A Collaborative and Semantic Data Management Framework for Ubiquitous Computing Environment *

Weisong Chen, Cho-Li Wang, and Francis C.M. Lau

Department of Computer Science, The University of Hong Kong
{wschen, clwang, fcmlau}@cs.hku.hk

Abstract. One fundamental task to realize the envisioned ubiquitous computing paradigm is the proper management of the data generated in this environment. The special characteristics of high distribution, heterogeneity, mobility, and autonomy of the ubiquitous computing environment introduce great difficulties in data management, which cannot be easily overcome using existing solutions. We propose a collaborative and semantic data management framework that is incentive-based. The incentives encourage contribution from and foster cooperation among different devices in the environment. Devices that contribute more to the successful information accesses of other devices will gain more routing knowledge, which in turn improves the service of their own queries. We suggest using ontology-based metadata to explicitly and formally describe data semantics. Our routing scheme would redirect queries to make full use of cached data available in the environment. Experiment results show that our system can serve information accesses in the ubiquitous environment with less communication costs than other similar systems.

1 Introduction

Ubiquitous computing is an emerging computing paradigm that promises continuous and seamless access to information anytime, anywhere, via any device [1]. Constant and rapid advances in hardware and communication technologies are bringing us closer to this envisioned paradigm. We are starting to experience some flavor of ubiquitous computing, though the complete realization is still not completely within reach.

One fundamental task to make ubiquitous computing a reality is the proper handling of the data generated in this environment. In order to support continuous and seamless information access, the underlying data should be carefully stored, distributed, and indexed. The special characteristics of ubiquitous computing, namely, high distribution, heterogeneity, mobility, and autonomy [2], introduce tremendous data management challenges, which cannot be easily overcome by existing solutions.

* This research is partly supported by HKU Large Equipment Grant 01021001 and Hong Kong RGC Grant HKU-7519/03E.

In this paper, we present a data management framework for ubiquitous computing environment. One guiding principle behind our design is to encourage contribution from and foster cooperation among devices owned by different users in the environment. The people joining the environment are expected to agree to share their devices and contribute to the networked community in which they reside. In addition to serving their owners, user devices share their data, as well as knowledge about their data, with other users' devices. Devices that contribute to the success of others' information accesses will benefit through acquiring useful routing knowledge in the process, which can enhance their ability to serve subsequent queries. Routing refers to a query traveling from node to node until finding an answer, and also the ensuing process of locating the desired data. The more contribution a device makes, the more knowledge it will gain. Hence, we have an incentive scheme for devices to participate in the activities of others.

We use the following techniques to address the data management challenges in ubiquitous computing environment.

- ***Ontology-based Metadata*** An ontology is an explicit specification of a conceptualization [3]. We suggest using ontology-based metadata to explicitly and formally describe data semantics, which should be an effective approach to dealing with data diversity in the ubiquitous environment.
- ***Incentive-based Routing Protocol*** We propose a routing protocol which provides incentives for devices to contribute to others' information accesses. The more contribution is made, the more knowledge will be gained. Devices interact in a collaborative manner that generates many mutual benefits.
- ***Cooperative Caching*** User devices maintain local cached copies of the downloaded data and share them with others. Popular data will be widely cached and unused data will fade away eventually. No explicit distribution control on the data sources is required.

The rest of the paper is organized as follows. In Section 2 we present the design of our system. Section 3 discusses the experiments, which are used to evaluate the performance of our system. We briefly discuss related work in Section 4. Section 5 concludes the paper.

2 System Design

This section discusses the essential aspects of our design, including system overview, ontology-based metadata, metadata similarity function, and incentive-based routing protocol.

