Multilevel Predictions for the Aggregation of Data in Global Sensor Networks

Andreas Benzing, Boris Koldehofe, Marco Völz, Kurt Rothermel
Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Stuttgart, Germany
{firstname.lastname}@ipvs.uni-stuttgart.de

***Abstract*—Real-time diagnostic simulations are one challenging application domain that is expected to introduce high requirements to global sensor applications. Besides having hard constraints on latency bounds at which data needs to be processed, such simulation applications will impose high requirements with respect to available bandwidth. Predictors, originally introduced in the domain of wireless sensor networks for energy saving, are one appealing solution to provide real-time estimates and at the same time significantly reduce the data rates. While in the setting of wireless sensor networks many prediction models have been analyzed, their behavior and use is unclear when applied to distributed data streams where aggregation results are typically processed over multilevel hierarchies.**

In the context of weather simulations, we propose a distributed R-Tree-based aggregation algorithm that allows for efficient reuse of aggregate queries. In the setting of real temperature readings taken from weather stations during one month, we study the trade-off between updates of the prediction model and the precision of the predicted values. Our evaluations indicate that even in situations where complex prediction models are expected to perform best, simple prediction models give higher benefits with respect to saving bandwidth while providing similar data accuracy.

***Keywords*-Global Sensor Networks; Distributed Stream Processing; Predictors**

I. Introduction

With the increasing deployment of local sensor networks it has become possible to use their data for a wide range of applications on a global scale as part of a *global sensor network* (GSN). Especially the simulation of natural habitats and ecosystems has gained attention over recent years to better understand global environmental changes. As the popularity of such applications and the number of users increases, it becomes important to provide middleware solutions and expressive continuous query paradigms for efficient data retrieval. This includes optimized bandwidth usage and the need to avoid the overload of data sources. Several *Distributed Stream Processing Systems* (DSPS) [1], [2], [3], [4] have been proposed that process data streams inside the network. In-network processing like filtering and aggregation reduces traffic in the system and therefore improves its efficiency.

While those approaches are an important step in contributing to a scalable deployment for GSN applications, major open research challenges are the provision of timely data access as well as the handling of large bursts of data. Such bursts need to be addressed particularly in real-time simulations where it is important to directly investigate the impact of environmental disasters on habitats and ecosystems to take countermeasures. However, traditional approaches proposed to enable real-time processing such as network reservation protocols are typically not available at the global scale. Moreover, the occurrence of unforeseen data bursts may still cause overload on the reserved capacity on the underlying communication links.

In this paper we propose and evaluate an alternative approach to real-time communication based on prediction models. *Predictors* were originally proposed in the domain of Wireless Sensor Networks (WSN) [5] to prolong the lifespan of individual sensors that are only scarcely equipped with resources. To achieve this, the amount of data reported to the WSN gateway is reduced by using predicted values instead of actual readings at the expense of slightly less accurate values. Additionally, an estimated measurement can be obtained instantly without introducing any communication delay. In the WSN domain various prediction models [5], [6], [7], [8], [9] have been proposed, widely differing in their employed algorithms and complexity.

However, in a global system for sensor information, single hop communication between sensors and applications prevents the reuse of data for multiple recipients. Therefore we present a new approach to multilevel data sensor data aggregation which allows efficient reuse of results. In this setting of multilevel hierarchies, the different accuracy in predicted values and applicability of predictors remain unclear. Even for sensor networks there exists no classification, which prediction model is best-suited for a certain type of data (e. g. temperature readings or wind speed). This question becomes even more difficult with in-network processing in a global multilevel context.

We demonstrate for a distributed R-Tree-based aggregation algorithm optimized for data reuse, how multilevel predictions can be integrated into distributed stream processing systems. The performance is evaluated on the basis of temperature sensor readings of the National Oceanic and Atmospheric Association (NOAA) [10] and the German Weather Service (DWD) [11] taken during a whole month. In addition, the Intel Lab Data set [12] was used to compare the results to existing approaches. Complex prediction models were expected to perform best since they capture the daily recurring features of sensor data. However, our results show that computationally simple models provide the best performance with respect to saving bandwidth.

In the remainder of the paper, Section II discusses related work. Section III presents our approach to a distributed R-Tree and the multilevel predictions along with the underlying system model. Our evaluations results are shown in Section IV. Finally, Section V concludes the paper.

II. Related Work

The Global Sensor Networks middleware (GSN) [13] proposes virtual sensors. By specifying local physical sensors or other remote virtual sensors as input and supplying algorithms for data processing, the result can be published as new virtual sensor to other users. However, GSN does not support direct queries to sensors without defining a virtual sensor for the set of sensors of interest and lacks the possibility to automatically find sensors of interest. Publish/subscribe [14], [15], [16] has emerged as a generic powerful many-to-many communication paradigm. It is applied for efficient decoupling of data sources (sensor nodes in a GSN) and subscribers to data streams mostly in event processing systems. More recent approaches also consider the delay and bandwidth constraints of subscribers [17], [18]. Applied in a GSN however, publish/subscribe lacks the possibility to perform in-network processing required to greatly reduce the high data rates.

To exploit the advantages of in-network processing of data, several distributed stream processing systems (DSPS) like Borealis [1] have been proposed. However, its query interface requires manual stream modeling and distribution of operators to available nodes. Other approaches like IrisNet [2] and HiFi [3] focus on the special properties of sensor data and provide data filtering and preprocessing close to the sensors. Hourglass [4] provides robust so called circuits on intermittent connections between nodes. Although they all support information source lookup and automatic routing of data streams, they lack the reuse of streams and intuitive range querying as required for global sensor data. SBON [19] introduces a layer between the DSPS and the physical network to optimize placement for network usage. The approach was improved towards optimal mapping of operators with respect to the underlying network [20]. Although similar operators might be placed on the same node, explicit reuse is not supported by this approach.

