers
W_p	Prediction Window

Table 1: Notations

2.2 Design Requirements

Our objective is to maximize the spatio-temporal data compression with accuracy guarantees for continuously monitored phenomenon. Conceptually, our requirements are:

1) The sink should be able to reconstruct the signal (sample values) of each node from collected compressed data.
2) The accuracy of the reconstructed signal should be within the application-driven error bound.
3) The number of messages to report data to the sink should be minimal.
4) The algorithms should adapt to evolving attribute dynamics and be independent of network properties such as node distribution, topology and routing protocols.
5) The data compression models should be efficiently computable on resource-limited sensor nodes.

3. THE PROPOSED ADAPTIVE SPATIO-TEMPORAL COMPRESSION (ASTC)

We propose a decentralized adaptive composite compression technique while exploiting application latency tolerance. Before discussing ASTC in detail, we first describe the redundancies in sampled values and how we can conceptually exploit them for data compression.

3.1 Redundancies and Correlations in a WSN

WSN generally exhibit both spatial and temporal redundancies due to redundant node deployment and spatial and temporal correlations between the sampled values. It is also substantiated by our observation of real world data [11] where we noticed that the trends/patterns are similar for a large number of nodes in each other's physical neighborhood, as shown in Fig. 1 by gray shaded regions. This key observation suggests that we can construct a single model for the nodes (spread over a large area) that exhibit similarity in sampled values. The redundant deployment and dynamic phenomenon highlight two key aspects of correlations between nodes as (1) the nodes in closest neighborhood (1-hop distance) have the highest spatio-temporal correlation and the correlation within 1-hope cluster is fairly persistent, and (2) correlations between 1-hop clusters change over time due to changing phenomenon distribution over nodes. Hence, a certain group of 1-hop clusters may be correlated for a certain time duration and after some time, when the phenomenon has changed, another group of 1-hop clusters may be correlated, as shown in Fig. 1 at time $t + \Delta t$ the clusters belong to different regions. Hence, we propose to exploit the spatial redundancy in two stages, i.e., proactive 1-hop cluster formation that are highly correlated and then merging the 1-hop clusters to larger regions/clusters. Spatial compression in our approach differs from contemporary approaches, such as [8] [22], that form large monolithic clusters. ASTC exploits temporal redundancy by constructing simple models on 1-hop clusters.

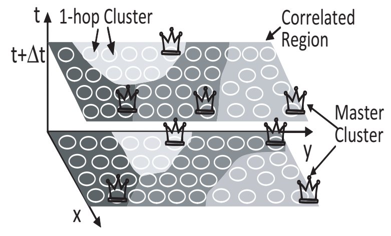

Figure 1: Phenomenon Distribution

The asymmetry in correlations between the nodes in close proximity and the nodes farther away makes the spatio-temporal compression a challenging problem. The nodes in close proximity of 1-hop distance generally exhibit persistently high correlations in sample values. However, the correlations between the 1-hop clusters are generally nonpersistent and dynamic when observed over a long time window. The existing literature addresses this asymmetry by limiting the modeling to just 1-hop clusters [18], assuming the attribute dynamics to be constant [4], requiring assistance from the sink [27], [28] or by clustering based on the instantaneous values rather than models [29], [30], [8].

3.2 A Guide through the ASTC Approach

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

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

In Stage 1, correlated sensor nodes form small 1-hop clusters based on a short history of the attribute values to exploit strong local correlation. 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. 1 bearing crown). Each constructed model is limited to the respective master clusters and approximates the sampled values of all the member sensor nodes of the master cluster.

In Stage 3, we propose schemes 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 the model. The clusters accepting the model merge to form a correlated region or larger clusters and further propagate the model to their neighboring clusters. 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 spatial compression in ASTC is a two level hierarchical clustering. The first and second hierarchy levels are formed during Stages 1 and 3 of ASTC. 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 sections, we detail the ASTC stages.

3.3 Stage 1: 1-hop Cluster Formation

In contrast to existing approaches, we initially construct small 1-hop clusters, instead of large monolithic clusters, to exploit local spatial correlations. The 1-hop clusters are later (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 high maintenance costs as we will elaborate in Section 3.5. As 1-hop cluster formation and inter-cluster routing is a well explored area in WSN [1], we only briefly describe them here. 1-hop clusters are formed based on the similarity of short history of the values that are transmitted by the nodes that candidate to be the cluster head. All nodes are initially eligible to be a cluster head candidate until some surrounding node announces its intent to be a cluster head as triggered by random timer. The candidate cluster head transmits a short history of the sampled values. The sensor nodes join the cluster for which the the difference between their sampled values and received sample values is within the accuracy requirement set by the application. The nodes unable to join a cluster, initiate their own cluster formation requests. Hence, a 1-hop cluster (C_i) is a set 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., $C_i = \{S_{CH}, S_1, S2, \ldots, S_r\}, \wedge\forall S_i \in C_i\ hopdist(S_{CH}, S_i) = 1$. Members of one cluster might be able to listen to more than one cluster heads but exclusively belong to one cluster, i.e., $C_i \cap C_j = \emptyset,\ i \neq j$. Each 1-hop cluster head discovers immediate 1-hop neighboring clusters (C_N) around it. In the discovery process they exchange the cluster Ids (same as cluster head Id) and hop distances of cluster head from the sink. They also fix the gateway nodes that are used to communicate between the two neighboring cluster heads. The 1-hop cluster formation process is performed only once. We do not explicitly refresh 1-hop clusters to maintain the correlations between the nodes within these clusters. Rather, we design Stages 2 and 3 of ASTC 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 3.5.5.

