Challenges in Ubiquitous Data Management

Michael J. Franklin

EECS Computer Science Division
University of California,
Berkeley, CA 94720, USA
franklin@cs.berkeley.edu

Abstract. Ubiquitous computing is a compelling vision for the future that is moving closer to realization at an accelerating pace. The combination of global wireless and wired connectivity along with increasingly small and powerful devices enables a wide array of new applications that will change the nature of computing. Beyond new devices and communications mechanisms, however, the key technology that is required to make ubiquitous computing a reality is *data management.* In this short paper, I attempt to identify and organize the key aspects of ubiquitous computing applications and environments from a data management perspective and outline the data management challenges that they engender. Finally, I describe two on-going projects: *Data Recharging* and *Telegraph*, that are addressing some of these issues.

Introduction

The confluence of ever smaller and more powerful computing and communication devices, improved connectivity in both wired and wireless environments, and emerging and accepted standards for data transfer and presentation (e.g., HTML, XML, HTTP, WAP, etc.) are leading to a world in which computers are playing an ever increasing role in peopleís daily lives. It is reasonable to expect that ultimately, such devices will be so pervasive and so critical to our activities that they will be simply taken for granted − effectively disappearing into the background. This idea, known as ìUbiquitous Computingî, has been a motivating vision underlying much Computer Science research in the past decade, going back to the seminal writings of Mark Wieser in which the term was coined [Wieser 91].

Of course, as anyone who depends upon computers or PDAs today realizes, there is still quite some progress to be made before this vision of ubiquitous computing is achieved. The necessary technology, however, is improving at an impressive rate and advances furthering the realization of ubiquitous computing are announced almost daily. Much of the research and development activity in this area is focused on improving the devices themselves and on the technologies they use to communicate. For devices, the emphasis has been on improving functionality, while reducing size, cost, and power requirements. For communications, the emphasis has been on improving bandwidth and coverage, and on developing protocols that are more tolerant of the error and connectivity characteristics of wireless and mobile devices.

R. Wilhelm (Ed.): Informatics. 10 Years Back. 10 Years Ahead, LNCS 2000, pp. 24-33, 2001.

Improved hardware and networking are clearly central to the development of ubiquitous computing, but an equally important and difficult set of challenges revolve around *Data Management* [AK93]. In order for computing to fade into the background while supporting more and more activities, the data required to support those activities must be reliably and efficiently stored, queried, and delivered. Traditional approaches to data management such as caching, concurrency control, query processing, etc. need to be adapted to the requirements and restrictions of ubiquitous computing environments. These include resource limitations, varying and intermittent connectivity, mobile users, and dynamic collaborations.

In this paper we first discuss the main characteristics of applications that ubiquitous computing aims to support and then focus on the requirements that such applications impose on data management technology. We then examine several different aspects of data management and how they are being adapted to these new requirements.

Applications and Data Management Requirements

While there is wide agreement on the great potential of ubiquitous computing, it is not yet clear what the ìkiller applicationsî (i.e., the uses that will result in widespread adoption) will be. Many researchers and product developers have created example scenarios to demonstrate the potential of the technology. Due to the integrated and universal nature of ubiquitous computing, these scenarios tend to include a large number of functions rather than any one single application. Thus, some in industry have begun to talk in terms of delivering a certain type of ìuser experienceî rather than a particular application or suite of applications. These scenarios tend to involve users with several portable devices, moving between different environments (e.g., home, car, office, conference). The devices typically take an active (and often annoying) role in reminding the user of various appointments and tasks that are due, provide access to any and all information that may be relevant to these tasks, and facilitate communication among groups of individuals involved in the tasks.

Categories of Functionality

Rather than specify yet another such scenario, it is perhaps more useful to categorize the functionalities that such scenarios imply. This categorization can then be examined to determine the requirements that are imposed on data management. The functionalities can be classified into the following:

1) *Support for mobility* ñ the compactness of the devices combined with wireless communication means that the devices can be used in mobile situations. Thus, existing applications must be able to operate in varied and dynamic communication and computation environments, possibly

moving from one network or service provider to another. Furthermore, new applications that are location-centric will also be developed.

2) *Context awareness* ñ if devices become truly ubiquitous, then they will be used constantly in a wide range of continually changing situations. For the devices to be truly helpful, they must be aware of the environment as well as the tasks that the user is performing or will be performing in the near future. Context aware applications range from intelligent notification systems that inform the user of (hopefully) important events or data, to ìsmart spacesî, that is, rooms or environments that adapt based on who is present and what they are doing.

