On Data Management

in Pervasive Computing Environments

Filip Perich, Anupam Joshi, Timothy Finin, and Yelena Yesha

University of Maryland, Baltimore County

1000 Hilltop Circle, Baltimore, MD 21250

phone: +1 (410) 455 - 2668

fax: +1 (410) 455 - 3969

{fperic1, joshi, finin, yeyesha}@cs.umbc.edu

This work was supported in part by NSF awards IIS 9875433 and 0209001, and DARPA contract F30602-00-2-0591

Abstract

This paper presents a framework to address new data management challenges introduced by data-intensive, pervasive computing environments. These challenges include a spatio-temporal variation of data and data source availability, lack of a global catalog and schema, and no guarantee of reconnection among peers due to the serendipitous nature of the environment. An important aspect of our solution is to treat devices as semi-autonomous peers guided in their interactions by profiles and context. The profiles are grounded in a semantically rich language and represent information about users, devices and data described in terms of "beliefs", "desires", and "intentions". We present a prototype implementation of this framework over combined Bluetooth and Ad-Hoc 802.11 networks, and present experimental and simulation results that validate our approach and measure system performance.

Keywords

Mobile Data Management, Pervasive Computing Environments, Data and Knowledge Representation, Profile-Driven Caching Algorithm, Profile Driven Data Management, Data-Centric Routing Algorithm

I. Introduction

Data-intensive pervasive computing environments introduce significantly new challenges not addressed by traditional mobile data management solutions. In this paper, we motivate the problem of data management in these environments and identify its challenges. We address these challenges by articulating the requirements for, and presenting an implementation prototype of, a robust framework enabling serendipitous querying and profile-driven data management. We also present simulation results in order to show the efficacy of our solution.

The paradigm of pervasive computing has already begun to influence our lives. We interact, daily, with tens of computationally enabled devices. We carry cellular phones allowing us to interact with other people and to connect to the Internet infrastructure anytime from anywhere. We use personal digital assistants to remind us of our scheduled appointments and for storing personal contacts. We use watches, which, in addition to displaying time, monitor our heartbeat or even

take photographs of our surroundings. We use transportation passes with embedded processors that automatically open gates upon entering a subway station or going through a highway toll-booth. Furthermore, we drive in cars equipped with on-board computers providing navigational assistance and that monitor the engine, fluid levels and the environmental systems. These examples represent only a small subset of devices we interact on a daily basis but they serve as a starting point for predicting what the future holds.

As manufacturers are able to develop increasingly powerful devices while simultaneously decreasing their size and cost, this trend will only continue to grow. The hardware developments coupled with the widespread deployment of wireless short-range ad-hoc networking technologies, such as Bluetooth [3] and UWB [36], will truly realize the vision of pervasive computing. Ad-hoc networking technologies are natural successors of wireless communication systems, including cellular telephone networks, low earth orbiting satellites and wireless local-area networks (WLAN), *e.g.*, IEEE 802.11 a, b and g standards [18]. WLAN can operate both in infrastructure-based and ad-hoc modes. In the former case, wireless devices interact with an access point. Access points control traffic on wireless subnets and act as bridges to a wired network. In an ad-hoc mode, all wireless devices communicate directly with others in their vicinity.

The deployment of wireless communication technologies has led to significant research in the area of mobile data management. The research is dominated by the client-proxy-server model. In this model, mobile devices are able to connect to the Internet and serve as client end-points, initiating actions and receiving information from servers on the network. The primary focus of existing research is on the development of protocols and techniques that deal with disconnection management, low bandwidth and device resource constraints. The aim is to allow applications built for the wired world (*e.g.*, World Wide Web and databases) to run in wireless domains using

proxy based approaches ([2], [22]). In systems based on the cellular infrastructure or WLANs, the traditional client-proxy-server interaction is perhaps an appropriate model where the "client" database can be extremely lightweight [4], has a (partial) replica of the main database on the wired side [19], [34] or where selected data is continuously broadcasted into the environment [1], [16].

With the widespread use of short-range ad-hoc networking technologies, alternative data management mechanisms are necessary in order to enable devices to spontaneously interact with others in their vicinity. In this approach, all mobile devices can be both sources and consumers of information and must be able to cooperate with other devices in their vicinity in order to pursue individual and collective tasks. This will allow mobile devices, including handhelds, wearables, computers in vehicles, computers embedded in the physical infrastructure, and (nano)sensors to become more autonomous, dynamic and adaptive with respect to their environment. For instance, a car running low on gas can query those traveling in the opposite direction if they know of a gas station within ten miles, or directly get this information from an electronic highway sign. Of course, this could be done by the car relaying its GPS coordinates to some centralized server over a cellular infrastructure-supported wireless network. This approach is, however, likely to be inefficient. The large number of requests generated by all cars present on the highways may lead to congestion in the wireless networks and a bottleneck on the yellow page hardware. The congestion may in turn result in a higher response latency of the information than already introduced by wireless networks. Additionally, the congestion will cause a high risk of system failure.

Such highly dynamic environments, consisting of spontaneously networked computationally enabled devices in a vicinity that are both producers and consumers of information, represent something very different from traditional mobile computing. They have recently been described as *data-intensive* pervasive computing environments [26]. These environments introduce additional

challenges on top of issues related to data management in traditional mobile environments: (i) As we increasingly rely on the use of information in electronic form, we expect and require instant and complete access to any information at anytime, anywhere. Thus, devices and their applications must cope with highly dynamic environments where both data and source availability vary over location and time. As such, devices must adapt to currently available sources and constraints. (ii) The pervasive computing environment inverts the traditional sense of data management in distributed databases, which was based on the idea of "passive databases" and "active users" [33]. There data is stored in a database in one or some well known locations and queries/users would "come and go". We can instead model the new environment as a scenario where data objects resident on mobile devices that actively move throughout the system and each device is continually processing a vast amount of incoming data while looking for information of interest to its user.

An important aspect of our framework is a cross-layer interaction between a data management layer and an underlying networking layer. Our framework, MoGATU, treats all devices as equal semi-autonomous peers guided in their interactions by profiles and context. We assume that peers can communicate in a half-duplex mode using Bluetooth or Ad-Hoc 802.11 technologies. The framework abstracts all devices in the environment in terms of Information Providers, Information Consumers, and Information Managers: (i) Information Providers represent data sources present on devices. Each Information Provider holds a partial set of heterogeneous data, a *fragment*, available in the whole environment and annotated in a semantic language. (ii) Information Consumers represent entities that can query and update data. These entities can represent humans as well as autonomous software agents. (iii) Lastly, an instance of an Information Manager (InforMa) must be present on every device. Information Managers are responsible for the underlying network communication and for most of the data management functions. Each InforMa maintains infor-

mation about peers in its vicinity. This information includes the types of devices and the types of information they can provide. Each InforMa also maintains a data cache for storing data obtained from other mobile devices and by its local Providers. Additionally, each InforMa may include a user profile reflecting some of the user's beliefs, desires, and intentions, a model that has been explored in multi-agent interactions [5]. The InforMa uses the profile to adapt its caching strategies and to initiate a collaboration with peers in order to obtain desired information. Additionally, MoGATU uses profiles to describe content of data objects, their ownership information and both access preconditions and postconditions.

The remainder of the paper is structured as follows: We illustrate the need for a data management infrastructure in pervasive computing environments in Section II. We define the new fundamental data management challenges and briefly describe traditional issues of mobile data management in Section III. We present the design and implementation of our infrastructure for addressing these challenges in Section IV. We measure the performance of our implementation in Section V. We present related work in Section VI and conclude with Section VII.

