Dissemination of Models over Time-Varying Data

Yongluan Zhou
University of Southern Denmark
zhou@sdu.dk

Zografoula Vagena
Rice University
zv1@rice.edu

Jonas Haustad
University of Southern Denmark
johau05@student.sdu.dk

ABSTRACT

Dissemination of time-varying data is essential in many applications, such as sensor networks, patient monitoring, stock tickers, etc. Often, the raw data have to go through some form of pre-processing, such as cleaning, smoothing, etc, before being disseminated. Such pre-processing often applies mathematical or statistical models to transform the large volumes of raw, point-based data into a much smaller number of piece-wise continuous functions. In such cases, the necessity to distribute data models instead of raw data may arise. Nevertheless, model dissemination has received very little attention so far. In this paper, we attempt to fill this gap and propose a model-agnostic dissemination framework that can handle different models in a uniform manner. The dissemination infrastructure is built on top of a tree-based overlay network, reminiscent to the ones employed in publish/subscribe systems, which are known to scale well to the number of data producers and receivers. To adequately deal with the vast model variation and receivers' very different accuracy requirements on the models, we have developed optimized model routing algorithms, which are intended to minimize data traffic and avoid bottlenecks within the dissemination network. The extensive experimental evaluation over a prototype system that we have built shows that our methods are both effective and robust.

1. INTRODUCTION

In recent years we have witnessed the proliferation of dynamic or time-varying data. Examples of such data include stock market prices, currency exchange rates, patient monitoring data and real-time traffic information. Such time-varying data usually take the form of (multidimensional) streams of numerical values and are characterized by their large volume and the frequent updates on their values.

To facilitate the processing and analysis of time-varying data, mathematical and/or statistical models built from the raw values need to be created and disseminated. Those models serve varying purposes, such as data cleaning (e.g. [22]), data aggregation (e.g. [5]), identification of important characteristics in the data, e.g. central tendency, spread, skewness, seasonality, trend, outliers, heavy hitters etc (e.g. [6]), event summarization (e.g. [17]), data anonymization (e.g. [**?**]) and value prediction (e.g. [7]). At the same time, many emerging applications of dynamic data have a large number of interested parties. The receivers of such data may have very different information and accuracy requirements. For example, a stock broker, whose buy/sell decisions heavily depend on precise and up-to-the-minute quotes, demands the exact stock quotes with very tight accuracy and timeliness requirements, while a financial observer may be interested in a model that enables her to analyze long-term market movements with reasonable degree of accuracy. Meanwhile a regulatory watchdog may only need the stock quotes that have diverged significantly from their previously reported values. Finally, the general public often desires to detect interesting patterns or trends in such data and be notified when those patterns or trends occur. Designing an information dissemination solution that effectively satisfies all the aforementioned requirements is a challenging task.

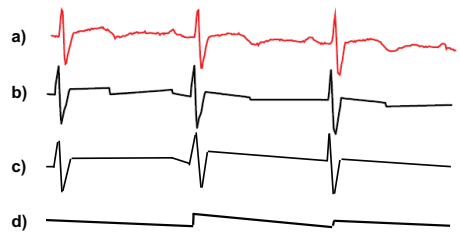


Figure 1: Piece-wise linear approximation (PCA).

EXAMPLE 1. *As an illustrative example of the diverse requirements that may arise, assume that we need to disseminate models derived from the time-series shown in Figure 1a. Also assume that there exist three end users: the first and second users are interested in receiving piece-wise linear approximations (PLAs, see e.g. [15] for a definition) of the time-series, for the purposes of (a) fast exact similarity search (as described, for instance, in [14]) and (b) relevance feedback ([16] respectively). The first user has more strict accuracy requirements than the second user and requires the more exact PLA of Figure 1b. The second user, on the other hand, can receive either of the PLAs shown in Figures 1b and 1c. Finally, the third user is only interested in identifying the trend of the data sequence within a specified error threshold and will be content with the PLA of Figure 1d.*

Proceedings of the VLDB Endowment, Vol. 4, No. 11

State-of-the-art dissemination infrastructures assume that the end users or applications are interested in, and can obtain access to, the raw data. However, as hinted above, in many cases the end users are only interested in mathematical or statistical models derived from the data. In such cases sending the actual data may be impractical (e.g. resource constrained clients may not have the necessary resources to create those models) or even impossible (e.g. when access to the raw data is restricted due to cost, privacy or regulatory restrictions). In such cases, creating the models at the data source (or an intermediate data aggregator) site and disseminating them to the clients may be the only viable option. Distributing those models, as opposed to actual data values, requires a rethink of the dissemination infrastructure as the content of the data is not available/accessible anymore. Nonetheless, to the best of our knowledge, no existing work has so far considered this angle.

In this paper we attempt to fill this gap and devise a framework for the effective dissemination of mathematical or statistical data models. As a first cut of the problem, we attempt to design a model-agnostic dissemination framework that can handle different mathematical or statistical models in a uniform manner. Such a framework is needed to handle arbitrary user defined models. We expect more future work on optimizations for specific models. In summary, we have made the following contribution in this paper:

- Formulate and tackle the problem of model dissemination. To the best of our knowledge this is the first work that describes how to disseminate data models, as opposed to raw values, to a large number of interested parties, with varying accuracy requirements.
- Propose a model-agnostic dissemination framework that can handle different mathematical or statistical models in a uniform manner.
- Devise a centrally-controlled routing algorithm that decreases the volume of transmitted data by exploiting opportunities of model sharing among the data receivers. Subsequently, we describe methods that perform the routing task in a distributed fashion without sacrificing much of the efficiency of the centralized algorithm.
- Empirically evaluate our algorithms on a prototype, model-aware dissemination system that we have built. The results show that our methods can disseminate the necessary models to the interested parties in a resource (in terms of both CPU utilization and network bandwidth) efficient manner.

The rest of this paper is organized as follows: In Section 2 we give some background information and formally state the problem that is investigated in this paper. In Section 3 we describe in detail our proposed solution and we experimentally evaluate the proposed methods in Section 4. Finally, in Section 5 we present some additional related work and we conclude the paper in Section 6.

2. PROBLEM FORMULATION

In this section we formulate the problem that we tackle in this paper and provide necessary background information to understand our solution. For simplicity of presentation we restrict our discussion on a single data source. Nevertheless our solution makes no such assumption and it can be applied to multiple (possibly dependent) data sources in a straightforward manner.

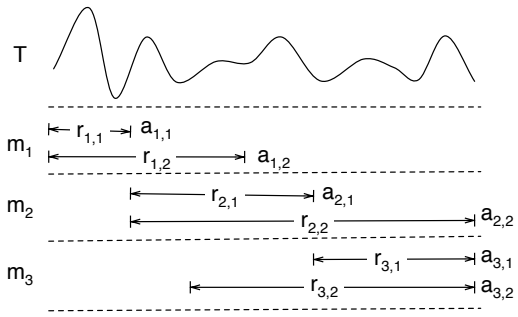


Figure 2: Models.

2.1 Time-Varying Data and Models

Time-varying data from a data source can be represented with an infinite univariate time-series, i.e. a sequence of numerical values measured at successive times of the form $(.., x_i, x_{i+1}, .., x_{i+N}, ...)$. Given a time-series, data receivers may request the creation of statistical or mathematical models over a specific finite snapshot of that time-series, T, defined as $(x_i, x_{i+1}, .., x_{i+N})$, where $-\infty < i, N < +\infty$. We call $T[p:q] = (x_p, x_{p+1} \ldots, x_q)$, where $p \geq i$ and $q \leq i+N$, a *segment* of T.