3) *Support for collaboration* ñ another key theme of ubiquitous computing applications is the support of groups of people. This support consists of communications and conferencing as well as the storage, maintenance, delivery, and presentation of shared data. Collaborations may be performed in real-time, if all of the participants are available, or may be done asynchronously otherwise. In addition to supporting on-going collaboration, access to and analysis of traces of past activities is also required.

Adaptivity and User Interaaction

These functionalities provide a host of challenges for data management techniques, but one requirement is present across all of them, namely, the need for *adaptivity*. Mobile users and devices, changing contexts, and dynamic groups all impose requirements for flexibility and responsiveness that are simply not addressed by most traditional data management techniques. Thus, adaptivity is a common theme of the techniques that we discuss in the remainder of the paper.

It is also important to note that because ubiquitous computing is intended to augment human capabilities in the execution of various tasks, the nature of these applications is that the user is typically interacting in real-time with the computers. We are able to exploit this fact as part of the solution to adaptivity by, in some cases, depending on the users to make dynamic choices or to cope with some degree of ambiguity. A concrete example of such a design choice is the way that many groupware systems handle concurrent access and update to shared data. Rather than impose rules that restrict the types and degrees of interaction that users can have, as is done by concurrency control mechanisms in traditional database systems, a groupware data manager will typically impose less stringent rules. The relaxation of these rules limits the extent to which the system can autonomously handle conflicts. Thus, such systems typically handle whatever cases they can, and when they detect a conflict that cannot be handled automatically, they simply inform the user(s) that the conflict has occurred, and allow them to resolve it based on their knowledge of the situation. Thus, having users ìin the loopî can be leveraged to provide more adaptive and flexible systems.

Requirements Due to Mobility

Other data management requirements are less universal across the three categories but yet must be addressed in order to support a comprehensive ubiquitous computing environment. For example, the issue of mobility raises a number of issues. First, the fact that the terminals (i.e. devices) are constantly moving, and often have limited storage capacity means that a ubiquitous computing system must be able to deliver data to and receive data from different and changing locations. This results in the need for various kinds of proxy solutions, where users are handed off from one proxy to another as they move. Protocols must be constructed in such a way as to be able to tolerate such handoffs without breaking. Mobility also raises the need for intelligent data staging and pre-staging, so that data can be placed close to where the users will be when they need it (particularly in slow or unreliable communications situations).

Secondly, mobility adds location as a new dimension to applications that does not typically play a role in stationary scenarios. For example, some of the most useful applications for mobile devices are *location-centric*. Consider a system that can answer questions such as ì find the drugstores within 2 miles of my current locationî. Such a system must track the location of the current user and be able to access information based on relative locations and distances. On a broader scale, servers may have to track large numbers of moving objects (people, cars, devices, etc.) and be able to predict their future locations. For example, an automated traffic control system would have to track numerous cars, including their current positions, directions, and velocities. Location-centric computing requires special data structures in which location information can be encoded and efficiently stored, as well as ones in which the dynamic positions of objects can be maintained.

Requirements Due to Context-Awareness

Context-awareness imposes significant demands on the knowledge maintained by the system and the inferencing algorithms that use that knowledge. In order to be context aware, a system must maintain an internal representation of usersí needs, roles, and preferences, etc. One example of such a system is a smart calendar that routes information to a user based on knowledge of the userís near-term schedule (as can be determined from the userís PIM calendar). If, for example, a user has a meeting with a particular client scheduled for the afternoon, such a system could send the user information that would be highly relevant to that meeting, such as data about the clientís account, results of previous meetings with that client, news articles relevant to the topic of the meeting, etc.

More sophisticated systems might use various types of sensors to monitor the environment and track usersí actions so as to be able to assist in the tasks the user is performing. Such sensor-based systems require the ability to process *data streams* in real-time and to analyze and interpret such streams. Thus, data-flow processing will play a key role in ubiquitous computing.

Whether the system obtains its context information from sensors, user input, PIM (personal information management) applications, or some combination of these, it

must perform a good deal of processing over the data in order to be able to accurately assess the state of the environment and the intensions of the user. Thus, context-aware applications impose demanding requirements for *inferencing and machine learning* techniques. These processes will have to cope with incomplete and conflicting data, and will have to do so extremely efficiently in order to be able to interact with the user in a useful and unobtrusive manner.

