Wireless Netw (2011) 17:1833–1850
DOI 10.1007/s11276-011-0382-3

Metadata-guided evaluation of resource-constrained queries in content caching based wireless networks

Ruilin Liu · Xiuyuan Zheng · Hongbo Liu · Hui Wang · Yingying Chen

Published online: 28 August 2011

Abstract Recent years have witnessed the emergence of data-centric storage that provides energy-efficient data dissemination and organization in mobile wireless environments. However, limited resources of wireless devices bring unique challenges to data access and information sharing. To address these challenges, we introduce the concept of content caching networks, in which the collected data will be stored by its contents in a distributed manner, while the data in the network is cached for a certain period of time before it is sent to a centralized storage space for backup. Furthermore, we propose a metadata-guided query evaluation approach to achieve query efficiency in content caching networks. By this approach, each cache node will maintain the metadata that summarizes the data content on itself. Queries will be evaluated first on the metadata before on the cached data. By ensuring that queries will only be evaluated on relevant nodes, the metadata-guided query evaluation approach can dramatically improve the performance of query evaluation. We design efficient algorithms to construct metadata for both numerical and categorical data types. Our theoretical and empirical results both show that our metadata-guided approach can accelerate query evaluation significantly, while achieving the memory requirements on wireless devices.

Keywords Resource-constrained queries · Content caching networks · Wireless networks · Metadata · Efficient query evaluation

R. Liu · H. Wang (✉)
Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ 07030, USA
e-mail: hui.wang@stevens.edu

R. Liu
e-mail: rliu3@stevens.edu

X. Zheng · H. Liu · Y. Chen
Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA
e-mail: xzheng1@stevens.edu

H. Liu
e-mail: hliu3@stevens.edu

Y. Chen
e-mail: yingying.chen@stevens.edu

1 Introduction

With the advancement of wireless technologies, wireless devices are blended into our daily life and spend much time with us when we are working, attending meetings, participating in classes, or socializing. We anticipate that these advances will continue, leading to a world where continuous wireless connectivity will support the collection, storage and sharing of information—thereby driving pervasive computing applications. Further, the increasing sensing capability on wireless devices (e.g., smart phones and bluetooth devices) supports the data-centric nature of the collected information. In data-centric storage, the collected data is stored by attributes or types (e.g., geographic location and event type) at nodes within the network [1–3]. Queries for data with a particular attribute will be sent directly to the relevant node(s) instead of performing flooding throughout the network, therefore, data-centric approach enables efficient data dissemination/access.

However, comparing to centralized servers, wireless devices have limited storage and are energy-constrained. To ensure efficient data access and information sharing, we introduce the concept of *content caching networks*, where the collected data will be stored by its *content* in a

distributed manner, while the data in the network is *cached*, i.e., the data will be stored on each node of the network for a certain period of time before it is sent to a centralized storage space for backup. The advantage of using content caching networks is three-fold. First, storing the data by content enables efficient evaluation of queries commonly raised on the data content, for example, the queries for specific participants, events, and locations. Second, caching enables real-time query evaluation and eliminates the existence of centralized storage that may become a bottleneck for query evaluation or a single target for attacks. Third, by uploading data in a lazy fashion (i.e., once in a while), it avoids frequent data transfer from the wireless devices to the centralized storage, and consequently reduces massive battery power consumption and the communication overhead of the network.

A content caching network may consist of a large number of nodes, moving in and out of the area of interest. To support such large-scale and dynamic content caching networks, it is essential to achieve efficient query evaluation. Although there has been research in wireless sensor networks that are related to data-centric storage [1, 2, 4, 5], most of the work focus on *stable* network topology, assuming data dissemination in a predefined manner, thus are not applicable to mobile wireless environments. In this work, we propose to use *metadata* to guide query evaluation so that only the nodes whose data may contribute to query answers will evaluate the queries. Although it is popularly studied in the scope of statistical network (e.g., P2P network) [6], content caching in mobile network with metadata is novel. To the best of our knowledge, how to design the metadata from the collected content with respect to the resource limitation (e.g., memory space) is not studied by any existing content caching work.

Our approach of metadata-guided query evaluation can: (1) support efficient query evaluation by only returning relevant nodes containing requested data, and (2) provide rich expressiveness to describe various types of data in content caching networks. On the other hand, the introduction of metadata will add additional overhead on the already memory-limited wireless devices. To address this issue, we propose efficient data clustering and compression algorithms for metadata construction. In particular, to meet the memory requirement, we developed *Clustering, Balancing, and Compression (CBC)* algorithm for numerical data, and *Clustering, Expanding, and Compression (CEC)* algorithm for categorical data. The design of our algorithms aim to minimize the information loss incurred by data compression.

To evaluate the effectiveness and efficiency of our approach, we conduct simulation using trajectory data generated from a mobile wireless network deployed in a city environment in Germany. By examining three representative networks (10-node, 50-node, and 500-node), our results show that our metadata-guided query evaluation approach can dramatically improve query evaluation performance with low false positive rate and high precision of query answers.

Our previous work [7] initialized the research on metadata-guide query evaluation in content caching networks that contain numerical data only. In this paper, we extend it significantly by the following:

- We extend the metadata construction techniques to cover categorical data (Sect. 3.5). In particular, we design an algorithm called *Clustering, Expanding, and Compress (CEC)* to construct metadata for categorical data. For the clustering step, we propose three clustering approaches, *Top-Down (TD), Bottom-Up (BU)*, and *Hybrid (HB)*. For the expanding step, we propose two expanding approaches, namely *GreedyMax* and *GreedyMin*.
- By exploring the property that the updates in content caching based wireless networks are insertion-only, we design an efficient algorithm to deal with updates on the cached data for both numerical and categorical data types (Sect. 4). Our update algorithm suppports efficient updates on the metadata with minimal maintenance cost.
- We conduct an extensive set of experiments to evaluate the performance of our metadata construction approach for categorical data (Sect. 6.3). Our results prove the efficiency and effectiveness of the approach.
- Besides two networks with 10-node and 50-node, we setup a new network of 500 nodes, and evaluate the query performance on it (Sect. 6). Our results show that our metadata-guided query evaluation approach is scalable; the performance of our meta-data guided query evaluation is fast with large network sizes.
- We add a detailed discussion of the related work (Sect. 7).

The rest of the paper is organized as follows. In Sect. 3.1, we provide the background of XML. In Sect. 3, we present metadata design principle in content caching networks and develop our metadata construction algorithms for both numerical and categorical data. In Sect. 4, we discuss how to deal with metadata with presence of update on the collected data. We next provide an analysis of metadata-guided query evaluation in Sect. 5. We present the simulation evaluation of our approach in Sect. 6. In Sect. 7, we discuss recent work related to this paper. We conclude our work in Sect. 8.

2 Preliminaries

XML is an HTML-like language with an arbitrary number of user-defined tags [8]. An XML dataset is a collection of

```
<Data>
  <Region>
    <lx> x1 </lx>
    <ux> x2 </ux>
    <ly> y2 </ly>
    <uy> y1 </uy>
    <lt> t1 </lt>
    <ut> t2 </ut>
  </Region>
  <Region>
    ...
  </Region>
...
<Data>
```

Fig. 1 An example of XML metadata to represent trajectory data

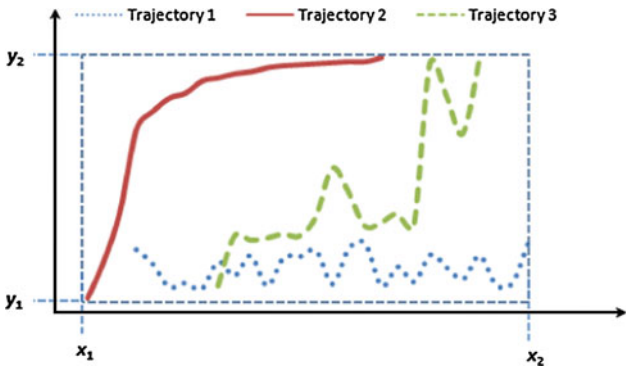


Fig. 2 Illustration of region in metadata versus trajectories in data

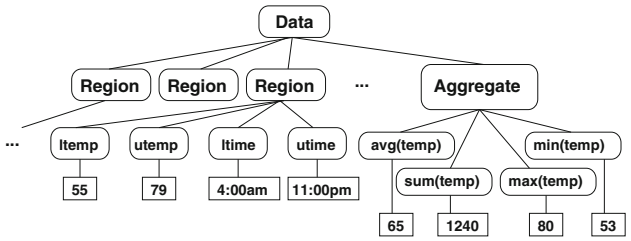


Fig. 3 Metadata with aggregate operators

data values marked with self-defined tags, which are used to describe the semantics of the data values. The flexibility of tag definition in XML makes it possible to built different semantic layers on top of data. Therefore, XML can be tailored to various categories of applications. Further, XML has been widely accepted and used as the standard for information integration and exchange on the Web. These advantages encourage us to use XML as the representation of the metadata in content caching networks. Figure 1 illustrates an example of XML metadata. It describes the spatial and temporal information of the trajectory data that the sensor node stores. In particular, the spatial information is described by the lx, ux, ly and uy elements, while the temporal information is described by the lt and ut elements. We note that the flexibility of XML makes it easy to extend to support various data types. We only use trajectory database as a running example in this paper to explain the basic idea of our approach. Our approach can be applied to other types of data, e.g., user information data.

Answers of XML queries are formalized by using matchings. Informally, a *matching* of a query to the XML metadata is a mapping between the query and the XML data such that both the structural constraints and the value-based constraints in the queries are preserved in XML data. For instance, an XML query "//Region[lx()=x_1]" consists of the structural constraint "//Region[lx]", which specifies the child element lx under the element *Region*, and the value-based constraint "lx()=x_1", which specifies the value of the element lx. By evaluating this query on the XML data in Fig. 1, it returns the first *Region* element.