3.4 Stage 2: Temporal Modeling on Clusters

Next, we elaborate a temporal compression scheme to model the sampled values of all the member nodes within the 1-hop cluster. We emphasize that the model construction is carried out only by master clusters (selection of master clusters is explained in Section 3.5.1), while the rest of the clusters use the models constructed by the master clusters to approximate their sampled values.

Definition 1. *$x(t)$ is an Autoregressive Moving Average Process $ARMA(p,q)$ of order (p,q) $p,q \in \mathbb{N}$ and $\forall$ t,*

$$x(t) = \Phi(\phi, \theta, x(t)) = \begin{array}{l} \phi_1 x(t-1) + \cdots + \phi_p x(t-p) + \\ \theta_1 z(t-1) + \cdots + \theta_q z(t-q) \end{array} \quad (1)$$

where ϕ and θ are model coefficients, $z(t)$ is white noise with mean zero and variance σ^2, denoted as $WN(0, \sigma^2)$ and $\Phi(\phi, \theta, x(t))$ (or in short Φ) denotes a model.

The use of computationally complex 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. However, sampled values can be decomposed into short duration segments. Each segment can be dynamically modeled as $\mu + x(t)$ (where μ is the mean or linear component and $x(t)$ is the random component), allowing the nodes to adapt to unforeseen changes in the attribute dynamics without requiring prior knowledge of dynamics. Random component can be modeled by simple Autoregressive Moving Average (ARMA) models (Def. 1) or even simpler variants, i.e., Autoregressive (AR) and Moving Average (MA) models [3]. These models allow adaptability but are simple enough to be evaluated on common sensor nodes. We next describe how a master cluster constructs the model.

3.4.1 Master Cluster Model Construction

While all the sensor nodes sample the desired attribute, only the master clusters construct the models. The master cluster head maintains a queue V of n sample values to train the model. We set $q = 0$ and $p = 3$ in Eq. (1) to reduce the complexity of the model further. Eq. (1) reduces to

$$x(t) = \sum_{p=1}^{3} \phi_p x(t-p) \quad (2)$$

to construct a third order AR (AR3) model. The sampled queue is broken into mean and random components, i.e., $\mu + x(t)$. The AR3 model fitting error is minimized in mean-square error sense for the random component $(V - \mu)$ in the training queue as given by the following equation

$$\frac{\partial}{\partial \phi_p} E(t) = \frac{\partial}{\partial \phi_p} (\sum_{i=1}^{n} (x(t) - \sum_{p=1}^{3} \phi_p x(t-p)) = 0 \quad (3)$$

The predicted value based on model parameters is given by

$$\hat{V}(t) = \mu + \sum_{p=1}^{3} \phi_p \times (V(t-p) - \mu) \quad (4)$$

A model can predict only a certain number of sample values within accuracy bound referred to as *the prediction window*. Beyond the prediction window the model is rendered invalid. In order to adapt to the changing attribute dynamics a new model is constructed in the same fashion using the new sampled values in the sample queue.

Naturally, the sampled values may contain spurious and noisy values that result in modeling errors. However, accuracy of the model is important as it is used to approximate not only the master cluster member nodes sampled values but also the surrounding clusters. Hence, the cluster head involves the cluster members in model building by requesting the member nodes also to build the model. A few nodes are randomly selected to send their models to the cluster head. The cluster head broadcasts the models to the member nodes. Each member node fits each model to their sampled values and report back error for each model. The model with the minimum error is selected. The final ASTC scheme constructs a batch of models, referred to as *model cache*, instead of one model. This is detailed in Section 3.4.4.

3.4.2 Guaranteed Accuracy

We define two parameters to ensure the data reproduced from models on the sink to be within tolerable error bound, (1) an error upper bound (ϵ), and (2) the allowed maximum number of outliers $O_{\#max}$ that a node can tolerate to accept a model. ϵ is the maximum allowed error between the value predicted by the model and the actual sampled value.

$$\hat{x}(t) = \begin{cases} x(t), \text{if} \, |\hat{x}(t) - x(t)| > \epsilon; \\ \hat{x}(t), \text{otherwise} \end{cases} \quad (5)$$

If the model estimated value differs from the sampled value by ϵ, the sampled value is classified as an outlier (Eq. (5)). The cluster head gathers and reports the outliers to the sink to maintain accuracy within ϵ. Accordingly, $O_{\#max}$ is directly proportional to the desired accuracy level (ϵ).