Contrary to these DSPS, our approach is to exploit the similarities in data streams by actively avoiding redundant data transmission. To reuse data streams we extended the R-Tree [21] to be organized in a distributed fashion that can be used to locate data as well as to route data streams. Previous approaches for distributed R-Trees like NR-Tree [22], Peer-Tree [23] or TPR*-Tree [24] only covered query routing and data source lookup but not the routing of the actual data streams. The DR-Tree [25] does handle routing in publish/subscribe to minimize false positives. However, it does not optimize reuse of data processing.

The optimization for reuse of data and reduction of bandwidth usually results in additional hops in the network communication and therefore increased delay. To provide real-time estimated results, we use the concept of predictors as introduced in the context of WSN [5]. First successful approaches focused on compression of multiple measurements before transmitting them as a whole. Between the reception of measurement blocks, the WSN gateway supplies predicted values to the user [6]. With the increasing computational power, the predictions were calculated directly on the sensor nodes using simple models. By simultaneously calculating them on the gateways only differing measurements actually had to be transferred from sensor nodes to the gateway, thereby reducing communication and hence energy consumption [26], [27], [7], [8]. More complex SARIMA models are used in PRESTO [9]. However, these models are too complex for sensor nodes and have to be computed on the WSN gateway. Further projects tried adaptations of control theory and Kalman filters for prediction [28]. Although these filters are versatile, they require manual modeling for each sensor and its individual characteristic. This is unfeasible for a global sensor network with a huge number of sensors where further challenges arise when using predictors [29]. However, besides the most basic prediction in the form of caching [30], the Kalman filter was the only approach that has yet been applied in DSPS [31].

III. Prediction-Based Aggregation of Data

In the following, we introduce our approach to support real-time estimates on the aggregated sensor data of a geographical region. We use this setting to evaluate prediction models applied over multilevel hierarchies. In Section III-A, we first introduce the basic system model and describe the participating nodes. Section III-B shows how aggregation streams can be established over an R-Tree-based overlay to enable i) efficient reuse of existing streams to reduce network load on the sensor network gateways and ii) fair distribution of this load according to the interest of peers. Finally, Section III-C shows how to integrate different prediction techniques into our system to save additional bandwidth and provide real-time estimations.

A. System Model

A GSN is formed by a set of *gateways* and *peers*. A gateway is associated with a local sensor network and can answer queries on the data available in that sensor network. In case of a failure of a gateway its associated WSN could not be queried at all. We therefore assume reliably available gateways and use them to build up a basic indexing structure for data source lookup. Peers are all other nodes participating in the GSN with a query which they can perform by either contacting a gateway or another peer of the GSN. Every peer that queries the GSN is then automatically integrated in the system and contributes to the maintenance of the GSN. This way, the task of data dissemination can be distributed among peers in a fair fashion and the reuse of data streams becomes possible.

We approximate the real world using a plane since coordinates can then be easily converted to latitude and longitude values. As already described it is unfeasible to explicitly address every single sensor to query in a global sensor network with a huge number of sensors. Therefore, we chose a query model which focuses on the specification of a region of interest rather than individual sensors. The region is used to find nodes which have the desired information available and finally resolved to single sensor nodes.

Applications specify the geographical area of interest as a rectangular shape using the lower left and upper right corners as identification marks. Besides these coordinates, a query includes the requested type of sensor data and an aggregation function which should be used to combine the measurements of single sensors. In particular, we consider in our evaluations only the average aggregate function although our architecture can easily be extended to support arbitrary aggregating operators. It is also possible to only gather the information and provide raw data without further processing. The result of a query is a continuous data stream of sensor information about the area of interest. The approach presented in this paper implicitly defines the routing of the data stream according to the reuse relationship between multiple queries.

Predictors provide expected future measurement values based on the as yet observed data. They are established on every data link in the system to provide real-time estimated values. With the forwarding of reused results between peers, a multilevel prediction hierarchy emerges. To exploit predictors not only for real-time purposes but also for data reduction, a maximum tolerated deviation can be additionally specified in the query. Both sender and receiver agree on the same predictor setup and synchronize their operation. When the predictor on the sender produces an estimation that differs less from the actual measurement than the given threshold no data needs to be transferred to the receiver. The detailed description of the operation of predictors is given in Section III-C.

B. Distributed Data Aggregation

Our query model allows for queries to specify a geographical area rather than explicitly stating single data sources. Therefore, we need an efficient index structure for source discovery. If all queries that are already running are integrated into this index, it can be exploited to allow the reuse of the associated data streams by identifying intersections between new and already existing queries. To achieve this, all peers and gateways are integrated into an R-Tree which is commonly used for indexing of points and rectangular shapes in database systems. On the way from a leaf to the tree's root, each parent vertex manages a rectangle that is a representation of all its children merged together, called *minimum bounding rectangle* (MBR). In other words, each node manages a rectangular shape which is becoming smaller from the root to the leaves. The rectangle managed by a node *contains* all of its children; this is called the *containment relationship* of an R-Tree. In our scenario, the leaves in the R-Tree consist of MBRs of single sensor networks represented by their respective gateway which then manages access to the single sensors. The tree structure is distributed into the network by mapping the logical vertices

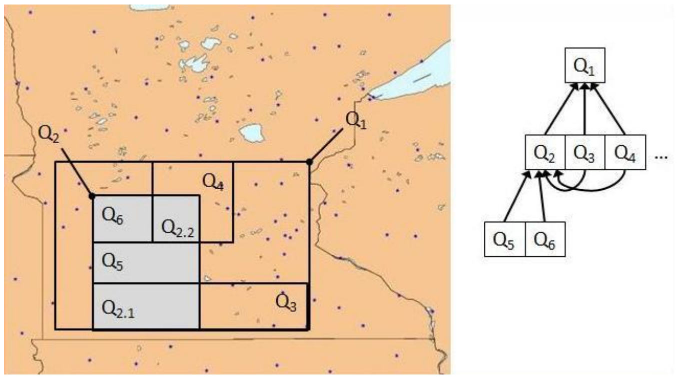


Fig. 1. The connection between the geographical alignment of queries and their positioning in the R-Tree

of the tree to available physical nodes (namely gateways and peers) forming a *distributed R-tree index*. To exploit the distributed R-Tree index for reuse of data streams the construction, insertion, and routing mechanisms were modified as described in the following.