In this study, we consider XPath [9], a node-selecting query language central to most core XML-related technologies. Recent work has shown that XPath queries can be evaluated in polynomial time [10]. In particular, XPath queries in general are evaluated with time complexity $O(|D|^4|Q|^2)$ and space complexity $O(|D|^2|Q|^2)$ [11]. For the core of XPath queries, which includes the logical and path processing features of XPath but excludes arithmetics and string operations, the time complexity is $O(|D||Q|)$ [12], i.e. linear in the size of the query and of the data.

3 Metadata construction algorithms

We assume that the schema of the metadata is pre-defined based on the prior knowledge of the data that is collected in the network. For instance, for the network that collects the trajectories of the mobile devices, assuming in the collected data each trajectory is of the format (x, y, t), where x and y denote the x- and y-coordinates at time point t. Then the collected data on the nodes is a set of trajectories $\mathcal{T} = \{T_i | T_i = ((x_{i_1}, y_{i_1}, t_{i_1}), \ldots, (x_{i_m}, y_{i_m}, t_{i_m}))\}$. Given such set of trajectories, an example of its metadata is shown in Fig. 1. Each *Region* element specifies the region, represented with the leftmost and the rightmost x-coordinates (the values x_1 and x_2) as well as the bottom and the top y-coordinates (the values y_1 and y_2), that covers all the trajectories, as well as the range of the timestamps in $\mathcal{T}$ (the values t_1 and t_2). Figure 2 illustrates the relationship between the region in metadata and the trajectories in the data.

Our metadata also supports several types of aggregate operators such as min, max, sum, and average on numerical data. Figure 3 shows an example of the metadata with aggregate operators on one node. In this figure, the *Region*

element is a summary node that holds the information of each region, including the lowerbound of temperature (stored in the "ltemp" sub-element) and the upperbound of timestamp (stored in the "utime" sub-element). There may have several *Region* elements in a metadata. Beside storing *Region* elements which contain temperature and time information, the metadata contains aggregate value of the data on the local node. For example, the maximum value of temperature 80°F of the local node is stored in the "max(temp)" sub-element.

In this section, first, we introduce the principle of designing metadata (Sect. 3.1). Second, we discuss the requirements (Sect. 3.2) and memory constraints (Sect. 3.3) of metadata construction. Third, we discuss the details of how to construct the metadata for both numerical data (Sect. 3.4) and categorical data (Sect. 3.5).

3.1 Metadata design principle

Sensors and their respective networks are becoming an essential source of information for planning, risk management and other scientific applications. These networks are composed of a large number of nodes, densely deployed within or very close to a phenomenon of interest. With aid of small, lightweight, and energy-efficient sensors, the network can be deployed with a spatial distribution that best fit the scientific requirements for gathering various kinds of data. To evaluate queries on such network, a naive way is to flood queries over the network. However, since it is possible that only a small portion of sensor nodes that contain the answers, evaluating queries on all sensors may incur large amounts of unnecessary query evaluation overhead. Then how to efficiently evaluate queries on large scale sensor networks (with regard to both number of nodes and the amount of data) becomes a challenge. In this paper, we propose to use metadata to guide query evaluation so that only the sensors whose data may contribute to query answers will evaluate the queries on their data.

To efficiently evaluate queries in large-scale and dynamic content caching networks, the metadata should have the following properties to support the data-centric approach:

- *Rich expressiveness*: Due to the diversity of data on content caching networks, the metadata should be able to describe data of various types.
- *Interoperability*: To integrate the data from heterogeneous content caching networks, the metadata should enable exchange of data among distributed heterogeneous wireless devices and with other kinds of information systems.
- *Efficient processing*: To help discover the wireless devices that have the required data, the metadata should support efficient query evaluation.

Unfortunately, most of the existing index mechanisms (e.g., ring-based index [4] and GHT [5]) cannot satisfy all the above requirements. Specifically speaking, both ring-based index and GHT do not satisfy either the rich expressiveness or the interoperability requirement. Thus, instead of the index technique, we propose to use metadata. In this study, we choose extensible markup language (XML) as the representation of metadata, due to its advantages of flexibility, self-description and support for information integration.

3.2 Requirements of metadata construction

In general, the XML metadata acts as the succinct and complete "summary" of the data. Given a dataset D and the metadata M of D, we say M *conforms* D if for every query Q, if $D(Q)$ (i.e., the answers of evaluating Q on D) is not empty, then $M(Q)$(i.e., the answers of evaluating Q on M) is not empty either. We say D is *relevant* to Q if $M(Q)$ returns non-empty answer. In other words, if there is no answer in $M(Q)$, then D definitely does not satisfy Q. Thus whether there is any answer in $M(Q)$ is the *necessary* condition of whether D has answers to Q. However, it is not the *sufficient* condition; the fact that $M(Q)$ returns non-empty answer does not imply that D must have the answer to Q too. When $M(Q)$ is non-empty but $D(Q)$ is empty, it incurs *false positive*. Thus given the data D, our goal of XML metadata design is two-fold:

- *Requirement 1*: the size of the metadata plus the size of D cannot exceed the available space on sensors, and
- *Requirement 2*: the metadata must conform to D with minimized possibility of false positives.

There exists trade-off between these two requirements; the metadata that is compressed too much (for example, describing all temperatures as a range $[-100, 100]$) will produce many false positives, while the metadata that is too detailed may be too large and exceed the available memory on wireless devices.

3.3 Memory constraints

Given the data D on a wireless device S, both the size of D and the total space N on S are fixed. Thus there exists an upper bound $U = N - |D|$ for the size of the metadata. Then Requirement 1 can be formalized as the following: given the data D and an upper bound U, how to construct the metadata M whose size is no larger than U?

To address this, we quantify the number of elements in the XML metadata. Let e be the average size of the elements in the XML metadata. Then the total size of the metadata is e^*k. Our goal is to make $e^*k \leq U$. Due to the fact that e is fixed, we can always fix the value

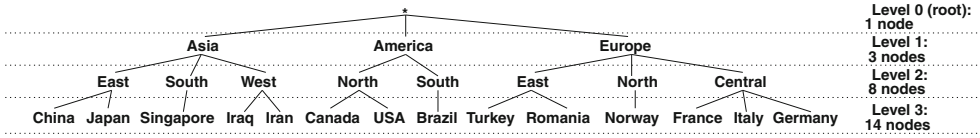


Fig. 4 An illustration of a taxonomy tree storing region and country information

$$k = U/e, \tag{1}$$

i.e., we can meet Requirement 1 by controlling the value of k, the number of XML elements in the metadata.

3.4 Constructing metadata for numerical data

We developed the *CBC* algorithm for metadata construction, which consists of three steps: *Clustering, Balancing*, and *Compression*. First, the data is clustered to k groups. Second, the data in each cluster is balanced by transferring. Third, each group is compressed to a single element in the XML metadata. For instance, all three trajectories in Fig. 2 are compressed to a single element as shown in Fig. 1. The detailed algorithm is presented below.

Step 1. Clustering: To address Requirement 2, we cluster the similar data together in the same group, so that the compressed metadata is as close to the original data D as possible, and the amount of false positives can be minimized. To cluster the data into k groups based on their similarity on trajectory, we use the k-means clustering analysis [13]. The k-means analysis takes the input parameter k, and partitions a set of n objects into k clusters so that the resulting intra-cluster similarity is high but the inter-cluster similarity is low. Using the trajectory data as an example, we define the distance between two trajectories $T_1((x_1, y_1, t_1), \ldots, (x_n, y_n, t_n))$ and $T_2((x'_1, y'_1, t_1, \ldots, (x'_n, y'_n, t_n))$ as the Euclidean distance:

$$Dist(T_1, T_2) = \sum_{i=1}^{n} \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}. \tag{2}$$

The k-means approach clusters the trajectories based on their distances. After clustering, we have k clusters, each containing trajectories that are close to each other. We must note that the distance function in the CBC algorithm can be replaced with other appropriate ones for various types of input data.

Step 2. Balancing: The resulting size of each cluster can be highly unbalanced. For example, some clusters may contain much more trajectories than others. To achieve a balanced compression of metadata, we compare the size of each cluster to the averaged size. If the number of elements in a cluster is smaller than the averaged number, ignore it; whereas if the number of elements is larger than the averaged number, we transfer the excessive number of elements to one or more other clusters by searching for the clusters that are of the least cluster distance to it and with the number of elements smaller than the averaged number. From Xiuyuan: After the clustering is done, since the trajectories in different clusters are highly unbalanced now, thus we need to balance every cluster to make it suitable for the metadata compression. What I do is to go over every cluster, if the number of trajectories are smaller than averaged number, ignore it; if number of trajectories are larger than averaged number, search for the cluster which has least cluster distance of it, and check if it is smaller or larger than average. If smaller, transfer some certain number of trajectories until make it up to the average. Do it recursively and finally we will obtain a balanced clustered trajectories within a node.

Step 3. Compression: In this step, each cluster is compressed into an element in the metadata. In particular, for each dimension (i.e., attribute) of the data in the cluster, let its data be $D = \{d_1, \ldots, d_n\}$. Then D is compressed as a range $[d_{min}, d_{max}]$, where d_{min} and d_{max} are the minimum and maximum value of D. For example, as shown in Fig. 1, the leftmost and rightmost x-coordinates are collected as the minimum and maximum value on the x-dimension of the trajectory data. Given the fact that all data values within the same cluster are similar, their generalized ranges must be close to the real values. Thus the possibility of false positive is minimized.

By the data-centric property of the network, all data of the same types/contents are collected on the same device. This enables that the updates on the data still fits the metadata, and thus will not cause much updates on the metadata.

3.5 Constructing metadata for categorical data

Categorical data is a common data type in the content caching networks. We assume that there exists a taxonomy tree that describes the semantics of categorical data. Typically the taxonomy tree is organized by supertype-subtype relationships. In such an inheritance relationship, the subtype presents the specialized data values that have the same properties, behaviors, and constraints as the generalized supertypes. Figure 4 shows an example of taxonomy tree for location data. Due to its tree structure, the taxonomy tree can be used to cluster the collected categorical data and construct the XML metadata.