II. Motivating Example

To illustrate the research focus and motivate the need for our framework consider the following example. It is an excerpt from Bob's daily activities used for experiments in Section V. It represents an ideal environment that our framework should provide:

It is 5:40 in the afternoon and Bob's work day is just ending. As he is getting ready to leave the office his phone rings. It is Alice asking him to meet her at the local shopping mall. Bob agrees to meet her and notifies his mobile device about the appointment. While he is walking through the building toward the parking lot, his device is able to connect to the office network infrastructure using a WLAN and fetch the directions through an information broker [7]. In the car, the mobile

device sends directions to the on-board computer that displays them on the area map. While en-route, Bob feels that the traffic is not moving fast enough and would like to get to his destination via some quicker route. He instructs his device, which can now connect to cars around him, to ask these if they know of a faster route. The device contacts its neighbors and returns with an alternate map. Although the suggested route is longer it circumvents afternoon traffic jam building up on the original route. Bob takes the different roads and arrives at the mall's entrance twenty minutes earlier. He decides to use the extra time by checking out local stores for a bargain on some small gift for Alice. His mobile device finds out from the mall directory server about stores that sell such gifts, contacts them to find out possible gifts within a $25 and presents ideas to Bob. Similar, the mobile devices has a *wish list* of other garments Bob wishes to purchase and proactively searches for these bargains as well. Upon Alice's arrival, Bob asks his mobile device to suggest available restaurants (the device cached such information during the mall exploration) and lets Alice pick one. She chooses the closest Italian restaurant that indicates it has an available table with no waiting period [30]. Thus, they are seated immediately and spend several hours eating and chatting.

The developments in networking and hardware have reached a point making this scenario technically possible. Cars are being equipped with desktop-like computers and wireless connectivity. Moreover, handheld mobile devices, such as HP iPAQ H5455, include 400MHz Intel processors with 64MB SDRAM and integrated Bluetooth and 802.11b wireless technologies. This provides even handheld devices with relatively large computing and networking resources enabling them to process and exchange data among peers. The related data management challenges, however, still need to be thoroughly addressed. Ideally, Bob's mobile device adapts its interaction mode with its environment based on the *knowledge* encoded in Bob's profile and the current *context*. The device must be able to describe and reason over Bob's preferences. The device must be able to utilize

locally available resources, *i.e.*, the device must posses the means to discover and interact with the resources in its vicinity. For example, while Bob walks through the mall, the device should cache local restaurant advertisements in anticipation of Bob's future query since his *calendar* calls for dinner at a restaurant within the next hour.

III. Data Management Challenges in Pervasive Computing Environments

The pervasive computing environment is a special type of mobile distributed database system; however, it is a far more complex than the conventional client-proxy-server model. We can illustrate how they are related by classifying the pervasive model along four orthogonal axes representing autonomy, distribution, heterogeneity and mobility of mobile databases [13]. The pervasive model is highly autonomous since there is no centralized control of the individual client databases. It is heterogeneous as we only assume that entities can *speak* to each other in some neutral format. The model is clearly distributed as parts of data objects may reside on different devices and there is replication as entities cache data and their respective metadata. Mobility is of course given – in ad-hoc networking environments devices can change their locations and no fixed set of entities is *always* accessible to a given device. The last point is perhaps the most important in distinguishing pervasive environments from traditional mobile information access. It is also the main reason why a direct use of solutions developed for mobile information access is inappropriate. In mobile distributed systems disconnections of mobile devices from the network are viewed only as temporary events. Additionally, these systems often assume that all data *managers* are located at fixed positions in the wired network and that their locations are known by every client a priori [6], [24], [25]. Finally, an additional limitation is the naming schema for defining data and for locating both data and devices in the traditional system. Here each client must know the precise server location as well as its corresponding database schema in order to utilize the data properly.

Additionally, the pervasive environment imposes the following issues that are primarily related to the randomness of every device's neighborhood at any instance of time. The neighborhood, also referred to as vicinity, consists of all reachable devices that a particular (mobile) device can communicate with and all available data that is accessible at that time.

- *Spatio-temporal variation of data and data source availability.* As devices move, their vicinity changes dynamically affecting data and data source availability. For example, depending on the specific location and time Bob asks for an alternate route his device may obtain different answers or none at all. Additionally, current wireless networking technologies cannot support stable connections under high mobility. In our framework, we address this issue by enabling mobile devices to proactively gather and cache data and by employing a data-based routing protocol.
- *Lack of a global catalog and schema.* As the neighborhood changes dynamically, a mobile device has no prior knowledge of the current set of available data. For example, while en-route, Bob's device does not know ahead of time which car to query for a different route. There is no global catalog that it may contact and ask for a location of a given data item. In our framework, we address this issue by requiring each device to describe and optionally advertise its capabilities to its neighbors using heterogeneous ontologies, defined via a common vocabulary encoded in a Semantic Web language. This allows every device to dynamically construct a subset of the local catalog. At the same time, each data source may have its own ontology that in our framework can be addresses by providing a translation service present on some devices in the vicinity.
- *No guarantee of reconnection.* When a device moves away from a current neighborhood it may affect any ongoing interaction among other devices of that neighborhood. As there is no guarantee that the mobile devices will ever again be able to communicate among themselves, this may cause an inconsistent global state. For example, when Bob's device is querying local restaurants it may be

unable to make a reservation. We address this issue by relying on proactively cached information, a data-based routing algorithm and by providing best-effort service only.

- *No guarantee of collaboration.* The issues of privacy and trust are very important for a pervasive environment where random devices interact in random transactions [35]. A device may have reliable information but refuses to make it available to others. A device may be willing to share information; however that information is unreliable. Lastly, when a device makes information available to other devices questions regarding protection of future changes and sharing of that data arise. For example, Bob's device is not guaranteed that the alternate route in fact reaches the mall. Currently, we do not address this issue and only assume a best-effort service. In related efforts, [23] provides a work on trust management for mobile environments.

One consequence of these challenges is that query answering will be highly serendipitous. The answer obtained will depend on information sources accessible in the current vicinity. Our data management system avoids such a situation and strives to provide answers to user queries. Consequently, each device in our framework should gather information proactively and much of the interaction among devices should happen in the background, without an explicit human intervention [14]. This requires that devices adapt themselves to the needs and preferences of their users and the current context. For example, Bob's device should start caching bargain offers matching Bob's interests when he is inside or near a shopping mall. Accordingly, Bob's preferences and needs, modulated by context as well as battery power and storage space, should allow the mobile device to determine what data to obtain proactively and its relative worth.