Models. In this paper, we focus on piece-wise mathematical or statistical models, which divide a time-series snapshot, $(x_i, x_{i+1}, .., x_{i+N})$, into segments and build a model on top of each segment [15]. Many algorithms, such as [15], have been proposed to optimize the segmentation of time series. While our algorithms are applicable to other models, considering their relative popularity for data management purposes (e.g. [8, 11]), we mainly evaluate our approach with piece-wise regression models. A discussion of the extension of our techniques to the other models is presented in Appendix D.

Common measures of model **a**ccuracy **r**equirements (ar) include but not limited to:

- Direct value approximation, such as ranges of the form $[v-\epsilon, v+\epsilon]$, where v is the actual raw value and ϵ is the permissible (exact numerical) deviation from that value.
- Cumulative error measures, such as the mean square error, distance metrics etc.
- Probability-based accuracy constraints, such as confidence intervals, variance requirements etc.

As explained in the next section, our model dissemination algorithms make no specific assumption on the definition of accuracy requirements. The only assumption is that the accuracy requirements can be quantified as *ratio values*, i.e. values among which a total ordering exists. Moreover, this ordering should reflect the relative accuracy orderings and the degree of the differences between accuracy requirements.

Public Model Interface. As the dissemination system is model agnostic, we need to define a generic public interface that the dissemination system can access. This interface, will provide the system with the necessary information to effectively route the models and at the same time hide any specific internal mechanisms of the individual models.

This interface can capture models that can be represented as piece-wise functions, which partitions the time-series into segments each modelled by a function. We make no restriction on the form of the function.

We denote the models defined by all the users over a given time-series as M and the ith model as m_i. Our solution requires that a model m_i should expose the following information and methods to the system:

- **Model parameters:** for the purposes of computing the cost of model dissemination, the system has to know the number and schema (i.e. types) of the individual model parameters.
- **Model accuracies and valid ranges:** each model m_i should be associated with a list of **a**ccuracy and valid **r**ange **p**airs, $ARP_i = \{arp_{i,1}, \cdots, arp_{i,n}\}$, where $arp_{i,j} = (a_{i,j}, T_{i,j}[p_{i,j} : q_{i,j}])$ and $a_{i,j}$ is the accuracy of m_i within the segment $T_{i,j}[p_{i,j} : q_{i,j}]$ of the time-series T. For brevity, we call $T_{i,j}[p_{i,j} : q_{i,j}]$ the *valid range* of m_i w.r.t. the accuracy $a_{i,j}$ and denote it as $r_{i,j}$ hereafter. Figure 2 shows an example. Here we build three models to represent the time-series T and there are two *arp*s for each model m_i. For instance, model m_1's accuracy over the segment $r_{1,1}$ is $a_{1,1}$. If $a_{i,j}$ is more stringent than $a_{i,k}$, we denote it as $a_{i,j} < a_{i,k}$. Data receivers can define their **a**ccuracy **r**equirements (ar) according to their information needs and the models that they expect. We call a model m_i is *consistent* with the accuracy requirement ar_j as long as there exist an $arp_{i,j}$ such that $a_{i,j} \leq ar_k$. Furthermore, we say that model m_i is *applicable* to node n_k if m_i is consistent with ar_k. As we will see soon, we can leverage the information of models' accuracies and valid ranges to minimize the model dissemination cost.
- **Model generators:** functions for creating the models over a time-series with accuracy requirement constraints should be provided to the system. As shown in the later sections, we take use of two functions: (1) Create models over a time-series T with a particular accuracy; (2) Determine the valid ranges of given models based on a particular accuracy. These two functions will be used by the algorithms described in the next section to generate the necessary models and minimize the models to be disseminated through the network.

We will see later in the next section how this interface can be utilized.

2.2 Model Routing Infrastructure

Our routing infrastructure consists of an overlay network of N nodes. Those nodes can be either *model producers* or *model receivers*. The model producers collect the raw data from the data source and generate the data models. Those nodes are the only nodes that have access to the raw data. As we explain in the next section, depending on the dissemination method that is employed the model producers may perform additional (routing related) tasks. The data receivers, on the other hand, host the end users of the models, which have specific information and accuracy requirements and expect models that satisfy those requirements. In addition to that, the data receivers may forward models to neighborhood nodes (thus, we also call them *router nodes*).

Following many existing publish/subscribe systems, the nodes of the dissemination network are structured into *routing trees* [1, 20, 25]. A routing tree could be some sort of minimum distance tree covering a particular subset of the dissemination infrastructure. The root of such a tree is a model producer, while the rest of the nodes are data receivers. For its construction, appropriate distance measures

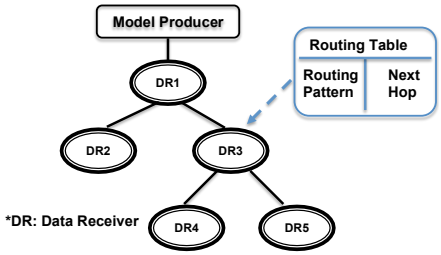


Figure 3: Routing Tree.

such as geographic distance among the nodes, link quality, network traffic etc may be taken into consideration [20, 25]. When a node (either a model producer or a data receiver) wants to join, it usually does so heuristically through a close by node that already belongs to the dissemination network. Periodically the entire routing structure may be restructured so that global routing quality requirements, such as load balancing and maximum throughput for the entire system, are achieved [25]. In Figure 3 an example of a routing tree is presented, where a model producer is connected with five data receivers. Two of the data receivers ($DR1$ and $DR3$) perform model routing as well to the rest of data receivers.

The dissemination of information from model producers to the receivers is accomplished through messages that are exchanged among the nodes. The end user requirements are described with *user profiles*, which are installed in the dissemination infrastructure during system bootstrap. Depending on the actual message routing method that is employed in the system, the information that is captured in the profiles may vary from direct data receiver addressing (e.g. IP address), indirect data receiver specification (e.g. data receivers with accuracy requirement stricter or equal to ar), content-based (e.g. predicates on the disseminated models) or any combination of those. A *routing table* is maintained at each router node, which compactly translates from user profiles to next hops. As in the case of model producer, depending on the routing algorithm, additional metadata information may be maintained at each router node.

2.3 Optimization Problem

A key task of the model routing algorithms is to decide what models to send to each data receiver. We formalize this task with the following optimization problem, called Min-Model-Dissemination , that we attempt to solve in this paper. Note that, to simplify the problem, we take the routing structure as the input of our problem and assume it is constructed by existing algorithms, such as [25].

Min-Model-Dissemination*: Given a finite time series T, a set of models $\{m_1, \cdots m_n\}$ over any segment of T, the accuracy requirement ar_k of each client node n_k, and the routing structure of the overlay network, choose the set of models M_k that are applicable to n_k, such that the T is totally covered by the valid ranges of the models in M_k and the total number of models that are sent over the network is minimized.*

In the above statement, we call a set of segments of T, $\{T_1, T_2, \ldots, T_i\}$, *covers* T, if their union is equivalent to T. Furthermore, the Min-Model-Dissemination problem is an NP-hard problem (Appendix B).

