Ïnfïnïty : A Generic Platform for Application Development and Information Sharing on Mobile Devices

Alvin Cheung, Tyrone Grandison, Christopher Johnson
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
{alvin, tyroneg, johnsocm}@us.ibm.com

Stefan Schönauer*
University of Helsinki
PO Box 68, 00014 Helsinki, Finland
schoenauer@acm.org

ABSTRACT

Personal mobile devices, such as cell phones and PDAs, are undergoing continual improvements in computing power, storage, and connectivity. This trend presents many opportunities for information sharing and collaboration among massively distributed data sources. However, the current mobile environment, which consists of incompatible operating systems, development models, and communication options, has hindered application development and data sharing over ad hoc networks of mobile devices. In this paper, we present a middleware framework, called Ïnfïnïty, that unifies techniques for local (and remote) device data access, automatic communication channel selection, and application deployment across heterogeneous platforms. Ïnfïnïty enables information sharing and collaboration among mobile devices in a privacy-preserving manner. We describe the prototype implementation of Ïnfïnïty and demonstrate how it eases application development and supports a wide variety of application scenarios.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Domain-specific architectures

General Terms

Design, Management

Keywords

Mobile Middleware, Information Sharing, Collaboration, Privacy Preservation

1. INTRODUCTION

Mobile data sources have become truly ubiquitous. Cell phones and PDAs feature rapidly growing storage capacities. These devices no longer contain only addresses and phone numbers, but also pictures, music, and various other personal data. In addition, they offer a wealth of connectivity options, such as GPRS, WLAN, and Bluetooth. The increasing amount of data stored on these devices presents great opportunities for information sharing. For example, medical institutions would like to access patient health information generated by pervasive monitoring devices for treatment and research purposes, automobile guidance systems would benefit from more accurate and up-to-date information about traffic congestion, and rescue workers would like the ability to search current information about potential victims at a disaster site. Each of these scenarios requires the exchange and analysis of recent, location-dependent information. However, there are many obstacles to the seamless sharing of information stored on mobile devices, including: (1) heterogeneity of data formats, (2) differences in communication means and protocols, and (3) privacy concerns of data subjects.

Application development is also difficult in these mobile ad hoc networks. Because mobile devices use various operating and storage systems, application developers must accommodate many different platforms to enable widespread data sharing. This also complicates deployment of applications, since every user must install a system-dependent application for his or her own device.

To overcome these challenges, we propose a middleware platform, called Ïnfïnïty, that facilitates application development for data sharing and distributed query applications on mobile devices. Ïnfïnïty translates among various data formats, automatically chooses appropriate communication channels, enables privacy-preserving query processing and data sharing across multiple devices, and provides a device-independent means to share and deploy applications.

1.1 Example Scenarios

To illustrate the utility of Ïnfïnïty, we describe three example scenarios involving application development and information sharing in ad hoc networks of mobile devices.

Evacuation Routing Suppose that an enterprise would like to provide its employees with a service on their mobile devices to facilitate evacuation routing in case of an emergency. The employees could install a simple application on their cell phones or PDAs that contains a map of the building in which the employee is situated, including the locations of available exits. The evacuation application could query other local devices to determine the location and concentration of evacuees in each exit area to allow employees to bypass congested areas. The aggregate query results returned to each mobile device would provide current and reliable information about the status of evacuation, allowing each employee to find the most expedient route out of the building. The company or its employees could also send the evacuation routing application, and current mapping information, to new or non-resident employees upon entering the building.

*Research was done while author was with IBM Research.

MobiDE'07, June 10, 2007, Beijing, China.

Recommendation Service A second application of Ïnfïnïty is to support recommendation services for restaurants, automobile repair shops, and other service-oriented businesses. Similar recommendation of services are popular on the worldwide web, but may not contain accurate or up-to-date information. They also may not be available to users at the time they need them, such as to find a restaurant or repair shop on the road in an unfamiliar area. Further, many users may not bother to input detailed reviews or ratings long after the fact. As an alternative, mobile device users could install an application containing the GPS locations of certain businesses, such as restaurants. They could then bookmark and rate the restaurant immediately after completing their meal using a simple location-aware interface. Other users of the recommendation service could query local devices to see how many local users have bookmarked a particular restaurant and obtain a composite rating of the restaurant from others in the area. This would provide an immediate answer to the user's query based on recent and reliable information from other mobile device users.