Index Construction: Initially, the R-Tree is constructed using the gateways of the available sensor networks which are specified by their MBRs. As a result, certain gateways will not only appear as leaves inside the tree but additionally serve as intermediate vertices. However, the intermediate vertices that are represented by gateways are only used for sensor data lookup and not the actual routing. As queries will be integrated during runtime, the querying peers will step by step replace them. The root of the tree represents an MBR of all known sensor networks and therefore covers the complete world as represented in our system. To avoid overloading the node responsible for the root vertex, it can be replicated to multiple nodes.

Inserting a New Query: A new peer can send a query to any node already participating in the R-Tree. On reception of a new query, the receiver compares the query to the MBR it is currently managing. If there is no match, i.e. the query is not contained inside the MBR, the receiver passes the query to the parent in the R-Tree. Eventually, the query will reach a node whose MBR completely contains the query. Note that the root vertex comprises the entire world. From this point on, the query is passed down the tree by forwarding the query to the child which potentially still contains the requested area. When a node is reached where the containment relationship to the new query is no longer given, the node initiates the insertion of the new peer into the distributed R-Tree index. The peer will be assigned an MBR which matches exactly its query rectangle. If there is a gateway node managing the same MBR in the tree, the new peer replaces the respective gateway. Figure 1 shows an example matching between the areas of different queries on the left and the corresponding tree structure on the right.

Routing for Efficient Reuse of Sensor Data: The structure to route sensor data can easily be constructed from this R-Tree. Figure 1 shows, how data routing is handled between intersecting queries: The peers share the matching areas with another and create horizontal routes. Apart from these special cases, the routing structure resembles the R-Tree created for indexing. Using this routing scheme, the number of outgoing links is inherently limited: Links are built by descending the tree, thus a peer receiving a certain data stream is discovered before the actual gateway. Instead of creating a second route to the source, the already receiving peer has to share his data stream by forwarding the appropriate sensor readings. Gateways provide data of the sensor network they are connected to on the lowest level of the tree. Therefore, any gateways serving as intermediate nodes are left out while creating the routes, sending the data directly to the next peer.

Aggregate Calculation: In a system of global scale, a new query will receive mostly partial results from other queries instead of getting data directly from a gateway. Duplicate sensitive aggregations like the average function require additional computation when they are computed in multiple steps. As each of the reused results is an aggregation of values over a differently sized area, it is necessary to weigh the incoming values accordingly (cf. Figure 1: Query Q_2 receives partial results $Q_{2.1}$ and $Q_{2.2}$ from queries Q_3 and Q_4 respectively). The individual weights w_i for each of the k incoming results are calculated according to the area of the reused result. In our approach the result r^t for a query at the time t is then computed by weighting the values of n reused partial results according to their area A_i with respect to the total queried area A_q:

$$r^t = \sum_{i=1}^{k} w_i d_i^t \text{ , with } w_i = \frac{A_i}{A_q}$$

denoting the weight for the i^{th} incoming data stream and d_i^t denoting the value of the i^{th} stream at the time t. This assignment of weights is adequate for temperature data with a reasonably equal density of sensors in all areas. However, other adapted average calculations like median estimation are likely to be better suited by weighting according to the number of sensors instead of area size. Different alternatives have already been investigated for sensor networks (e.g. [32]) and their integration is subject to future work.

Fair Load Distribution: The additional load of sharing data streams is distributed in a fair manner across all peers. Every node only has to provide the data, it has actually queried. Therefore, the load induced by this distribution is proportional to the amount of data the respective peer acquires from the system. Furthermore, the number of outgoing links for each peer is limited to a system wide maximum. Although the system could also work with a single outgoing link for each peer, a higher number of links reduces the average number of hops required.

With our approach we satisfied the two main goals: first, the reuse of data streams is inherent in the R-Tree structure by the containment property which is maintained between a parent and its children. By mapping the vertices that correspond to queries of the peers that issued them, those peers can provide their results to their fathers and siblings. Gateways are thereby alleviated of distributing their data to each and every consumer by themselves. Our insertion procedure can furthermore ensure that each gateway only has to maintain a single outgoing data link by shifting additional load to the peers in a fair manner.

C. Multilevel Prediction

While the aggregation structure introduced in Section III-B provides efficient sensor lookup and reuse of data streams, it does not optimize for network latency. To overcome this and provide real-time estimated results, predictors are integrated into our stream processing system.

Predictor Models: A predictor uses a certain *predictor model* to generate an estimated value of the next expected measurement. This is done by combining a set of *model parameters* u_i with a set of recently received sensor measurements d^t, where t denotes the time the measurement was received. The number of model parameters depends on the *order* of the predictor. In our system we focused on a seasonal autoregressive integrated moving average (SARIMA) model [33], [9] as a representative for more complex approaches. The SARIMA model originates from the time series analysis domain where long term trend analysis is more important than fast generation of model parameters. The least-mean-square-algorithm (LMS) [7] provides a simple and fast alternative with low memory and computational overhead.

The LMS algorithm uses a simple approach to update the model parameters in each time step. For each parameter its new value u_i' is calculated based only on the absolute error of the last prediction $e^t = d^t - p^t$ and its corresponding last measurement x^{t-n+i}. The adaptation speed of the model is controlled by μ, which depends on the variability of the input data.