We propose *Clustering, Expanding*, and *Compression (CEC)* algorithm to construct the metadata for categorical data. The CEC algorithm consists of three steps: *Clustering, Expanding*, and *Compression*. The basic idea is that, first, the cache data is clustered into $n \leq k$ clusters by following the taxonomy tree, where k is the maximum number of clusters allowed by the available memory valued by Equation 1 (Sect. 3.3). However, the number of the created clusters may be much less than k. Thus second, the n clusters are expanded to $n' > n$ clusters if it is necessary, with n' much closer to but still less than k. Third, the XML metadata is constructed by compressing each cluster to a single element in the metadata.

Before go to the details of the algorithm, we explain two basic operations, namely *node expansion* and *node merge*, that are used in the algorithm. The node expansion operation unfolds a parent node in the metadata tree to its children. The node merge operation is opposite; the children nodes are compressed to their parent node (Algorithm 1).

Algorithm 1 Metadata construction: CBC algorithm

Require: Clustered data piece;
Ensure: XML metadata;
1: $metadata \leftarrow \{\}$;
2: $i \leftarrow 0$;
3: **for all** $i <$ the size of *cluster* **do**
4: new *metanode*;
5: $j \leftarrow 0$;
6: **for all** $j <$ the size of $cluster_i.attr$ **do**
7: $k \leftarrow 0$
8: $metadata_j.max \leftarrow cluster_i.attr_j.k$;
9: $metadata_j.min \leftarrow cluster_i.attr_j.k$;
10: k++;
11: **for all** $k <$ the size of $cluster_i.attr_j$ **do**
12: **if** $attr_j.k > metadata_j.max$ **then**
13: $metadata_j.max \leftarrow cluster_i.attr_j.k$;
14: **else if** $attr_j.k < metadata_j.min$ **then**
15: $metadata_j.min \leftarrow cluster_i.attr_j.k$;
16: **end if**
17: **end for**
18: **end for**
19: $metadata \leftarrow metadata \cup metanode$;
20: **end for**
21: $i \leftarrow 0$;
22: create XML root;
23: **for all** $i <$ *the size of metadata* **do**
24: create XML metanode of $metadata_i$;
25: append XML metanode to XML root;
26: **end for**
27: Return XML metadata.

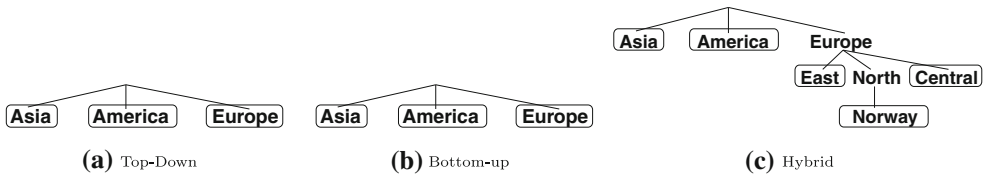
3.5.1 Step 1: Clustering

In this step, the categorical cache data will be clustered according to the given taxonomy tree. We propose three approaches, namely *top-down (TD), bottom-up (BU)*, and *hybrid (HB)* approaches, to cluster the categorical data. In particular, given k the maximum number of clusters allowed by the available memory valued by Eq. 1 (Sect. 3.3), both TD and BU approaches find a proper level of the taxonomy tree at which there are n nodes, where n is the maximum such number that is less than k, while the HB approach finds the nodes in the taxonomy tree that scattered at multiple levels of the taxonomy tree, so that the total number of nodes is the maximum such number that is less than k. To illustrate these three approaches, we use the taxonomy tree in Fig. 4. We only consider the categorical data on a single attribute. But our methods can be easily extended to multiple attributes.

Top-down (TD) Approach: Starting from the root of the taxonomy tree, the TD approach traverses the taxonomy tree in the breadth-first style, until it reaches a level h at which the total number of nodes at this level is greater than k. The level $h - 1$, i.e., the one 1 level higher at which locates the maximum number of nodes that is less than k, is returned as the result. For example, given the taxonomy tree in Fig. 4 and $k = 5$, the clustering result by the TD approach is shown in Fig. 5(a).

Bottom-up (BU) approach: opposite to the TD approach, the BU approach starts from the leaves of the taxonomy tree and traverses the taxonomy tree in a bottom-up fashion, until it reaches the level h at which the total number of nodes is less than k. The h is returned as the result. Using the taxonomy tree in Fig. 4 and $k = 5$ as an example again, the clustering result by the BU approach is shown in Fig. 5(b). Note that both TD and BU approaches produce the same result for expansion; however, they may incur different time overhead. In particular, when the located level is closer to the root of the taxonomy tree, the TD approach will be faster than the BU approach, and vice versa when the located level is closer to the bottom of the taxonomy tree. We have more empirical comparison details of these two approaches in Sect. 6.

Hybrid (HB) approach: Unlike the TD and BU approaches that traverse the taxonomy tree by levels, the HB approach traverses the tree by following the nodes that have the maximum number of children. In particular, starting from treating all leaves of the taxonomy tree as the candidates, the HB approach repeatedly picks the parent node N of the candidates from the taxonomy tree that has the maximum number of children, updates the candidate set by inserting node N and removing all children of N, until the size of candidate set is less than k. The nodes in the candidate sets are returned as the clustering result.



(a) Top-Down **(b)** Bottom-up **(c)** Hybrid

Fig. 5 EC algorithm: results of step 1 clustering ($k = 5$)

Algorithm 2 CEC algorithm: the HB clustering

Require: Taxonomy Tree T, The expected number of clusters k;
Ensure: The clustered nodes;
1: $CS \leftarrow$ all leaves of T;
2: **repeat**
3: $N \leftarrow$ the parent node of nodes in CS with the maximum number of children;
4: $CS \leftarrow CS$ -{all children nodes of N};
5: $CS \leftarrow CS \cup \{N\}$;
6: **until** $|CS| <= k$
7: Return CS.

Clearly, it is possible that the returned nodes are at different levels of the taxonomy tree. During the traversal, if there are more than one node that satisfy the maximum-children condition, we randomly pick one to break the tie. The pseudo code of the HB approach is shown in Algorithm 2.

Using the taxonomy tree in Fig. 4 and $k = 5$ as an example again, initially the candidate set contains all leaves of the taxonomy tree. In the first round, the *Central* node (of three children) is picked. The candidate set is updated as the *Central* node being inserted and its three children leave nodes being removed. The procedure repeats until the size of the candidate set shrinks to be no larger than k. The clustering result by the HB approach is shown in Fig. 5(c). The squared nodes (e.g., Norway and East) are the ones appear in the metadata.

3.5.2 Step 2: Expanding

It is possible that the total number of clustered nodes by Step 1 can be much less than the required k (i.e., the maximum number of clusters allowed by the available memory), especially for the TD and BU approaches. This will lead to high inaccuracy for later query evaluation by using the metadata. Therefore, we take Step 2 to further expand the clusters by Step 1, so that the total number of clusters will be much closer to k. To achieve this goal, we propose two approaches, namely *GreedyMin* and *GreedyMax*, to expand the clusters. Intuitively, the GreedyMin approach always picks the nodes of the minimum number of children to expand, while the GreedyMax approach always chooses the nodes of the maximum number of children for expansion.

(a) GreedyMin **(b)** GreedyMax

Fig. 6 CEC algorithm: results of step 2 expansion (expanding from the clusters in Fig. 5a); the squared nodes are the ones appeared in the metadata)

GreedyMin expanding: Given the cluster nodes created by Step 1 as the initial candidates, the GreedyMin approach repeatedly picks the node of the minimum number of children from the candidates to expand. If there are more than one such node, we randomly pick one to break the tie. The expanded cluster node will be replaced with its children nodes in the candidate set. The procedure repeats until the total number of cluster nodes is greater than k. The clusters of the previous step before termination will be returned as the result.

We use the clustered nodes by the TD approach (shown in Fig. 5(a)) to illustrate the GreedyMin approach. The initial candidates for expansion include the *Asia, America*, and the *Europe* nodes. Then the *America* node is chosen to be expanded to *North* and *South*, as it has the minimum number of children. We repeat the procedure until the number of total cluster nodes reaches the required constraint $k = 5$. The result of expansion is shown in Fig. 6(a).

GreedyMax expanding: the GreedyMax approach is similar to the GreedyMin approach. The only difference is that the GreedyMax approach picks the node of the minimum number of children every time for expansion. Using the cluster nodes of the TD approach (shown in Fig. 5a) again, the result of the GreedyMax expansion approach is shown in Fig. 6(b).

3.5.3 Step 3: Compression

After Step 1 & 2, we obtain the cluster nodes whose total number is very close to k. At last, in Step 3, we construct the XML metadata from these cluster nodes. In particular, similar to the CBC algorithm for the numerical data, each cluster node is compressed into an element of the XML

metadata. By such procedure, each piece of cache data can match one element in the metadata. The details of how to use the constructed metadata for query evaluation will be explained in Sect. 5.

4 Efficient metadata updates

In content caching networks, updates of the cached data are common. As the sensors keep collecting new data and storing them in the cache all the time, the only update operation on the cached data on each sensor is insertion. Clearly, insertion of new data into cache will lead to the updates on the metadata. The naive approach that constructs the metadata from scratch for the updated data will incur unnecessary overhead. Thus our goal is to design efficient and light-weight mechanism that maintain the metadata with the presence of data updates. Next, we discuss the details of such maintenance mechanism for both numerical and categorical data.

4.1 Update on numerical data

Intuitively, inserting new data values into metadata will incur the change of the ranges that the metadata nodes represent. For instance, merging an integer 100 into the range [70, 99] will change the range to be [70, 100]. First, we formally define the change on the range.