IV. MoGATU: Design and Implementation

In an effort to address the challenges above we have designed and implemented a framework for handling data management in pervasive computing environment. We call the implementation

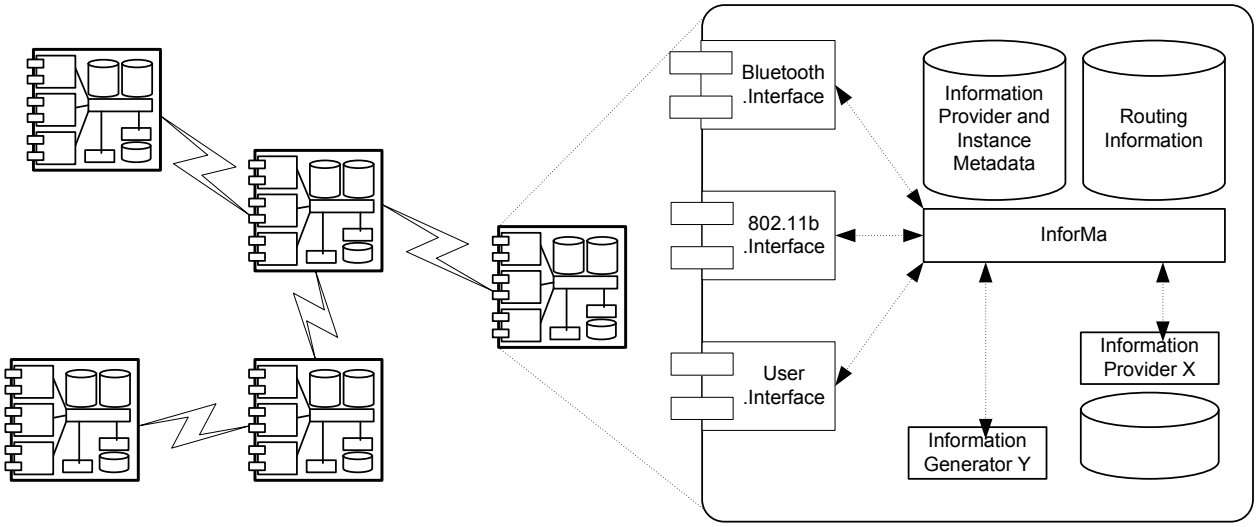

Fig. 1. Entity Details and Interaction in the MoGATU Framework

prototype *MoGATU*. Our framework treats all devices as equal semi-autonomous peers. It can be classified as *chained* architecture with *random* replication and local *incremental* policy [37]. The framework abstracts devices in terms of Information Providers, Information Consumers and Information Managers (InforMas). Additionally, the framework employs Communication Interfaces for supporting multiple networking technologies and Profiles for enabling a proactive device behavior. Figure 1 depicts an overview of the framework and the integration among various devices and their resources. In our framework, we address the challenges from Section III in the following manner:

- *Autonomy.* We treat all devices as independent entities acting autonomously from others.
- *Mobility.* We do not pose any restriction on the mobility patterns of devices.
- *Heterogeneity.* Pervasive computing environments are highly heterogeneous in terms of devices, data resources and networking technologies. We address this by abstracting each device with an InforMa. Each resource of stored or dynamically generated information is abstracted by an Information Provider. Lastly, networking technology is abstracted by a Communication Interface allowing peer InforMas to interact regardless of the underlying network.

- *Distribution.* Mobile devices may have multiple Information Providers, each holding a subset of the global data repository. We allow devices to advertise, solicit, exchange and modify such data with their peers. Thus, our framework is highly distributed.
- *Lack of a global catalog and schema.* We do not depend on any global catalog or schema. Instead, we use a semantically rich language for defining base ontologies – a set of common vocabularies. These ontologies enable us to describe information provided by any Information Provider. These ontologies are also used to advertise, discover and query such information among devices.
- *No guarantee of reconnection.* To remedy the effects of reconnection, our framework operates on a best-effort basis and relies on proactively cached information. We also employ a data-based routing algorithm allowing *closer* devices to provide answers to queries placed by their peers.
- *Spatio-temporal variation of data and data source availability.* We address this issue by enabling devices to proactively gather information without a human interaction. We employ a user profile also annotated in a semantically rich language. Each InforMa uses the profile to adapt its caching behavior and to proactively query its vicinity.

A. Information Providers

Every device may hold one or more Information Providers. An Information Provider manages and provides an interface to a subset of the global data repository. The data can be stored on the device or generated on-demand. The managed subset may be inconsistent with other *copies* on other devices as there is no guarantee that these devices can interact and the subset may even be empty. In our framework, any entity is an Information Provider if it is able to accept a query and generate a response. For example, Bob's device has at least two Providers: one representing current time and another for Bob's calendar. Accordingly, the response of each Provider is based on the query, stored data and mechanisms, including reference rules, for generating new data.

Each Information Provider describes its capabilities in terms of ontologies defined in a semantically rich language. For our implementation we use DAML+OIL [11]. Moreover, we base our design on the DAML-S standard [10] that attempts to comprehensively describe services for the WWW. We do not use DAML-S directly as it has not been finalized yet. Using this approach, however, each Provider can describe itself by defining the service model it implements, the process model that provides the information, and the input (query) restrictions/requirements [10].

We use a semantically rich language for describing Information Providers and data they can provide in order to overcome the heterogeneous nature of the pervasive computing environment and the lack of global schema. It enables devices in our framework to interact among themselves as they are only required to *speak* using a common set of vocabularies. Each device wishing to interact must only be able to process, *i.e.*, parse, the information annotated in the language at a syntactic level. Moreover, the language supports more efficient discovery and matching approaches required for locating Information Providers (or cached answers) for answering queries [7]. Lastly, the use of DAML+OIL allows mobile devices to also use resources available on the Semantic Web.

Upon start-up, each Information Provider registers itself with the local InforMa by sending a registration message including the service model s, process models p and input restrictions I:

$$reg = (s, p, I, t, a)$$

Each Provider also specifies the lifetime t it will be available and if it is willing to process queries for remote devices, denoted as a. Each Provider, however, communicates only with its local InforMa, which routes messages between the Provider and other devices in the vicinity. The InforMa adds this Provider into its cache of local providers and discards the entry once the *lifetime* expired and the Provider has not renewed its registration. Additionally, InforMa may advertise the

Provider to other devices in the vicinity if the Provider is willing to process queries for remote devices. The advertisement frequency is a tunable parameter for each InforMa.

B. Information Consumers

Information Consumers represent entities that can query and update the data. In the current design, Information Consumers primarily represent human users that ask their mobile devices for context-sensitive information but may also represent autonomous software agents. Similar to Information Providers, Consumers register with local InforMa; however, they are not advertised to remote devices. Each Consumer sends queries to its InforMa, which routes them to appropriate local Information Providers or those on remote peer devices for processing, and awaits a response.

C. Information Manager (InforMa)

An InforMa is responsible for most of data management functions and for an underlying network communication. From the data management perspective, InforMa must be able to discover available sources, construct dynamic indexes and catalogs, support queries, and provide caching mechanisms for addressing the dynamic nature of the environments. From the networking perspective, InforMa must be able to discover devices, interact with them and route messages.

Each InforMa maintains information about Providers and Consumers present on the same device as the InforMa. This information includes the *lifetime* of each Provider and its service model, process models and query restrictions. Each InforMa also maintains information about peers in its vicinity. This information includes the ID of devices and types of information they can provide, *i.e.*, Provider advertisements. Lastly, InforMa maintains a data cache for storing information obtained from other mobile devices as well as the information provided by its local Providers, *i.e.*, answers to previous queries. Additionally, each InforMa may include a user profile reflecting the user's

preferences and needs. The InforMa uses the profile to adapt its caching strategies and to initiate a collaboration with peers for obtaining the desired information.