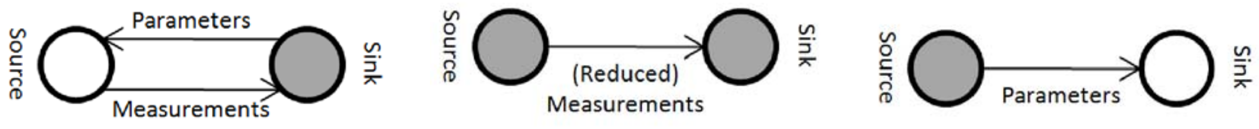
$$u_i' = u_i + \mu e^t d^{t-n+i}$$

In this case, the parameters and measurements can then be evaluated to a prediction of the expected future measurement as a weighted sum:

$$p^{t+1} = \sum_{i=1}^{n} u_i d^{t-n+i}$$

In contrast, a SARIMA model is specified by two statistical processes one of which covers the seasonal part while the other models the general trend. Both processes consist again of an auto-regressive and a moving average part. Furthermore, SARIMA models differ in the order of differencing that is used to estimate the current trend in the observed data for the general and seasonal part. The order of a SARIMA model is given by $(p, d, q) \times (P, D, Q)_S$, where p and d denote the orders of the auto-regressive and moving average processes, respectively. d is the order of differencing used, the upper-case letters stand for the according values of the seasonal components. The model parameters can be estimated using several strategies [33]. A detailed description of this estimation is, however, out of the scope of this paper.

Note that parameter estimation takes a considerable amount of computational effort and is therefore only done during



(a) Sink-based operation (b) Synchronous operation (c) Source-based operation

Fig. 2. Three operation modes: Model parameters are generated on the gray node

an initial training phase. Approaches for periodically refining the model parameters to capture qualitative changes in the observed measurements have also been proposed for the PRESTO architecture [9]. It was also observed that the optimal order of the SARIMA model for temperature data is $(0, 1, 1) \times (0, 1, 1)_S$.

The computation of a prediction for the SARIMA model takes into account historical measurements and their respective prediction errors. S thereby denotes the length of the seasonal period of the observed phenomena, for example, one day in the context of temperature measurements. With the model order proposed in the PRESTO architecture, predictions are computed as follows:

$$\begin{aligned} p_{t+1} &= x_t + x_{t-S+1} - x_{t-S} \\ &+ u_1 e_t - u_2 e_{t-S+1} + u_1 u_2 e_{t-S} \end{aligned}$$

Operation Modes: To use predictors to reduce the number of measurements transmitted from a data source to a corresponding sink, they agree on the same prediction model along with the according set of model parameters. They also have to agree on the same history of measurements and prediction errors in order to provide the same predictions. This can be done during an initial setup phase where measurements are directly sent to the sink until a certain history of data is available at both, the source and the sink.

During normal operation, in each time step, they independently estimate the next measurement. Only if the source detects that the actual measurement deviates more than the user-given threshold from the prediction, it sends and update to the sink. If the prediction was sufficiently precise, it is added to the set of recent sensor measurements and the model parameters remain unchanged.

Although the generation of model parameters is the most expensive part of predictors with respect to the processing time, each peer in GSN is powerful enough to perform it locally. Model parameters can therefore be computed on the source, the sink, or both. This also allows a flexible decision on how much information is exchanged for an update between two correspondent predictors. Possibilities range from single measurements over a set of model parameters to a complete history of data. By using a modular design, all operation modes can be combined with arbitrary prediction models in our system.

The sink-based operation shown in Figure 2(a) requires the transfer of measurements to the sink where predictor model parameters are computed. Model parameters are then transferred back to the data source where the prediction is calculated and compared to the actual measurement. It is well suited for the resource-constrained environment of WSN to unburden the sensor nodes from expensive computations. However, in distributed stream processing the reduction of transmitted data is more important and such constraints are not present. Therefore, we did not focus on the sink-generated approach due to its additional traffic caused by communicating measurements and model parameters.

If the model parameters are generated locally on both ends of the communication link, their transmission can be avoided at cost of increased computational load. The location of the parameter generation and transfer of measurements is shown in Figure 2(b). The major problem is to keep the models in sync, since a single lost update can cause increasing deviation in the two predictions over time. Depending on the order of the predictor and the properties of the measured data, the models can either be synchronized using single updates or larger blocks of measurements. With data that can be predicted with high accuracy over long time periods, larger blocks of measurements can be profitable over single measurements. A trade-off between the reduction of messages and transmitted information has to be found that allows the most effective operation of the predictor.

Another possible mode of operation is source-based, as can be seen in Figure 2(c). In this case, model parameters are only computed on the data source, where all actual measurements are available. This approach requires the transmission of the whole parameter set to the sink node. The amount of data that has to be transmitted is therefore higher than for the synchronous approach, but if an update is lost, the parameters are completely replaced with the next update. However, since the measurement histories also have to be in sync, updates are extremely costly in terms of bandwidth. We expect this approach to perform best for more complex prediction models that can encode more information in the set of model parameters.

Multi-Level Operation: In our multilevel aggregation setup, predictors are employed on every data link between peers. Since measurements may arrive late, each peer sends its own estimation of the actual sensor measurement to its successor and provides real-time results. As a result, the theoretically possible deviation increases with each predictor hop in the system. However, in each time step, the predictor cannot deviate more than the user-defined threshold. It takes