3.4.3 Cluster Synchronization

We use the model constructed on the master cluster to also approximate the sampled values of surrounding cluster nodes. Though, the attribute dynamics are generally similar over a large area but change slightly from one cluster to another. Hence, when the model constructed on a master cluster is used to approximate the sample values in other clusters, the prediction window of the model can be different in different clusters. Different lengths of the prediction window results into approximating different number of samples with the same model in different clusters. This results into desynchronization of the clusters in terms of the approximated sample values. Once the clusters are desynchronized one cluster can not use the model of an other cluster (master cluster), as nodes in surrounding clusters could be approximating sample values ahead or behind the values for which the model was constructed. Hence, to avoid such desynchronization we restrict the prediction window length to a fixed value of W_p. Even if a model approximates the values within ϵ beyond W_p, the next values are considered to be in the next segment and are approximated by the next model. If the prediction window is shorter than W_p, the nodes compensate it with the outliers. If the model can not approximate the values even after allowing $O_{\#max}$ outliers, the model is rejected by the neighboring cluster node.

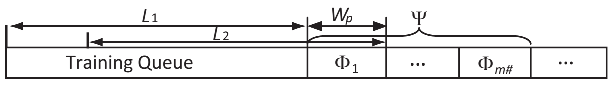

Figure 2: Model Caching

3.4.4 Model Caching

As per our design requirements (Section 2.2) we use simple models. Consequently, the prediction window is also expected to be short (in comparison to more complex models that may not be computable on sensor nodes) reducing the temporal compressibility. We assume tolerance in reporting latency. Hence, in order to increase the temporal compressibility, the master cluster heads construct a consecutive batch of models referred to as *model cache*. We denote the model cache by Ψ, which is a set of $m_\#$ models $\Psi = \{\Phi_1, \Phi_2, \ldots \Phi_{m\#}\}$. The model cache is constructed in the same fashion as a single model as explained in Section 3.4.1 but $m_\#$ models are constructed at the same time instead of just one as shown in Fig. 2. Each model in the model cache has a prediction window of W_p. Hence, to model $m_\#$ models the cluster head waits until $(m_\# - 1) \times W_p$ values are sampled in addition to the initial training queue. The first model Φ_1 is constructed from the sampled queue equal to the training length (Fig. 2, L_1). The next model Φ_2 is constructed from the next sampled attribute values (Fig. 2, L_2). It is repeated until the complete model cache Ψ has been constructed. To construct the next Ψ, additional $(m_\# - 1) \times W_p$ values are sampled and similarly modeled.

3.5 Stage 3: Merging 1-hop Clusters

In Stage 1, we exploited limited but strong spatial correlations to form 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 instantaneous value based monolithic clustering [1] [8] [22] to exploit the spatial correlation. They generally are very costly as they incur continuous maintenance cost. We on the other hand use models instead of instantaneous values while exploiting the spatial correlation. Large monolithic clusters based on models require tracking of all the 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 generally nodes are highly correlated. The second hierarchical clustering level is constructed by the 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. 2). 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 ASTC. Before we describe 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 surrounding clusters nodes.

3.5.1 Master Cluster Head Selection

The master cluster merges with the surrounding clusters based on its model cache to form cliques or correlated regions. We are interested in determining the smallest number of correlated regions in the network or the smallest set of model caches that best approximates the largest network parts in order to 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 2. *We define a clique as a network subgraph Q_C, a subset of the cluster heads, such that each cluster head in Q_C satisfies the model cache Ψ.*

Finding the master clusters that can construct the best model caches that form largest coverage clique is NP complete [2]. 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, hence 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 the model cache to the neighboring clusters. If we allow otherwise, i.e., 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 3.3. The cluster heads use this information to figure out the farthest cluster and hence the master cluster locally. In case of cluster heads having 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 3.4.4) can be used to approximate the sampled values of the nodes in the surrounding 1-hop clusters.

3.5.2 Model Cache Acceptance by 1-hop Clusters

The master cluster constructs the model cache as described in Section 3.4 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 3.4.2). To evaluate whether the received model cache approximates the sampled values, each member node predicts the values as:

$$\hat{V}(t) = \mu_{local} + \sum_{p=1}^{3} \phi_p \times (V(t-p) - \mu_{local}) \qquad (6)$$

where V are the sample values of the node, μ_{local} is the sample mean of the local cluster head and model parameters are as receiving 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 predict the values using received model parameters using Eq. (6) and the error, $\hat{V}(t) - V(t)$, is calculated and the outliers are counted. Total count of the outliers for each model Φ in the model cache Ψ for the prediction length W_p should be less than the maximum number of allowed outliers ($O_{\#MAX}$) for model acceptance (Section 3.4.2). Hence, for each model in the model cache to be accepted, the node counts the outliers for each model as:

$$count\,|x_i(t) - \hat{x}_i(t) > \varepsilon| \leq O_{\#max}, \hat{x_i} \in \Phi, j = 1...W_p \qquad (7)$$