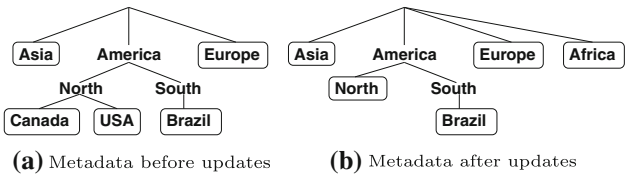
Definition 1 Given a node N in the numerical metadata M and a new numerical data value v, let $[l, u]$ be the range that N originally represents, and $[l', u']$ be the range that N represents after merging with v. Then the change c on the range on N by merging v equals

$$c = (u' - l') - (u - l). \tag{3}$$

Our update approach works as following. For each newly inserted data value v, first, we look for the node N in the metadata M that v falls into the range N representing. If there exists such a node, there is no update on M; v is merged into the node N. Otherwise, we pick the node N' on which the merge of v will incur the minimal change on the range that N' represents.

4.2 Update on categorical data

Similar to the updates on numerical data, for the newly inserted categorical data value v, we first look for the node N in the metadata M that v matches the category N representing. If there exists such a node, v is merged into N, and there is no update on M. However, if there is no such node, we look for the node N' in M that is of the closest supertype to v, and merge v into N'. For example, given the metadata in Fig. 6(a) and the new data value *Mexico* that will be inserted into the metadata, *Mexico* will be merged with the *North* node under the *America* node. If there is no such node N' existing in M, we insert a new node for v into M. In the worst case, the insertion may lead to a metadata whose size exceeds the limitation of memory. To address this, we will pick a set of nodes in the metadata who have the same parent in the taxonomy tree, and replace them with their parent node. To minimize the updates on the metadata, we pick the nodes whose common parent has the smallest number of children.



(a) Metadata before updates **(b)** Metadata after updates

Fig. 7 Updates of metadata: only squared nodes appear in the metadata

For example, consider the metadata shown in Fig. 7(a), assume $k = 5$. We also assume that in its corresponding taxonomy tree, there is an "Africa" element at the same level of "Asia", "America" and "Europe". If a new data value "Africa" is inserted into the data, a new node "Africa" will be inserted into metadata. Consequently there will be six nodes, which violates the memory constraint as $k = 5$. Therefore, we replace "Canada" and "USA" with their parent node "North", and have five nodes (i.e.,"Asia", "North", "Brazil", "Europe", and "Africa") in the metadata. The metadata after the update is shown in Fig. 7(b). The total update cost is 4, including deletion of "Canada" and "USA" nodes, and insertion of "North" and "Africa".

4.3 Discussion of update cost

We have the following theorem regarding the update cost for both types of data.

Theorem 1 *Given the metadata M,*

- *if M is of numerical type, any inserted data value v leads to at most one node in M to be updated;*
- *if M is of categorical type, any inserted data value v leads to at most $m + 2$ nodes in M to be updated, where m is the number of children of the node who has the smallest number of children.*

Proof For the newly inserted value v, there are two cases: (1) v can be fit into an existing node in the metadata, and (2) v does not match any node in the metadata. For Case (1), it is straightforward that there is no change on the metadata. For Case (2), we discuss how the metadata will

be updated for two types of data values. If v is of numerical type, v will be merged with an existing node n in the metadata whose range is the closest to v. Accordingly, the range of the node n will be changed after the merge of n. If v is of categorical type, the update of the metadata will be more complicated than the numerical case, since in the worst case, inserting a new node for value v may lead to a metadata whose size exceeds the limitation of memory. Then, we pick a node such that: (1) it is not a cluster node while all its children nodes are, and (2) the number of its children is miminum among all nodes which satisfy (1). We replace all its children node with this node. Therefore, we insert 2 nodes and removed m nodes, where m is the number of leave nodes whose parent has the smallest (leave) children. The update cost $m + 2$ then follows. □

Our merge approach ensures the minimal update on the metadata.

5 Query evaluation analysis

In this section, we explain the query evaluation procedure by using metadata, and theoretically analyze the effectiveness of the metadata.

5.1 Query types

In this paper, we focus on three types of queries:

- *Point-based query*. This type of queries asks for data that contains a specific value. E.g., a query returns all locations that have data at 4:00pm.
- *Range-based query*. This type of queries asks for data whose value meets a given range. E.g, a query returns all nodes whose observed temperature is between 60 and 65°F.
- *Aggregate query*. This type of queries asks for aggregate results. We support four kinds of aggregate operators: min, max, sum, and average. E.g, a query returns the average temperature of the day.

5.2 Metadata-guided query evaluation

For point-based queries and range-based queries, the purpose of using metadata is to discover the nodes S that contain the data relevant to queries in content caching networks, so that the queries will not be evaluated on all devices, but only on S. To reach this goal, our query evaluation procedure consists of two steps.

Step 1: Relevance check on metadata. By this step, the input query Q is translated to the query Q_M on XML metadata. The translation is straightforward; the conditions are defined as the value-based constraints in the queries, with the corresponding structural constraints from the schema. For example, for Q that looks for the trajectories containing the coordinates (x, y), it is translated as Q_M: $//Region[lx() \leq x \ \ AND \ \ ux() \geq x \ \ AND \ \ ly() \leq y \ \ AND \ \ uy() \geq y$, which looks for the region that (x, y) falls into. Then we evaluate the query Q_M on the metadata M.

Step 2: Query evaluation on data. For those nodes whose metadata returns non-empty answer to Q_M, the input keyword query Q is evaluated on their data D. Since the metadata is only a synopsis, it is possible that the returned answers contain false positives. However, since we respect the similarity of data when we construct the metadata, our approach minimizes the possibility of false positives.

The evaluation of aggregate queries also follows the 2-step procedure described above. First, the value-based constraints of the aggregation in the queries will be evaluated against the aggregate operators in the metadata. Only the nodes whose metadata matches the aggregate constraints in the queries will be considered as relevant for further query evaluation. For example, for a query that asks for all nodes whose average temperature is greater than 70°F, the node whose metadata is shown in Fig. 3 will be considered as not relevant since its average temperature is only 65°F; it will not be eligible for further query evaluation.

5.3 Analysis of efficiency

To analyze the effectiveness of using metadata for query evaluation, we consider two scenarios, no metadata is present and the metadata is available, and compare the query evaluation performance in these two scenarios.

When there is no metadata, the query Q will be propagated to the whole network for evaluation. Let n be the total number of sensors in the network, D_i $(1 \leq i \leq n)$ be the size of the data on the i-th sensor node and $t(D_i)$ be the time that Q is evaluated on D_i. The total time of evaluating Q without any metadata is $T_1 = \sum_{i=1}^{n} t(D_i)$.

When there is metadata, the query Q will be first translated to Q_M. Then Q_M will be propagated to the whole network for evaluation. Only the sensors whose metadata satisfies Q_M will evaluate Q on their data to return the final answers. Let m be the number of sensors whose metadata return non-empty answer for Q_M, M_i $(1 \leq i \leq n)$ be the size of the metadata on the i-th sensor node, and $t(M_i)$ be the time that Q_M is evaluated on M_i. The total time of evaluating Q with the presence of metadata is $T_2 = \sum_{i=1}^{n} t(M_i) + \sum_{i=1}^{m} t(D_i)$.

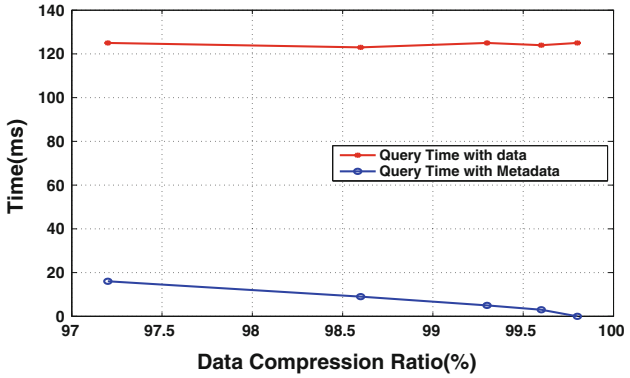


Fig. 8 Comparison of query time on metadata and data itself

It is straightforward that Q always can be evaluated by scanning D_i once. As shown in Fig. 8, our experiments confirm that $t(M_i)$ is always dominated by $t(D_i)$. Therefore, we approximate $t(M_i)$ as a portion of $t(D_i)$, i.e., $t(M_i) = k_i * t(D_i)(k_i < 1)$. We assume the network is of balanced load so that the data on different sensor nodes is of roughly the same size. As the result, all k_is $(1 \leq i \leq n)$ are of similar value. This also applies to $t(D_i)$. Thus we can approximate the ratio of T_2 over T_1 as:

$$T_2/T_1 = k + m/n. \tag{4}$$

This result applies to both aggregate and un-aggregate queries. For the worst case that $m \approx n$, T_2/T_1 is always greater than 1. In other words, for those queries whose answers locate on a significant portion of sensor nodes in the networks, evaluating them directly on the data is more efficient than using metadata. However, in practice, most of query answers are stored on only a small number of sensor nodes, i.e., m/n is negligible. Therefore, $T_2/T_1 \approx k$, which is always less than 1. Therefore, in theory, we show that using metadata can achieve more efficient query evaluation than without using metadata.

6 Experimental evaluation

In this section, we first describe our metrics, and then present experimental methodology followed by the results that evaluate the effectiveness of our approach.

6.1 Metrics

We utilize the following metrics to evaluate the performance of our metadata-guided approach.

False positive rate and precision. For each query, we define the *false positive* as when the XML metadata on a node returns true, but the subsequent search on the data stored on this node returns zero tuples. Correspondingly, the *true positive* is defined as when the XML metadata on a node returns true, the subsequent data search returns at least one tuple that match the query.

We present false positive in two ways. First is the *false positive rate*, which is the percentage of nodes that results in false positive among those nodes return zero tuples in the network in a batch test. Second, we measure *precision*, the percentage of nodes return true positive out of those nodes, which return true from metadata in a batch test. In our study, false positive rate is an important measure of the information loss due to the data compression of the metadata construction (from Xiuyuan). In our experiment, we measure false positive in two ways. First, we show the rate of false positive as the percentage of nodes are false positives. Second, we measure the precision as the percentage of returned relevant by metadata are true positives.