Since every device in the environment, ranging from sensors to laptops, must implement an instance of InforMa, the framework does not enforce that each device implements all functionalities for its InforMa. In the simplest case, type 0, InforMa can maintain only one local Provider. It does not cache any remote information and it does not possess any reasoning or parsing mechanisms. This InforMa only periodically *broadcasts* data sent to it by the Provider. This type of InforMa is well-suited for resource limited devices, such as gas station beacons. On the other hand, devices wishing to interact in more intelligent manner and those that possess more resources must implement an InforMa that is able to maintain information about multiple local and *remote* Providers. These types of InforMa must also be able to parse messages, route message to other peer devices and proactively query peers. Moreover, we can differentiate among additional four types of InforMa based on the collaboration level: (1) InforMa does not cache any remote advertisements or answers to queries, (2) InforMa caches remote advertisements only for the *lifetime* specified in the message or until replaced by another entry, (3) InforMa caches both advertisements and answers, and (4) InforMa also caches all advertisements/answers and makes them available to other peers. The type 4, hence, can serve as a temporary sub-global catalog for peers in the current vicinity.

C.1 Query Processing

One of main objectives of InforMa is to provide querying capabilities. An Information Consumer sends a query annotated in DAML+OIL. Similar to Provider advertisements, a query specifies the required service model s and the input values i:

$$query = (s, i)$$

Algorithm 1 InforMa_Process_Query($src, query$)

```
(s_q, i_q) ← parse_query(query)
for each a ← cached_answer do
   (s_a, i_a) ← parse_answer(a)
   if s_q ≃ s_a and i_q = i_a then
      return a
   end if
end for
for each p ← local_provider do
   (s_p, I_p) ← parse_provider(p)
   if s_q ≃ s_p and i_q ∈_restriction I_p then
      return ask_provider(p, query)
   end if
end for
if src is local Consumer then
   for each p ← remote_provider do
      (s_p, I_p) ← parse_provider(p)
      if s_q ≃ s_p and i_q ∈_restriction I_p then
         dst ← provider_location(p)
         return InforMa_Route_Query(src, dst, query, 0, do_not_intercept)
      end if
   end for
end if
return error
```

An InforMa matches the query against entries in its cache. Algorithm 1 shows the pseudo-code. Each entry in the cache represents an unexpired answer to a previous query (otherwise it would be removed), or an advertisement for some local or remote Provider. The InforMa parses the query and each entry according to DAML+OIL rules and relationships specified in the employed ontologies. The InforMa compares service models and validates the query against inputs restrictions. For cached answers, the InforMa matches input values of the query against those in the cached answer. Our approach is equivalent to DAML-S using traditional forward chaining method also used by other DATALOG / Prolog based query processing techniques [15], [17]. The InforMa first tries to find and return a cached answer. Otherwise, the InforMa tries to find a local or some remote Provider.

C.2 Caching

Each InforMa stores query answers together with advertisements and registrations of local and remote Providers in a cache. In order to provide answers to, at least the expected, user queries, *i.e.*, to overcome the spatio-temporal variation of data and data source availability, an InforMa must utilize the cache in the most effective manner. Our framework supports the traditional LRU and MRU replacement algorithms; however, as shown in Section V, an InforMa must use the user profile in order to improve cache *effectiveness* – the percentage of successfully answered queries:

$$\text{effectiveness} = \frac{1}{n}\sum_{i=1}^{n} hit(query_i)$$

We model a profile in terms of beliefs, desires, and intentions, a model that has been explored in multi-agent interactions [5]. Each InforMa treats all user *beliefs* as query restrictions that are assigned *utility* and *reliability* functions, *e.g.*, one of Bob's restrictions is that he prefers Chinese restaurants. The *utility* and *reliability* functions use current time, location and other information in the profile as inputs to calculate *importance* value for comparing against other entries in the cache. Each InforMa then uses an *intention* to represent a standing query. The query is assigned a starting point and time period during which the InforMa attempts to obtain and cache an answer. The standing query also has the *utility* and *reliability* functions in order to prioritize the InforMa's activity. Lastly, the InforMa uses *desires* to represent user's wishes that cannot be directly converted into standing queries but instead require additional user-provided rules.

Similar to queries and descriptions of Providers, we annotate profiles in DAML+OIL. An InforMa employs a DAML+OIL based forward-chaining reasoning engine defined in CLIPS [32]. The reasoning engine parses the profile according to DAML+OIL and our profile DAML-S like relationships. Additionally, we have developed sample rules for converting beliefs and intentions

into query restrictions and standing queries, respectively. We use these rules for experiments in Section V, *e.g.*, one rule creates a standing query to search for gas stations while a person is driving in a car low on gas. The InforMa's reasoning engine uses these rules together with the current time, location and the profile to create standing queries and to adapt the caching strategy.

For caching, the InforMa uses the profile in two ways: (i) to preallocate space for specific data type and (ii) to assign utility value to each entry. In the first case, the InforMa uses the profile to determine types of standing queries. The InforMa uses these types to reserve portions of the cache for the related data types, *e.g.*, traffic. We use the first heuristic to create hybrid LRU+P and MRU+P algorithms. These algorithms find a cache victim entry e_i for new entry n of type D (given allocated cache E_D of maximum size max_D) according to following:

$$v = \begin{cases} e_i & e_i = \emptyset \\ e_i & \text{cache full} \wedge |E_D| \geq max_D \wedge e_i = \text{LRU}(\forall e \mid e \in E_D) \\ e_i & \text{cache full} \wedge |E_D| < max_D \wedge e_i = \text{LRU}(\forall e \mid e \in E_C \wedge C \neq D \wedge |E_C| > max_C) \end{cases}$$

Additionally, we implement a simple semantic cache algorithm S+P employing both heuristics. The algorithm also enforces space allocation for different data. The algorithm then uses the *utility* function U to replace cache entries instead of a simple timestamp-based approach. The functions are defined by a user in her profile for each query restriction and standing query:

$$v = \begin{cases} e_i & e_i = \emptyset \\ e_i & \text{full} \wedge |E_D| \geq max_D \wedge e_i \in E_D \wedge U(e_i) = U_{min}(\forall e \mid e \in E_D) \wedge U(n) > U(e_i) \\ e_i & \text{full} \wedge |E_D| < max_D \wedge e_i \notin E_D \wedge \\ & U(e_i) = U_{min}(\forall e \mid e \in E_C \wedge C \neq D \wedge |E_C| > max_C) \wedge U(n) > U(e_i) \\ \emptyset & \text{otherwise} \end{cases}$$

C.3 Discovery and Routing in Multi-Hop Networks

An important aspect of the framework is to discover local and remote Providers. The discovery allows each InforMa to construct a temporary catalog representing current data and data sources in the vicinity. Our framework supports both push and pull based approaches, *i.e.*, each InforMa can

Algorithm 2 InforMa_Route_Query($src, dst, query, r, h, i$)

```
if dst is local Provider then
    a ← find_cached_answer(query)
    if a = ∅ then
        a ← ask_provider(dst, query);
    end if
    if a ≠ ∅ then
        return a
    end if
else {dst is remote Provider}
    if src is local Consumer then
        r ← calculate_route(dst)
    end if
    if i = intercept_queries then
        a ← find_cached_answer(query)
        if a ≠ ∅ then
            return a
        end if
    end if
    h = h + 1
    if willing_to_forward and h ≤ max_hops then
        n ← next_hop_or_short_cut(dst, r)
        if n = ∅ and src is local Consumer then
            n ← broadcast
        end if
        if n ≠ ∅ then
            forward_to(n, src, dst, query, r, h)
            return
        end if
    end if
end if
return error
```

advertise its capabilities or solicit capabilities of other peers. We study the impact of different frequencies in Section V. In order to restrict the number of messages in the environment, solicitation and advertisements are, however, limited to only *one-hop* neighbors.

