Exploiting Correlations for Efficient Content-based Sensor Search

Richard Mietz*, Kay Römer*
* Institute of Computer Engineering
University of Lübeck
23562 Lübeck,Germany
Email: {mietz,roemer}@iti.uni-luebeck.de

***Abstract*— Billions of sensor (e.g., in mobile phones or tablet pcs) will be connected to a future Internet of Things (IoT), offering online access to the current state of the real world. A fundamental service in the IoT is search for places and objects with a certain state (e.g., *empty* parking spots or *quiet* restaurants). We address the underlying problem of efficient search for sensors reading a given current state - exploiting the fact that the output of many sensors is highly correlated. We learn the correlation structure from past sensor data and model it as a Bayesian Network (BN). The BN allows to estimate the probability that a sensor currently outputs the sought state without knowing its current output. We show that this approach can substantially reduce remote sensor readouts.**

I. Introduction

By equipping more and more real-world entities (e.g., things, places, and people) with Internet-connected sensors, the state of the real world becomes accessible online and in real time. As searching for static information is a fundamental service in the Internet today, finding real-world entities with a given current state (e.g., *free* parking spots, supermarkets with *short* waiting queues etc.) will be of great importance in the IoT. The underlying problem of finding sensors with a given current output is a challenging one due to the huge number of sensors in a future IoT and due to the highly dynamic nature of sensor output. In addition, wireless sensors are often battery powered, such that communication with them must be minimized. This renders traditional techniques such as streaming sensor output to a search engine or indexing the output of sensors useless. In previous work [1], we have investigated prediction techniques, where based on the past output of a sensor we learn a prediction model to estimate the probability that a sensor outputs a sought value at the time of query without knowing the current value of the sensor. Sensors are read in decreasing order of probability until enough matches have been found, thus substantially reducing the number of sensor readouts. However, many sensors are hardly predictable (such as those measuring weather phenomena) and the method cannot be applied there. Fortunately, colocated sensors often produce correlated output (consider two close-by weather stations) and this is what we exploit in our current work.

II. Approach

In this section we first give a brief overview about Bayesian Networks. Afterwards we will present our approach by formalizing the system and prediction model.

A. Bayesian Networks

A Bayesian Network (BN) is a directed acyclic graph to model probabilities of random variables and their conditional dependencies. Every node represents a random variable and every directed edge between two nodes their conditional dependencies. Parents of a node n are all nodes which have an edge to that node. If two nodes are not connected then they are conditionally independent of each other. Each nodes' probability distribution is calculated from the conditional probabilities of its parents. These are often represented as probability tables. If some variables are observed and this evidence is entered in the network, this knowledge can be propagated and so the probabilities can be refined. Solving this inference problem exactly is $\mathcal{NP}$-hard, but there exist several approximation algorithms to solve this problem for different kinds of networks.

B. System Model

We now introduce a formal model of our system. A sensor s from a set of sensors $\mathcal{S}$ which monitors real-world states is described by the function

$$s : \mathcal{T} \mapsto \mathcal{V}$$

where $\mathcal{T}$ is the set of discrete points in time and $\mathcal{V}$ is the discrete, finite, ordered set of output values of the sensor, e.g., the noise level of a place could be $\mathcal{V} = \{$*silent*, *adequate*, *loud*$\}$. That is, our system assumes sensors output a discrete, finite set of states, which are, in practice, inferred from continuous low-level measurements of one or more different sensors.

The prediction model is represented as

$$P : \mathcal{S} \times \mathcal{T} \times \mathcal{V} \mapsto [0, 1]$$

meaning that the model returns the probability that a sensor $s \in \mathcal{S}$ outputs a value $v \in \mathcal{V}$ at a point in time $t \in \mathcal{T}$.

A formal definition of a query to our system is given by

$$Q : \mathcal{T} \times \mathcal{V} \times k \mapsto \mathcal{R}.$$

We want to find k sensors which output a certain state $v \in \mathcal{V}$ at a point in time $t \in \mathcal{T}$, which are returned in the set $\mathcal{R} \subseteq \mathcal{S}$.

In the optimal case, the prediction model returns high probabilities for sensors matching the query and low probabilities for ones that do not match. Thus, ideally the query process only needs to contact the first k sensors on top of the list to read their current output state. Note that this only holds as long as there are at least k matching sensors and these are ranked highest by the model among all sensors. If the ranking is not optimal, the query process will need to contact more than k sensors. In the worst case of less than k matching sensors, even if the ranking is optimal, the query process contacts every sensor until it can assert that there are not enough hits. Hence, the metric to evaluate the quality of our proposed approach is the number of contacted sensors to find k matching sensors.