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

Algorithm 1 Model Cache Acceptance

```
1: msg.type='$\Psi fit$';
2: msg.$\Psi$ = $\Psi$;
3: timer.start();
4: broadcast(msg)
5: function msgReceive(msg)
6: add $\Psi$ to $\Psi\_List$;
7: function timer.expired()
8: $\Psi$.accepted=($\Psi\_List$ total accepts> $\%r, r \in C$)
9: broascast($\Psi$.acceptance);
10: if $\Psi$.accepted then
11:     remove member that rejected $\Psi$ & add the new requesting
        member
12: end if
```

In Alg. 1 and 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 locally calculated means) to the cluster members (Alg. 1, L. 1-4). The cluster head starts a timer to wait for the responses to be collected (Alg. 1, 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. 3). Each node must accept all the models in the model cache to accept the model cache. The cluster head is not notified of the acceptance of the model cache (positive vote), rather when the timer expires on the cluster head it assumes the model cache to be accepted by the particular node. The negative votes are, however, explicitly reported along with the outliers to be reported to the sink (Alg. 2, L. 5). Withholding report for positive votes saves messages because most of the nodes in the 1-hop cluster accept the model cache. The members nodes send outliers to the cluster head instead of sending them to the sink (Alg. 2, L. 6). The cluster heads reports the outliers efficiently to the sink by sending multiple outliers in one message. The cluster head counts the votes for model cache once the timer has expired (Alg. 1, L. 7-8). A model cache is accepted as a model cache for the cluster if it received acceptance from at least $k\%$ member nodes (Alg. 1, L. 8).

3.5.3 Self-Adaptive Clique Formation

So far, we have described the formation of the 1-hop clusters (Section 3.3), the model cache construction on the master clusters (Section 3.4.1) and how a model cache constructed on the master cluster can be used to approximate the values of sensors in the surrounding clusters (Section

Algorithm 2 Model Cache Evaluation

```
1: function msgReceive(msg)
2: if msg.type==’$\Psi fit$’ then
3:     vote=(solve Eq. (7) for $\Phi$, $\Phi \in \Psi$)?reject:accept
4:     if $vote == reject$ then
5:         unicast(vote,outliers[],$C_H$);
6:     else if count($outliers > 0$) then
7:         unicast(outliers[],$C_H$);
8:     end if
9: end if
```

3.5.2). Here, we describe the clique formation based on the model caches constructed on 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 (Section 3.5.2). If the neighboring cluster head accepts the model cache it joins the clique and sends the model cache to its neighboring clusters and the process repeats. Hence, the clique formation is essentially controlled flooding of model cache over the cluster heads around the master cluster. The flooding stops once the model cache is rejected by the cluster heads. Next, we describe this scheme in detail.

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.

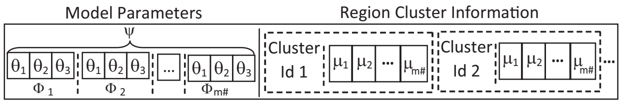

Figure 3: Join Message Payload Format

The master cluster initiates the clique formation by constructing the model cache (Section 3.4.4). Fig. 3 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 algorithm (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 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_{Msg}) and transfers it 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.

Algorithm 3 Clique Expansion

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

The clique expansion continues until Alg. 2 returns positive join responses for the new clusters. A cluster head knows the clique expansion has stopped locally around it 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. The join message is flooded from the master cluster till the final boundary of the clique. Hence, 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.

Algorithm 4 Clique Joining by Neighboring Cluster

```
1: function receive(msg)
2: if msg.type=’Join’ then
3:     $Resp_{Msg}$.accept=’false;
4:     Call Alg. 1 to evaluate msg.$\Psi$
5:     if msg.$\Psi$.accepted then
6:         $Resp_{Msg}$.accept=’true’;
7:         clique.add(msg.$C_{RequestCH}$);
8:         me.Boundary=true;
9:         Execute Alg. 3
10:    end if
11:    1HSend($Resp_{Msg}$, $C_{HR}$)
12: end if
```

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 an other 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 clusters head 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 clique formation. On the expiration of the wait time period the cluster head initiates clique formation. 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 such a restriction because the two cliques are growing based on two different model caches.

3.5.4 ASTC 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 large cluster body will be very costly, because it is built using the model cache, which requires the clusters to agree on a sequence of values over time. Hence, cliques are reconstructed rather than being maintained. We show in Section 4.1 that the cost 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 changes in 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 3.4.4. The process of the model cache construction and the clique formation is repeated and the sink receives accurate continuous data.

