Abstract

Mobile devices need two basic renewable resources — power and data. Power recharging is easy; data recharging is a much more problematic activity. It requires complex interaction between a user and a collection of data sources. We provide an automatic data recharging capability based on user profiles written in an expressive profile language. A profile identifies relevant information and orders it by its usefulness. In this article, we discuss the issues involved in designing a profile language for data recharging.

Expressing User Profiles for Data Recharging

Mitch Cherniack, Brandeis University
Michael J. Franklin, University of California at Berkeley
Stan Zdonik, Brown University

We are rapidly heading toward a world in which the computing infrastructure will contain billions of devices that are carried or worn by their users as they go through their daily routines. These devices require two key resources to function: power and data. The mobile nature of such devices combined with the economic limitations of size and cost makes it impractical to keep them continually connected to fixed sources of either resource. Mobile devices cope with disconnection from fixed sources of power and data by "caching." For power, devices typically use rechargeable batteries; batteries act as a cache of power from the fixed power grid. Likewise, data from the "information grid" (i.e., the Internet) is cached in device-local storage (memory, flash memory, disk, etc.) for use by the applications running on that device.

Periodically, a device's batteries must be recharged by connecting with the fixed power grid. Performing such recharging is easy: the device can be plugged into any electrical outlet that is available; the charge can be interrupted at any time —- the longer it is plugged in, the *better* the charge gets, until it is fully charged; and most important, the process happens with minimal intervention by the user.

Data recharging is a service that aims to provide the analogous functionality for recharging the device-resident *data cache*. As with battery recharging, the goal is for data recharging to function in a flexible and geographically independent manner. That is, mobile devices should be able to efficiently recharge their data by connecting at any point in the global Internet, using whatever bandwidth is available at that point, connecting for whatever amount of time they can spare, and doing so with little or no help from the user.

While the analogy between recharging power and recharging data is an appealing one, obviously it can only be taken so far. A key place where the analogy breaks down is that while electrons are basically fungible, the data needed on a particular device is highly dependent on the user of that device and the tasks they will be performing when they need the data. Thus, the choice of data that must be sent to a device in order to recharge it is dependent on the *semantics* of the applications. The data recharging infrastructure must therefore maintain and exploit information about the data needs of users. The key technology used for communicating and representing such information in a data recharging system is the *user profile*.

Profiles for data recharging must contain two types of information. First, the profile must describe 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 bandwidth, device-local storage, and recharging time limitations, only a bounded amount of information can be sent to a device during data recharging. Thus, the profile must also express the user's preferences in terms of priorities among data items, desired resolutions of multiresolution items, consistency requirements, and other properties. A key challenge for data recharging, therefore, is the development of a suitable language and processing strategy for these highly expressive user profiles. In the remainder of this article we describe these challenges in more detail, and present our initial approaches toward solving them. First, however, we describe the data recharging service and infrastructure in more detail.

Data Recharging

Motivation

Mobile devices in the form of laptops and personal digital assistants (PDAs) have achieved widespread importance as tools for providing data access to users on the move. With current technology, the data stored on the devices must be manually maintained and synchronized with the fixed data sources; such synchronization can typically be performed only at specific locations, such as the user's (stationary) workstation. These restrictions currently place severe limitations on the range and sophistication of the applications that can run on mobile devices. Our approach to data recharging is aimed at removing these limitations by making the process of "recharging" data as simple and flexible for future mobile users as recharging power is for today's users. Below we briefly give two example scenarios where such support is needed.

1070-9916/01/$10.00 © 2001 IEEE

The Business Traveler — Consider an executive who is traveling for a series of meetings with current and potential customers. While on the road, she would like to be kept apprised of important developments in her company as well as relevant news articles related to her business. Furthermore, as her schedule of meetings evolves during the trip, she would like to obtain background information on the people and accounts with whom she will be meeting. Finally, she would also like to have information related to her travel, including local maps, rental car and shuttle information, directions to her hotel, and so on.