Automobile Navigation A third application of Ïnfïnïty is to improve the accuracy of automobile navigation services. Suppose that a car navigation system uses stored map information to determine the fastest route for a specified trip. It relies on recent and accurate information about traffic density on the current route and alternative routes to adjust the route to current traffic conditions. Current systems rely on passively collected traffic density information provided by a central service. However, navigation systems can plan more efficient routes by exchanging traffic information with navigation devices in numerous other cars in the area. Each car can query neighboring cars about traffic information on the road ahead, forwarding queries to other cars within communication range, as appropriate. This would provide timely and location-specific information, allowing the system to gather density information as well as intended routes of other travelers. This would lead to significantly more accurate traffic models and better routing decisions.

These scenarios underscore the need for new technology to enable many different mobile data sharing applications.

1.2 Paper Organization

In section 2, we define the requirements for the Ïnfïnïty middleware platform to enable this seamless query processing in a network of massively distributed data sources. In section 3, we discuss the state of the art in mobile computing and the technical primitives required for this next generation middleware. We present the Ïnfïnïty system architecture in section 4 and describe our prototype implementation in section 5. We outline future research directions in section 6 and conclude in section 7.

2. REQUIREMENTS

To support easy development of applications, the Ïnfïnïty middleware platform must meet several requirements. First, it must provide a communication interface to allow data transfer among devices. This involves transforming the data into a platform-independent format, ensuring reliable data transfer, and automatically choosing the optimal communication channel (*e.g.* GPRS, Bluetooth) for other devices and applications.

Second, the middleware must support transparent local and distributed data access in a uniform manner. Thus, the data access component must be able to determine which remote data sources to contact, based on the query and the current system context.

Third, as mobile devices only have a limited number of applications installed at any given time, users should be able to share applications among devices to facilitate collaboration and information sharing for arbitrary tasks. To reduce the risk of distributing viruses or other malware, applications should not be able to access restricted information and should only execute allowed operations. The middleware should also provide a means of transferring applications and installing them on remote devices. It should provide the user with reliable information about the application so the user can make an informed decision regarding whether to allow those operations.

Fourth, because mobile devices store sensitive personal information, privacy protection and selective information sharing are essential for a data sharing middleware platform. Privacy policy enforcement should be transparent to the query processing and any other application. It should also be flexible enough to support complex privacy rules that are scalable to large networks.

Finally, the data sharing middleware should not interfere with the normal device operation or require installation of a completely new software environment. Instead, it should leverage the mobile device's existing hardware and software infrastructure.

3. RELATED WORK

Although mobile ad hoc networks are an active area of research, the challenges of developing privacy-preserving, location-aware data sharing applications have only been explored in other contexts.

CarNet [5] is an example of an ad hoc wireless network architecture for traffic data exchange among automobiles. While it supports data exchange among mobile devices and forwarding of queries to the appropriate data sources, CarNet relies on special hardware and is restricted to specific applications. Further, although it addresses the privacy problem of tracking the movement of mobile devices, the CarNet architecture does not address privacy problems associated with the shared data, as in Ïnfïnïty .

Internet peer-to-peer systems [2, 7, 3] address the problem of accessing data in a volatile set of users. Research on peer-to-peer systems has also explored data privacy concerns [2, 3], but these systems rely on a constantly available network infrastructure and global indexing services. Such privacy measures are unsuitable for mobile ad hoc networks with the constantly changing connectivity of devices. Also, peer-to-peer systems typically do not address application sharing.

There are several systems for service discovery in pervasive environments, including MSDA [6] and Jini [8]. These systems enable service discovery for applications following a client-server model in heterogenous environments, but do not address data sharing support for applications on top of their systems or the associated privacy concerns. Jini does include the means to share applications among partners.

Sensor databases and systems, such as IrisNet [**?**], web service platforms [**?**, **?**], specialized mobile application platforms, for activities like emergency evacuation and route planning, and collaborative filtering technology for mobile environments assume the consolidation of data, from many sources, into a centralized coordination module or repository. This design choice, which is not made in the Ïnfïnïty middleware, limits the usefulness of related technology when this central service is down, due to natural or man-made disasters.

Enforcement of privacy-preserving access and disclosure policies in relational databases has been introduced in [1, 4]. We leverage similar techniques for this work, which represents a novel application in the ad hoc mobile network space; as other approaches to privacy preservation enable controls at the application rather than the data level.