3.5.5 1-hop Cluster Dynamic Rearrangement

The 1-hop cluster members usually stay correlated. However, due to changes in physical phenomenon, the correlations even at the 1-hop cluster level 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 3.5.2 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 rest of the surrounding clusters. The 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. Such a rare condition arises due to a new phenomenon developing in this 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.

4. EXPERIMENTS AND DISCUSSION

We now evaluate the proposed spatio-temporal compression technique both analytically and via simulations.

4.1 Complexity and Compressibility Analysis

Our main objective is to perform maximal spatio-temporal compression with minimal number of messages without sacrificing data accuracy. The message overhead of our proposed scheme is very low. The maximum number of messages that a 1-hop cluster must exchange to process a model cache can be given by $Msgs = 2 \times \Psi_{Msg} + \sum_{S_k \in C_j, S_{k_{O\#}} > 0} Out_{Msg} + Resp_{Msg} + NBC_{Msg}$. The first message is the broadcast message (Ψ_{Msg}) to the member nodes to evaluate the model. The member nodes respond only if there is an outlier to be reported (Out_{Msg}). The cluster head reports to the requesting cluster head (through the gateway nodes) whether it accepted the model cache ($Resp_{Msg}$). If the model cache is accepted the cluster head broadcasts the model cache message again (Ψ_{Msg}) to the neighboring clusters (through gateway nodes). If the cluster is no longer on the boundary, it broadcasts the 'not-boundary-cluster message' (NBC_{Msg}) to the neighboring cluster through the gateway nodes. Hence, in the worst case the maximum number of messages a cluster head has to transmit are four. For a member node that is not the gateway node it is only one (Out_{Msg}, if there are outliers). For a member node that is also a gateway node it transmits four messages ($Out_{Msg} + \Psi_{Msg} + Resp_{Msg} + NBC_{Msg}$). The master cluster has an additional round of one message to construct a model cache.

The compression achieved within 1-hop cluster can be calculated as $Total_{bytes} = S_{\#\in C_i} \times W_p \times m_{\#}$. If there are $S_{\#\in C_i} = 7$ sensor nodes in a 1-hop cluster, the prediction window is $W_p = 15$, the number of models in Ψ are $m_{\#} = 3$, then one model cache is able to compress 315 sample values within one 1-hop cluster. For each 1-hop cluster joining the clique we achieve similar compression rate. Practically achieved compression is slightly lower due to a few outlier values (which is also bounded by the maximum number of allowed outliers $O_{\#max}$).

The computation overhead to achieve the compression is minimal. Only the master clusters construct the model cache, hence the master cluster heads have to compute matrices A and B for 3 unknowns of a linear system $AX = B$. The other nodes (either in the master cluster or the clusters joining the clique) need to solve a third order linear equation (AR3) and compare it with the sampled values.

The collection of data to construct Ψ introduces latency, which can be calculated as $Latency = m_{\#} \times W_p \times T$, where T is sampling period. Hence, Ψ and W_p are selected while keeping in consideration the maximum desired latency.

4.2 Experimental Performance Evaluation

ASTC Simulation Settings: We use a publicly available real-world data set [11], containing traces for temperature, humidity, light and voltage. We selected the temperature readings for four continuous days because, as compared to the other attributes, the temperature readings are not monotonic. The temperature continuously changes during day and night and hence provides a good opportunity to test the adaptability to the changing dynamics. The network simulations were performed in TOSSIM. The signal reconstruction at the sink is conducted using MATLAB. The main overhead of our scheme is to transport the *model caches* and the *outliers values*. Hence, we measure the efficiency by counting the total number of messages required to transport the model caches and the outliers. The network consists of 51 nodes with a data set of more than half-million data samples with the sampling period of 31 sec. We selected the AR3 model, set the training history length to 100, $O_{\#max} = 5$, $W_p = 20$, $m_\# = 3$ and $\varepsilon = 0.2$ (less than 1% mean error). In the following evaluation studies, one parameter is varied at a time and the others are fixed to the default values. To cover the complete spectrum for spatial and temporal compressions, we compare ASTC to its two variants. The first variant is the temporally limited case when the model cache is limited to one model only to compare to the temporally limited schemes. The second variant is the spatially limited schemes when spatial compression is limited to 1-hop cluster only. We also compare ASTC to PAQ [18] for PAQ's ability to dynamically adapt to unknown attribute dynamics. In order to compare to the schemes using no models we simulated for instantaneous value based clustering and further compare to continuous raw data collection to establish a baseline to measure the compressibility.