Compression ratio. Since each node is memory-constrained, the size of the XML metadata comparing to the size of the original data stored on the node is an important factor to evaluate our approach. For numerical data, we define the *Compression Ratio* (CR) as

$$\text{CR} = 1 - \frac{S_{metadata}}{S_{data}}. \tag{5}$$

This formula computes the percentage of the data that has been compressed to the XML metadata.

Resource satisfaction factor. When constructing metadata of categorical data, it is not easy to create exactly k clusters, where k is calculated from the memory requirement. Indeed, our algorithm produces the metadata that consists of n clusters, where n is the closest number to k that is allowed by the taxonomy tree. Thus, to measure how well the available memory has been used for the constructed XML metadata, we define the *Resource Satisfaction Factor* (RSF) as following:

$$\text{RSF} = 1 - \frac{n}{k}. \tag{6}$$

where k is the expected number of clusters calculated from the memory requirement (Formula 1 in Sect. 3.3), and n is the number of clusters constructed by our algorithm. Ideally, the closer RSF is to 0, the better that the memory has been used.

Query time. We measure the efficiency of query evaluation by time. When without metadata, the data stored on all nodes must be accessed. Thus, the query time is equivalent to the total time of evaluating a query on the data of each node in the network. When under the scenario with XML metadata, the query time consists of two parts: the time of evaluating a query on the metadata of each node in the network and the time of evaluating the query on data stored in relevant nodes.

Fig. 9 The experimental data sets are generated based on the city and its vicinity in Germany

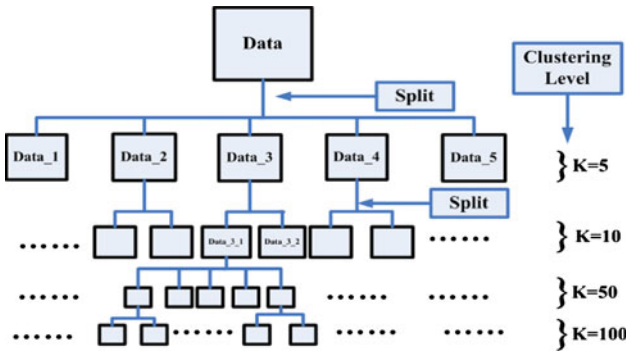


Fig. 10 CBC algorithm: *k*-means clustering is performed based on the data containment relationship

6.2 Evaluating effectiveness for numerical data

6.2.1 Methodology

Data generation. We evaluate the feasibility of our approach using the trajectory data in mobile wireless networks. We note that our approach is generic and can be applied to other types of data as well. We conducted experiments based on mobile nodes generated from a city environment and its vicinity in Germany [14, 15] as shown in Fig. 9. The size of the area is 25,000 m × 25,000 m and the mobile nodes are moving along the roads in the city randomly at the walking speed (3 feet/s).

Each node collected its own trajectory data locally while it is moving. We investigated three networks of 10 nodes, 50 nodes, and 500 nodes. To simulate a typical memory-constrained wireless node, we assume the memory on each mobile node is 500KB, similar to a sensor node [16]. In practice, the memory on a mobile node can be larger. Thus, each node stores around $S_{data} = 500$ KB trajectory data, which includes 330 trajectories on average. In our study, each trajectory lasts for 50 time points.

Metadata construction. We apply the CBC algorithm to construct metadata using various k values ($k \in \{5, 10, 50, 100\}$) during k-means clustering. The smaller the k, the more the data is compressed to metadata. The resulting metadata size is $S_{metadata} \in \{1, 2, 7, 14\}$ *KB*. Our data compression is based on the data containment relationship for each level of k as shown in Fig. 10.

Query evaluation. We perform query evaluation under the scenarios with or without XML metadata on a PC with a 2 GHz Intel Core 2 CPU and 2 GB RAM. When with XML metadata, our evaluation program accesses the XML metadata first. If the metadata returns true, which means there exists a possible answer in the current node, we then read the data stored in this node and search for the query result. Under the scenario without XML metadata, the evaluation program just simply scans all the data stored in every node in the network to answer the query. We run batches of queries with each batch involving different number of nodes. We call the involved nodes in each query *hit node*. Our results are the average of 20 times for each batch test.

6.2.2 Results

False positive rate and precision. Figure 11 presents the false positive rate and precision of metadata-guided queries under various compression ratio for the networks of 10, 50, and 500 nodes, respectively. The key observation is that the false positive rate is low and stable, below 6%, under high compression ratio, above 97%. Further, the false positive rate only increases slightly from 4 to 6% when the compression ratio increases from 97 to 99.8%. This is very encouraging as the low false positive rate indicates that the information loss due to the metadata construction is small. On the other hand, we observed high precision, above 90%, under high compression ratios. This indicates that the introduction of metadata only has small impact on the precision when querying data, which may be ignored.

Further, we observed similar trends in all the simulated networks, including 10, 50, and 500 nodes, indicating that our approach is not sensitive to the network size. Due to the space limitation, we will only present the results from the 10-node network in the rest of the paper.

Query efficiency. Figure 12 shows the comparison of query time without metadata to that with metadata guidance under various data compression ratios in the 10-node network. The metadata-guided query time is presented with different percentage of hit nodes in the network. We observed that queries without metadata guidance always takes more time than those with metadata guidance. Further, we found that the higher compression ratio corresponds to the less query evaluation time, i.e., there is a slight decreasing trend of the query time when the

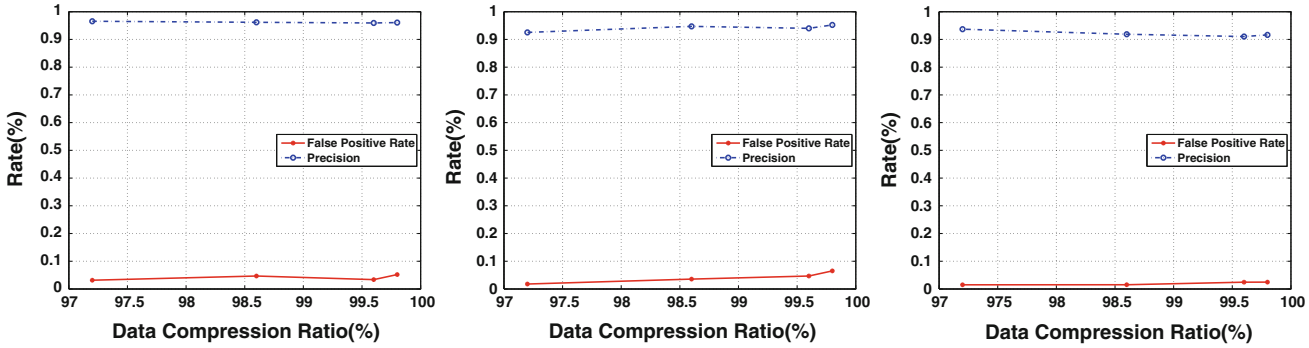


Fig. 11 False positive rate and precision under metadata-guided query

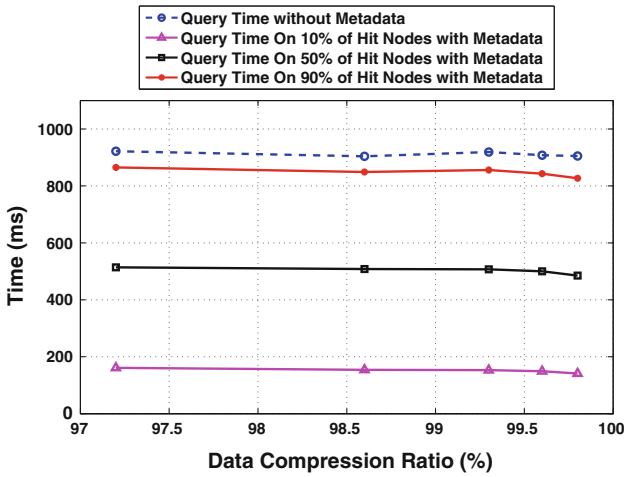


Fig. 12 Query time comparison under various compression ratio of metadata (10-node network)

compression ratio increases. This is because higher compression ratio results in smaller size of the metadata, which needs less time for query evaluation.

Moreover, when using metadata guidance the less percentage of hit nodes, the more efficient our approach is. This observation is further confirmed in Fig. 13, which presents the query evaluation time versus the percentage of hit nodes in the network with and without metadata guidance. The query evaluation time without metadata guidance is almost a constant, above 900ms. Under metadata guidance, when the percentage of hit nodes is 10%, the query time is less than 200 ms, only 22% of that without metadata guidance. The query time increases as the increasing percentage of hit nodes. This is because in the metadata-guided approach, only those nodes contain the relevant data will be searched for query results, whereas without metadata guidance, the data on all nodes in the network will be evaluated.

Additionally, we observed that when the percentage of hit nodes reaches 100%, the metadata-guided query evaluation time exceeds that without metadata guidance by about 30 ms. This is the overhead introduced by the metadata-guided approach, which is only about 3.3% of the query evaluation time on data. Thus, our metadata-guided approach is highly efficient during query evaluation. We believe that with the large-scale data, the performance optimization by our approach metadata will be more significant.

6.3 Evaluating effectiveness for categorical data

6.3.1 Methodology

Data generation. To evaluate the effectiveness of our CEC algorithm, we use a synthetic temperature range dataset. The ranges are stored as strings. We then generate a taxonomy tree to represent the hierarchical relationships between the ranges. The tree has six levels and the number of nodes on level 1 to level 6 are 1, 3, 8, 35, 97, 329. We use the taxonomy tree to guide the clustering of categorical data.

We investigated three networks in different sizes as we did on numerical data (10-, 50-, and 500-node). To simulate a typical memory-constrained wireless node, we assume the memory on each mobile node is 500 KB, similar to a sensor node [16].