One possible solution would be to minimize the number of models built for each individual receiver under the

constraint of the receiver's accuracy requirement. However this approach fails to exploit the sharing opportunities of model dissemination. Consider a simple example that we disseminate the models over the time-series shown in Figure 2 to two nodes: n_i and its child node n_j. Suppose $ar_i = a_{1,1} = a_{2,1} = a_{3,1}$, $ar_j = a_{1,2} = a_{2,2} = a_{3,2}$ and $ar_i < ar_j$. Then models m_1, m_2 and m_3 can be sent to n_i. One can also see that n_i can actually forward m_1 and m_2 to n_j instead of disseminating a different set of models from the publisher all the way to n_j.

To exploit such sharing opportunities, there are two particular challenges. First, generating the models is a costly job and hence generating all the possible models is infeasible. Therefore, we have to selectively generate models that are more likely to be shared by more nodes. Second, there might be a lot of data receivers and there could be a large number of different accuracy requirements. This could also incur high cost to generate the models as we have to produce the different valid ranges for different accuracies. An efficient way to reduce such an overhead is needed.

3. MODEL DISSEMINATION

In this section we describe our model dissemination methods, which try to explore model sharing, and at the same time not clutter the data receivers with redundant models.

3.1 Centrally-Controlled Routing

The algorithm assumes that global information on the structure of the dissemination network as well as the accuracy requirements of each data receiver are available at the model producer. Thus, the model producer has all the necessary information to solve the optimization problem described in Section 2.3. However, as we explained in that section, the optimal solution is computationally impractical to achieve. Consequently, we propose a heuristic way of generating the required models for each data receiver that quickly provides a sufficient and viable compromise.

The method identifies a subset of data receivers whose models, if shared among the rest of the data receivers, have potential to reduce the amount of information that needs to be transmitted in the network. We call the accuracy requirements of those data receivers the *vantage accuracies* from now on. Once the vantage accuracies have been chosen and their consistent (Section 2.1) models have been generated, the models for the rest of the accuracy requirements are created by appropriately extending the valid ranges of those initial models.

The first step of the algorithm is to choose the vantage accuracies. We choose them in such a way that their consistent models can be re-used for the other accuracy requirements. To accomplish that, we employ the following simple lemma:

LEMMA 1. *Suppose a model m_i over a time-series T is consistent (see Section 2.1) with accuracy requirement ar_k and $[s, e]$ is the valid range of m_i with regard to ar_k. Then the same model m_i is also consistent with ar_j in the range $[s - \epsilon_1, e + \epsilon_2]$ as long as $ar_k \leq ar_j$, where $\epsilon_1, \epsilon_2 \geq 0$.*

In the lemma, the relation $\leq$ signifies that the left side accuracy requirement is more stringent than the right side accuracy requirement. From that lemma, it becomes obvious that the models created for a data receiver with accuracy requirement ar can serve any other data receiver whose accuracy is less stringent than ar.

A solution to the problem of choosing the vantage accuracies would be to quantitatively characterize and formally compute the degree of model sharing as a constraint satisfaction/optimization problem. However, as our experiments reveal, merely maximizing the degree of model sharing leads to suboptimal solutions. This is due to the fact that generating the models for a data receiver with accuracy requirement ar_j by extending the ranges of the models consistent with ar_i, where $(ar_i \leq ar_j)$, may result in more models than generating the models for ar_j from scratch. As a result, an optimal solution to the choice of vantage accuracies requires a computationally expensive global optimization process.

We experimented with forming the vantage accuracies in a heuristic fashion by choosing, for each of the children of the model producer, the data receiver with the most stringent accuracy requirement in that branch of the routing tree. The intuition behind this heuristic is that, for each branch, the models for the most stringent accuracy requirement in that branch will have to be generated anyway. Consequently, if we choose those models as the basis for the creation of the rest of the models we reuse their computation for free. This heuristic, although very simple, can achieve, as shown in our experimental evaluation, considerable model sharing.

Once the vantage accuracies have been chosen and their consistent models have been generated, the method employs the previous lemma to create candidate models for the rest of the accuracy requirements. In particular, for each of those accuracy requirements, ar_k, it (a) chooses an appropriate vantage accuracy ar_j and (b) extends the valid range of each model that has been generated for ar_j as much as possible with the constraint that the model is consistent with ar_k. The chosen vantage accuracy is the one which (a) is more stringent than ar_k and (b) maximizes model sharing within the dissemination network. The lemma ensures that the expanding is a valid process.

After having identified the the previous models, the algorithm has to prune some of those so that it identifies the smallest number of models that need to be disseminated to each data receiver. The reason why pruning is required is because the range extension that was performed in the previous step may result in models whose valid ranges may overlap in arbitrary ways and as a result some of those models are redundant. This is true if the ranges of the rest of the models fully cover them. To efficiently solve this problem we map it to a set covering problem as follows:

LEMMA 2. *Given the models M over T, to minimize the number of models that are needed by a receiver with ar_k, we have to choose the minimum number of ranges that covers T from the subset $\{r_{i,j} | a_{i,j} \leq ar_k\}$.*

Although the set-covering problem is known to be NP-hard, there exists a well-known, greedy-based approximation algorithm [4]. We adopt that algorithm in our implementation.

Once the model creation phase completes, the generated models are disseminated to the corresponding data receivers in a batch fashion; during that phase, identical models along a particular path, are only transmitted once.

3.2 Distributed Routing

As shown in our experimental evaluation, the previous algorithm does a good job in minimizing the number of models to be routed. Nevertheless it requires the model producer to eliminate redundant models by solving the set covering

problem that we described in Section 3.1 for each accuracy ar_k. With a large number of accuracies, that might incur a large computation overhead and hence turn the model producer into the bottleneck for the entire system.

We propose to reduce the load of the model producer by offloading some of the work to the router nodes. In order to identify what can be delegated we make two observations: First, the only computations that need to take place at the model producer are those involving the original data sequence T. Moreover, once a model has reached a data receiver n_k with accuracy requirement ar_k, the same model can be forwarded to any descendant, n_j, of that data receiver, with $ar_k \leq ar_j$. Such model-recycling opportunities make forwarding additional models to n_k, in order to serve the aforementioned descendants, unnecessary.

In fact we can do even better than that: as we explained in the previous section, some of the models that are sent to the data receiver, n_i, may be redundant for any descendant data receiver, n_j, with $ar_i \leq ar_j$. In the centralized algorithm, redundant model elimination is performed at the model producer and it requires the ARP list associated with each model. If a data receiver has this information then it can perform the elimination, thus saving the set cover computation at the model producer. That information has to be generated at the data source site, as it requires access to T and as a results needs to be forwarded to the data receivers.

The routing algorithm begins by generating the valid ranges at the model producer. Then the producer will perform the routing to its child nodes. The models that are sent to each child are chosen based on the strictest accuracy ar_j in the subtree of the child. The redundant model elimination algorithm is run to fulfill this task. The similar process is run at each node once it receives the models from its parent. Note that if ar_j happens to be in the vantage accuracies that are chosen by the producer, we can simply pick those models that are originally generated based on ar_j without running the redundant model elimination algorithm.

3.3 Accuracy Clustering

For applications that have no restrictions on the choice of accuracy requirements, the system could receive a large number of different accuracy requirements, $\{ar\}_n$, and as a result need to generate a large number of models. That may incur high overhead in generating and routing those models.