The main differentiators of Ïnfïnïty over all the related work is in the articulation of how to enable a web-styled application platform, the cohesive integration of the different technologies needed

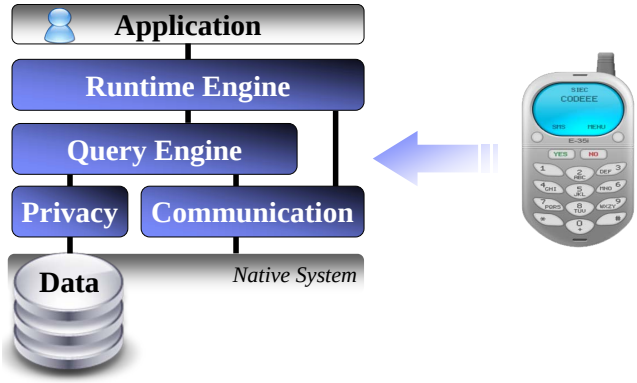

Figure 1: System Architecture of the Ïnfïnïty Middleware.

to create the foundation of a powerful, agnostic mobile device and the inclusion of novel technology to address privacy concerns. This also represents this paper's contribution.

4. SYSTEM ARCHITECTURE

To address the requirements outlined in Section 2, we propose a new middleware architecture that consists of: (1) a communication module, (2) a runtime engine, (3) a query processor, and (4) a privacy enforcement engine. These modules each address a specific requirement outlined above. Together, they provide a comprehensive framework for easy development and deployment of privacy-preserving, data sharing applications in mobile ad hoc networks. But the components can also be used individually to support development of native applications for specific platforms.

Figure 1 illustrates the Ïnfïnïty solution architecture. We describe the component parts and their interdependencies in the following sections.

4.1 Communication Module

The communication module provides information exchange capabilities to the other Ïnfïnïty modules, including methods to transform objects to and from a format that is device and operating system independent. This format supports all data types that Ïnfïnïty applications exchange and the applications themselves. This is necessary to transmit information among different hardware and software platforms.

The communication module also provides resource discovery. It monitors the available partners on different communication channels and provides resource information to the other modules to make appropriate routing decisions. Information about the available communication partners is important in ad hoc networks because the set of partners can be extremely volatile over time.

The communication module further ensures that the transmitted data is correct and provides the results of transmissions to the query module or the runtime module through a publish-subscribe mechanism. Each module maintains two queues – one for incoming messages and another for outgoing messages. These queues allow information transfer among the communication module, the query module, and the runtime module.

Finally, the communication module transparently chooses the locally optimal channel for data transmission, based upon the capabilities of the recipient and channel characteristics such as cost, available bandwidth and reliability.

4.2 Query Processor

The main function of the query processor is to provide transparent data access. Transparency is essential for application development because it obscures the complexity of the network structure from the developer. The query processor receives queries from the runtime engine, a native application, or the communication module (in the case of remotely initiated queries). It then enables local and remote applications to process these queries, irrespective of the local data representation and the current storage location of the requested data. The query processor supports restrictions on queries of the local data store when the application only needs access to local data.

To achieve transparency, the query language must be declarative rather than procedural. That is, queries should describe what data needs to be retrieved rather than the method of retrieval. The Ïnfïnïty prototype uses SQL as its query language. We extended the query syntax with an optional parameter to indicate either local or global data access.

Transparent data access also requires a virtual global data schema so that queries can be formulated without precise knowledge of the distributed data placement. Since a global data schema cannot be sufficiently comprehensive to cover all possible data objects for any possible application, it must be seamlessly extended with application specific data. We accomplish this by allowing applications to define additional relations, which the query engine automatically creates at the first invocation of the application. Query engines in devices without the specific application installed will simply return empty results for any queries of those relations. The names of these extensions must be globally unique, e.g., by extending the relation names with the application name and version.

Although Ïnfïnïty uses a global data schema to allow transparent data access, mobile devices can store data in a different local schema. Indeed, this will be the most common case, as various mobile operating systems and platforms store information in different formats. Therefore, the query engine translates queries formulated against the global data schema into ones formulated against the local data schema. It then translates the results back into the global schema.