C. Prediction Model

We design a prediction model that exploits correlations between sensors. We define the correlation between two sensors s_i and s_j as

$$C_{i,j} = \frac{\sum\limits_{\forall t \in \mathcal{T}} 1 - \frac{|s_i(t) - s_j(t)|}{|\mathcal{V}| - 1}}{|\mathcal{T}|}$$

where the difference of two states is the difference of their indices in the ordered set $\mathcal{V}$. $C_{i,j}$ is a value between 0 and 1. The higher $C_{i,j}$, the higher the correlation between the two sensors.

Based on the computed correlations between all pairs of sensors, we create a BN. Every sensor s in our model is represented by a node in the BN. Because of the acyclicality of BNs, it is not possible to create a complete graph representing our network. Furthermore, the exponential growth of the probability table in the number of parents and states per parent constricts the creation of dense connected graphs. Hence, we decide to create a simple structure, namely a list structure, where every node has only one predecessor except the root. With this structure we still have small probability tables at every node while having every node connected to the network, thus correlations between any pair of nodes can be modeled as the dependency relationship in BN is transitive. Because evidence propagation has more impact on nearby nodes, we always connect strongly correlated nodes.

We illustrate the construction of the BN using the following example with four sensors and their states at four points in time as shown in Table Ia. The state is either *empty* (e) or *full* (f), thus $|\mathcal{V}| = 2$. The calculated correlation between every pair of sensors is shown in Table Ib. To construct the mentioned list structure, we sort the pairs by $C_{i,j}$ in descending order. First, the highest-ranked pair in the sorted list is chosen. If there is more than one pair with the same correlation, one is chosen randomly. In the example $C_{1,3}$ is selected as the first pair to construct the BN. Thus, s_1 is now the father of s_3. In the next step every pair which includes sensor s_1 is deleted from the list. Afterwards, among all pairs including sensor s_3, the one with the highest correlation, i.e., s_2, is the next to add to the BN. The process continues until the list is empty, i.e., every sensor is included in the BN. The resulting structure is $s_1 \rightarrow s_3 \rightarrow s_2 \rightarrow s_4$ where $\rightarrow$ represents the parent relationship and s_1 is the root node.

s	t_1	t_2	t_3	t_4
1	e	e	f	f
2	f	e	f	e
3	f	e	f	f
4	f	f	e	e

(a) Four sensors with their states at four points in time.

s_i	s_i	$C_{i,j}$
1	2	$\frac{1}{2}$
1	3	$\frac{3}{4}$
1	4	0
2	3	$\frac{3}{4}$
2	4	$\frac{1}{2}$
3	4	$\frac{1}{4}$

(b) Correlation between each pair of sensors.

TABLE I: Example with four sensors.

After construction of the BN, we fill in the probability table of every node which contains the probability for each state $v \in \mathcal{V}$ to occur. The root node of the network has no dependencies and the probability table is therefore identical to the probability distribution of the states of the respective sensor, which can be calculated for each state v by

$$P(s, v) = \frac{\sum\limits_{\forall t \in \mathcal{T}} \begin{cases} 1, & \text{if } s(t) = v \\ 0, & \text{else} \end{cases}}{|\mathcal{T}|}$$

In contrast, to populate the probability table for nodes with parents, we have to calculate the conditional probability for each combination of states of the node and its parent, i.e., for the example above for sensor 3 we need to calculate the probability of state *empty* if sensor 1 is in state *empty* and *full* as well as the probability of state *full* if sensor 1 is in state *empty* and *full*. Hence we have to calculate

$$P(s, v) = \sum_{\forall v, w \in \mathcal{V}} P(s, v, w)$$

where

$$P(s, v, w) = \frac{\sum\limits_{\forall t \in \mathcal{T}} \begin{cases} 1, & \text{if } s(t) = v \wedge pa(s, t) = w \\ 0, & \text{else} \end{cases}}{|\mathcal{T}|}$$

where $pa(s,t)$ returns the state of the parent of s at time t.

Whenever evidence for a node is available, the probability table of that node is updated, i.e., the probability of the current state of that node is set to 1 and the probability of all other states are set to 0. Using EPIS sampling [2] on the BN, the information is propagated through the network, updating probabilities for each state at every node.

D. Query algorithm

Algorithm 1 shows the query algorithm. The query, placed by the user, consists of the sought state v and a number k of matching sensors to find. $\mathcal{S}$ denotes the set of sensors and $\mathcal{C}$ the set of sensors which have already been contacted. Hence, in the beginning $\mathcal{C}$ is empty. In each iteration, the algorithm sorts the sensors from the set $\mathcal{S} \backslash \mathcal{C}$, this means all non-contacted sensors, by their probability to match the users' query using the probability tables in the BN. The sensor with the highest probability is then contacted to read its actual state. The sensor is added to the set of contacted sensors and if its actual state is the searched one, also to the solution set. At the end of the iteration the new evidence, i.e., the known value of the sensor, is inserted into the Bayesian Network and the network is updated. The iteration is repeated until enough matching sensors are found or all sensors have been contacted. The second case occurs if not enough matching sensors can be found in the network. Finally, the solution set with all matching sensors is returned to the user.