We employ a hybrid mechanism combining discovery and routing for routing queries (*i.e.*, data discovery) among peer InforMas in multi-hop networks. The algorithm uses a source-initiated approach similar to AODV [28] and DSR [21]. The algorithm works on a best-effort basis. It

attempts to rebuild disconnected routes; however, it does not guarantee message delivery. The pseudo-code is shown in Algorithm 2. In *data-intensive* environments the answer can often be provided by more than one device, *e.g.* cached by InforMas or available by local Providers. The querying source may not be interested in who it interacts with as long as it receives a correct answer. We modify the AODV approach to intercept routed query/discovery messages and process them when possible. Additionally, each InforMa intercepts all messages it receives or routes in order to provide *shortcuts* for cached routes. Here each InforMa maintains a route entry for *one-hop* peers. The InforMa also maintains a route entry for peers more than one hop away if those peers are used in on-going interactions or the InforMa is caching advertisements for those peers.

D. Communication Interfaces

To support multiple types of networking interfaces and to abstract these types from an InforMa, each device implements at least one Communication Interface. A Communication Interface provides a common set of interfaces for discovering neighboring devices and for communicating with them. Every Communication Interface registers its capabilities with InforMa. Accordingly, an InforMa is still *network aware* as it can infer the network constraints and requirements from the information contained in the registered capabilities.

The current implementation prototype of the framework supports two types of Communication Interfaces: (i) Bluetooth and (ii) Ad-Hoc 802.11. We use the Bluetooth protocol stack developed by Axis Communications Inc. [20] and Bluetooth modules developed by Ericsson. For the Ad-Hoc 802.11 network we use off-the-shelf wireless cards and a standard Linux communication stack.

Our design requires every device be capable of broadcasting messages, which is not supported by Bluetooth. Broadcasting in Bluetooth is restricted to messages used for device discovery. In order to exchange application level messages, a device must first establish a link level connec-

tion with its peers. Additionally, every communicating device must be either a *master* or a *slave*. Thus, simultaneous link level connections cannot be established between a pair of devices. To solve this problem we use the connect-transmit-disconnect procedure, a half-duplex communication. Accordingly, each device connects to a peer, sends a message and disconnects. For sending responses, the peer device follows the same protocol.

V. Performance Experiments

In this section, we show the improvement of the system performance of the MoGATU framework when using the semantic cache replacement algorithm S+P defined in Section IV-C.2: (i) We measure how close a cache reflects user needs using this cache replacement algorithm. (ii) We measure its *effectiveness* in answering user queries. We compare the S+P algorithm with traditional timestamp-based LRU and MRU approaches and with the extended LRU+P and MRU+P algorithms defined in Section IV-C.2. Additionally, in this section we examine the practicality of MoGATU: (iii) We measure the average response time for peer interaction over both Bluetooth and Ad-Hoc 802.11 networks and provide a preliminary validation of the data-based routing protocol defined in Section IV-C.3.

We have implemented the MoGATU framework using C language. The code runs on Linux-based Pentium II and III laptops. We provide the connectivity support for MoGATU via Bluetooth using Ericsson cards connected on serial ports on the laptops and IEEE 802.11 PCMCIA cards operating in an Ad-Hoc mode. Additionally, as future work we are developing the MoGATU framework to function as a part of the GloMoSim simulator [38] in order to enable simulations and measurements on a larger scale than currently supported by our hardware resources.

For the experiments, we either vary or measure the following parameters:

- *Query Frequency* – This represents a rate at which a user asks her mobile device an identical

question. We use the frequency to simulate a scenario where a user repeatedly asks the same question during the current activity, *e.g.*, while on the road Bob is asking for traffic conditions. For our experiments, we use two frequencies: (i) *unlimited* and (ii) 5 minutes. The unlimited frequency represents a scenario where a query is randomly placed just once during an activity. We call this scenario *Single Queries*. The second scenario allows multiple questions to be asked depending on the duration length of a particular activity. This represents a less serendipitous case since each algorithm has more chances of satisfying each distinct query. We refer to this scenario as *Repeating Queries*. We use the period of 5 minutes as the shortest activity in our simulation lasted 15 minutes.

- *Cache Size* – In our experiments the Cache Size allows an InforMa to store up to 64 different information entries. These entries represent non-expired advertisements for local and remote Providers and answers to previously issued queries. As described in Section IV, the lifetime for each entry is specified by a Provider and InforMa removes each entry after its lifetime elapses.
- *Cache Replacement Algorithm* – We use the five different cache replacement algorithms described in Section IV-C.2. These include the traditional timestamp-based LRU and MRU replacement policies, profile-based LRU+P and MRU+P and the semantic S+P algorithm.
- *Cache Allocation* – In our experiments we distinguish among eight information types: merchandise, gas, dining, directions, subway, parking, traffic and others. We measure the Cache Allocation for each type throughout the simulation in order to compare S+P algorithm with the other four approaches. For that we measure the ideal cache allocation that answers all queries a user asks. This *omniscient* scenario assumes that an InforMa has a complete prior knowledge of every step of its user. We compare the results obtained for each algorithm *a* to this omniscient case *o* using a cosine similarity *A*. We define the similarity using the following formula where vector a_t and o_t

represent the allocation for each information type for a specific time of the day t:

$$A_o(a) = \frac{a \cdot o}{\|a\|\|o\|} = \frac{\sum_{i=1}^{n} a_i o_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} o_i^2}}$$

- *Cache Hit Success Rate* – The Average Cache Hit Success Rate represents a fraction of queries that an InforMa is able to answer using its cache – the cache *effectiveness*. Initially, every InforMa has an empty cache. The InforMa can obtain data by either accepting *bulk* advertisements from other mobile devices in the vicinity or by explicitly querying them. We measure the rate using the formula defined in Section IV-C.2.
- *Cache Update Period* – The Cache Update Period represents a frequency at which each InforMa is willing to update its cache. This period represents the mobile device's preferred refresh rate in order to prolong its battery life. Additionally, this period can be used to represent a rate at which remote sources appear and disappear. We vary the period from 1 to 128 minutes.
- *Peer Query Response Time* – The Peer Query Response Time represents a period it takes for one device to query another and obtain an answer. We measure the average response time in seconds for devices querying over Bluetooth and 802.11 networks.

We use the results of these metrics to measure the implementation performance, namely we measure the effects of:

1. Cache Replacement Algorithms vs. Cache Allocation (Section V-B)
2. Cache Update Rate vs. Cache Hit Success Rate for Single Queries (Section V-C)
3. Cache Update Rate vs. Cache Hit Success Rate for Repeating Queries (Section V-D)
4. Networking Technology vs. System Performance (Section V-E)

A. Experimental Setup

To measure the effectiveness of the various cache replacement policies, we extend the motivation example from Section II. Here we simulate a twelve-hour period in which Bob, equipped with his mobile device, travels between three distinct cities to attend meetings. In our experimental scenario, Bob begins the day at 8AM by driving from one city to another while attending meetings. Some of the meetings are scheduled a priori and recorded on Bob's mobile device in a calendar application. Other meetings/activities are scheduled as the day progresses, *e.g*, dinner with Alice or finding an alternate route. Additionally, the mobile device has Bob's profile that can be converted into query restrictions and standing queries for the eight different types of information listed above. We distribute information providers with different types of information along the traveled path. These providers simulate cars, electronic lights on office buildings, gas stations, subway infrastructure and other people's mobile devices. Some of these providers hold data closely related to Bob's current activity while other providers have *less useful* data.