2.1 System Overview

In the ubiquitous computing environment, there are two types of devices, shared devices providing public access and private user devices owned by particular users. Shared devices, such as sensors and server systems, generate or/and store data that can be accessed by different users in the environment. Traditionally,

when two nearby user devices issue queries for similar data, two searches will occur independently; and the search result in one user device cannot be shared and reused by the other device. In our model, people joining the environment will agree to share their devices and contribute to the infrastructure. Whenever possible, devices would contribute to information accesses of other devices, and thereby gain routing knowledge to their own advantage. As such, the devices of different users form a peer-to-peer community based on mutual benefits. Devices in the community share not just data but also knowledge.

Ontology-based metadata are used to explicitly describe data semantics. Metadata can be widely propagated. Through metadata propagation, a device advertises its knowledge about certain data. By receiving metadata, a device incrementally builds up its routing knowledge. Devices request information independently. Queries are forwarded to the nodes that have the closest matching metadata using a similarity function. Once results are found, the corresponding metadata are sent to the device that initiated the query, following the query path in reverse direction. The intermediate nodes along the query path are benefited too—they copy the metadata to their own store and update their routing knowledge accordingly. The initiating device, based on the received metadata, then makes a direct request to the data source for the desired data and maintains a cached copy. Future similar queries received by the intermediate nodes will be directly forwarded to the initiating node, where the cached data has been stored.

2.2 Ontology, Metadata, and Query

Ontology is the formal and explicit conceptualization of a particular domain. It includes a set of concepts and their inter-relationships. Based on [4], we define ontology structure as $O = \{C, P, H^C, R\}$. Using the example ontology in Fig.1, we explain each component of the ontology structure.

- ***Concepts*** (C): well-defined terms referring to classes (or types) of objects in a particular domain. In the sample ontology, $C = \{$*Publication, Staff, Institute, Report, ...*$\}$.
- ***Relations*** (P): properties of concepts defining the concept semantics. In the sample ontology, $P = \{$*Publish, Author, Work, Graduate, ...* $\}$.
- ***Concept Hierarchy*** (H^C): a hierarchy of concepts that are linked together through relations of specialization and generalization. H^C*(Report, Publication)* means that *Report* is a sub-concept of *Publication*.
- ***R***: a function that relates two concepts non-taxonomically, using the relations in P. $R(P) = (C_1, C_2)$ is usually written as $P(C_1, C_2)$. For example, *R(Publish)= (Staff, Publication)*, is written as *Publish(Staff, Publication)*.

There are other terms relating to the usage of ontology, defined as follows.

- ***Concept Instance***: An object is an instance of a concept if it is a member of the class denoted by the concept.

– ***Relation Instance***: relations relating two concepts in the ontology can be used to relate concept instances of these two concepts. Instances of these relations are called relation instances.
– ***Ontology Instance***: comprising concept instances and relation instances.

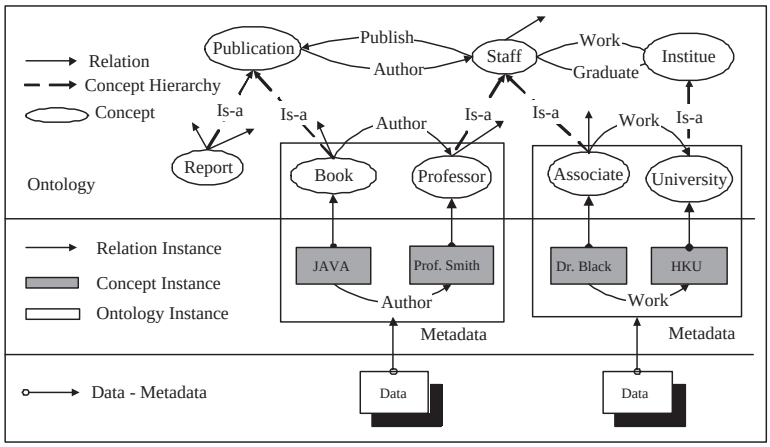

Fig. 1. Ontology, Metadata and Data

In our system, we use ontology instances as metadata to describe data semantics. A metadata structure is a 6-tuple $M = \{O, I, C, PI, I^C, I^R\}$, where O is the referenced ontology, I the set of concept instances, C the set of concepts (a subset of the concepts in the ontology), PI the set of relation instances, $I^C : I \rightarrow C$ the function that relates concept instances to the corresponding concepts, and function $I^R : PI \rightarrow I \times I$ the function to relate concept instances using relation instances; $I^R(PI) = (I_1, I_2)$ is usually written as $PI(I_1, I_2)$.