Requirements Due to Collaboration

The final set of requirements we discuss here are those that result from the need to support collaborative work by dynamic and sometimes ad hoc groups of people. As stated above, a prime requirement that stems from such applications is adaptivity. In addition, however, there are several other types of support that are required beyond what has already been discussed. First, there is a need for synchronization and consistency support. At the center of any collaborative application is a set of shared data items that can be created, accessed, modified, and deleted by participants of the collaboration. This coordination function must be lightweight and flexible so that many different types of interactions can be supported, ranging from unmoderated chat facilities, to full ACID (Atomic, Consistent, Isolated, and Durable) transactions, as provided by traditional database systems.

A second requirement stemming from collaborative applications is the need for reliable and available storage of history. In particular, if the collaboration is to be performed in an asynchronous manner, users must be able to access a record of what has happened earlier in the collaboration. Likewise, if the participants in the collaboration can change over time (e.g., due to mobility, failures, or simply due to the nature of the collaboration), then a durable record of participants and their actions is essential to allow new members to join and come up to speed. Such durable storage is also useful for keeping a log of activity, that can be used later to trace through the causes of various outcomes of the collaboration, or as input into learning and data mining algorithms which may help optimize future collaborations.

Example Data Management Technologies ñ On-Going Projects

The preceding discussion addressed some of the *data management* challenges that must be addressed to support ubiquitous computing scenarios and outlined the application properties from which they arise. In this section, we briefly describe two on going projects that are addressing some of these aspects. The first project, called Data Recharging, exploits user interest and preference information to deliver data updates and other relevant items to users on their portable devices. The second project, called Telegraph, is building an adaptive data-flow processing architecture to process long-running queries over variable streams of data, as would arise in sensor-based and other highly dynamic data environments.

Data Recharging: Profile-Based Data Dissemination and Synchronization

Mobile devices require two key resources to function: power and data. The mobile nature of such devices combined with limitations of size and cost makes it impractical to keep them continually connected to the fixed power and data (i.e., the Internet) grids. Mobile devices cope with disconnection from these grids by "caching". Devices use rechargeable batteries for caching power, while local storage is used for caching data. Periodically, the device-local local resources must be ``recharged'' by connecting with the fixed power and data grids. With existing technology, however, the process of recharging the device resident data is much more cumbersome and error-prone than recharging the power. Power recharging can be done virtually anywhere in the world, requires little user involvement, and works progressively − the longer the device is recharged, the better the device-stored power becomes. In contrast, data ``recharging'' has none of these attributes.

The Data Recharging project is developing a service and corresponding infrastructure that permits a mobile device of any kind to plug into the Internet at any location for any amount of time and as a result, end up with more useful data than it had before [CFZ00]. As with power recharging, the initiation of a *data charge* simply requires "plugging in" a device to the network. The longer the device is left plugged in, the more effective the charge. Although similar to battery recharging, data recharging is more complex; the type and amount of data delivered during a data charge must be tailored to the needs of the user, the capabilities of the recharged device, and the tasks that the recharged data is needed to support.

Different mobile users will have different data needs. A business traveler may want updates of contact information, restaurant reviews and hotel pricing guides specific to a travel destination. Students may require access to recent course notes, required readings for the next lecture and notification about lab space as it becomes available. Data recharging represents specifications of user needs as *profiles*. Profiles can be thought of as long-running queries that continually sift through the available data to find relevant items and determine their value to the user.

Profiles for data recharging contain three types of information: First, the profile describes the types of data that are of interest to the user. This description must be *declarative* in nature, so that it can encompass newly created data in addition to existing data. The description must also be flexible enough to express predicates over different types of data and media. Second, because of limitations on bandwidth, device-local storage, and recharging time, only a bounded amount of information can be sent to a device during data recharging. Thus, the profile must also express the userís preferences in terms of priorities among data items, desired resolutions of multi-resolution items, consistency requirements, and other properties. Finally, user context can be dynamically incorporated into the recharging process by parameterizing the user profile with information obtained from the device-resident Personal Information Management (PIM) applications such as the calendar, contact list, and To Do list.

Our previous work on user profiles has focused on 1) efficiently processing profiles over streams of XML documents (i.e., the ìXFilterî system) [AF00], 2) learning and maintaining user profiles based on explicit user feedback [CFG00], and 3) development of a large-scale, reliable system for mobile device synchronization

[DF00]. Data recharging can build upon this work, but requires the further development of a suitable language and processing strategy for highly expressive user profiles (i.e., that include user preference and contextual information), and the development of a scalable, wide-area architecture that is capable of providing a data recharging service on a global basis to potentially millions of users.