B. Cache Replacement Algorithms vs. Cache Allocation

In the first experiment, we measure how profile knowledge affects the cache allocation for the eight different data types defined above. We use eight types of information as opposed to smaller values because it allows us to better analyze the cache behavior for different replacement algorithms. As we show in the next two experiments, a cache preallocation plays an important role in improving cache hit success rate for both single and repeating queries.

We measure the cache allocation during the twelve-hour period for the standard timestamp-based LRU algorithm and the semantic S+P algorithm defined in Section IV-C.2. We obtain the measurements by recording a snapshot at one minute intervals. For the LRU algorithm we use numbers reflecting actual data in the cache while for the S+P algorithm we use numbers inferred by the In-

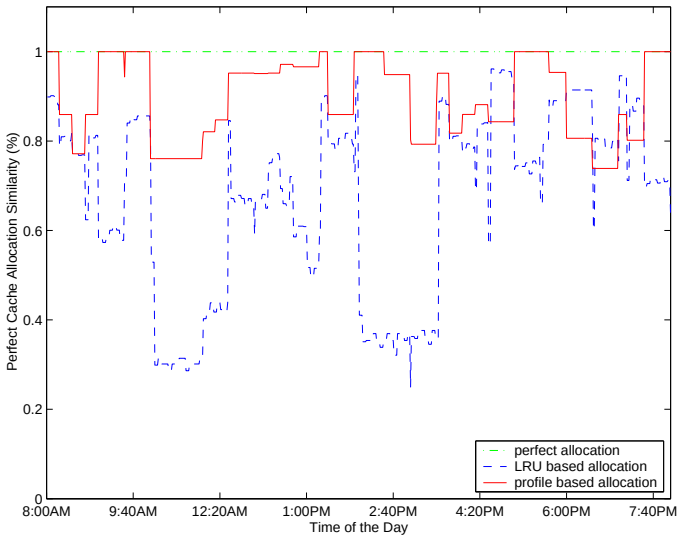

Fig. 2. Effects of Cache Replacement Algorithms on Cache Allocation

forMa. Each snapshot is described by an eight-dimensional vector representing number of entries allocated for each specific data. Accordingly, we obtain a set of 720 vectors for each algorithm. We also measure the optimal cache allocation from the *omniscient* scenario. We use the optimal allocation as a base for comparing the LRU profile-less and S+P profile-based algorithms. For that we use the cosine similarity formula. We do not measure cache preallocation for approaches that use profiles with static utility values [8]. These profiles do not change with context. Hence, their initial cache preallocation remains the same from the start until the end of a simulation.

Figure 2 shows the results. The semantic S+P cache replacement algorithm is, on average, 89.5% similar to the optimal allocation, while the traditional LRU algorithm is only 67.1% similar. The positive results of S+P based cache allocation are partly due to the fact that the mobile device knows or will learn throughout the simulation about most of the meetings/activities. The allocation for the LRU algorithm is, on the other hand, in many cases quite different from the optimal case, sometimes only 29.9% similar. Moreover, the LRU algorithm frequently fluctuates its cache allocation. This is due to the fact that the LRU approach ignores the information types. Instead it is based on timestamping only. This way the LRU-related allocation depends on the order of received

information as well as the expiration time of each entry. This may cause an important information, which is later needed, to be removed from the cache prematurely. Consequently, this will cause a cache miss. Hence, to overcome the cache-allocation serendipity, it is necessary to utilize context and profile knowledge in order to narrow the content of a cache at each point of time.

C. Cache Update Rate vs. Cache Hit Success Rate for Single Queries

In the next experiment, we measure the cache *effectiveness* in answering user single queries. We vary the period at which each mobile device can update its cache from 1 to 128 minutes. This period represents the device's preferred refresh rate in order to prolong battery life. Additionally, the period indirectly represents how much work each mobile device must perform. During the simulation period, we assume that the person asks at most four distinct single queries during each activity. This results in 54 unique queries that Bob asks during the twelve-hour period. To improve chances of answering these queries we, however, pose a restriction that a query can be asked only when the device has previously received a valid answer and, if cached, the answer has not expired. This way we guarantee that with an optimal performance each query can be satisfied. Figure 3 illustrates the performance for caches employing the traditional LRU and MRU approaches, the hybrid LRU+P and MRU+P approaches, and the semantic S+P approach.

As shown in Figure 3, the S+P algorithm always outperforms other techniques. This is expected since the algorithm uses the context and user-profile knowledge to preallocate cache space and to assign utility values for each entry. This way the algorithm is able to, for example, allocate enough cache space for traffic-related information while the person is driving. Moreover, the algorithm assigns different values to each traffic update and caches only the most valuable ones, resulting in a higher success hit rate. We see that its performance degrades with larger update periods. This is also expected since here the activity changes more frequently than the cache can reflect.

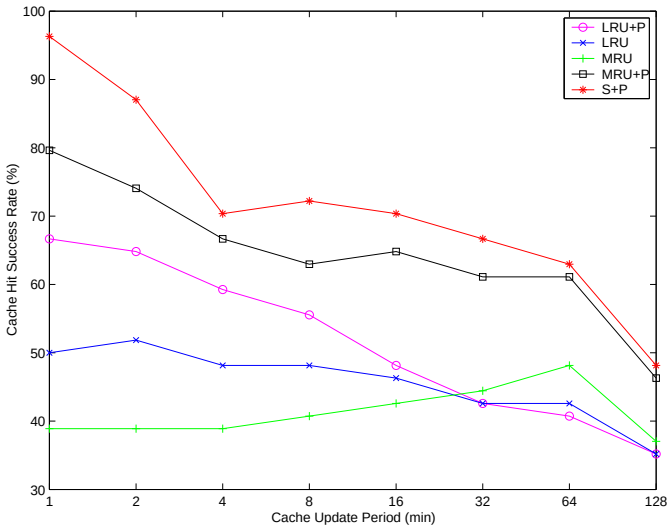

Fig. 3. Effects of Cache Update Rate on Cache Hit Success Rate for Single Queries

Additionally, the timestamp-based LRU algorithm outperforms the MRU approach. At its peak, at 2 minute cache update intervals, the LRU is able to satisfy 51.9% of user queries. By extending it to utilize the profile knowledge, *i.e.*, by using LRU+P approach, the cache would, however, improve its performance by 12.9%. Moreover, a cache using S+P algorithm is 87.0% successful for the same update period – an improvement of 45.1%. Moreover, we see a stable performance for most cache replacement policies for update intervals between 4 and 16 minute. Consequently, a mobile device can prolong its battery life by using a larger update interval and still maintain a performance level equivalent to 4-minute update periods.

D. Cache Update Rate vs. Cache Hit Success Rate for Repeating Queries

In this experiment, we also measure the cache *effectiveness* in answering user queries. We, however, extend the previous experiment by asking repeating queries instead of single queries. This experiment represents a less serendipitous case where each algorithm has more chances of satisfying each distinct query. A repeating query is a distinct query asked more than once per activity. We use a query frequency equivalent to 5 minutes in order to guarantee that each query

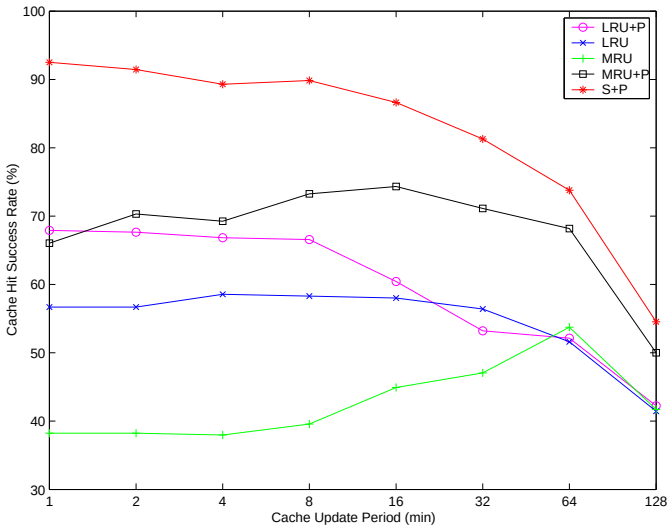

Fig. 4. Effects of Cache Update Rate on Cache Hit Success Rate for Repeating Queries (5 min interval)

is asked at least three times given an activity, and the shortest activity in our simulation lasted 15 minutes. We show the measured cache hit success rates for each algorithm in Figure 4.