With current technology, this information, if obtainable at all, would have to be assembled manually and transmitted to the executive's mobile device at a point when it can be connected to the network for a sufficient amount of time over a sufficiently high-bandwidth link. The collection of such information is labor-intensive and error-prone. Furthermore, if the connectivity of the device is interrupted unexpectedly, it is likely that some important information required by the executive will not be delivered to her. Even worse, it is quite possible that the information she stored previously on the device will be left in a corrupted state due to the incomplete synchronization with the fixed data sources. Because of such problems, mobile devices today are relegated to serving primarily as replacements for paper-based tools such as day planners, and their potential to serve as active personal information assistants remains largely untapped.

The Mobile CoursePak — Consider a large undergraduate course in which small teams of students must collaborate on a group project. With data recharging support, mobile devices could serve as a key coordination tool for the course and lecture material as well as the group projects. First of all, the calendar functions of the mobile devices would be used to store the schedule of lecture topics and assignments. Lecture notes, assignments, grades, discussions, and so forth would be published as a *CoursePak* that is posted electronically and continually updated during the course of the semester. Data recharging, keyed off of the class calendar and tailored to the needs of each student, would allow the relevant parts of the CoursePak to be downloaded to a student's mobile device when they connect to the network. Currently, such connection would occur at a physical network access point (at a library, lab, classroom, etc.) but soon many campuses will provide wireless connectivity, allowing such recharging to be done on a more continuous basis.

Another important use of recharging technology in this scenario is for coordination within project groups. When recharging, a group member could obtain the latest updates of document sections or code modules from the other group members. Likewise, the devices could be used for scheduling and coordinating group meetings at various places on or near the campus. Finally, the instructors and TAs could also use the recharging functionality, for example, to obtain the latest relevant information for an upcoming lecture. Data recharging issues similar to those described for the business traveler above will arise in this scenario as well, due to the increasing prevalence of distance learning in which students may reside in different campuses, companies, or countries from where the course is physically being taught.

Data Utility

In the examples described above, students and business travelers who recharge their mobile devices will receive very different data, despite having recharged their devices using the same data recharging service. What determines the result of data recharging for each client is her *user profile*. As discussed in a later section, many information distribution systems provide some type of support for describing user interests. For data recharging, however, user profiles must specify not only the user's data interests, but also the priorities and preferences the user has regarding those items. This latter information is needed to cope with the resource limitations that arise in a data recharging scenario. These limitations include those inherent in the device being recharged (e.g., limited storage, limited bandwidth, limited screen size) as well as those that arise as part of the recharging process itself, such as a limited connection time for performing recharging.

Data recharging 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. 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. If available resources are few or a charge ends prematurely, received data should still be consistent and uncorrupted. More resources and longer charges should only enhance the utility of the data provided by a charge.

Data utility can be influenced along many dimensions, including resolution, priority, compression, currency, and fidelity. For any of these dimensions, more useful data requires more available resources during the data charge. The resolution of data refers to its level of detail. Low-resolution forms of an image (which reduce the number of pixels) or a document (which might exclude figures and formatting) might be less useful than their higher-resolution equivalents, but would require fewer resources (e.g., device space, charge time) to transmit. Prioritization would charge a device with data by order of importance. In this case, data utility is measured by the number of priority classes transferred as a result of the charge. Compressed data might be less useful than uncompressed data if the compression were lossy, but would require transmission of fewer bits. Currency ties the utility of data to its age. Recent versions of data might be more useful than stale ones, but require more resources to retrieve than cached older versions. Fidelity refers to the accuracy of data. More accurate data might result from larger samples of data sources or corroboration with alternative sources, but both require use of more resources prior to transmission.