To solve the problem, we propose to partition the ars into a fixed number of clusters, say N (where $N << n$). All the data receivers whose ars are in the same cluster will receive the same set of models that are generated based on the strictest ar in the cluster. Thus, each data receiver will get models that are consistent with their accuracy requirements.

We denote the ith accuracy in the jth cluster as ar_i^j and the strictest ar in the jth cluster as sar_j. The clustering algorithm attempts to cluster accuracies based on their proximity to each other. To achieve that we compute, for each cluster, the sum of the differences of each accuracy from the strictest accuracy in that cluster. Our objective function is then, the total sum over all clusters, i.e. $\sum_j \sum_i (ar_i^j - sar_j)$ and the clustering objective is to minimize that function.

Existing clustering algorithms could be adapted to solve this problem, which has a different objective function in comparing to existing clustering problems. As the study of cluster optimization is beyond the scope of this paper, we adopt a simple and low cost distributed clustering algorithm in the current implementation of our system. As shown in the experiment results, such a simple algorithm can already achieve a great improvement.

The algorithm runs in a bottom up fashion and starts at each leaf node in parallel. Initially each cluster only contains one ar value. If the number of clusters is smaller than N, then the clusters will be forwarded to the parent node. For each cluster c_i, we only maintain two values: the strictest ar in the cluster, i.e. sar_i, and the number of elements in this cluster, ne_i. If the number of clusters is larger than N, then the algorithm would selectively merge two clusters at each iteration until the number of clusters is equal to N.

4. EXPERIMENTAL EVALUATION

This section presents our major experimental results. We prototyped the model dissemination system described in Section 2.2 in Java 6 using *Java RMI* for distributed communication. Additional experimental settings and results can be found in Appendix C.

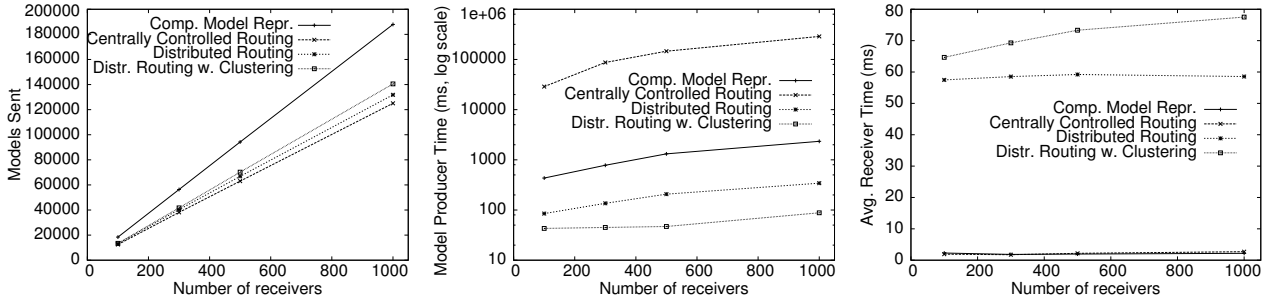
Routing Methods. We implemented and compared the following routing methods:

- *Compact Model Representation (CMR)*, i.e. the simple solution that was described in Section 2.3. This algorithm tries to minimize the number of models for each individual receiver. Due to the lack of prior work we use this as the baseline.
- *Centrally Controlled Routing (CCR)*, which performs all the routing decisions at the model producer and was presented in Section 3.1.
- *Distributed Routing (DR)*, which offloads some of the routing decisions to the router nodes and was described in Section 3.2.
- *Distributed Routing with Accuracy Clustering (DRAC)*, which groups the accuracy requirements into a small number of clusters while still performing distributed routing decisions and was detailed in Section 3.3.

Data. We used the publicly available sensor measurements from Intel Berkeley Research ([12]). In particular, we experimented with the temperature data from sensor 45.

Measures. We examined both the resource consumption at the nodes of the dissemination infrastructure (i.e. model producer and data receivers) and network usage. To accomplish that we have measured:

- **Network bandwidth**: To estimate the required network bandwidth, we count the total number of unique models that are routed along all the edges of the routing tree. Depending on the employed method, some additional metadata may also be need. The required space for this additional metadata can be amortized due to the batching that we employ during the actual model routing. However for each model we have to send the potentially large number of parameters (most of which will require floating point precision). As a result, those parameters constitute the bulk of the space requirement. Taking into consideration that the models within a batch all have the same form, their number is directly proportional to their space requirements.
- **Running time at each node**: Our setup consists of homogeneous, dedicated nodes. As a result by measuring running time of the algorithms (i.e. wall time), we can estimate the CPU consumption at those nodes.



(a) Network Bandwidth Consumption (b) Model Producer Time (c) Avg Receivers Time

Figure 4: Sensitivity to #Receivers.

4.1 Number of Receivers

We first vary the number of data receivers from 100 to 1000, while keeping the other parameters fixed. The results are shown in Figure 4, where the X axis represents the number of data receivers.

As shown in Figure 4(a), the Compact Model Representation (CMR) approach incurs the highest bandwidth consumption (which is represented by the total number of models sent) in comparison to the other three approaches. The main reason for this is that although it generates the minimum number of models that are required per data receiver, those models provide small sharing opportunities with models of other data receivers. In contrast, the rest of the algorithms make model sharing a principal target.

As shown in the same figure, the difference in bandwidth consumption becomes larger as the number of data receivers increases, due to the fact that there are more opportunities for model sharing with a larger number of receivers.

As expected, the CCR method performs best in minimizing the bandwidth consumption. Nevertheless, the distributed routing approaches perform only slightly worse, although they can only make local decisions. Those results show that the distributed routing and accuracy clustering techniques do not lower much the optimization quality.

Furthermore, as shown in Figure 4(b), although the CCR method performs the best in terms of bandwidth consumption, it takes a very long running time at the model producer, which could, because of that, become a bottleneck of the system. That is because the algorithm depends on the model producer to control the entire model routing process. That entails (a) generating the models and (b) performing the redundant model elimination step for all the data receivers which are time-consuming tasks. On the other hand, the two distributed approaches reduce the producer's running time by more than three orders of magnitude over CCR and up to an order of magnitude over the baseline (CMR) method. The $DRAC$ approach further improves the producer's performance by reducing the number of accuracy requirements that the latter has to handle.

Moreover, as shown in Figure 4(c) both distributed approaches require more computation at the data receivers. This is because they share the routing decisions with the model producers. Nevertheless, the more balanced workload among the model producers and the data receivers help to avoid potential bottlenecks that may otherwise occur.

Two last observations have to be made from the figures: (a) the running time of all three of our methods is not greatly affected by the number of data receivers, while (b) the number of required models increases linearly with the former number. Those are very encouraging results as they hint for the scalability of our methods.