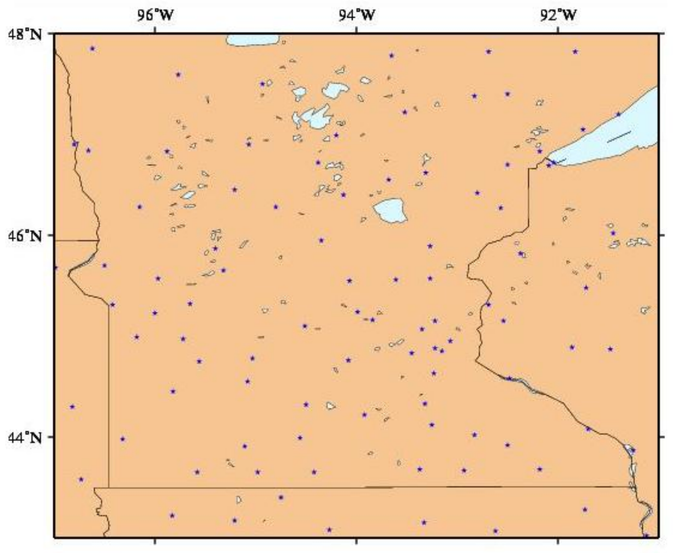


Fig. 3. The sensor stations considered in the area of Minnesota

as many consecutive time steps without update as hops in the system to reach this theoretical bound. Additionally, the prediction error has to be only marginally smaller than the received data in each of these time steps. Due to the variance in measurements, this threshold is never met over a longer period in practical use. Since the predictors operate on aggregated data, the prediction model can also benefit from the smoothed input data. Especially by using the average operator, prediction errors with opposite sign mutually compensate and result in smaller prediction errors on the outgoing data link.

IV. Evaluation

The system described in Section III has been evaluated with multiple real-world datasets. More specifically, we selected a total of 184 sensor stations freely available from the NOAA [10] in the area of 97°W to 91°W and 43°N to 48°N (NOAA dataset). This is a rough approximation of the state of Minnesota, for the concrete placement of sensor stations, see Figure 3. The sensor readings were collected over a period of one month, December 2008, on an hourly basis. Two further sets of data include all stations from the German Weather Service (DWD) [11] during April 2008. One set includes data from 489 stations on an hourly basis (DWDsy dataset), the other consists of measurements from 184 (DWDmu dataset) stations that gather data every 10 minutes. To provide a comparison to the originally proposed predictors, the set of Intel Lab Data [12] measurements (mitlab dataset) was also used for evaluation. In general, evaluations have shown very similar results for all datasets and therefore only the DWDmu dataset is shown in most figures for clarity.

Following our goals, the experiments intend to investigate the following issues:

- What is the impact of the smoothing effect of building an aggregated average over sensor data?
- How efficient can predictors reduce traffic in a multilevel environment?
- What is the user-experienced deviation from the actually measured values?
- Which prediction model performs best in terms of saving network bandwidth with respect to the used type of sensor data?

To clearly stress the properties of predictors in this scenario, we set up an idealized aggregation tree for a single query over all available sensors. Data is transmitted to a single node using predictors. The average over all sensor predictions is then calculated and forwarded using a second predictor. The overall prediction error was determined for each of the links originating directly from the sensors as well as for the overall average. Although this setup only involves two successive prediction steps, our results clearly show the applicability of different predictor models.

The two most common predictor models, LMS and SARIMA, have been chosen for evaluation. To investigate whether the achieved performance really originates from the model and not the general prediction approach, a simple model was also evaluated. This simple model always provides the last received measurement as prediction without considering any history or trend of the data.

Evaluations have shown that none of the approaches provides a sufficiently high prediction horizon to compensate for additional transmission of complete parameter sets or blocks of measurements. Therefore, only the synchronous operation mode, which provides the intended benefit of reduced traffic, is shown in the results. In particular, only single measurements are transmitted when the desired deviation is exceeded. Transmission of multiple values for updates has shown to cancel out the data reduction properties of a predictor.

To show the ability of different predictor models to reduce traffic, their hit ratio has been investigated. A prediction is sufficiently precise if it differs less than the user given threshold from the measurements. The hit ratio is defined as the number of sufficiently precise predictions divided by the total number of measurements over the whole experiment. As measurements are only transferred when the deviation is exceeded, this corresponds directly to the reduction in traffic. In other words, the hit ratio quantifies the portion of traffic that is saved using a predictor.

A. *Smoothing of Data by Aggregation*

Since high and sudden deviation usually cannot be covered by predictors, high noise in input data usually leads to poor predictor accuracy. As already described in Section III-C, smoothed input data is therefore expected to lead to better prediction results. Two values characteristic for the noise in input data were considered: variance and mean input power. The mean input power is also used to estimate the parameter μ for the LMS predictor which controls its adaptation speed.

We evaluated the smoothing effect of averaging temperature measurements on a large scale to show the resulting significant

Dataset	Avg. Var.	Var. of Agg.
DWD sy	18.4	13.8
DWD mu	17.8	13.9
mitlab	9.5	1.7
NOAA	50.2	33.6

TABLE I
REDUCED VARIATION DUE TO AGGREGATION

Dataset	Avg. MIP	MIP of Agg.
DWD sy	77.3	68.7
DWD mu	74.6	66.7
mitlab	477.6	455.9
NOAA	179.3	142.7

TABLE II
REDUCED MEAN INPUT POWER (MIP) DUE TO AGGREGATION

reduction in variation and mean input power. Tables I and II show the results of our evaluation.

The data was averaged over the complete available data sets to show the maximum possible impact. The reduction of both, variation and mean input power, is clearly visible for all datasets. In fact, both, mean input power and variance, are smaller for the aggregated data than for any single sensor. Of course, this smoothing effect is smaller when fewer sensors are considered. However, on a system of global scale, we believe that queries will usually cover many sensors to improve reliability of results and reduce the effect of outlying measurements. The following sections discuss the impact on the different predictor models.

B. Reduced Amount of Data

Results for different predictor models are shown in Figure 4. The graph shows the average hit ratio over single hop predictors for all stations.

Since the fast adaptation speed of predictor model of lower order is best suited for highly variable sensor data, the LMS model considered is of order 1. Higher order models can be used to fit the underlying trend in measurements but still do not achieve the same hit ratio. As a higher order directly results in increased memory usage and higher computational effort for parameter estimation, models of order 1 provide the best overall characteristics. While the LMS model performs best considering the overall hit ratio, the good performance of the simple approach indicates that the measured values only change slowly over time.