Metadata construction. We construct the metadata of the categorical data by using various k values ($k \in \{5, 10, 50, 100\}$). Similar to the result on numerical data, the smaller the k is, the more the data is compressed to metadata. The resulting metadata size is $S_{metadata} \in \{1, 1, 3, 6\}\, KB$.

Query evaluation. We evaluated queries under the scenarios with and without XML metadata on a PC with a 2.4 GHz Intel Core 2 CPU and 4 GB RAM. When we evaluating queries with XML metadata, the query evaluation program will access the XML metadata first. Similar to the query evaluation on metadata for numerical data, a

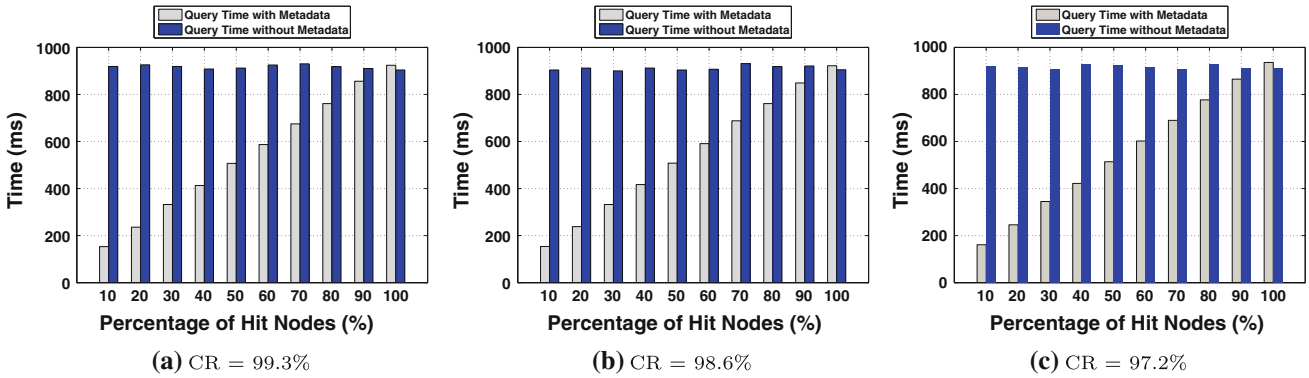


Fig. 13 Query time comparison under different percentage of hit nodes (10-node network)

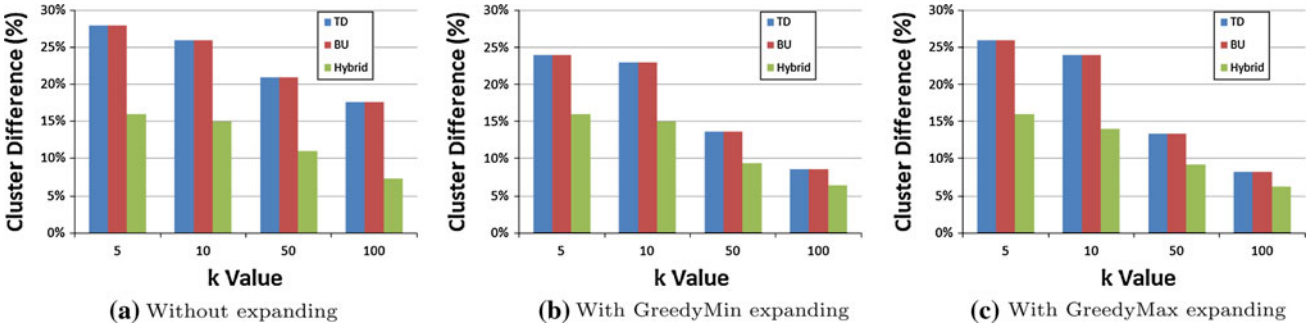


Fig. 14 Resource satisfaction factor (*RSF*) comparison of three approaches (50-node network)

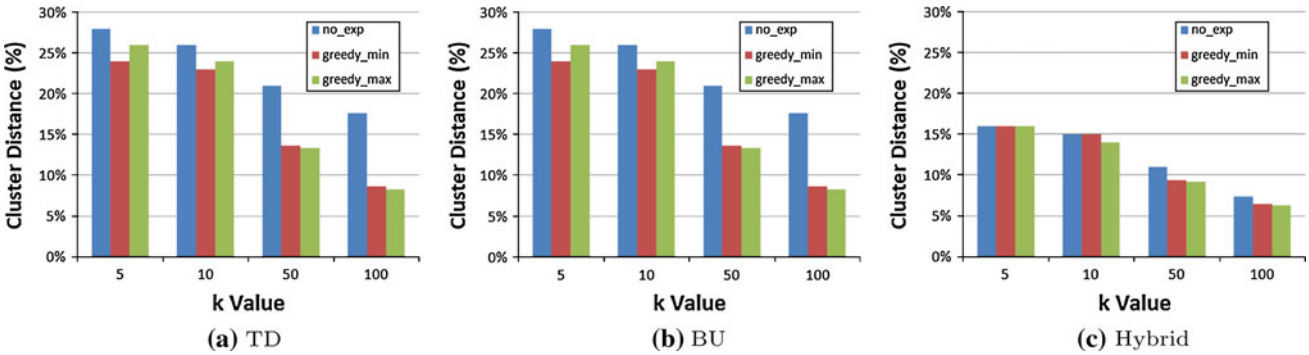


Fig. 15 Resource satisfaction factor (*RSF*) comparison of expanding methods (50-node network)

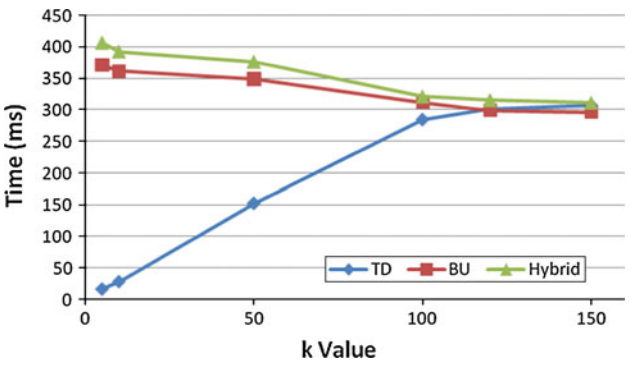


Fig. 16 Metadata construction time comparison (50-node network)

query evaluation will be only performed on the database on the "hit nodes". When without XML metadata, the process is simply performing query evaluation on every node in the network. Our experiment results are the average of 20 times running the test.

6.3.2 *Results*

Resource satisfaction factor. To evaluate the effectiveness of various clustering approaches, we compare the resource satisfaction factor (RSF) of our top-down (TD), bottom-up (BU), and hybrid (HB) approaches. The results are shown in Fig. 14. First, we observe that larger k (i.e., the

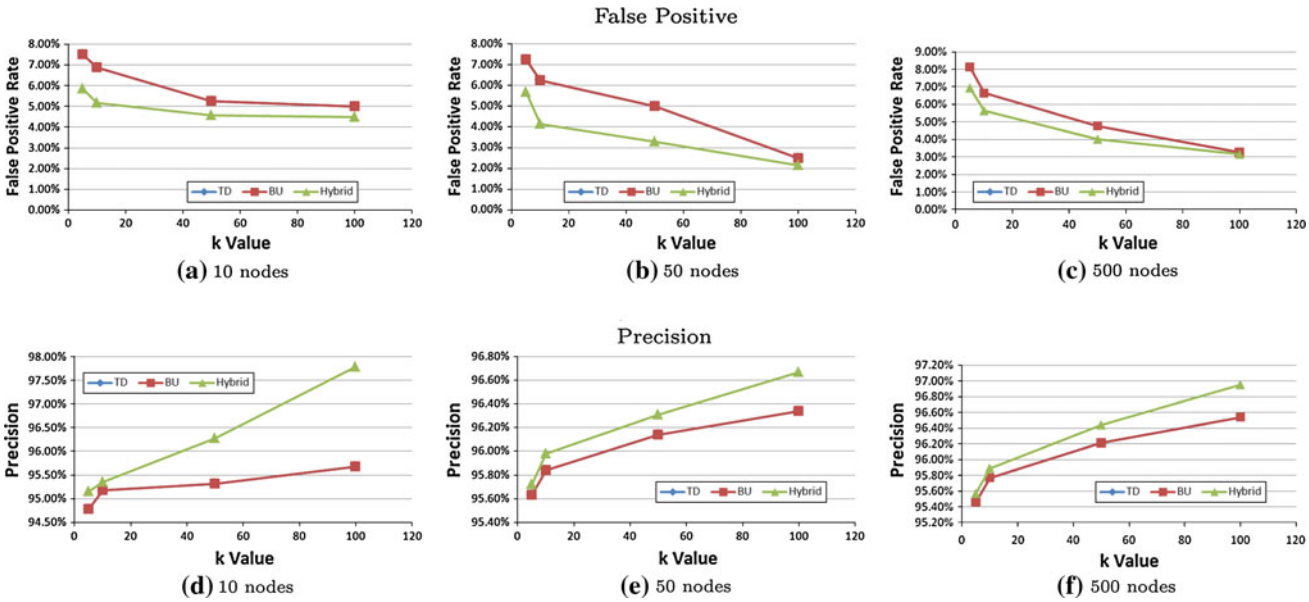


Fig. 17 False positive ratio and precision comparison of three approaches

maximum number of clusters allowed by the memory constraints) results in smaller RSF. This is because with larger k, the number n of generated clusters by our clustering approaches has higher probability to get closer to k. Second, we observe that the hybrid approach always produces the best RSF, since its expansion allows clustered nodes locating at different levels of the taxonomy tree, which may achieve larger number of clusters than that of TD and BU approaches. Third, TD and BU always achieve the same RSF, as they always produce the same clustering result. Finally, Fig. 14(b), (c) show that our two expanding methods can improve the RSF.

We also compare RSFs of our two expanding methods. Figure 15 shows that both GreedyMin and GreedyMax expanding methods improve RSFs; the improvement of RSFs is more significant for larger k value. However, there is no absolute winner out of these two approaches regarding RSFs.