The actual processing of a query also contributes to transparent data access. Upon receiving a query from the runtime engine (or communication module, in the case of a remotely initiated query), the query processor analyzes the query to determine whether it can be answered completely with local data. If so, the query processor translates the query into the local data schema and forwards it to the privacy enforcement module, which rewrites the query to be compliant with the privacy policy (avoiding violations) and then executes the query on the local data (Step 3a in Figure 2).

If remote data sources are necessary to answer the query, the processor splits the original query into two parts. The first part includes the portion of the query that can be answered with local data and is treated like any other local query. The second part includes those portions of the query that require access to remote data. The query processor determines which data sources are necessary to answer the query, exploiting locally available information such as the current availability of other devices. It then augments the query to contact the minimum necessary remote devices and forwards the augmented query to the communication module. The module forwards the query to the appropriate remote devices and returns any results to the query engine (Steps 3b and 4 in Figure 2).

After receiving the local query results and any remote results, the query engine unites these partial results. This may also entail post-processing steps, such as calculating aggregates over the partial results. The engine then translates the united results back into

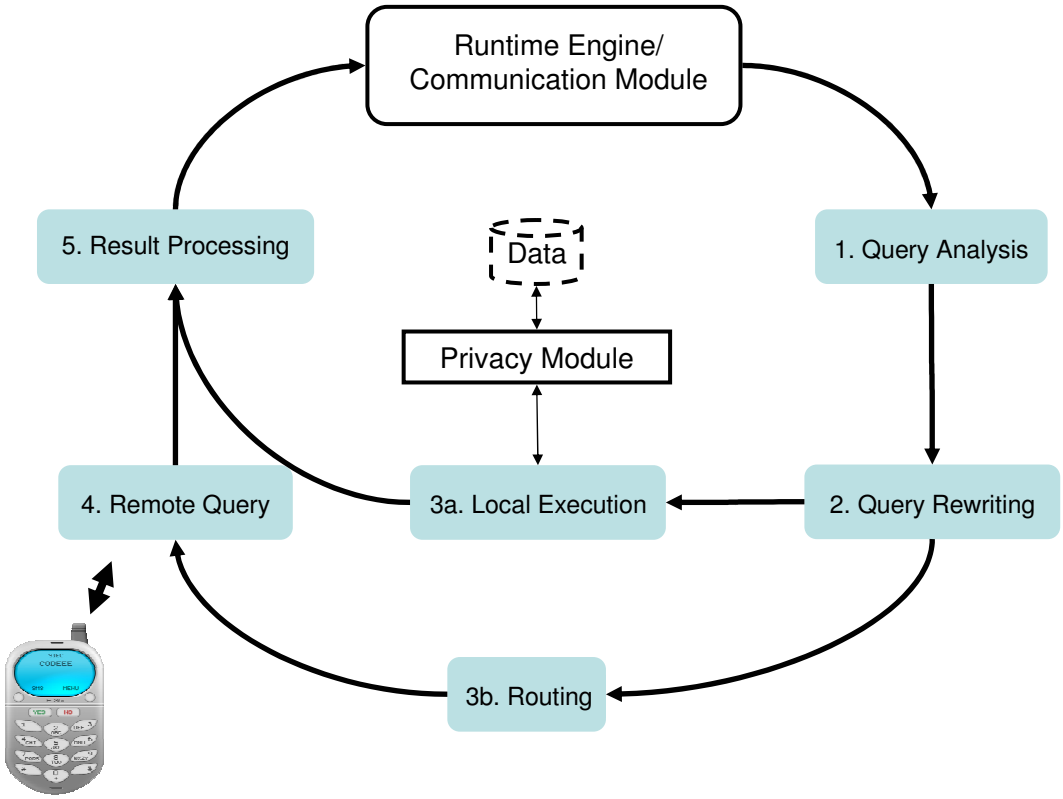

Figure 2: Query processing in the Ïnfïnïty middleware.

the global schema and returns them to the query source, *i.e.* the runtime engine or the communication module.

We refer to this method of executing a query locally to the extent possible and then passing the remaining parts of the query to the appropriate remote devices as "process and forward." This method results in queries being propagated through the network of devices, with the query results returned to the originator.

4.3 Runtime Engine

As identified in Section 2, one of the design goals of Ïnfïnïty is the ability to share applications across devices regardless of their hardware or operating system. This requires two functions. First, applications must be specified in a platform-independent way to enable transfer. Second, each device must contain an environment to execute these applications. Both functions are performed by the Ïnfïnïty runtime engine.