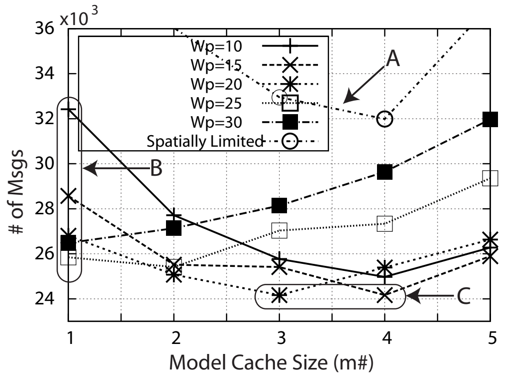

Figure 4: Message Cost for Varying W_p and $m_\#$

Spatio-Temporal Compression in Terms of ASTC Parameters: In order to elaborate the results we explain the spatio-temporal relations, as introduced in Section 3.1, in terms of parameters introduced in our scheme. The temporal compressibility can be controlled by changing both model cache size ($m_\#$) and prediction window (W_p). With each increment in $m_\#$ we increase one model in the model cache that compresses W_p additional samples as explained in Section 3.4.4. By increasing W_p we approximate more samples with the same model, which may not be efficient if W_p is too long and hence could result in more outliers. Hence, increasing W_p in comparison to $m_\#$ is not as efficient because for $m_\#$ we construct an additional model that approximates the new dynamics. The temporal compression for the given parameters is given by $m_\# \times W_p$ as shown in Section 3.4.4. The temporal compression scope influences the achievable spatial compression (hence the clique size). The short temporal compression implies the clusters have to agree for a short duration of time, which is highly likely. However, we will require more model caches to compress the complete sampled history. Long temporal compression scope can compress the whole history in fewer iterations but requires clusters to agree for a long duration of time, which results in smaller cliques. Hence, we have to find the compromise between the temporal and spatial compression to optimize the message cost. Our simulations provide a guide to choose the parameters for the optimal results.

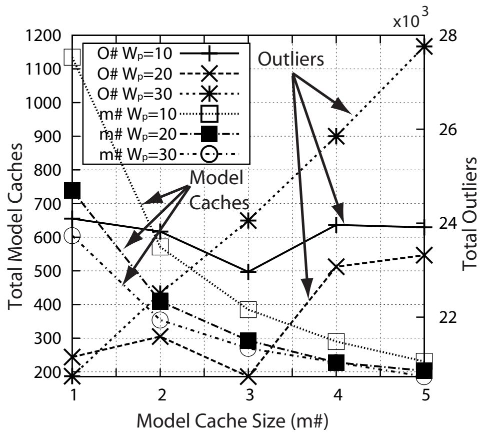

Figure 5: Total Number of Outliers and Model Caches

ASTC Efficiency and Compressibility: Fig. 4 depicts the message cost for $m_\#$ varied from 1 to 5 models per model cache and W_p varied from 10 to 30 approximated samples per model. Fig. 5 helps us understand the reason for changes in message cost as it depicts the generated model caches and the outliers with varying $m_\#$ and W_p. In Fig. 4 we initially observe a clear drop in the number of messages required to transport the data as we increase the models in $m_\#$, because each additional model compresses W_p more sample values. Fig. 5 confirms this by showing a clear drop in the model caches with increasing $m_\#$ and initial drop in the outliers for the short prediction window.

The number of messages, after the initial drop, start to increase in Fig. 4 for the shorter prediction window ($W_p = 10$ *to* 20) and model cache with the large number of models ($m_\# = 4$ *to* 5). This is due to (a) the limited size of 28 bytes for the message payload in TOSSIM, as we increase models we require more parameters and more messages to transmit all the parameters as shown in Section 3.5.3 (b) larger $m_\#$ and W_p increase the temporal compression making it difficult to approximate the cluster farther away from master cluster. Hence, it results in more outliers and smaller cliques. This is also evident for the long prediction window (30), where the message cost starts increasing very early on. Fig. 5 confirms the increase in the message cost by showing a clear increase in outliers for the large W_p and $m_\#$.

In Fig. 4 (Label A) we also illustrate the spatially limited case when the spatial compression is limited to only 1-hop cluster. The message cost is so high that we had to reduce the scale to show ASTC optimized results. PAQ [18] costs also lie on this line when $m_\# = 1$, however it is still high enough (47,215 messages) that it can not be displayed in the graph. The simulation results for instantaneous value based clustering required 109,862 messages. The continuous raw data collection resulted in 39,33,345 messages.

Fig. 4 (Label B) can also be used to study the temporally-limited case when we limit $m_\# = 1$ and allow clique formation. We can again see the same trend, i.e., the message cost decreases with increasing W_p as more clusters accept the model cache and increase only when the W_p gets too long (30) as it results in increased outliers and decreased spatial compression scope, resulting in smaller cliques.

ASTC has low overhead of outliers and low message cost (high compressibility). For example, in Fig. 5 for $m_\# = 3$ and $W_{max} = 20$ outliers on the cluster heads amount to roughly 5% of the raw data. Taking continuous raw data collection as a baseline to measure the message cost, ASTC requires around 0.6% of the messages required to transport the complete sampled history.