For each piece of metadata, there is one concept instance that serves as the identifier of the described data. This concept instance is called the central concept instance of the metadata, denoted as M^I. The corresponding concept of the central concept instance is called the central concept, denoted as M^C. The central concept identifies the class of objects that the described data belong to. Other concept instances, together with the relation instances, describe the properties of the central concept instance. The properties are also called attributes of the central concept instance.

While metadata describe data semantics, queries state the desired properties of the requested information. The query structure and the meaning of each element in this structure are the same as those of the metadata. They can be used in the operations where metadata are applicable. The only difference is that query allows wildcard instance (denoted as I^*), i.e., any instance of a particular concept.

2.3 Metadata Similarity

To determine the degree that metadata M_2 is similar to metadata M_1, we first independently calculate the degree that the concept instances in M_2 is similar to their corresponding concept instances in M_1. The following formula is to determine the corresponding concept instances between two metadata. If the corresponding concept instance does not exist, we denote it as I_{NIL}. Suppose the concept instance set in M_2 is I_{M2} and relation set is P_{M2} respectively, the central concept instances in M_1 and M_2 are M_1^I and M_2^I respectively, for any concept instance I_1 in M_1 , whose relation with M_1^I is P,

$$CP(I_1) = \begin{cases} M_2^I & \text{if } I_1 = M_1^I \\ I_2,\ s.t.\ I_2 \in I_{M2} \text{ and } P(M_2^I, I_2) & \text{if } P \in P_{M2} \\ I_{NIL} & \text{otherwise} \end{cases}$$

Other ontology research projects [5] have defined a numerical function to measure the similarity level between two concepts in a concept hierarchy. The basic principle is to exploit the shared concepts that are super-concepts for both concepts. A concept's super-concepts (SC) in the ontology can be determined using the following formula,

$$SC(C_i) = \{C_j | C_j \in C,\ H^C(C_i, C_j)\} \cup \{C_i\}$$

The degree that concept C_j is similar to concept C_i, denoted as $C_{Sim}(C_i, C_j)$, is given by the following formula, where ρ is a tuning parameter with a range of [0,1].

$$C_{sim}(C_i, C_j) = \rho \frac{|SC(C_i) \cap SC(C_j|}{|SC(C_i)|} + (1-\rho)\frac{|SC(C_i) \cap SC(C_j|}{|SC(C_j)|}$$

The similarity level between two concept instances is given by the following formula,

$$I_{sim}(I_1, I_2) = \begin{cases} 0 & \text{if } I_1 = I_{NIL} \text{ or } I_2 = I_{NIL} \\ 1 & \text{if } I_1 = I_2 \\ C_{sim}(I^C(I_1), I^C(I_2)) & \text{if } I_1 = I^* \text{ and } I_2 \neq I^* \\ \frac{C_{sim}(I^C(I_1), I^C(I_2))}{2} & \text{otherwise} \end{cases}$$

The degree that metadata M_2 is similar to M_1 is given by the following formula, where I_{M2} denotes the concept instance set of M_2, excluding the central concept instance M_2^I.

$$M_{sim}(M_1, M_2) = I_{sim}(M_1^I, M_2^I) * \frac{\sum I_{sim}(I_i, I_j), I_j \in I_{M2} \text{ and } I_j = CP(I_i)}{|I_{M1}|}$$

2.4 Incentive-based Routing Protocol

As nodes participate in routing processes, they continuously receive metadata from other nodes. The received metadata are classified according to their central

concepts $(C_1, C_2, \ldots, C_n)$. A list (L_i) will be used to store the metadata with the same central concept. The lists are indexed by the central concepts of the stored metadata. Associations between the metadata and the nodes providing the metadata are created and the metadata are inserted into the corresponding lists. Newly received metadata will be inserted at the head of the lists. When receiving a query (Q), the node searches the routing table to find the list (L_i) that is indexed by the central concept of the query. For each metadata (M_i) in (L_i), we compute the metadata similarity using the formula $M_{sim}(Q, M_i)$. The query is forwarded to the node that has the most similar metadata.