The SARIMA model performs very well for a small threshold in maximum deviation. With increasing threshold, the model gets too few updates to provide sufficiently precise predictions. This behavior can be improved by using other operation modes and by transmitting a block of measurement updates instead of single values. However, outdoor temperature data does not have the seasonal nature required for efficient operation of SARIMA predictors.

Figure 5 shows the comparison of the SARIMA and LMS models to the simple model for a predictor operating on

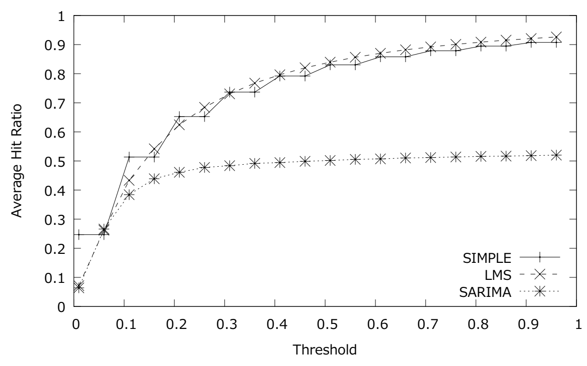


Fig. 4. Average hit ratio of different predictor models for single hop using the DWDmu dataset

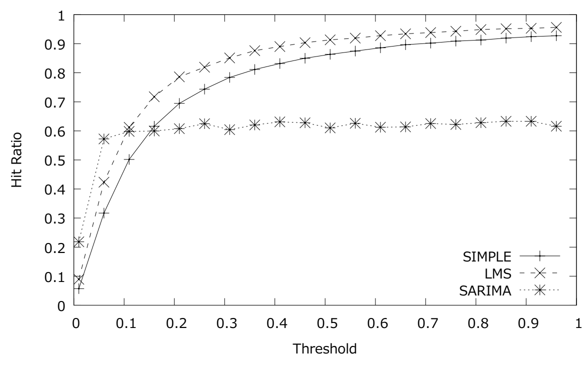


Fig. 5. Hit ratio of a predictor using the aggregated DWDmu dataset

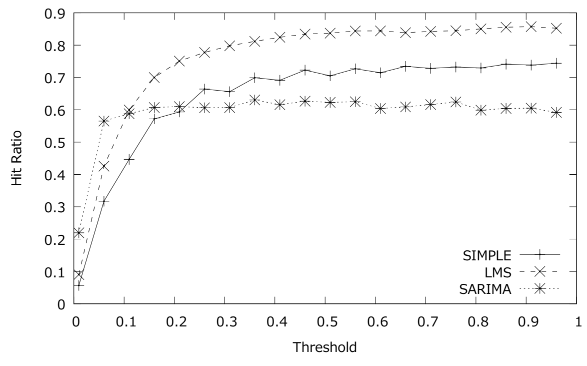


Fig. 6. Hit ratio of different predictor models for a two hop predictor on the DWDmu dataset

the data aggregated over all stations of the DWDmu dataset. All predictor models significantly benefit from the smoothed input values and provide a higher hit ratio. As expected, the SARIMA model gains most from this effect especially for low thresholds. The LMS model clearly outperforms the simple approach in this setting because the current trend in measured data can be taken into account.

The hit ratio of the considered prediction models in the two hop setup is shown in Figure 6. Surprisingly, the SARIMA model performs equally well as in single hop operation while

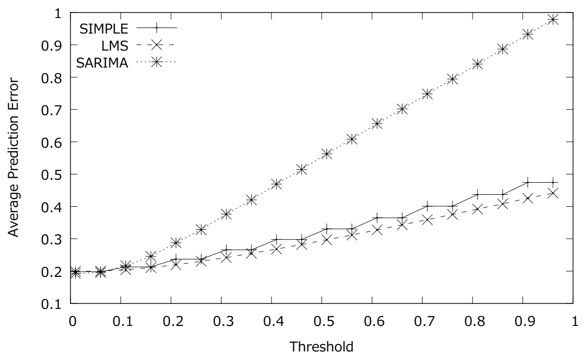


Fig. 7. Average prediction error of different predictor models for single hop using the DWDmu dataset

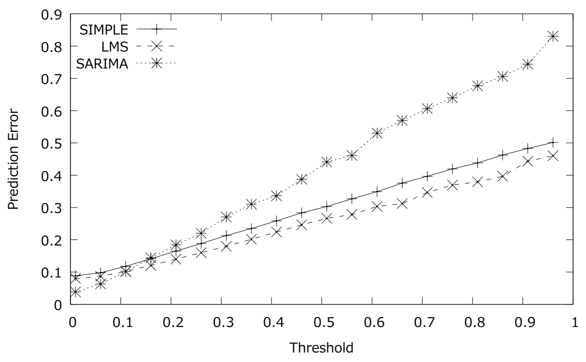


Fig. 8. Prediction error of different predictor models using the aggregated DWDmu dataset

the other approaches show a decrease in their hit ratio. Especially the simple model cannot provide the hit ratio as in single hop operation. The reason for this decrease is that it cannot cover any kind of trend in the observed data and therefore lags behind the actual measurements. Opposing to this, the SARIMA model relies on the daily trends in the observed data which provide more information about the data than a simple trend analysis.

C. Errors in Predictions

While the maximum deviation is given by the user-defined threshold, the user-experienced error presents the more important information. The average absolute deviation between the data delivered by the predictor and the actually measured data is shown in Figure 7 for the non aggregated data of each station in the dataset. Since the data provided by the stations is given at a resolution of 0.1°C the error does not decrease further even for a lower threshold.

The LMS and simple approaches provide a prediction error below the actual threshold because the changes in the trend are very low. In contrast, the SARIMA model relies on the information from the last season, e.g. the day before. It thereby generates several significantly higher prediction errors as outdoor temperature greatly varies between days.