To enable these functions, we designed an application model similar to the web services model. Each runtime engine acts like a web server, listening on a configurable local port for HTTP requests and delivering content through this port. All applications are web applications written in HTML and JavaScript, running inside any web browser available on a mobile device. The applications and the runtime engine exchange queries and results using standard HTTP messages. To further simplify application development, Ïnfïnïty also provides a set of JavaScript constructs that aid in the communication with the runtime engine and ease the processing of query results by the application.

An application must be accompanied by a machine-readable XML specification file to be installed on the Ïnfïnïty middleware. This specification contains the description of any data schema extensions used by the application as well as a description of all of the queries the application may generate. The runtime engine reads and processes the application specification upon invocation. It then ensures that the schema extensions are created locally, if necessary, and sets up the runtime environment for the application.

The second function of the runtime engine is to package applications and deploy them on remote devices using the communication module. This function may be invoked by the user. It may also be invoked automatically if, for example, a remote device is unable to process an information request because certain schema extensions are not available on the device. In this case, the runtime engine on the local device would bundle the application specification and the actual application and send them to the remote device via the communication module. Once the application has arrived at the remote device, the runtime engine on the remote device inspects it and specially processes the application specification. It then presents the results of this analysis to the user of the remote device and asks for permission to install the application. To prevent the user from being overwhelmed by application installation requests, the middleware can be configured to block certain or all such requests.

Since the runtime engine only allows the application to issue queries listed in the specification file, the user can decide, based on those queries, whether to reveal information sought by the application. Alternatively, the user can amend her privacy policy to restrict data access at a finer granularity or completely deny the installation of the particular application. If the user grants permission, the received application is stored on the device and started normally through the runtime engine. The mobile device can then participate in the newly installed application.

4.4 Privacy Enforcement

The privacy enforcement module is a key component of the Ïnfïnïty architecture, particularly in the volatile environments of mobile ad hoc networks. Ïnfïnïty provides reliable privacy enforcement by ensuring that all access proceeds through its privacy enforcement module.

To be useful and effective in mobile ad hoc networks, privacy preferences should be enforced automatically with minimal user intervention. Otherwise, the constant adaptation of privacy settings due to network and application changes would make the system unusable. For this reason, we have designed a policy-based privacy enforcement module for the Ïnfïnïty middleware. The privacy module automatically enforces user-defined privacy rules in determining whether to grant access to remote applications.

A privacy policy is a set of rules specifying the conditions under which certain data may be accessed and disclosed. The policy states which information may be accessed by whom, for what purposes, and under what conditions. It must be possible to specify individual privacy rules in broad categories or on a fine-grained basis. While broad categories are often sufficient, fine-grained controls may be necessary to implement more specific privacy preferences. For example, a broad rule may prohibit sharing of information in the personal address book, but a more specific rule may allow sharing of certain addresses with specified family members. Broad categories are used to design rules for the base cases, while more specific rules may be defined for specific persons or data.

Purpose specification is also important in designing privacy policies. It allows owners to control access to their data further based upon the intended use of the data. This purpose can usually be inferred from the application requesting access to the data. For example, a user could specify that personal addresses are not to be shared other than for emergencies. If a rescue vehicle sought access to a particular address, the policy engine could infer an emergency purpose in deciding whether to grant access. To allow sharing in ad hoc networks, purpose classes would have to be widely known and therefore standardized. To overcome the limitations of a small set of standard purposes delivered with the base installation of the middleware, users can add more fine-grained conditions to the privacy rules.

Ïnfïnïty uses a negative base policy, which provides that no information may be disclosed by default. Additional positive rules may affirmatively specify the conditions under which certain information may be accessed. This base configuration ensures that the middleware is safe from unwanted intrusion "out of the box." To enable selective sharing, though, a set of base rules can be defined to disclose minimally intrusive information, such as allowing the sharing of general business addresses.

Once the rules of the privacy policy are specified, the Ïnfïnïty middleware automatically enforces them for every data access. We employ the same technique as defined in [4], to transform queries so that only data permitted by the privacy policy is retrieved. The privacy enforcement module analyzes and rewrites incoming queries to that that they only access data items that the appliable policy allows to be revealed. Only the rewritten query is executed against the local data repository. This ensures that only the minimal amount of data necessary to answer the query is retrieved, enhancing both the security and the performance of the system.