In our model, nodes that help forward queries will obtain more knowledge about data and their locations in the network, thus enhancing the ability of these nodes to serve future queries. When forwarding queries, nodes record the nodes that initiated the queries. When passing the query results to the initiating nodes, the nodes record the nodes providing the results. This way, nodes obtain knowledge about the cached data in the network, and knowledge of the actual data sources. This knowledge will prove to be useful in serving subsequent queries. The nodes that participate more in forwarding queries will have more knowledge, and incentive scheme attracting participation from nodes.

3 Performance Evaluation

We modified the NeuroGrid [7] system to support ontology-based metadata and incorporate our proposed routing protocol. The parameter settings of our simulation system are based on the observation provided by [12]. Table 1 shows the detailed meanings and default values of the key parameters used in our simulation system. We evaluate our proposed framework based on the performance metrics hit ratio and average path length. Hit ratio measures the percentage of the queries got served within bounded time-to-live (TTL), and average path length estimates the cost of finding the data.

Table 1. Parameter Settings

Parameter	***Base Value***
Total number of concepts in the ontology	150
The number of shared devices	50
The number of user devices	50
Total number of data objects	3000
The size of the cache memory	1 MB
The size of the routing table	100 KB
Staring TTL of the queries	7
Total number of queries by all user devices	10000
Disconnection probability of shared devices	20%

3.1 Ontology Vs. Keyword Searching

Our first experiment is to show the performance of our system, using two test cases. One uses ontology-based searching. The other uses keyword-based searching. The performance results are shown in Fig.2. We can see that, in both cases, as more queries are issued, the cached data contribute more to the overall hit ratio. However, the ontology-based searching has far superior performance, compared with keyword-based searching.

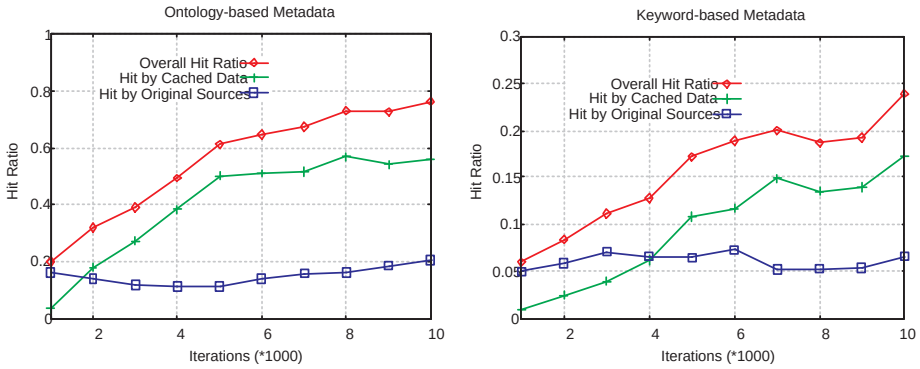

Fig. 2. Overall Performance

3.2 Effect of Cache Replacement Policies and Query Patterns

In the second experiment, we adopt three cache replacement policies, namely, First In First Out (FIFO), Least Recently Used (LRU), and Least Frequently Used (LFU). Cache replacement occurs when a cache is full and there are new data coming in to be cached. We study their effect on the system performance by running simulation for each policy. The results are shown in the left part of Fig. 3. We can see that when the number of searches is small, the hit ratio monotonically increases and there is no major difference among these three policies. This is because the caches in the user devices are not full. Then, after around 4000 iterations, the caches become full. New data will start to push away other data, and different cache replacement policies start to show different effects on the system performance. We found that FIFO has the worst performance, due to the fact that it indiscriminately replaces cached data solely based on the time of caching, which may swap out popular data needed by other devices.