Performance. Next, we evaluated the performance of our CEC algorithm by measuring the construction time of metadata. Since the expanding time is negligible, here we only show the time performance of the three clustering approaches in Fig. 16. The results show that, first, although the TD and BU approaches produce the same metadata, their time performance vary. When k, the upperbound of the number of clusters, is less than 100, TD is the fastest out of the three approaches. This is because for small k values, the TD approach traverses the fewest levels of the taxonomy tree by starting from the root of the tree. When k is larger, the performance of TD increases dramatically, while both BU and hybrid approaches spend much less time since they can terminate in traversing fewer tree levels. When k is larger than 120, TD is the slowest out of the three approaches, since the total number of nodes that it traversed exceeds that of BU. Furthermore, albeit the hybrid approach traverses similar number of nodes as that by BU, it iterates more times, thus its time performance is slightly worse than that of the BU approach.

False positive rate and precision. We use false positive rate and precision to demonstrate the effectiveness of XML metadata on categorical data on 10-, 50- and 500-node network as we did for numerical data.

First, we measure the false positive rate and precision on our three clustering methods. The results are shown in Fig. 17. First, the false positive rate of the three methods is always less than 9%, which proves the effectiveness of our clustering methods. Second, the false positive rate and the precision of both the TD and BU approaches are the same, as they produce the same clusters. Meanwhile, the hybrid approach always achieves the best false positive rate and precision. This proves that the more clusters that are constructed, the smaller false positive rate is. Third, Fig. 17(a)–(c) show that the false positive decreases with larger k. Similarly, from Fig. 17(d)–(f) , we observe that the precision is increasing with k. This is because larger k results in larger n, the number of produced clusters, which can increase the precision.

We also investigated the improvement of false positive rate and precision by applying our two expanding methods. Figure 18 shows that both GreedyMax and GreedyMin approaches can improve the false positive rate and precision. However, whether the GreedyMax method or the

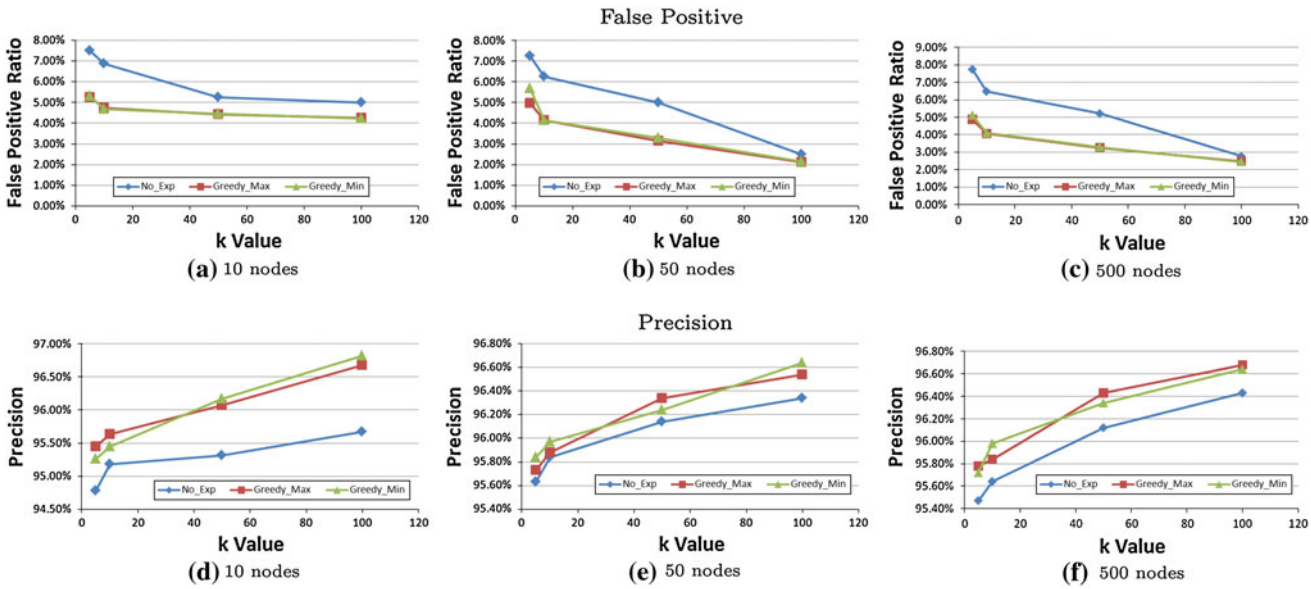


Fig. 18 False positive ratio and precision with expanding

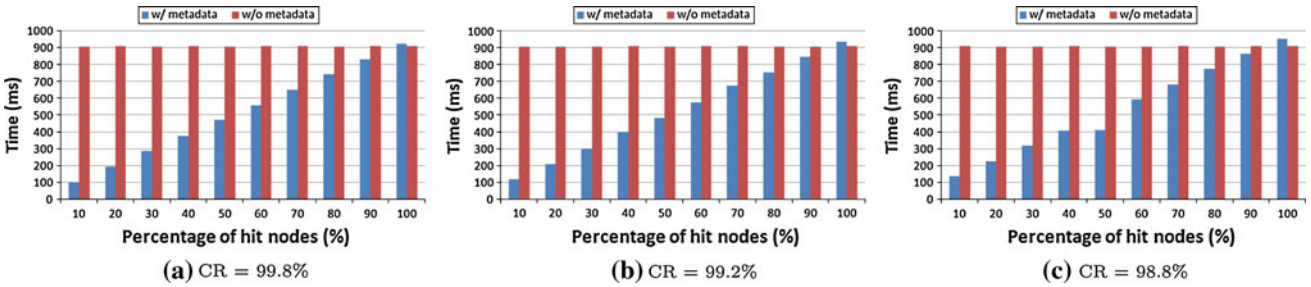


Fig. 19 Query time comparison under different percentage of hit nodes (10-node network)

GreedyMin method wins regarding the precision depends on the value of k and the dataset. For example, Fig. 18(d) shows that GreedyMax wins when k is equal to 10, while loses when k is equal to 50.

Query efficiency. Figure 19 shows the comparison of query time without metadata to that with metadata guidance under various data compression ratios on categorical data. The query evaluation time without metadata guidance is almost a constant, which is around 909ms. The query evaluation time with assistance of metadata is increasing alone with the increasing of percentage of hit nodes. From this figure, we can observe similar phenomenon that the metadata-guided query time is always faster than the approach that without help of metadata while not every node has answer. Furthermore, we found that the higher compression ratio results in less query evaluation time, because higher compression ratio on metadata results in smaller size of the XML metadata, which needs less time for query evaluation. The optimization by our method is more significant for smaller hit rate. On the other hand, our approach still can reach at least 10% saving even when the hit percentage reaches as large as 90%. That proves the effectiveness of our method.

7 Related work

7.1 Metadata in networks

With the rapid development of the Internet, metadata initially proposed to solve the problems of managing, searching, accessing, retrieving, sharing, and tracking complex resources in various formats. However, metadata has some disadvantages includes maintainence cost, unreliability, lack of authentication, and lack of interoperability with respect to syntax. In recent years, few work has been done on metadata research to overcome these disadvantages and extend its application. Extensible Markup Language (XML) and its associated technologies [8, 17] make it possible to exchange data in a standard structured format. [18] provides a powerful query language for XML documents. The Open Archives Initiative (OAI) [19] provides a

protocol to build a framework for archives that using metadata to provide "value-added services". Our work is an attempt of innovative use of metadata for optimizing query evaluation in wireless networks.

7.2 XML-based data dissemination in wireless networks

There has been some recent work on applying XML technique in wireless networks. SensorML [20] was proposed to handle data disseminate for satellite and other sensor networks. [21] uses an XML-based markup language to describe the data from various types of sensor in order to make sensor networks more accessible to non-professional user and make inter-network communication more easier. However, none of these consider use XML techniques to create matadata for accelerating query evaluation performance.

7.3 Resource-constrained query processing in networks

There has been some recent work on query processing in sensor networks with focus on the dissemination of selection and aggregation queries in the network to reduce power consumption [22–24]. They assume that data is named using attribute-value pairs, which does not stand in our project. [5] proposes a data-centric data dissemination scheme, in which data dissemination is handled in a predefined manner regardless of queries. Hence, it may introduce many unnecessary data transfers when the querying rate is low. [4] proposes an index-based data dissemination scheme, based on the idea that data are collected and stored at the nodes close to the detecting nodes. The location information of these storing nodes is recorded on some index nodes. However, we consider a fully distributed framework that does not include such index nodes. [25] presents the Tiny AGgregation (TAG) service for aggregation in low-power, distributed, wireless environments. [24] proposes an Acquisitional query processing (ACQP) which provides a framework for addressing issues of when, where, and how often data is sampled and which data is delivered in distributed, embedded sensing environments. [26] introduce aggregation query into wireless network to build energy-efficient aggregate query evaluation algorithm based on the node caoability. Diffusion [27] is the first description of the software architecture that supports named data and in-network processing in an operational, multi-application sensor-network. When combined with dense deployment of nodes, this kind of named data enables in-network processing for data aggregation, collaborative signal processing, and similar problems. These approaches are essential for emerging applications such as sensor networks where resources such as bandwidth and energy are limited. [28] presents a system called DIMENSIONS that constructs multi-resolution summaries and progressively ages them to lower the communication overhead and provide long-period storage capability. [29] presents a sensor network query processing architecture that can lower the in-network processing costs and optimize query evaluation. However, all of them only consider aggregate queries which only return a "summary" the data. Our algorithm can support all kinds of queries.

7.4 Distributed data storage

Data centric storage (DCS). Several work [30–32] have been done on data centric storage. Data-centric storage was first explicitly proposed in [30]. The data-centric storage based on the GPSR routing algorithm and an efficient peer-to-peer lookup system. [31] makes data-centric storage more practical by removing the need for point-to-point routing. [32] uses a Geographic Hash Table (GHT) system for DCS on sensor-nets. Our work is built on DCS model.