4.2 Sensitivity to Accuracy Requirements

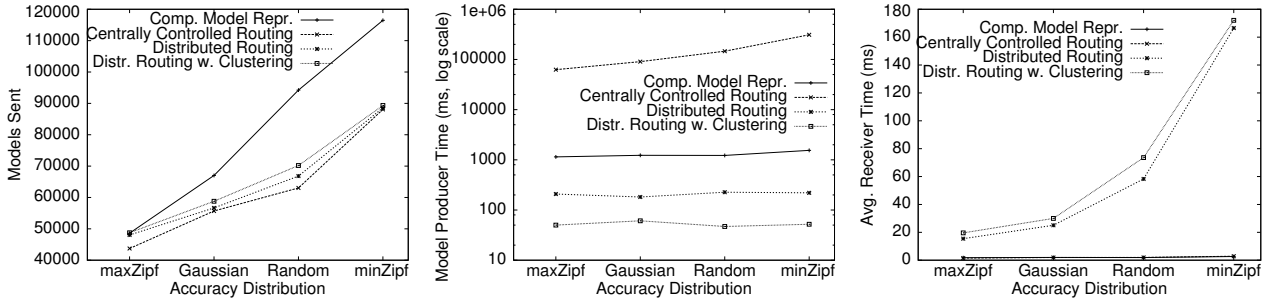
Next we examine how the algorithms perform when we vary the characteristics of the accuracy requirements (ar). We first vary the data distribution of the receivers' accuracy requirements (ar). The range of possible ar's is fixed to the default value (i.e. $1\% - 30\%$) and we use the following data distributions to generate a data receiver's ar value:

- **MaxZipf**: we use a zipf distribution with the parameter as 1.1 and the higher end of the accuracy range as the most popular value. This is to simulate the situation that most data receivers require low accuracies.
- **Guassian**: a Gaussian distribution with the median value in the accuracy range as the mean. This is to simulate the situation that most data receivers require a medium accuracy.
- **Random**: a uniform distribution over the whole range of accuracies. This is the default in other experiments.
- **MinZipf**: similar to MaxZipf, except the lower end of the accuracy range is set as the most popular rather the higher end. In this case, most data receivers have high accuracy requirements.

The results are illustrated in Figure 5. From Figure 5(a), one can observe that when the data receivers have higher accuracy requirements, all the approaches consume more network bandwidth. The CRM again consumes more bandwidth than the three proposed approaches and the difference becomes much larger when the data receivers become more demanding. The reason is that the demanding data receivers require the system to disseminate more models and hence the need to explore model sharing becomes more important. In addition, as shown in Figure 5(b) and 5(c), the distributed approaches manage to balance the routing task load among the model producer and the data receivers and hence avoid bottleneck.

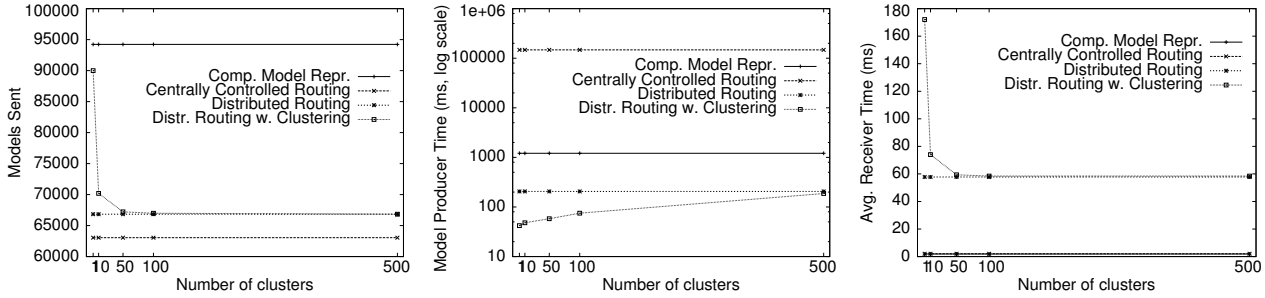
4.3 Accuracy Requirement Clustering

In the final experiment we examine the system's sensitivity to the number of accuracy clusters. For that purpose, the number of clusters is varied from 1 to 500. The results are shown in Figure 6. As can be seen from the figure, with 50 clusters, we can reduce the CPU consumption by more than half and only increase the bandwidth consumption very slightly. Moreover, the data receivers' running time is practically unaffected by the clustering.



(a) Network Bandwidth Consumption (b) Model Producer Time (c) Avg Receivers Time

Figure 5: Sensitivity to the Distribution of Accuracy Requirements.



(a) Network Bandwidth Consumption (b) Model Producer Time (c) Avg Receivers Time

Figure 6: Sensitivity to the Number of Accuracy Clusters.

5. RELATED WORK

This paper is related to research efforts in the area of large-scale information dissemination systems as well as the modeling and processing of time-varying data. We have already related our work to many previous efforts. In this section we present some additional reference material.

Publish/Subscribe systems. Information dissemination is the focus of distributed publish/subscribe systems. Publish/subscribe [9] is an asynchronous messaging paradigm, where no direct communication between message publishers and subscribers takes place. In addition, many efforts have focused on extending publish/subscribe systems to support complex data and subscription types [2, 24, 27]. Due to the advantages of $P2P$ systems in terms of scalability and fault tolerance, recent efforts have focused on developing publish/subscribe systems on top of them [21, 10]. Such techniques can be easily adopted into our system as the latter does not require any specific message routing structure.

Dynamic Data Dissemination. Conceptually either of the publish/subscribe or peer-to-peer systems can be used to disseminate dynamic data. However, the high frequency of data updates and the large number of data objects pose extra challenges to its scalability. Therefore, researchers have studied how to exploit users' tolerance of data inaccuracy to reduce the amount of network traffic [18, 19, 20, 26, 25]. Our approach is complementary to those works as it aims at optimizing the dissemination of models by exploiting opportunities of sharing them among the interested parties.

Data Modeling and Processing. In the area of data acquisition, the use of models to reduce the amount of data that needs to be transferred from the data source to the base station [3, 7, 13, 23] has been studies. The basic idea is that if the base station can use a model to predict the updates of sensor values, the sensors do not need to send the update. Models that have been examined include the Kalman Filter [13], AR models [23], and the multivariate Gaussian distribution [3, 7]. In our work we investigate an orthogonal, yet related problem, namely that of disseminating data models to a large number of receivers. In that setting the optimization goal is to share models among receivers, as opposed to the minimization of amount of information gathered from the data sources.

6. CONCLUSIONS

In this paper, we have introduced the problem of distributed dissemination of models over time-varying data. We have proposed the architecture of a large-scale model dissemination system and solved an important optimization problem that has risen, i.e. minimizing the network bandwidth consumption by exploiting the sharing of models among receivers. As the number of possible models is large and the model generation is an expensive task, effective heuristics have been proposed to selectively generate models that have good sharing potential.

Additionally, both central and distributed model routing algorithms have been proposed. As experimentally shown, although the centralized algorithm performs slightly better in minimizing bandwidth consumption, it incurs high computation cost at the model producer. The distributed algorithm, on the other hand, successfully shares the routing task load among all the nodes. Finally, to further limit the cost of model generation, we proposed to group the accuracy running time by more than 70% with 1000 receivers.

7. REFERENCES

[1] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event

notification service. *ACM Transactions on Computer Systems (TOCS)*, 19:3:332–383, Aug. 2001.

[2] B. Chandramouli, J. Xie, and J. Yang. On the database/network interface in large-scale publish/subscribe systems. In *Proc. ACM International Conference on Management of Data (SIGMOD)*, pages 587–598, Chicago, USA, 2006.

[3] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *Proc. IEEE International Conference on Data Engineering (ICDE)*, Atlanta, USA, 2006.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms, 2nd edition, 2001.

[5] G. Cormode, T. Johnson, F. Korn, S. Muthukrishnan, O. Spatscheck, and D. Srivastava. Holistic udafs at streaming speeds. In *Proc. ACM International Conference on Management of Data (SIGMOD)*, pages 35–46, Paris, France, 2004.