Figure 8 shows the results for the aggregated dataset. Again,

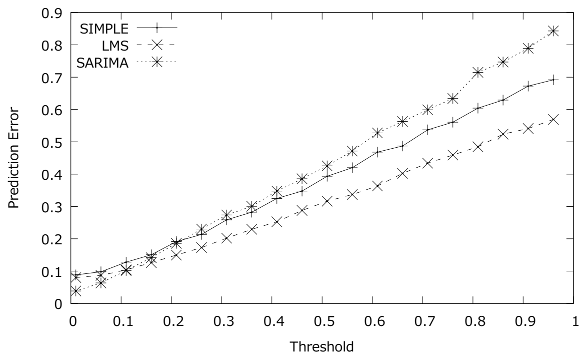


Fig. 9. Prediction error of different predictor models for a two hop predictor on the DWDmu dataset

the improved performance of predictors can also be seen in terms of reduced prediction error. As in the case of improved hit ratio, the gain depends on the prediction model. The SARIMA model can again benefit most when using low error thresholds and provide better prediction accuracy.

With two hop operation of predictors the prediction error perceived by the user increases, as shown in Figure 9. Again, the result in prediction error is very similar to that of the hit ratio.

Note that the error increases proportionally with the allowed deviation, indicating that the variation in temperature is still significantly higher. This also means that the information provided to the user still contains the queried information despite the high reduction in data transmitted.

Since an update is always sent when the maximum specified deviation is exceeded, the maximum prediction error will be eventually constrained when the update is received by the user. However, the average prediction error of the LMS and simple models is smaller than the provided maximum deviation for a sufficiently high threshold. This indicates that the predictions can be used as real-time estimate of the actual sensor data.

D. Applicability of Predictors for Different Sensor Types

The evaluations have shown that the LMS model is best suited as predictor for temperature data contained in the DWDmu dataset. Figures 10, 11, and 12 show the data reduction results for other types of sensor data for a two hop prediction setup. Instead of temperature the measurements of relative humidity, wind speed and air pressure of the same dataset were used as input.

While the seasonal nature of humidity variation is best fit by the SARIMA model, the highly dynamic non seasonal wind speed can still be fit by the LMS and simple models. Most of the predictions can, however, also be provided by a very simple approach that assumes sensor information to be constant. Other sensor types like air pressure, where the simple approach does not perform well, provide a better ground for prediction models like LMS and SARIMA. This clearly shows that predictors are not equally suited for different types of sensor information.

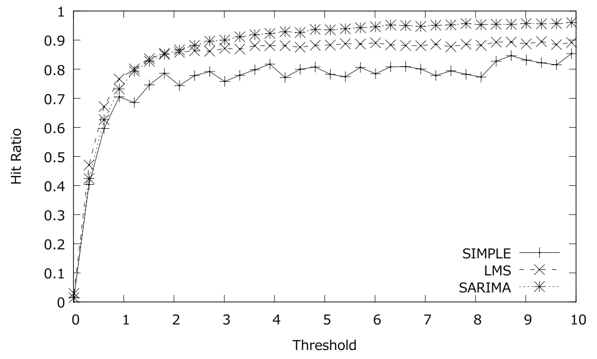


Fig. 10. Hit ratio for different predictor models for a two hop predictor using humidity data from the DWDmu dataset

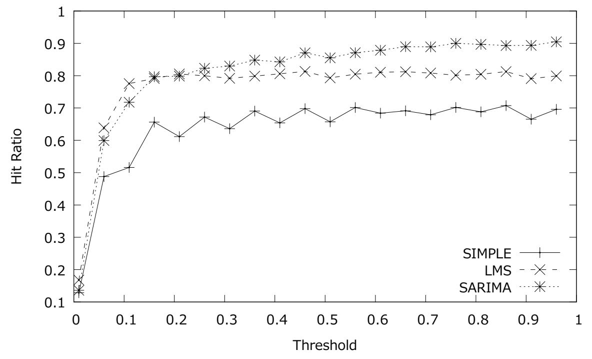


Fig. 12. Hit ratio for different predictor models for a two hop predictor using pressure data from the DWDmu dataset

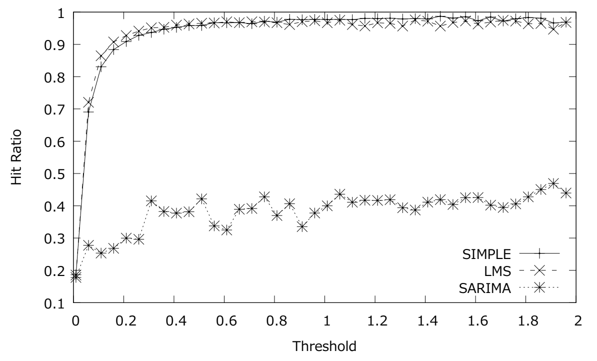


Fig. 11. Hit ratio for different predictor models for a two hop predictor using wind speed data from the DWDmu dataset

V. Conclusions and Future Work

We have shown how a distributed R-Tree approach can be used to provide a scalable solution for global indexing of sensors. By exploiting the containment relation inherent to the R-Tree structure, we achieved fair load distribution and efficient reuse of data streams. The basic concept of predictors can be used to greatly reduce traffic even in a multilevel aggregation scheme. Our evaluations show that predictor models that have been proposed for WSN in the literature also perform well in our multi-hop scenario. Especially the LMS predictor performs well not only for the original mitlab temperature measurements but also for other types of sensors and outdoor sensors. As in WSN, a low order of the LMS predictor provides the best results and provides a low computational overhead and requires little memory. The performance of predictors is even better for smoothed input values which significantly improve the hit ratio and prediction error for all predictor models.