We also want to know how different query patterns might affect the performance of our system. We test three different query patterns, namely, Random Query Pattern, Interest-based Query Pattern, and Popularity-based Query Pattern. As their names suggest, the random query pattern will generate queries

randomly without any predefined pattern. The interest-based query pattern will generate queries only for some limited number of concepts that are of interest to each device. The popularity-based query pattern will generate queries according to what are popular in the network. The effect of these three query patterns are shown in the right part of Fig. 3. As expected, the popularity-based query pattern has the best performance, as most of the queries can be served by the cached copies in the network.

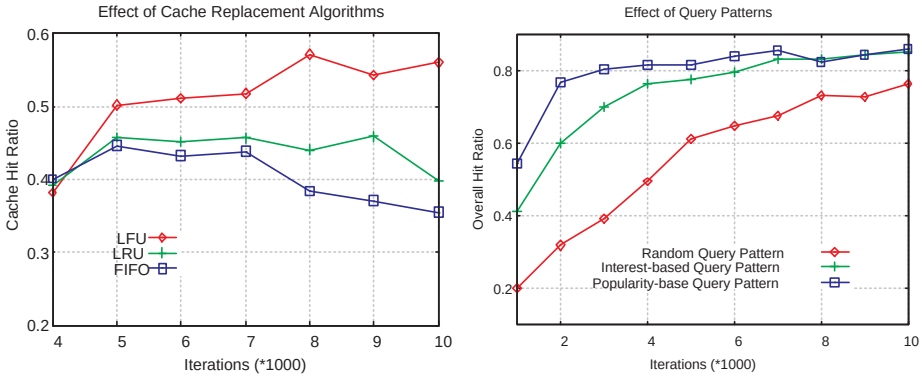

Fig. 3. Effect of Cache Replacement Policies and Query Patterns

3.3 Comparison with FreeNet, NeuroGrid, and Gnutella

Our third experiment is to evaluate the efficiency of our proposed routing protocol against some well-known protocols. We compared our system with FreeNet, NeuroGrid, and Gnutella, using the same values for the parameters listed in the above table. Our incentive-based routing protocol has very similar performance with FreeNet system, in terms of overall hit ratio and the number of messages transferred. Both systems have much better performance than NeuroGrid and Gnutella. FreeNet only supports exact ID matching; our proposed framework allows users formulate semantic descriptions of the queried data, which is more flexible. In addition, we can achieve higher hit rate and less messages when the number of iterations is getting larger. This is because we store routing knowledge, instead of caching the data themselves. Routing knowledge is much smaller compared with the data themselves and our system can accommodate much more knowledge for locating data.

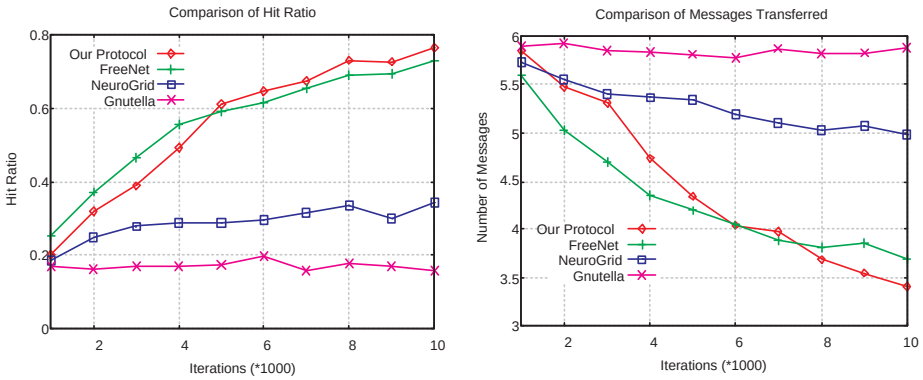

Fig. 4. Comparison with FreeNet, NeuroGrid, and Gnutella

4 Related Work and Discussion

There are some existing research work that have motivated us. In NeuroGrid [7], each node maintains routing knowledge by incrementally building up a list of queries together with which other nodes have been good at answering these queries in the past. In their implementation, keyword strings are used to index local resources and routing knowledge.