[6] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. In *Proc. ACM International Conference on Management of Data (SIGMOD)*, pages 155–166, Paris, France, 2004.

[7] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proc. International Conference on Very Large Databases (VLDB)*, pages 588–599, Newport Beach, USA, 2004.

[8] A. Deshpande and S. Madden. Mauvedb: supporting model-based user views in database systems. In *Proc. International Conference on Management of Data (SIGMOD)*, pages 73–84, 2006.

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35:2:114–131, 2003.

[10] A. Gupta, O. Sahin, D. Agrawal, and A. Abbadi. Meghdoot: Content-based publish/subscribe over P2P networks. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, page 273, 2004.

[11] S. Ilarri, O. Wolfson, E. Mena, A. Illarramendi, and P. Sistla. A query processor for prediction-based monitoring of data streams. In *Proc. International Conference on Extending Database Technology (EDBT)*, pages 514–426, Saint Petersburg, Russia, 2009.

[12] Intel Lab Data. `http://db.csail.mit.edu/labdata/labdata.html`.

[13] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using kalman filters. In *Proc. ACM International Conference on Management of Data (SIGMOD)*, pages 11–22, Paris, France, 2004.

[14] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3:3:263–286, Aug. 2000.

[15] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *Proc. IEEE International Conference on Data Mining (ICDM)*, pages 289–296, Berkeley, USA, 2001.

[16] E. J. Keogh and M. J. Pazzani. Relevance feedback retrieval of time series sata. In *Proc. ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 183–190, Berkeley, USA, 1999.

[17] J. Kierman and E. Terzi. Constructing comprehensive summaries of large event sequences. In *Proc. ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 417–425, Las Vegas, USA, 2008.

[18] S. Shah, S. Dharmarajan, and K. Ramamritham. An efficient and resilient approach to filtering and disseminating streaming data. In *Proc. International Conference of Very Large Databases (VLDB)*, pages 57–68, Berlin, Germany, 2003.

[19] S. Shah, K. Ramamritham, and P. J. Shenoy. Maintaining coherency of dynamic data in cooperating repositories. In *Proc. International Conference of Very Large Databases (VLDB)*, pages 526–537, Hong Kong, China, 2002.

[20] S. Shah, K. Ramamritham, and P. J. Shenoy. Resilient and coherence preserving dissemination of dynamic data using cooperating peers. *IEEE Transactions on Knowledge and Data Egnineering (TKDE)*, 16:7:799–812, july 2004.

[21] W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. Buchmann. A peer-to-peer approach to content-based publish/subscribe. In *Proceedings of the 2nd international workshop on Distributed event-based systems*, pages 1–8, 2003.

[22] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy. Probabilistic inference over rfid streams in mobile environments. In *Proc. IEEE International Conference of Data Engineering (ICDE)*, pages 1096–1107, Shanghai, China, 2009.

[23] D. Tulone and S. Madden. Paq: Time series forecasting for approximate query answering in sensor networks. In *Proc. European Workshop on Wireless Sensor Networks (EWSN)*, pages 21–37, Zurich, Switzerland, 2006.

[24] Y. Zhou, K. Aberer, and K.-L. Tan. Toward massive query optimization in large-scale distributed stream systems. In *Proc. ACM/IFIP/USENIX International Conference on Middleware*, pages 326–345, Leuven, Belgium, 2008.

[25] Y. Zhou, B. C. Ooi, and K.-L. Tan. Disseminating streaming data in a dynamic environment: an adaptive and cost-based approach. *The International Journal on Very Large Data Bases (VLDB J.)*, 17:6:1465–1483, nov 2008.

[26] Y. Zhou, B. C. Ooi, K.-L. Tan, and F. Yu. Adaptive reorganization of coherency-preserving dissemination tree for streaming data. In *Proc. IEEE International Conference on Data Engineering (ICDE)*, page 55, Atlanta, USA, 2006.

[27] Y. Zhou, A. Salehi, and K. Aberer. Scalable delivery of stream query results. In *Proc. International Conference of Very Large Databases (VLDB)*, pages 49–60, Lyon, France, 2009.

APPENDIX

A. ADDITIONAL ALGORITHM DETAILS

In this section, we will present the additional details of the algorithms described in Section 3. In particular, pseudocodes and their brief description will be presented.

A.1 Centrally-Controlled Routing

The pseudocode of the algorithm to generate the valid model ranges introduced in Section 3.1 is presented in Algorithm 1. As shown in line 6 of the algorithm, the first step is to choose the vantage accuracies. Once the vantage accuracies have been chosen and their consistent models have been generated (lines $7-10$ of algorithm 1), the algorithm create candidate models for the rest of the accuracy requirements.(lines $11-20$ of algorithm 1).

Algorithm 1 Generate Valid Model Ranges.

1: **Input:** *Data T, Accuracy requirements ar_i, $1 \leq i \leq n$*
2: **Output***: Models M, Vantage accuracies VA, $|VA| < n$*
3:
4: Pick k vantage accuracies VA, $1 \leq k < n$
5: **for each** $ar \in VA$ **do**
6: Create models M_{ar} over T,
7: such that $\forall\ m_i \in M_{ar}$, $a_{i,1} \leq ar$ and $r_{i,1} \subseteq T$
8: **end for**
9: **for each** accuracy $ar_k \notin VA$ **do**
10: Pick accuracy $ar_j \in VA$
11: **for each** model $m_i \in M_{ar_j}$ **do**
12: $r_{i,l} := r_{i,j}$ and $a_{i,l} := ar_j$
13: **while** $a_{i,l} \leq ar_k$ **do**
14: Expand $r_{i,l}$
15: **end while**
16: Insert $< a_{k,l}, r_{i,l} >$ into ARP_i
17: **end for**
18: **end for**

The centrally-controlled routing algorithm is shown in Algorithm 2.

Algorithm 2 Centrally Controlled Model Routing.

h
1: **Input:** *Data Sequence T, Accuracy requirements $\{ar_i\}$*
2: *Date Receivers $\{n_k\}$*
3: **Output***: A set of models M_k for each receiver n_k*
4:
5: $\{M, VA\}$:= Generate Valid Model Ranges (T, ar_i)
6: **for each** accuracy $ar_k \notin VA$ **do**
7: Compute min cover of T from the range set $\{r_{i,j} | a_{i,j} \leq ar_k\}$
8: Collect the corresponding models for the selected ranges into M_k
9: Send model M_k to the receiver n_k
10: **end for**

A.2 Distributed Model Routing

The pseudocode of the distributed model routing algorithm is presented in Algorithm 3. The algorithm begins by generating the valid ranges at the model producer (lines 7–9) and then the producer will perform the routing to its child nodes (lines 10–15). Each node in the dissemination tree will perform a similar procedure when it receives the models.

Algorithm 3 Distributed Model Routing.

h
1: **Input:** *Data sequence T (only at the producer)*
2: *Accuracy requirements $\{ar_i\}$*
3: *Data Receivers $\{n_k\}$, Model Producer mp*
4: **Output***: A set of models M_k for child receiver n_k*
5:
6: **if** this is the producer **then**
7: $\{M, VA\}$:= Generate Valid Model Ranges (T, ar_i)
8: **end if**
9: **for each** child node n_i **do**
10: ar_j := the strictest ar in the subtree of n_i
11: Compute min cover of T from the range set $\{r_{x,y} | a_{x,y} \leq ar_j\}$ {Note that this step can be saved away if ar_j is in the vantage accuracies chosen by the producer}
12: Collect the corresponding models for the selected ranges into M_j
13: Send M_j to n_i
14: **end for**

A.3 Accuracy Clustering

The pseudocode of clustering accuracies can be seen in Algorithm 4.