Adaptive Dataflow Query Processing

A second important aspect of ubiquitous computing environments is the variable nature of data availability and the challenges of managing and processing dynamic data flows. In mobile applications for example, data can move throughout the system in order to follow the users who need it. Conversely, in mobile applications where the data is being created at the endpoints (say, for example, a remote sensing application) data streams into the system in an erratic fashion to be processed, stored, and possibly acted upon by agents residing in the network. Information flows also arise in other applications, such as data dissemination systems in which streams of newly created and modified data must be routed to users and shared caches.

Traditional database query processing systems break down in such environments for a number of reasons: First, they are based on *static* approaches to query optimization and planning. Database systems produce query plans using simple cost models and statistics about the data to estimate the cost of running particular plans. In a dynamic dataflow environment, this approach simply does not work because there are typically no reliable statistics about the data and because the arrival rates, order, and behavior of the data streams are too unpredictable [UFA98].

Second, the exisiting approaches cannot adequately cope with failures that arise during the processing of a query. In current database systems, if the failure of a data source goes undetected, the query processor simply blocks, waiting for the data to arrive. If a failure is detected, then a query is simply aborted and restarted. Neither of these situations is appropriate in a ubiquitous computing environment in which sources and streams behave unpredictably, and queries can be extremely long-running (perhaps even ìcontinuousî).

Third, existing approaches are optimized for a *batch* style of processing in which the goal is to deliver an entire answer (i.e., they are optimized for the delivery of the *last* result). In a ubiquitous computing environment, where users will be interacting with the system in a fine-grained fashion, such approaches are unacceptable. Processed data (e.g., query results, event notifications, etc.) must be passed on to the user as soon as they are available. Furthermore, because the system is interactive, a user may choose to modify the query on the basis of previously returned information or other factors. Thus, the system must be able to gracefully adjust to changes in the needs of the users [HACO+99].

The *Telegraph* project at UC Berkeley [HFCD+00] is investigating these issues through the development of an adaptive dataflow processing engine. Telegraph uses a novel approach to query execution based on ìeddiesî, which are dataflow control structures that route data to query operators on an item-by-item basis [AH00]. Telegraph does not rely upon a traditional query plan, but rather, allows the îplanî to develop and adapt during the execution. For queries over continuous streams of data,

the system can continually adapt to changes in the data arrival rates, data characteristics, and the availability of processing, storage, and communication resources.

In addition to novel control structures, Telegraph also employs non-blocking, symmetric query processing operators, such as XJoins [UF00] and Ripple Joins [HH99], which can cope with changing and unpredictable arrival of their input data. The challenges being addressed in the Telegraph project include the development of cluster-based and wide-area implementations of the processing engine, the design of fault-tolerance mechanisms (particularly for long-running queries), support for continuous queries over sensor data and for profile-based information dissemination, and user interface issues.

Conclusions

Ubiquitous computing is a compelling vision for the future that is moving closer to realization at an accelerating pace. The combination of global wireless and wired connectivity along with increasingly small and powerful devices enables a wide array of new applications that will change the nature of computing. Beyond new devices and communications mechanisms, however, the key technology that is required to make ubiquitous computing a reality is *data management.* Data lies at the heart of all ubiquitous computing applications, but these applications and environments impose new and challenging requirements for data management technology.

In this short paper, I have tried to outline the key aspects of ubiquitous computing from a data management perspective. These aspects were organized into three categories: 1) support for *mobility*, 2) *context-awareness*, and 3) support for *collaboration*. I then examined each of these to determine a set of requirements that they impose on data management. The over-riding issue that stems from all of these is the need for *adaptivity*. Thus, traditional data management techniques, which tend to be static and fairly rigid, must be rethought in light of this emerging computing environment.

I also described two on-going projects that are re-examining key aspects of data management techniques: the *DataRecharging* project, which aims to provide data synchronization and dissemination of highly relevant data for mobile users based on the processing of sophisticated user profiles; and the *Telegraph* project, which is developing a dynamic dataflow processing engine to efficiently and adaptively process streams of data from a host of sources ranging from web sources to networks of sensors.

Of course, there are a number of very important data management issues that have not been touched upon in this short paper. First of all, the ability to interoperate among multiple applications and types of data will depend on standards for data-interchange, resource discovery, and inter-object communication. Great strides are being made in these areas, and the research issues are only a small part of the effort involved in the standardization process. Another important area, in which on-going research plays a central role, is the development of global-scale, secure and archival information storage utilities. An example of such a system is the OceanStore utility, currently under development at UC Berkeley [KBCC+00].