Parameter Selection: From the results in Fig. 4 and Fig. 5 we learn that the upper bound for $m_{\#}$ is dictated by the implementation platform as more models may require more messages due to limited payload size (in TOSSIM the maximum payload size is 28). However, W_p depends on the phenomenon. In our simulations $m_{\#}$ of size 4 and 5 and W_p 15 and 20 yielded the best results (Fig. 4, Label C).

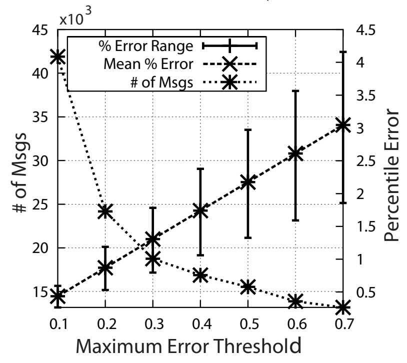

Figure 6: Total Communication Cost with Varying ε

ASTC Accuracy: Fig. 6 shows the effect of changing the error tolerance (ε) and maximum, minimum and average error induced after reconstruction. It further asserts that ASTC fulfills the accuracy requirements imposed by the user. Increasing the error threshold naturally results in less number of outliers and more spatio-temporal compressibility, hence the messages count decreases. Setting ϵ to 0.2 generates maximum 1.18%, minimum 0.53% and average error of 0.87%.

For maximum allowed outliers ($O_{\#max}$) neither large nor small values are beneficial as shown in Fig. 7. Too small $O_{\#max}$ requires strict matching of model to the sampled value, hence spatial compression is limited and smaller cliques are formed limited spatio-temporal compressibility. Allowing $O_{\#max}$ to be too large generates too many outliers requiring more messages to transport the outliers. In our simulations $O_{\#max} = 5$ yielded the best results.

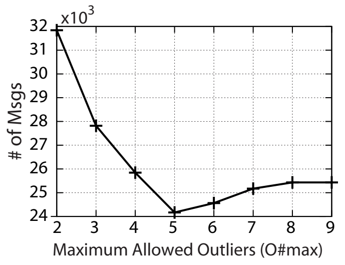

Figure 7: Total Communication Cost with Varying $O_{\#max}$

ASTC Latency: Given the sampling time of 31 sec the latency of constructing model cache in each ASTC round ranged from 310 sec for $m_{\#} = 1$ and $W_p = 10$, to the medium case of 1860 sec for $m_{\#} = 3$ and $W_p = 20$, to the extreme case of 4650 sec for $m_{\#} = 5$ and $W_p = 30$.

5. RELATED WORK

This paper focused 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 granularity while exploiting delay-tolerance in data collection. There is wide range of research work in WSN for data compression, simple aggregation, suppression, filtering such as TinyDB [12] and Cougar [9], to name a few. However, there is very limited work in the area of 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. [13] [10] [16] [24] are a few of many spatial compression techniques relying on compressive sensing, spatially correlated models, filtering and aggregation.

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 [14] and [23] construct models based on Markov-chains and time series, respectively.

Spatio-Temporal Compression There is very limited research in the area of spatio-temporal compression. We have attempted to give an exhaustive account of the state of the art in the field. In [19], 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 in WSN, which exploits these advantageous intrinsic features of the WSN paradigm. This work does not focus on continuous data collection contrary to our target applications. [29], [30] and [8] form large monolithic clusters based on instantaneous values or aggregates of the values. Hence, they continuously collect data from the member nodes to check for consistency and maintain these clusters by breaking up and merging them. We on the other hand use hierarchical clustering, i.e., (1) 1-hop cluster requiring rare maintenance and (2) clique that are not maintained rather reconstructed at a minimal message cost. Additionally, cliques (clusters) in our work are formed based on models rather than instantaneous values. [18] proposes realtime data collection through continuous modeling. It is closest to our proposed scheme in terms of adaptability to changing dynamics. Hence, we have included it for comparison in simulations. However, it is limited both in spatial and temporal compressibility. The focus of our proposed scheme, in contrast to [18], are the applications that are delay-tolerant in data collection. [25] uses Kalman filter for modeling within 1-hop clusters, which incurs heavy computational cost. [26] proposes to reduce energy consumption through hierarchical aggregation. Both [25] and [26] 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. [27] and [28] use centralized heuristics for cluster formation and nodes correlation determination. Our proposed scheme is completely distributed and does not require any information from the sink other than a few parameters initially required to setup the proposed distributed scheme as per the user requirements. [22] has two modes of operation, i.e., interactive mode is limited to spatial compression only and streaming mode performs both spatial and temporal compression. Streaming mode of [22], [20] and [4] construct probability density function (pdf) for modeling the attribute. These schemes construct pdf by collecting samples values of sensors on the sink once and are not updated afterwards. Hence, correctness of such schemes can not be guaranteed as the 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 [20]). We do not assume the system dynamics to be constant and construct the model dynamically based on the changes in the dynamics. Therefore, our technique can not only track the evolving attribute dynamic but also adapts in case of unexpected or unpredictable changes in the dynamics.