Similar to the previous experiment, the semantic-based S+P algorithm has the best performance. It has on average 1.5 times larger success rate than the LRU algorithm, which is the best *profile-less* approach. Each profile-based cache replacement algorithm, in fact, outperforms the traditional timestamp-based techniques. This is expected since the profile-based algorithms use context and user-profile knowledge to preallocate cache space and S+P uses the knowledge to even assign utility values for each entry.

E. Networking Technology vs. System Performance

In the last set of experiments, we test the effects of hardware on the framework and its performance in real world experiments. We are primarily interested if the implementation is feasible using the current hardware technology. Accordingly, in these experiments we examine the interaction between peer laptops over Bluetooth and 802.11 networks in Ad-Hoc modes. To provide similar conditions for both technologies, we assume a half-duplex interaction employing the connect-send-

disconnect mode. Additionally, each exchanged message has an average size of 1.0KB.

Our first set of real-fielded experiments simply validate the working of the system, namely we test the routing among devices. The test consists of four laptops interacting over a period of 100 minutes. In this experiment, device *A* is able to communicate with device *B* that, in turn, is in range of devices *C* and *D*. Device *A* provides weather information and device *D* has information about locations and prices of nearby gas stations. We evaluate the system by randomly selecting a query and assigning it to one of the four devices and monitoring information present at each cache. For example, when device *A* asked for the closest gas station, it was able to deduce that *D* contains the required information and that the query should be routed through *B*. Moreover, upon receiving the query *B* was able to immediately return a cached answer instead of first routing the request to device D. In this experiment, we did not perform any scalability measurements because of the limited number of available hardware. As future work we are, however, converting the MoGATU framework to function as a part of the GloMoSim simulator [38] in order to enable simulations and measurements on a larger scale than currently supported by our hardware resources. Moreover, for our future work we will also develop additional mobility models to better represent devices moving in a city-like environment, *e.g.*, cars and people.

Next, we study the impact on performance for reasoning over the cache entries by each InforMa in comparison to transmission time. In the current implementation an InforMa linearly scans the cache to find a matching DAML+OIL structure, *i.e.*, a Provider advertisement or a cached answer. For a 30K cache, however, the processing time was on average 5ms per query after 100 runs. On the other hand, the measured network transmission completely dominates this time. In Bluetooth environments it takes 4.56s to transfer a 1.0KB “query” and to send a response. This is partly due to the delay it takes to establish a connection between two peers and subsequently to disconnect.

For the much faster 802.11 devices, the time needed to send the query and receive the answer over the network was on average 27ms. This is, however, still five times longer than an InforMa spends on reasoning over its cache. Therefore, at this moment the cache reasoning does not impose a bottleneck on the system performance; however, the networking does. From the measured results it appears that the Bluetooth technology cannot be used for serendipitous querying in environments reflecting a high mobility, such as cars in a city environment traveling at 25mph. This is due to the fact that such mobility prevents peer devices from establishing a communication link before they are out of range for both querying and routing purposes. At the same time, the Bluetooth technology is fast enough to allow exchanges and interactions in relatively stable environments, *e.g.* people in a mall or cars traveling in the same direction.

VI. Related Work

The problem of data management in a distributed environment has been well researched, both in terms of wired infrastructure and infrastructure-based wireless networks. The work on distributed and federated databases is also well-known [27]. Accordingly, we present work related to data management in wireless networks and a short discussion of related work on user profiles.

Data Management in Wireless Networks: The problem of data management in wireless networks has drawn a significant degree of attention. The proposed solutions primarily address problems imposed by the underlying networking technology, such as low bandwidth and high probability of disconnection. They also address the issues related to the retrieval of location dependent information. Existing solutions often rely on the support of a fixed, wired infrastructure. These solutions place primary data on servers located within the wired infrastructure and treat mobile devices solely as clients. Chrysanthis *et al* [25] consider disconnected operations within mobile databases by presenting a mechanism, referred to as a “view holder” that maintains versions of

views required by a particular mobile unit. They also propose an extension to SQL that enables the profile- and capability-based programming of the view holders. Kottkamp and Zukunft [24] present optimization techniques for query processing in mobile database systems that include location information. They present a cost model and different strategies for query optimization incorporating mobility specific factors like energy and connectivity. Bukhres *et al* [6] propose an enhancement to the infrastructure-based mobile network model of Mobile Hosts (MHs) connected over a wireless virtual subnet and Mobile Support Stations (MSSs) connected to a wired static network. They recommend the addition of a mailbox serving as a central repository for the MHs that is maintained by the cellular provider and duplicated in all the MSSs. Pitoura [29] presents a replication schema based on augmenting a mobile database interface with operations with weaker consistency guarantees. Demers *et al* [12] present the Bayou architecture, which is a platform of replicated, highly available, variable-consistency, mobile databases for building collaborative applications.

In contrary to these approaches, our work assumes no support from the fixed infrastructure. When a mobile device requires instantaneous information (*e.g.* traffic updates or bad weather warnings), it may be more easily, or only accessible, from other "local" mobile devices and not from a fixed node. Moreover, a mobile device in our work is always in nomadic mode [6] and [24].

User Profiles: The data management community of late has been advocating the use of *profiles*, especially when dealing with pervasive systems and stream data. Ren and Dunham [31] represent a profile as a collection of continuous location dependent data queries. The location dependent data is described in terms of tuples in a single-relational database, *e.g.* all hotels and restaurants in a city. A user specifies her preferences by constructing several SQL queries based on the database schema. In a seminal work, Cherniak *et al* [8], [9] explore the use of profiles in the area of client/server based data recharging for mobile devices. They discuss the requirements for a successful profile as

well as describe the need for a formal language that enables expressing such profiles. Their profile consists of two sections: (i) a *domain* responsible for the data description and (ii) a *utility*, which is a numerical function denoting the data importance with respect to other information. The utility function is, however, immune to the current context.

While a step in the right direction, we argue that a profile explicitly enumerating data and its utility is not sufficient. As described in Section IV-C.2, we extend the profile in terms of *beliefs*, *desires*, and *intentions*. Here the notion of "domain" and "utility" is subsumed by the notion of "beliefs". Domains of interest, and their utility, are thus inferred from beliefs, desires, and intentions, as modulated by the current context. The domains, as well as their utilities, vary over time and context. This allows us to better adapt to the dynamic nature of the environment, such as constructing standing queries or changing cache replacement policies. Our model also allows us to better predict the future actions of the user and does framework can be more proactive.