In summary, ubiquitous computing raises a tremendous number of opportunities and challenges for data management research and will continue to do so for the foreseeable future. It is crucial to recognize that although it is tempting to focus on the latest ìgadgetî or communications protocol, data management plays a central role in the development of ubiquitous computing solutions and that advances in this area will ultimately depend on our ability to solve these new and difficult data management problems.

Acknowledgements

The ideas and opinions expressed in this article are the result of many discussions over a number of years with colleagues in both academic and industrial settings. The two projects discussed in this paper are on-going collaborations with several people: The Data Recharging project is a three-university collaboration with Mitch Cherniack of Brandeis University and Stan Zdonik of Brown University. Matt Denny and Danny Tom of UC Berkeley have made important contributions to the project. The Telegraph project is being done at UC Berkeley with Joe Hellerstein. Students currently working on the project include: Sirish Chandrasekaran, Amol Deshpande, Kris Hildrum, Nick Lanham, Sam Madden, Vijayshankar Raman, and Mehul Shah.

The work described in this paper is partially funded by DARPA as part of the ìEndeavour Expeditionî under Contract Number N66001-99-2-8913, by the National Science Foundation under the ìData Centersî project of the Information Technology Research program, and by research funds from IBM, Siemens, Microsoft, and the Okawa Foundation.

Thanks also to the organizers of the Dagstuhl 10th Anniversary Conference and in particular, to Reinhard Wilhelm for his hard work and patience in putting together an extremely successful symposium.

References

[AF00] Mehmet Altinel, Michael J. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information, *Proceedings of the International Conference on Very Large Data Bases*, Cairo, September 2000.

[AK93] Rafael Alonso, Henry F. Korth. Database System Issues in Nomadic Computing. *Proceedings of the ACM SIGMOD Conference*, Washington, D.C., June 1993, pp388-392.

[AH00] Ron Avnur, Joseph M. Hellerstein. Eddies: Continuously Adaptive Query Processing, *Procedings of the ACM SIGMOD Conference,* Philadelphia, PA, June 2000,

[CFG00] Ugur Cetintemel, Michael J. Franklin, and C. Lee Giles. Self-Adaptive User Profiles for Large-Scale Data Delivery, *Proceedings of the International Conference on Data Engineering*, San Diego, CA, February, 2000, pp 622-633

[CFZ00] Mitch Cherniack, Michael J. Franklin, Stan Zdonik. Expressing User Profiles for Data Recharging, *submitted for publication,* October, 2000.

[DF00] Matthew Denny, Michael J. Franklin. Edison: Database-Supported Synchronization for PDAs, *submitted for publication,* November, 2000.

[HFCD+00] Joseph M. Hellerstein, Michael J. Franklin, Sirish Chandrasekaran, Amol Deshpande, Kris Hildrum, Sam Madden, Vijayshankar Raman, Mehul Shah. Adaptive Query Processing: Technology in Evolution, *IEEE Data Engineering Bulletin,* June 2000, pp 7-18.

[HACO+99] Joseph M. Hellerstein Ron Avnur, Andy Chou, Chris Olston, Vijayshankar Raman, Tali Roth, Christian Hidber, Peter J.Haas. Interactive Data Analysis with CONTROL, *IEEE Computer 32(8),* August, 1999, pp. 51-59.

[HH99] Peter J. Haas, Joseph M. Hellerstein. Ripple Joins for Online Aggregation, *Procedings of the ACM SIGMOD Conference,* Philadelphia, PA, June, 1999, pp. 287-298.

[KBCC+00] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage, *Proceeedings of the Ninth Iinternational Conference on Architectural Support for Programming Languages and*
Operating Systems (ASPLOS 2000), November 2000.

[UFA98] Tolga Urhan, Michael J. Franklin, Laurent Amsaleg. Cost Based Query Scrambling for Initial Delays,*Procedings of the ACM SIGMOD Conference,* Seattle, WA, June 1998, pp. 130-141.

[UF 00] Tolga Urhan, Michael J. Franklin. XJoin: A Reactively-Scheduled Pipelined Join Operator. *IEEE Data Engineering Bulletin*, 23(2), June, 2000, pp. 27-33.

[Weiser 91] Weiser, Mark. The Computer for the Twenty-First Century. *Scientific American*. September 1991. pp. 94-104.