Algorithm 1 Query algorithm

```
Require: $v \in \mathcal{V}$ : State to search for
Require: $k$ : Number of matching entities to find
 1: $\mathcal{S}$ : Set of sensors
 2: $P(s,v)$ : Probability of sensor $s$ for state $v$
 3: $s(t)$ : State of sensor $s$ at time $t$
 4: $\mathcal{C} \leftarrow \emptyset$ {Set of sensors already contacted}
 5: $\mathcal{R} \leftarrow \emptyset$ {Set of fulfilling sensors}
 6: while $|\mathcal{R}| < k$ and $|\mathcal{C}| < |\mathcal{S}|$ do
 7:     $s_i \leftarrow \max_{\forall s \in \mathcal{S} \backslash \mathcal{C}} (P(s,v))$
 8:     if $s_i(t_{now}) == v$ then
 9:         $\mathcal{R} \leftarrow \mathcal{R} \cup s_i$
10:     end if
11:     $\mathcal{C} \leftarrow \mathcal{C} \cup s_i$
12:     Enter evidence and update Bayesian Network
13: end while
14: return $\mathcal{R}$
```

III. Evaluation

We evaluate our approach with a real-world dataset from Barcelona's Bicing bicycle rental system [3]. Each rental station contains a sensor to count the number of available bikes. We collected this data from 384 sensors over a period of six months. As Barcelona is located on the slope of a mountain, colocated stations tend to show similar usage patterns (see Fig. 1), as people typically ride down the hill but not the other way around. Hence, on top of the hill the rental stations are often empty or only a few bikes are available while in the valley there are a lot of bikes.

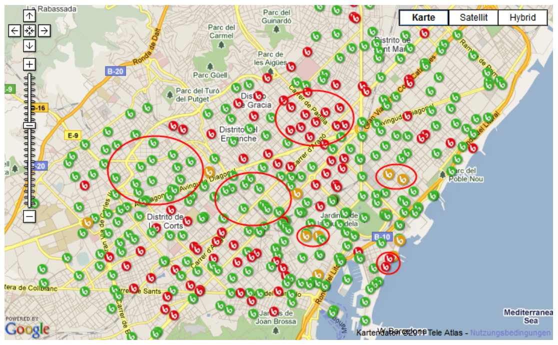

Fig. 1: Map of Barcelona's Bicing rental stations. Circles indicate sets of stations with highly correlated usage patterns.

A. Experiments

The search algorithm is implemented in Java and makes use of the SMILE reasoning engine developed by the Decision Systems Laboratory of the University of Pittsburgh. SMILE is a library for building and reasoning over probabilistic models such as Bayesian Network. The data logged from the website of the rental bike stations is stored in a SQL database for fast and easy access.

The method as described in Section II is compared with a naive approach where sensors are read out in random order until a given number of matches has been found. We repeatedly searched with different state $v \in \mathcal{V}$ for ten respectively one matching rental station, i.e., stations having a certain number of bikes left.

B. Results

The box plots in Figures 2 and 3 show the aggregated number of sensor readouts of 1000 runs. Each box plot shows minimum and maximum (the end of the "whiskers"). The box itself is representing the medial 50% of the data, i.e., the bottom of the box represents the 25th percentile respectively the lower quartile, the top the 75th percentile (upper quartile) and the middle-line the median. When searching for 10 matches (see Fig. 2), both our method and the baseline have to read all sensors in the worst case, but in the best and median cases, our approach is better by a factor of two. Our approach shows its real potential when only a very small number of matches is required. When only a single matching sensor is needed (see Fig. 3), our method is seven times better in the worst case and three times better in the median case – which is optimal (i.e., the first sensor that is read matches).

IV. Related Work

With Dyser [4] we presented a search engine for the IoT. In the system *sensors* construct prediction models from past sensors output. An *indexer* periodically indexes these prediction models. The user can issue a keyword-based query to the

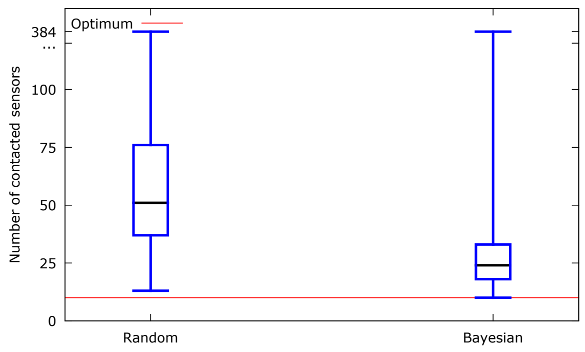

Fig. 2: Number of sensor readouts when searching for k = 10 matching sensors.