5. THE ÏNFÏNÏTY PROTOTYPE

To demonstrate the feasibility the Ïnfïnïty framework, we have implemented a prototype on several mobile devices, including two Cingular 8125 Pocket PCs and two Cingular 2125 Smartphones,

Figure 3: Deploying the Evacuation Routing Application.

using the Bluetooth communication protocol. We developed two sample applications on top of the middleware – one for an evacuation routing service and a second for a restaurant recommendation service.

We chose these scenarios because they require: (1) access to ad hoc networks with large user populations, (2) process-and-forward distributed query processing, (3) location-sensitivity, (4) privacy policy enforcement, and (5) availability of recent and continually updated data. Generally, these characteristics define application domains well-suited for the Ïnfïnïty middleware.

In this section, we describe the evacuation routing prototype, using the two Cingular 8125 Pocket PCs. As described in Section 1.1, this application uses map data and information about the current position of mobile devices to automatically determine the least crowded evacuation route. For the prototype, we provided map data for two test sites and used an external GPS receiver to provide location information. In an indoor application scenario as described below, position information could for example be acquired through triangulation using known positions of wireless senders like access points or RFID readers. In any case, the phones retrieve the location information as needed by the application. Consequently, the global schema for this prototype contains the following two additional relations (which are also the only ones accessed by the application):

- *Map* (containing the map information)
- *Position* (containing ID and position of devices)

The global and local schema for the map information are identical and there is no materialized local schema for the position information. Instead, the query engine maps each access to *Position* onto an access to the location sensing device.

5.1 First Steps

First, the middleware must be installed and initialized. This starts the runtime engine and query processor, allowing applications to be loaded. In Figure 3, the phone on the left contains the evaluation routing application and is deploying it to the phone on the right.

As the phone on the left attempts to share the evacuation application, the user of the phone on the right is prompted with an installation dialog asking whether she would like to install this application. After the user reviews the application specification and

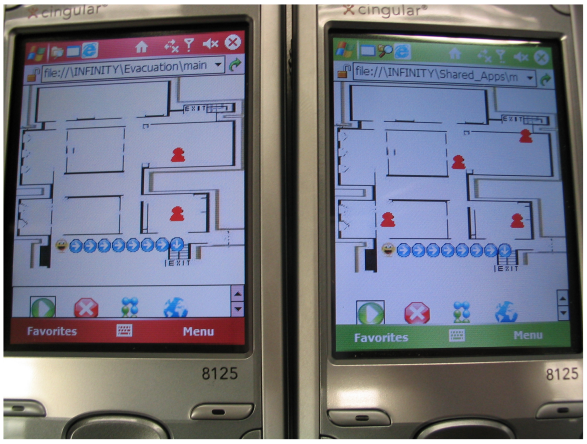

Figure 4: Initial Displays from Users of Evacuation Routing Application.

Figure 5: Re-Routing Users based on Query Results

accepts the request, the phone on the right installs and initializes the application. This includes the creation of the mapping between the *Map* and *Position* tables in the global schema and the position information.

This deployment step can occur at any time during normal operation.

5.2 Application Use

Once the application is running (Figure 4), it queries other users within its particular location (e.g., building, public venue, downtown, etc.) to find the least crowded and fastest exit route. To achieve this, the application invokes a query that requests the position of all users in the location. The runtime engine creates the necessary control structures, like buffers for query results and timers, to provide the application with the query results at a later time. It then forwards the query to the query processor.

The query that retrieves the positions of all mobile devices within range is of the form *"select coordinates from Postion"*. The query processor analyzes this query and splits it into two parts. It then translates the first part into calls to the positioning device to retrieve the current position of the mobile device. Before the query processor executes the query, it checks the query against the local privacy policy. The following table shows example privacy rules for the evacuation scenario (the first rule is actually used in the scenario):

Rule	Data	Accessor	Purpose	Condition
1	*Position*	any	emergency	-
2	*Position*	manager	work related	Mo-Fr 9am-5pm

Next, the second part of the query is sent out to other users in the same location to retrieve their exact position. In our simple example, the remote query is identical to the original query for the global schema. The query is passed to the communication module of the middleware, which transforms the query into a generic and machine independent textual representation, chooses the appropriate communication channel, and sends the query to remote devices. In our prototype implementation, Bluetooth connections are the only supported communication channel. At the remote device(s), the queries are received by the respective communication modules and passed to the query processor for execution against local data sources. When the original mobile device receives the remote query results, it accumulates them, processes them in the query processor, and forwards them to the runtime engine. In turn, the runtime engine transforms them into a suitable format to be retrieved by the application.