VII. Conclusions

The constant enhancements in capabilities of palmtop, embedded and wearable devices, together with the advent of pervasive connectivity, present a new paradigm for the way we think about interaction among devices. These devices will become both sources and consumers of information, and will be able to cooperate with other devices in their vicinity in order to pursue their individual and collective tasks. The emerging data-intensive pervasive computing environment, however, challenges the traditional data management models that were proposed for wired and wireless infrastructure-based networks. In addition to issues addressed by mobile distributed data management systems, the data and data source availability is no longer fixed. Instead it varies with location and time. Other issues include lack of a global catalog and schema, no guarantee of reconnection among peers, no guarantee of collaboration among peers, and the issues of commits

and aborts due to the serendipitous nature of the environment.

We have designed a data management framework that is applicable to the *data-intensive* pervasive computing environments. The framework treats all devices as peers and operates on a best-effort basis. The framework is network-agnostic as it can operate over any wireless or wired technology. In our current implementation prototype, the framework supports both 802.11 and Bluetooth networks. Additionally, the framework attempts to utilize as much of available information as possible to enhance the mobile device's functionality. It uses both static information, such as user's preferences and information about data sources, and dynamic information, such as current context description, to allow each device to behave proactively.

We have measured and compared the effectiveness of various cache replacement policies that can be used by each mobile device to cache currently available information. We have also shown that the implementation of the framework is feasible using current hardware technologies by implementing it on laptop computers equipped with IEEE 802.11 and Bluetooth networks. Moreover we have tested the networking aspect of the system by measuring the communication delays for the 802.11 and Bluetooth networks, and by verifying our simple query-based routing algorithm. We have, however, not performed any routing scalability measurements because of our limited number of available hardware.

For future work, we will convert the MoGATU framework to function as a part of the GloMoSim simulator [38] in order to enable simulations on a larger scale. As part of the GloMoSim-based simulator we will develop novel mobility behaviors to better represent devices moving in a city-like environment. Additionally, we will use the simulator to explore the related aspects of transactions, joins and more advanced data-centric routing approaches. We will also investigate the problem of expressing profiles and context information, which are an integral part of our system.

REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: data management for asymmetric communication environments. In *Proceedings of ACM SIGMOD*, pages 199–210, 1995.

[2] H. Bharadvaj, A. Joshi, and S. Auephanwiriyakyl. An Active Transcoding Proxy to Support Mobile Web Access. In *Proc. of the IEEE Symposium on Reliable Distributed Systems*, October 1998.

[3] Bluetooth SIG. The Bluetooth Specifications. http://www.bluetooth.com/.

[4] C. Bobineau, L. Bouganim, P. Pucheral, and P. Valduriez. PicoDBMS: Scaling down Database Techniques for the Smartcard. In *Proc. of the 26th International Conference on Very Large Databases*, 2000.

[5] M. Bratmann. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[6] O. Bukhres, S. Morton, P. Zhang, E. Vanderdijs, C. Crawley, J. Platt, and M. Mossman. A Proposed Mobile Architecture for Distributed Database Environment. Technical report, Indiana University, Purdue University, 1997.

[7] D. Chakraborty, F. Perich, S. Avancha, and A. Joshi. DReggie: Semantic Service Discovery for M-Commerce Applications. In *Workshop on Reliable and Secure Applications in Mobile Environment, SRDS*, October 2001.

[8] M. Cherniak, M. Franklin, and S. Zdonik. Expressing User Profiles for Data Recharging. *IEEE Personal Communications*, July 2001.

[9] M. Cherniak, E. Galvez, D. Brooks, M. Franklin, and S. Zdonik. Profile Driven Data Management. In *28th International Conference on Very Large Databases*, August 2002.

[10] The DAML Services Coalition. DAML-S: Semantic Markup For Web Services. http://www.daml.org/services/, 5 2001.

[11] DARPA. DARPA Agent Markup Language + Ontology Inference Layer (DAML+OIL). `http://www.daml.org.`

[12] A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and B. B. Welch. The Bayou architecture: Support for data sharing among mobile users. In *Proceedings IEEE Workshop on Mobile Computing Systems & Applications*, pages 2–7, Santa Cruz, California, 8-9 1994.

[13] M. Dunham and A. Helal. Mobile Computing and Databases: Anything New? *SIGMOD Record*, pages 5–9, 1995.

[14] M. Franklin. Challenges in ubiquitous data management. In *Informatics*, pages 24–33, 2001.

[15] T. Gaasterland and J. Lobo. Qualified Answers That Reflect User Needs and Preferences. In *VLDB*, 1994.

[16] D. Goodman, J. Borras, N. Mandayam, and R.Yates. INFOSTATIONS : A New System Model for Data and Messaging Services. In *Proc. of IEEE VTC'97*, volume 2, pages 969–973, 1997.

[17] J. Grant, J. Gryz, J. Minker, and L. Raschid. Semantic Query Optimization for Object Databases. In *ICDE*, 1997.

[18] IEEE 802.11 Working Group. Cool. `http://ieee802org/11`.

[19] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, pages 352–372, May/June 1997.

[20] Axis Communications Inc. OpenBT: An Open Source Bluetooth Stack For Linux. `http://sourceforge.net/projects/openbt/`, 2001.

[21] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[22] A. Joshi. On Proxy Agents, Mobility and Web Access. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2000.

[23] L. Kagal, J. Undercoffer, F. Perich, A. Joshi, and T. Finin. A Secure Architecture Based on Trust Management for Pervasive Computing Systems. In *Grace Hopper Celebration of Women in Computing*, 2002.

[24] H. Kottkamp and O. Zukunft. Location-Aware Query Processing in Mobile Database Systems. In *Selected Areas in Cryptography*, pages 416–423, 1998.

[25] S. Lauzac and P. Chrysanthis. Utilizing Versions of Views within a Mobile Environment. In *Proceedings of the 9th International Conference on Computing and Information*, 1998.

[26] *NFS IDM 2001 Workshop on Information and Data Management*, 2001.

[27] M. Oezsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, Inc., New Jersey, 2nd edition, 1999.

[28] C. Perkins and E. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[29] E. Pitoura. A Replication Schema to Support Weak Connectivity in Mobile Information Systems. In *Database and Expert Systems Applications*, pages 510–520, 1996.

[30] O. Ratsimor, V. Korolev, A. Joshi, and T.Finin. Agents2Go: An Infrastructure for Location-Dependent Service Discovery in the Mobile Electronic Commerce Environment. In *ACM Mobile Commerce Workshop*, July 2001.

[31] Q. Ren and M. Dunham. Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. In *ACM MobiCom'00*, 2000.

[32] G. Riley. CLIPS: A Tool for Building Expert Systems. `http://www.ghg.net/clips/`.

[33] M. Stonebraker. Position Paper on Monitoring Applications. In *NSF Workshop on Context-Aware Mobile Database Management (CAMM)*, January 2002.

[34] C. Tait, H. Lei, S. Acharya, and H. Chang. Intelligent File Hoarding for Mobile Computers. In *Proc. of the First ACM International Conference on Mobile Computing and Networking - MobiCom'95*, 1995.

[35] J. Undercoffer, F. Perich, A. Cedilnik, L. Kagal, and A. Joshi. A Secure Infrastructure for Service Discovery and Access in Pervasive Computing. *ACM MONET: Special Issue on Security in Mobile Computing Environments*, 2002.

[36] UWB Working Group. The Ultra Wireband Technology. http://www.uwb.org/.

[37] B. Yang and H. Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. In *27th VLDB Conference*, 2001.

[38] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Workshop on Parallel and Distributed Simulation*, 1998.