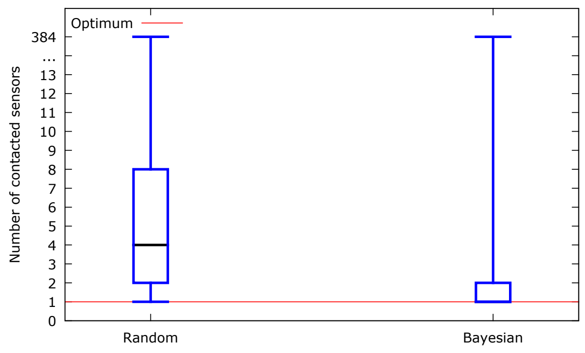

Fig. 3: Number of sensor readouts when searching for k = 1 matching sensors.

search engine which is segmented into a static and dynamic part. With the static part the *resolver* identifies potentially matching sensors. Afterwards the prediction models of these sensors are executed to obtain a ranked list of sensors which are most likely to match the dynamic part of the query. These sensors are then contacted in the order of their rank to read their current state until enough matching sensors are found. However, the prediction models used are based on the assumption of periodic output patterns of the observed sensors. In the present work we consider sensors which not necessarily output a periodic pattern.

There are other systems addressing search for dynamic content in the IoT such as [5], [6]. *Distributed image search* [5] gives the user the ability to specify an image and the system searches for camera sensor nodes which captured similar scenes. However, the system needs to contact all sensors, which will not scale in the growing IoT. *Objects Calling Home* [6] develops a search engine to locate lost objects. Although the system uses probabilities to search for the specified object, the goal is different from the idea of finding a subset of sensors having a certain state.

PRESTO [7] concentrates on providing predictions along with error bounds. The predictions are based on ARIMA time series models derived from continuous data and therefore assumes periodic output patterns.

ASAP [8] clusters the sensor network such that nodes with similar output are assigned to the same cluster. Out of each cluster, only a subset of nodes is reporting their values. The values of the other nodes are predicted by a model constructed based on temporal and spatial correlations inside the cluster. The difference to our appraoch is that ASAP is used for periodic reporting instead of search.

V. Conclusion

In this paper, we use correlations between sensors to improve the search for sensors that exhibit a given state at the time of the query. We achieve this by using prediction models based on Bayesian Networks to communicate only with sensors most likely matching the search criterion. Our results show that the number of sensor readouts needed to find the desired number of matching sensors is significantly lower than with the baseline method of random search.

VI. Future Work

In the future we want to overcome the limitations of Bayesian Networks by using other, more flexible, approaches to model the correlations between sensors. Additionally, we want to distribute the data collection and processing inside the network instead of using a centralized approach in order to achieve the scalability, that is needed to support a future Internet of Things.

Acknowledgment

This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01BK0905, GLab). The authors alone are responsible for the content of the paper.

The core of our implementation is based on the SMILE reasoning engine for graphical probabilistic model contributed to the community by the Decision Systems Laboratory, University of Pittsburgh (http://dsl.sis.pitt.edu).

References

[1] B. M. Elahi, K. Römer, B. Ostermaier, M. Fahrmair, and W. Kellerer, "Sensor ranking: A primitive for efficient content-based sensor search," in *IPSN 2009*. IEEE Computer Society, 2009.

[2] C. Yuan and M. J. Druzdzel, "An importance sampling algorithm based on evidence pre-propagation," in *In Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, 2003.

[3] Bicing. (2008) Bicing. [Online]. Available: http://www.bicing.cat

[4] B. Ostermaier, K. Römer, F. Mattern, M. Fahrmair, and W. Kellerer, "A real-time search engine for the web of things," in *Proceedings of Internet of Things 2010 International Conference (IoT 2010)*, 2010.

[5] T. Yan, D. Ganesan, and R. Manmatha, "Distributed image search in camera sensor networks," in *SenSys 2008*. ACM, 2008.

[6] C. Frank, P. Bolliger, F. Mattern, and W. Kellerer, "The sensor internet at work: Locating everyday items using mobile phones," *Pervasive and Mobile Computing*, vol. 4, no. 3, pp. 421–447, jun 2008.

[7] M. Li, D. Ganesan, and P. J. Shenoy, "Presto: feedback-driven data management in sensor networks," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1256–1269, 2009.

[8] B. Gedik, L. Liu, and P. S. Yu, "Asap: An adaptive sampling approach to data collection in sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, pp. 1766–1783, December 2007.