Algorithm 4 Accuracy Clustering.

1: **Input:** *Set of clusters $\{c_1, c_2, \cdots, c_k\}$*
2: *The targeted number of clusters N, $k > N$*
3: **Output***: New set of clusters $\{\underline{c}_1, \underline{c}_2, \cdots, \underline{c}_N\}$*
4:
5: **while** $\#clusters > N$ **do**
6: Pick clusters c_i and c_j such that
7: $sar_i \leq sar_j$ and $(sar_j - sar_i) * ne_j$ is MIN
8: Merge c_i and c_j
9: Propagate the clusters to the parent node
10: **end while**

B. PROBLEM HARDNESS

THEOREM 1. *The* MIN-MODEL-DISSEMINATION *problem is NP-hard.*

PROOF. If we restrict the MIN-MODEL-DISSEMINATION problem by ignoring the routing structure, then the problem becomes finding the minimum number of models applicable to each node, whose valid ranges completely covers T. This is equivalent to the set cover problem, which is NP-hard [4]. Therefore, the MIN-MODEL-DISTRIBUTED problem is an NP-hard problem. □

C. ADDITIONAL INFORMATION OF THE EMPIRICAL EVALUATION

In this section, we present additional information for the experimental evaluation that we performed that we have no space to present in the paper's main body.

C.1 Additional Details of the Experimental Setup

Data. As mentioned above, we used the publicly available sensor measurements from Intel Berkeley Research ([12]). As the default setting, in each round we disseminate the models derived from 400 sensor readings. We have also varied the number of readings from 100 to 1200. The conclusions are consistent with the experimental results that we discuss below. We repeated the same set of experiments on the *Adiac* dataset from the UCR time series archive [29] and obtained similar results (omitted due to lack of space).

Workload. As a default setting there exist 500 data receivers, structured in a balanced routing tree with a default fan-out of 3. The purpose of using a balanced tree is to easily control the height of the tree, which as shown later is critical in the performance of model sharing, by changing the fan-out. We have also used unbalanced tree structures and did not find any significant variations in the results. Finally, when accuracy clustering is required the default number of clusters is 10.

We use piece-wise polynomial (of varying degree) regressions as our models. The accuracy of a model is the maximum deviation between the model output and the raw data values within the valid range of the models.

To set the accuracy requirements for the receivers, we calculate the standard deviation of the values, sd. Then the accuracy requirement of each receiver is chosen randomly in the range of $[1\% \cdot sd, 30\% \cdot sd]$. Both the distribution and the range are varied. The random assignment is biased against our techniques, as we assume no specific clustering of data receivers based on their data accuracy requirements. In practice the receivers' requirements are taken into consideration while creating the routing tree [26, 25], and data receivers with similar requirements are clustered together so that the aggregate routing distance is minimized. That can potentially increase model sharing opportunities.

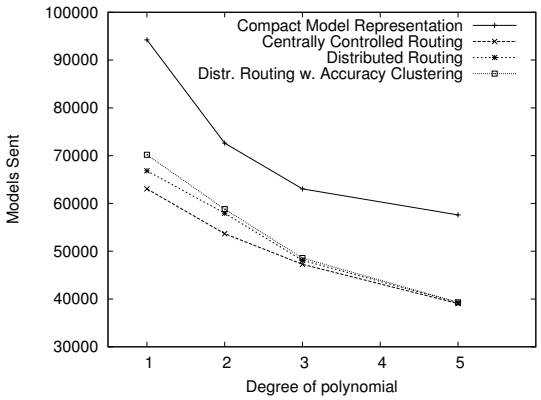
Prototype System. We have coded the dissemination infrastructure and the above methods in Java 6 using JAVA RMI for distributed communication.

Simulation Setup. The experiments were run on a Linux machine with an Intel Core 2 Quad CPU (3.0GHz) and 8GB RAM. As our measures are insensitive to whether the experiments are run in multiple machines or simulated in one machine we opted for simulating the distributed environment by running one instance of the system for each node in the network. We schedule one instance at a time in order to obtain an accurate running time estimation. By using simulation we can vary the scale of the experiments and control the parameters flexibly and in a time-effective way. Due to page limit, we only present a subset of the results; additional results can be found in the appendix.

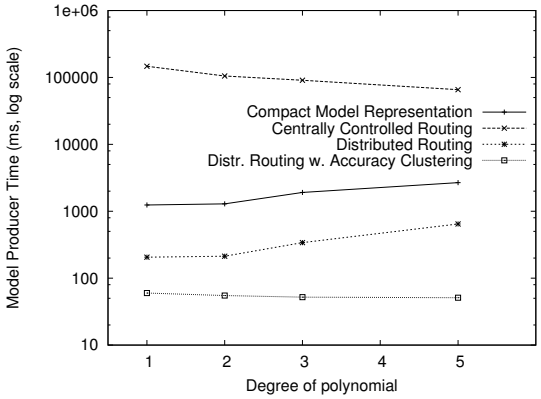
C.2 Degree of Polynomial

In this experiment we vary the degree of the polynomials that model the data. Note that the purpose of this study is not to determine which degree is better. Instead, our primary intention is to examine how the algorithms perform when different models are employed. We experiment with degrees from 1 (i.e. which represents linear regression) to 5 and we report the results in Figure 7.

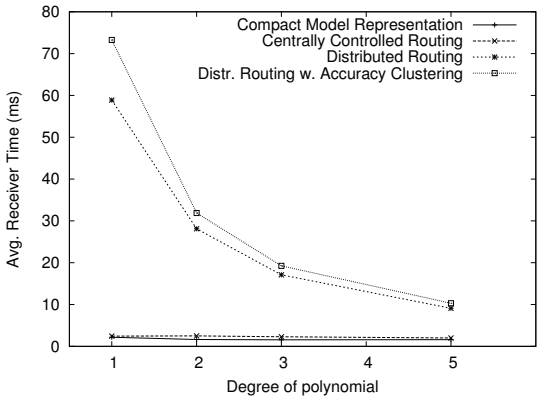
Figure 7(a) shows that, for the data that we used in the experiments, there are less models to be disseminated with a higher degree polynomial. Once again, our approaches