Other systems. On the other hand, with high-speed networks and powerful computer nodes, distributed data storage can provide high availability, capability to extending, efficiency. [33] presents a prototype of a highly available scalable and distributed data storage structures in high-speed networks that connect powerful computers. Bigtable [34] is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Cougar [35] uses database technology to control and access mobile devices in large scale network. They proposed a concept of a device database system where individual devices are modeled as database objects, which allows to access collections of devices with declarative queries.

8 Conclusion

In this work, we introduced content caching networks to support data-centric storage for pervasive computing applications in mobile wireless environments. The content caching networks use the metadata-guided query evaluation approach to achieve efficient data dissemination/access. The metadata-guided approach has the characteristic of rich expressiveness that can describe various types of data and capture the diversity of data from heterogeneous wireless devices. Therefore, using metadata in content caching networks can integrate the data from heterogeneous nodes and enable exchange of data among distributed heterogeneous wireless devices. We studied how to construct metadata for two types of data, numerical data and categorical data. For numerical data, we developed the CBC algorithm, which is

based on data clustering and compression during metadata construction. For categorical data, we developed the CEC algorithm which consists of three interchangeable clustering approaches (top-down, bottom-up, and hybrid) and two expanding approaches (GreedyMax and GreedyMin) to construct metadata. Both CBC algorithm and CEC algorithm can minimize the possible information loss due to data compression, while meeting the memory requirements on wireless devices. Both of our theoretical analysis and empirical results using data generated from a city environment show that the metadata-guided approach is effective in achieving efficient data query and only incurs small overhead, thereby providing strong evidence of the feasibility of content caching networks. Additional, we studied how to efficiently update XML metadata after data has been updated. All the effort built a complete framework to accelerate query evaluation in content caching networks using XML metadata.

For the future work, we will consider other types of resource constraints, e.g., energy consumption. We will investigate how to design metadata to meet these resource constraints. We will also extend our metadata design mechanism to include the semantics of the cached data.

References

1. Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., & Estrin, D. (2003). Data-centric storage in sensornets. *ACM SIGCOMM Computer Communication Review archive, 33*.
2. Ghose, A., Grossklags, J., & Chuang, J. (2003). Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proceedings of the 4th international conference on mobile data management*, pp. 45–62.
3. Shao, M., Zhu, S., Zhang, W., & Cao, G. (2007). pDCS: Security and privacy support for data-centric sensor networks. In *Proceedings of the IEEE international conference on computer communications (INFOCOM)*.
4. Zhang, W., Cao, G., & Porta, T. L. (2003). Data dissemination with ring-base index for wireless sensor networks. In *IEEE international conference on network protocols (ICNP)*.
5. RatNasamy, S., karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R. & Shenker, S. (2002). GHT: A geographic hash table for data-centric storage. In *ACM international workshop on wireless sensor networks and applications*.
6. Li, J., Chou, P., & Zhang, C. (2004). *Mutualcast: An efficient mechanism for content distribution in a peer-to-peer (p2p) network*. Microsoft Research TechReport (MSR-TR-2004-100), 100.
7. Wang, H., Liu, R., Zheng, X., Chen, Y., & Liu, H. (2009). To do or not to do: metadata-guided query evaluation in content caching networks. In *Proceedings of the 28th IEEE conference on global telecommunications*, pp. 4524–4529.
8. W3C, *Extensihle markup language (xml)*. http://www.w3.org/XM.
9. W3C, *Xpath 2.0*, http://www.w3.org/TR/xpat.
10. Gottlob, G., Koch, C., & Pichler, R. (2003). The complexity of xpath query evaluation. In *Proceedings of the ACM international conference on principles of database systems (PODS)*, pp. 179–190.
11. Gottlob, G., Koch, C., & Pichler, R. (2003). Xpath query evaluation: Improving time and space efficiency. In *Proceedings of the 19th IEEE international conference on data engineering (ICDE)*, pp. 379–390.
12. Gottlob, G., Koch, C., & Pichler, R. (2002). Efficient algorithms for processing xpath queries. In *Proceedings of the 28th international conference on very large data bases (VLDB)*.
13. Lloyd, S. (1982). Least squares quantization in pcm. In *IEEE transactions on information theory*, pp. 128–137.
14. Brinkhoff, T. (2000). Generating network-based moving objects. In *Proceedings of the 12th international conference on scientific and statistical database management*.
15. Brinkhoff, T. (2002). A framework for generating network-based moving objects. *GeoInformatica, 6*(2), 153–180.
16. Crossbow Technology Inc. White paper available at http://www.xbow.co.
17. W3C, *Xml schema language*, http://www.w3.org/XML/Schem.
18. W3C, *Xml query language*, http://www.w3.org/XML/Quer.
19. *The open archives initiative protocol for metadata harvesting 2.0*, http://www.openarchives.org/OAI/openarchivesprotocol.htm.
20. *Sensor model language*, http://vast.nsstc.uah.edu/SensorM.
21. *Tinyml: Meta-data for wireless networks*, http://www.cs.berkeley.edu/culler/cs294-f03/finalpapers/tinyml.pd.
22. Johannes, P. B., Gehrke, J., & Seshadri, P. (2001). Towards sensor database systems. In *Proceedings of the second international conference on mobile data management*, pp. 3–14.
23. Yao, Y., & Gehrke, J. (2002). The cougar approach to in-network query processing in sensor networks. *SIGMOD Record 31*, 2002.
24. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2003). The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD international conference on management of data*, pp. 491–502.
25. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2002). Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev. 36*, 131–146.
26. Tu Z., & Liang, W. (2005). *Energy-efficient aggregate query evaluation in sensor networks*, p. 3794.
27. Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., & Ganesan, D. (2001). Building efficient wireless sensor networks with low-level naming. *SIGOPS Oper. Syst. Rev., 35*, 146–159.
28. Ganesan, D., Greenstein, B., Perelyubskiy, D., Estrin, D. & Heidemann, J. (2003). An evaluation of multi-resolution search and storage in resource-constrained sensor networks. In *Proceedings of the ACM sensys*.
29. Galpin, I., Brenninkmeijer, C. Y., Jabeen, F., Fernandes, A. A., & Paton, N. W. (2009). Comprehensive optimization of declarative sensor network queries. In *Proceedings of the 21st international conference on scientific and statistical database management*, pp. 339–360.
30. Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., & Estrin, D. (2003). Data-centric storage in sensornets. *SIGCOMM Comput. Commun. Rev. 3*, 137–142.
31. Ee, C. T., & Ratnasamy, S. (2006). Practical data-centric storage. In *USENIX symposium on networked systems design and implementation*.
32. Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., & Yu, F. (2003). Data-centric storage in sensornets with ght, a geographic hash table. *Mob. Netw. Appl. 8*, 427–442.
33. Litwin, W., Moussa, R., & Schwarz, T. J. E. (2004). Lh*rs: A highly available distributed data storage system. In *Proceedings of the 30th international conference on very large data bases (VLDB)*, pp. 1289–1292.

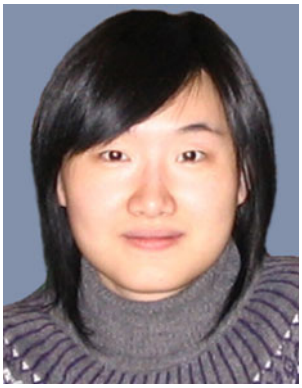
34. Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., & Gruber, R. E. (2006). Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th USENIX symposium on operating systems design and implementation*, pp. 15–15.
35. Johannesgehrke, P. B., & Mayr, T. (1999). *Query processing in a device database system*. Cornell University, Technical Report.

Author Biographies

Ruilin Liu is a Ph.D. student in Department of Computer Science at Stevens Institute of Technology. His research interest includes privacy-preserving data publishing, XML database, network access control and social network.

Xiuyuan Zheng is currently a Ph.D. student of the Electrical and Computer Engineering Department at Stevens Institute of Technology. His research interests include information security & privacy, wireless localization and location based services (LBS), wireless and sensor networks. He is currently working in the Data Analysis and Information SecuritY (DAISY) Lab with Prof. Yingying Chen. He was in the Master program in the Electrical and Computer Engineering Department at Stevens Institute of Technology from 2007 to 2009. He received his Bachelor's degree from Department of Telecommunication Engineering at Nanjing University of Posts and Communications, China, in 2007.

Hongbo Liu is a Ph.D. candidate of the Electrical and Computer Engineering Department at Stevens Institute of Technology. His research interests include information security & privacy, wireless localization and location based services (LBS), wireless and sensor networks. He is currently working in the Data Analysis and Information SecuritY (DAISY) Lab with Prof. Yingying Chen. He got his Master degree in communication engineering from Department of Communication and Information Engineering of University of Electronic Science and Technology of China in 2008. He received his Bachelor's degree from Department of Communication and Information Engineering of University of Electronic Science and Technology of China, China, in 2005.



Hui Wang received her Ph.D. degree in Computer Science from University of British Columbia, Vancouver, Canada. She has been an assistant professor in the Computer Science Department, Stevens Institute of Technology, since 2008. Her research interests include data management, database security, data privacy, and semi-structured databases.

Yingying Chen received her Ph.D. degree in Computer Science from Rutgers University. She is currently an assistant professor in the Department of Electrical and Computer Engineering at Stevens Institute of Technology. Her research interests include wireless and system security and privacy, wireless networking, and distributed systems. She has coauthored the book Securing Emerging Wireless Systems and published extensively in journal and conference papers. Prior to joining Stevens Institute of Technology, she was with Bell Laboratories and the Optical Networking Group, Lucent Technologies. She received the IEEE Outstanding Contribution Award from IEEE New Jersey Coast Section each year 2005-2009. She is the recipient of the NSF CAREER award. She is also the recipient of the Best Technological Innovation Award from the International TinyOS Technology Exchange in 2006, as well as the Best Paper Award from the International Conference on Wireless On-demand Network Systems and Services (WONS) in 2009.