The existing composite approaches require the signal and its statistical properties to be constant, require location information, are partially centralized or use instantaneous values instead of models for clustering. The common factor among all the schemes is that they target real-time or immediate data collection. None of existing works exploits the latency-tolerance of many applications, thus, loose efficiency potentials. 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.

6. CONCLUSION AND FUTURE WORK

We have developed ASTC, a fully distributed spatio temporal data compression technique for accurate sensor data collection in WSN. ASTC dynamically adapts to approximate the monitored attribute both in space and time. 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 attribute. Simple model batches combined with the first exploitation of data collection delay tolerated by the application is used to maximize compression ratio and significantly reduce the message cost.

In our simulations, we were able to reconstruct the signal from spatio-temporally compressed data on the sink with granularity of a single node, mean error less than 1% and data compression to 95% of the raw data samples. In our future work, we aim to (a) tolerate node crashes and spurious data and (b) extend to multi-variate compression.

7. REFERENCES

[1] A. A. Abbasi and M. Younis. A survey on clustering algrithms for WSN. *Comp. Comm.*, 30(14-15), 2007.

[2] J. Abello et al. On maximum clique problems in very large graphs. *External Memory Algorithms*, American Mathematical Society 1999.

[3] A. Ali et al. MPM: Map based predictive monitoring for sensor networks. In *AUTONOMICS*, 2009.

[4] D. Chu et al. Approximate data collection in sensor networks using probabilistic models. In *ICDE*, 2006.

[5] I. A. Dario et al. Underwater acoustic sensor networks: Research challenges. *Ad Hoc Netw.*, 3(3), 2005.

[6] A. Deshpande et al. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.

[7] Faizulkhakov, Ya. R. Time synchronization methods for wireless sensor networks: A survey. *Program. Comput. Softw.*, 33(4), 2004.

[8] B. Gedik et al. ASAP: An adaptive sampling approach to data collection in sensor networks. *IEEE TPDS*, 18(12), 2007.

[9] J. Gehrke and S. Madden. Query processing in sensor networks. *IEEE Pervasive Computing*, 3(1), 2004.

[10] A. Jindal and K. Psounis. Modeling spatially correlated data in sensor networks. *ACM TOSN*, 2(4), 2006.

[11] S. Madden. Intel lab data, 2003, http://db.cscail.mit.edu/labdata/labdata.html

[12] S. R. Madden et al. TinyDB: an acquisitional query processing system for sensor networks. *ACM TODS*, 30(1), 2005.

[13] M. Mahmudimanesh et al. Reordering for better compressibility: Efficient spatial sampling in wireless sensor networks. In *SUTC*, 2010.

[14] R. A. F. Mini et al. Prediction-based energy map for wireless sensor networks. *Ad Hoc Netw.*, 3(2), 2005.

[15] F. K. Shaikh et al. Generic information transport for wireless sensor networks. In *SUTC*, 2010.

[16] I. Solis and K. Obraczka. Isolines: efficient spatio-temporal data aggregation in senor networks. *Wire. Comm. and Mob. Comp.*, 9(3), 2009.

[17] Gilman Tolle et al. Macroscope in the Redwoods, In *SenSys*, 2005.

[18] D. Tulone and S. Madden. PAQ: time series forecasting for approximate query answering in sensor networks. In *EWSN*, 2006.

[19] M. C. Vuran et al. Spatio-temporal correlation: Theory and applications for wireless sensor networks. *Comp. Net. Journal*, 45(3), 2004.

[20] L. Wang and A. Deshpande. Predictive modeling based data collection in WSN. In *EWSN*, 2008.

[21] G. Werner-Allen et al. Monitoring volcanic eruptions with a wireless sensor networks. In *EWSN*, 2005.

[22] S. Yoon and C. Shahabi. The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM TOSN*, 3(1), 2007.

[23] C. Zhang et al. Model-aided data collecting for wireless sensor networks. In *HPCC*, 2006.

[24] J. Zhao et al. Residual energy scan for monitoring sensor networks. In *WCNC*, 2002.

[25] J. Min and C. Chung EDGES: Efficient data gathering in sensor networks using temporal and spatial correlations. *JSS*, 83(2), 2010.

[26] S. J. Baek et al. Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation. *IEEE JSAC*, 22(6), 2004.

[27] C. Liu et al. An energy efficient data collection framework for sensor networks by exploiting spatiotemporal correlation. *IEEE TPDS*, 18(7), 2007.

[28] H. Gupta et al. Efficient gathering of correlated data in sensor networks. In *MobiHoc*, 2005.

[29] H. Jiang et al. Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE TPDS*, 22(6), 2011.

[30] N. D. Pham et al. SCCS: Spatiotemporal clustering and compressing schemes for efficient data collection applications in WSNs. *Int. J. Comm. Sys.*, 23(11), 2010.