MoGATU [2] is a project explicitly designed to deal with data management in the ubiquitous computing environment. Profile and context information are used to guide the interactions among different devices. Caching and replication are deployed at the devices, according to profile and context information. MoGATU considers each device individually to serve their users' information accesses. Their results can be used to complement our system.

HyperCuP [6] organizes the peer nodes in the P2P network into a hypercube topology. It guarantees that each peer node receives exactly one message in any broadcasting request. HyperCuP was among the early research that adopted ontology. Peers with similar resources or interests are grouped into concept clusters. The concept clusters are organized into a hypercube topology. Concept clusters themselves are hypercubes or star graphs. Queries in the network are first propagated to the intended concept clusters, which in turn optimally broadcast the queries within the clusters.

5 Conclusion and Future Work

In the paper, we propose a collaborative and semantic data management framework to address the challenges of data access in the emerging ubiquitous computing environment. Experiment results have shown that our system can make efficient use of cached data in the network, and therefore it will not easily succumb to disconnection of original data sources. We studied the effect of different

cache replacement policies and query patterns on the performance of our system. We also compared our system with some similar projects. We have proved the efficacy of our proposed framework in providing effective and efficient information access to device users.

In this paper, we have assumed that complete ontology knowledge is available at each device, which is not always possible in the ubiquitous computing environment. We did not explicitly deal with ontology variations either. In the next stage, we will incorporate ontology management into our infrastructure to bring the system closer to the reality.

References

1. M. Satyanarayanan, "Pervasive Computing: Vision and challenges," IEEE Personal Communications, pp. 10–17, August 2001.
2. F. Perich, A. Joshi, T. Finin, and Y. Yesha. "On Data Management in Pervasive Computing Environments," IEEE Transactions on Knowledge and Data Engineering, May 2004.
3. T. Gruber. "A Translation Approach to Portable Ontology Specifications," Knowledge Acquisition, pp. 199-220, 1993.
4. A. Maedche, and V. Zacharias. "Clustering Ontology-based Metadata in the Semantic Web," in Proceedings of 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002.
5. T. Andreasen, H. Bulskov, and R. Knappe. "From Ontology over Similarity to Query Evaluation," 2nd CoLogNET-ElsNET Symposium, Questions and Answers: Theoretical and Applied Perspectives, 2003.
6. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. "HyperCuP–Hypercubes, Ontologies and Efficient Search on P2P Networks," International Workshop on Agents and Peer-to-Peer Computing, Bologna, Italy, July 2002.
7. S. Joseph, "NeuroGrid: Semantically routing queries in peer-to- peer networks," In Proceedings of the International Workshop on Peer-to-Peer Computing, 2002.
8. A. Crespo and H. Garcia-Molina. "Semantic overlay networks for p2p systems," Technical Report, Computer Science Department, Stanford University, October 2002.
9. I. Clarke, O. Sandberg, B. Wiley and T.W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System." In Designing Privacy Enhancing Technologies: Lecture Notes in Computer Science 2009.
10. F.M. Cca-Acuna, C. Peery, R.P. Martin, and T.D. Nguyen. "PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing," International Workshop on Peer-to-Peer Computing, Pisa, 2002.
11. M.J. Franklin. "Challenges in Ubiquitous Data Management," Informatics 10 Years Back, 10 Years Ahead, 2001.
12. M.T. Schlosser, T.E. Condie, and S.D. Kamvar, "Simulating a File-Sharing P2P Network." First Workshop on Semantics in P2P and Grid Computing, December, 2002.