In Figure 5, this process yields different routing plans for the users of the left and right phones due to the flow of people out of the building.

6. FUTURE WORK

The Ïnfïnïty middleware demonstrates that privacy-preserving data sharing applications for mobile ad hoc networks can be developed easily in this unified framework. However, many open questions remain for further research.

One open issue involves metrics for communication channel selection. While least-cost routing has been studied for stable wired as well a mobile networks, ad hoc networks require different strategies. Cost and reliability are not the only considerations. Transmission speed can become a decisive factor in such volatile networks, since the chance of losing connection to a device increases with the length of the required transmission window. This may rule out certain otherwise seemingly optimal channels.

Query routing also presents similar research problems. The performance and effectiveness of the process-and-forward approach to query processing depends on the effectiveness of routing decisions. The goal is to have a minimal number of recipients providing the maximum amount of information. Solutions for this decision problem must be evaluated, especially with respect to their scalability to larger networks with limited information about the participants.

Privacy and security also present interesting research topics. One such topic involves further investigating possibilities for user identification in mobile ad hoc networks. This is a precondition for the use of fine-grained and user-specific privacy policies. The short lifetime and high churn of mobile ad hoc networks poses severe constraints. A second topic concerns the ease of defining and maintaining privacy policies. Policy definition should be intuitive and quick, even on mobile devices with limited capabilities.

Another important research challenge is to develop caching strategies for query results. To avoid having a query unnecessarily propagated through the entire network, previous query results should be cached to answer similar queries from other devices. Determining which results should be cached and for how long has a strong

impact on the performance of the entire system.

While our prototype implementation provides initial approaches for some of these research problems, more thorough investigations are necessary.

7. CONCLUSION

The increasing popularity and diversity of mobile devices, miniaturization, decreasing hardware costs, and shifting consumer habits indicate the emergence of a network of massively distributed data sources. Such a network requires technology that can integrate information from these infinite data points, leverage available networks through local communication channels, address data privacy concerns, and support application deployment and use across heterogeneous platforms.

The Ïnfïnïty middleware platform provides a framework for information sharing and collaboration in ad hoc networks of mobile devices. We have demonstrated that it is feasible to develop and deploy applications on this platform. Ïnfïnïty accommodates a variety of practical scenarios, including traffic monitoring, disaster recovery and rescue tools, recommendation services, and many others. We hope that our approach will be a useful first step in developing platform-independent applications that leverage the vast amounts of recent, location-specific information available in mobile ad hoc networks.

8. ACKNOWLEDGMENTS

The authors would like to thank Rakesh Agrawal, Karin Kailing, and Jerry Kiernan for initial discussions on the work and Leonard Lee, Eva Shon, Yong Yao and Ian Yap for implementing the prototype of the Ïnfïnïty middleware.

9. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *VLDB '02: Proc. of 28th Int. Conf. on Very Large Data Bases*, pages 143–154, 2002.

[2] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Survey*, 36(4):335–371, 2004.

[3] R. Huebsch, B. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The Architecture of PIER: An Internet-Scale Query Processor. In *CIDR '05: 2nd Biennial Conf. on Innovative Data Systems Research*, pages 28–43, 2005.

[4] K. LeFevre, R. Agrawal, Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt. Limiting Disclosure in Hippocratic Databases. In *VLDB '04: Proc. of the 30th Int. Conf. on Very Large Data Bases*, pages 108–119, Toronto, Canada, 2004.

[5] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. Decouto. Carnet: a scalable ad hoc wireless network system. In *EW 9: Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 61–65, New York, NY, USA, 2000. ACM Press.

[6] P.-G. Raverdy, O. Riva, A. de La Chapelle, R. Chibout, and V. Issarny. Efficient context-aware service discovery in multi-protocol pervasive environments. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*, page 3, Washington, DC, USA, 2006. IEEE Computer Society.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of the ACM SIGCOMM 2001 Conference*, pages 149–160, 2001.

[8] J. Waldo. The jini architecture for network-centric computing. *Commun. ACM*, 42(7):76–82, 1999.