Although these results provide very powerful mechanisms for the handling of global sensor data, further research questions remain. First of all, the system has to cope with different quality of data requirements of users. When serving data at different precision levels the identification and reuse of matching data streams becomes far more complex. If the precision is determined by the system, the trade-off between user experienced precision and network load has to be investigated. Finally, the question remains whether the precision of data delivered to the user can be adapted on a finer granularity by pro-actively sending updates before the maximum deviation is exceeded.

Acknowledgments: The author A.B. would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

[1] D. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina *et al.*, "The design of the borealis stream processing engine," in *CIDR '05: Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research*, Asilomar, California, January 2005.

[2] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, "Irisnet: An architecture for a worldwide sensor web," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 22–33, 2003.

[3] M. Franklin, S. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. Wu, O. Cooper, A. Edakkunni, and W. Hong, "Design considerations for high fan-in systems: The hifi approach," in *CIDR '05: Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research*, Asilomar, California, January 2005.

[4] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh, "Hourglass: An infrastructure for connecting sensor networks and applications," Harvard University, Tech. Rep. TR-21-04, 2004.

[5] S. Goel and T. Imielinski, "Prediction-based monitoring in sensor networks: Taking lessons from mpeg," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 5, pp. 82–98, 2001.

[6] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *ICDE '03: Proceedings of the 19th International Conference on Data Engineering*, March 2003, pp. 429–440.

[7] S. Santini and K. Römer, "An adaptive strategy for quality-based data reduction in wireless sensor networks," in *INSS'06: Proceedings of the 3rd International Conference on Networked Sensing Systems*, 2006, pp. 29–36.

[8] D. Tulone and S. Madden, *Wireless Sensor Networks*. Springer Berlin / Heidelberg, 2006, ch. PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks, pp. 21–37.

[9] M. Li, D. Ganesan, and P. Shenoy, "Presto: Feedback-driven data management in sensor networks," in *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 23–23.

[10] NOAA. [Online]. Available: http://data.nssl.noaa.gov/

[11] Deutscher Wetterdienst. [Online]. Available: http://www.dwd.de

[12] Intel Lab Data. [Online]. Available: http://db.lcs.mit.edu/labdata/labdata.html

[13] K. Aberer, M. Hauswirth, and A. Salehi, "The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks," EFPL, Tech. Rep. 006, 2006.

[14] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 332–383, 2001.

[15] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.

[16] J. A. Briones, B. Koldehofe, and K. Rothermel, "Spine : Adaptive publish/subscribe for wireless mesh networks," *Studia Informatika Universalis*, vol. 7, no. 3, pp. 320–353, Oktober 2009.

[17] M. A. Tariq, B. Koldehofe, G. G. Koch, and K. Rothermel, "Providing probabilistic latency bounds for dynamic publish/subscribe systems," in *Proceedings of the 16th ITG/GI Conference on Kommunikation in Verteilten Systemen 2009 (KiVS 2009)*. Kassel, Germany: Springer, Januar 2009.

[18] M. A. Tariq, G. G. Koch, B. Koldehofe, I. Khan, and K. Rothermel, "Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints," in *The Sixteenth International Conference on Parallel Computing (Euro-Par 2010)*. Springer, August 2010.

[19] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-aware operator placement for stream-processing systems," in *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2006, p. 49.

[20] S. Rizou, F. Dürr, and K. Rothermel, "Solving the multi-operator placement problem in large-scale operator networks," in *Proceedings of the 19th International Conference on Computer Communication Networks*. Zurich: IEEE Communications Society, August 2010, Konferenz-Beitrag.

[21] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 1984, pp. 47–57.

[22] B. Liu, W.-C. Lee, and D. L. Lee, "Supporting complex multi-dimensional queries in p2p systems," in *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, June 2005, pp. 155–164.

[23] M. Demirbas and H. Ferhatosmanoglu, "Peer-to-peer spatial queries in sensor networks," in *P2P 2003: Proceedings of the Third International Conference on Peer-to-Peer Computing*, Sept. 2003, pp. 32–39.

[24] Y. Tao, D. Papadias, and J. Sun, "The tpr*-tree: An optimized spatio-temporal access method for predictive queries," in *VLDB '03: Proceedings of the 29th international conference on Very large data bases*. VLDB Endowment, 2003, pp. 790–801.

[25] S. Bianchi, P. Felber, and M. Gradinariu, *Euro-Par 2007 Parallel Processing*. Springer Berlin / Heidelberg, 2007, ch. Content-Based Publish/Subscribe Using Distributed R-Trees, pp. 537–548.

[26] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *VLDB '04: Proceedings of the 30th international conference on Very large data bases*. VLDB Endowment, 2004, pp. 588–599.

[27] P. Desnoyers, D. Ganesan, H. Li, M. Li, and P. Shenoy, "Presto: A predictive storage architecture for sensor networks," in *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*. Berkeley, CA, USA: USENIX Association, 2005, pp. 23–23.

[28] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Tech. Rep., July 2006.

[29] A. Benzing, K. Herrmann, B. Koldehofe, and K. Rothermel, "Identifying the challenges in reducing latency in GSN using predictors," in *WowKiVS 2009: Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen*, ser. Electronic Communications of the EASST, T. Margaria, J. Padberg, and G. Taentzer, Eds., vol. 17. Kassel, Germany: EASST, March 2009.

[30] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2003, pp. 563–574.

[31] A. Jain, E. Y. Chang, and Y.-F. Wang, "Adaptive stream resource management using kalman filters," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2004, pp. 11–22.

[32] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: New aggregation techniques for sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 239–249.

[33] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed., D. J. Balding, N. A. C. Cressie, G. M. Fitzmaurice, I. M. Johnstone, G. Molenberghs, D. W. Scott, A. F. M. Smith, R. S. Tsay, S. Weisberg, V. Barnett, J. S. Hunter, and J. L. Teugels, Eds. John Wiley & Sons, Inc., 2008.