(a) Network Bandwidth Consumption

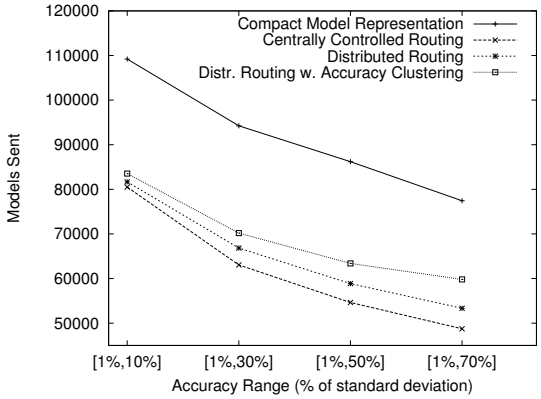


(b) Model Producer Time

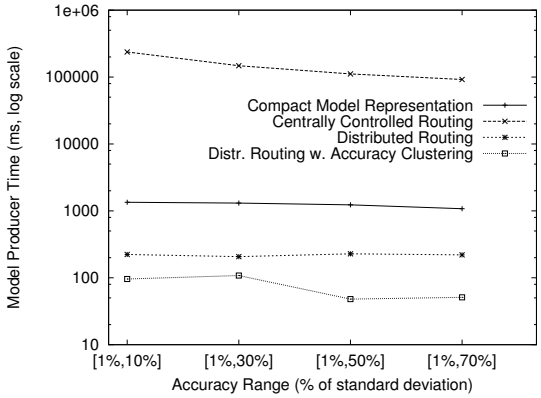


(c) Avg Receivers Time

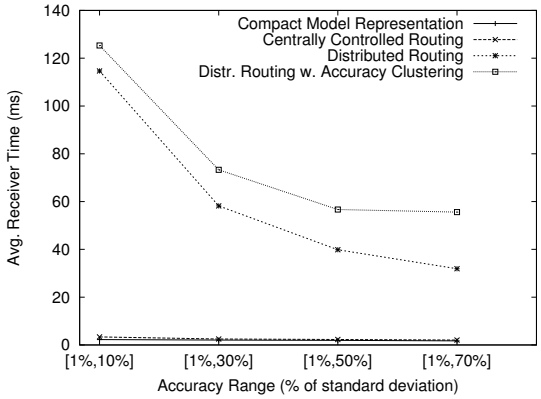
Figure 7: Sensitivity to the Degree of Polynomial.



(a) Network Bandwidth Consumption

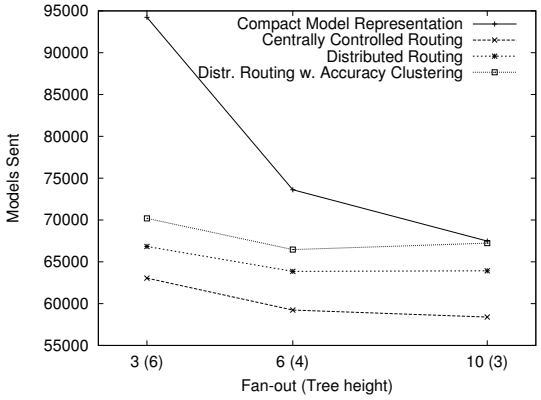


(b) Model Producer Time

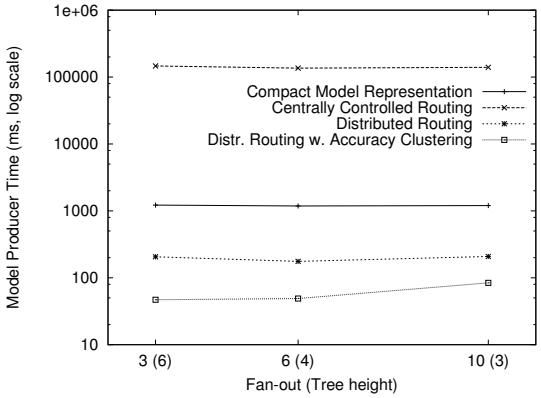


(c) Avg Receivers Time

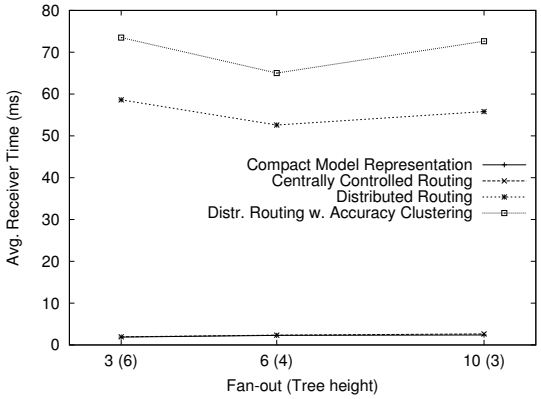
Figure 8: Sensitivity to the Range of Accuracy Requirements.



(a) Network Bandwidth Consumption



(b) Model Producer Time



(c) Avg Receivers Time

Figure 9: Sensitivity to Fan-Out.

outperform the baseline algorithm (CMR) consistently for all the different degrees. Another interesting fact is that, with higher degree of polynomials, both CCR and DR consume more and more CPU at the producer in comparison to $DRAC$. This is because higher degree polynomials are more complex to compute. Nevertheless, the accuracy clustering that is performed by $DRAC$ reduces the number of accuracies that are considered at the model producer and hence keep the computation time low at the producer.

C.3 Range of Accuracy Requirement

In this experiment we vary the range of accuracy requirements by varying the higher end of the range and investigate the effect of that range on the performance of the algorithms. In particular, we experiment with ranges of $[1\% - 10\%]$, $[1\% - 30\%]$, $[1\% - 50\%]$, and $[1\% - 70\%]$ and these percentages are in terms of the standard deviation of the test dataset. Figure 8 shows the results.

The observations from this experiment are similar with those in the previous experiments and reveal once again the superiority of the three proposed methods in terms of both network bandwidth consumption and running time at each node of the routing infrastructure. One additional trend worth mentioning is the decreased resource consumption when data receivers become more lenient in their accuracy requirements. This is as expected, because with less stringent accuracy requirements, the number of models that needs to be generated is potentially smaller as the same model can potentially represent a larger segment of the original data sequence.

C.4 Fan-out

In the forth experiment we varied the fan-out and hence the height of the routing tree. The results are shown in Figure 9(a). The figure shows that, with a higher fan-out, the CMR has a smaller network bandwidth consumption (however it is still outperformed by all three of the proposed algorithms). This trend hints that there is less to gain from model sharing with a shallow routing tree. Note that the number of receivers is set to 500. When the fan-out is equal to 10, there are only 3 levels in the routing tree, which is balanced. Therefore, as would be expected, the benefit of model sharing is not as high. In other words, it is more critical to exploit model sharing with a higher routing tree. Finally model producer's and data receivers' CPU consumption follows the same trend as those observed in previous experiments.

D. DISSEMINATION OF OTHER MODELS

In this paper, we have focused on piecewise polynomial approximations of time-varying data as the models to be disseminated. Nevertheless, any form of linear or non-linear regression can be easily employed to model a given partition. Other high level timeseries representations, such as symbolic (i.e. NLP-based or string based) representations (please see [28] for a description of such models), can be supported in a straightforward manner. More specifically, this can be done by (a) identifying the partitions of the timeseries that will enable maximum sharing and (b) applying the transformation on that partition. Another important class of models pertain to the signal-based analysis of timeseries (such as wavelet, Fourier and cosine transformations). In such cases, the timeseries is regarded as a time-varying functions which are then transformed into (infinite) weighted sums of some basis functions (i.e. sines and cosines in the case of Fourier and cosine transformations and wavelets in the case of wavelets). Subsequently, only the co-efficients of the most important bases functions are usually maintained. By identifying those co-efficients that are shared among multiple timeseries and adapting the definitions of accuracies accordingly, our dissemination techniques may be applied with minor changes. Finally, a number of classification models, such as the SVM and various of statistical models (e.g. hidden markov models or conditional random fields) have been extensively used for prediction and classification. Our techniques cannot be easily applied and hence further investigation needs to be conducted. We plan to study the possibility to effectively disseminate such models in the future.

E. REFERENCES

[28] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, San Diego, California, 2003.

[29] The UCR Time Series Classification/Clustering Page. `http://www.cs.ucr.edu/~eamonn/time_series_data/`.