Our current work on profiles for data recharging has focused on the problems of combining the specification of the data items that are of interest to a user with a description of the importance (i.e., "utility") of each of those items to that user. These issues are the focus of a later section of this article. The additional issues of multiresolution, compression, and currency are important aspects of our future work, and are discussed later along with several other advanced topics. Before going into detail on profiles, however, we first briefly describe the architectural context in which we expect the data recharging service to operate.

An Architecture Overview

The data recharging service is implemented as a distributed system that consists of a network of *profile managers* located throughout the Internet. Profile managers are responsible for retrieving needed data from data sources and packaging it for delivery to specific devices based on their recharging profiles. Our approach to developing this infrastructure is based on the notion of a dissemination-based information system (DBIS) [1] and is shown in Fig. 1. In our model there are three different roles that can be played by a profile manager:

- *Data collector* – These profile managers interact directly with the underlying data sources, requesting and, if necessary, reformatting the data.

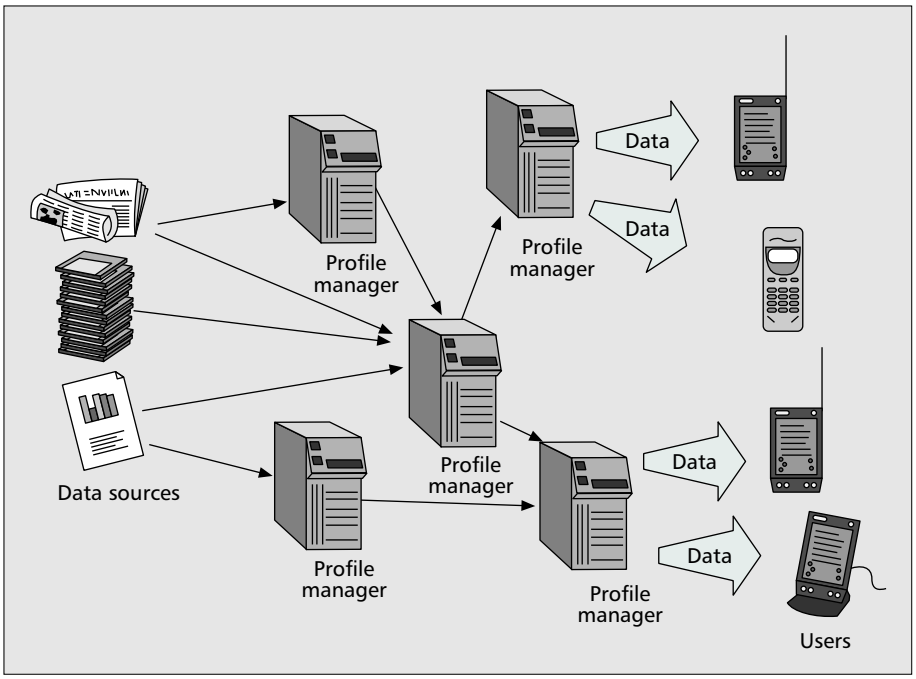

Figure 1. *Architecture.*

- *Edge node* – These profile managers interact directly with the user devices and are responsible for storing and managing profile information for individual users, and for delivering the data to devices during recharging sessions.
- *Broker* – These profile managers aggregate profiles from many devices and data from many sources in order to more efficiently distribute data through the network.

Profile managers can play one or more of these roles concurrently. In a sense, profile managers are application-level routers that move data through the network based on the detailed information available in the profiles. The profile information flows from the devices and edge nodes back through the infrastructure, being aggregated as it goes. The data collector nodes, to obtain relevant data from the data sources, ultimately use this aggregated profile information. The collected data is then spread through the network in the opposite direction, being distributed to the edge nodes from which it will ultimately be delivered to the devices.

Describing User Profiles

A business traveler and a student can plug in the same kind of device at the same location, and yet receive very different data as a result of data recharging. The individualization of data recharging is driven by the user profile. But what should a profile specify, and how should it specify it? In this section we consider an example profile for a business traveler to illustrate these issues.

What Should a Profile Specify?

Consider a profile for a business traveler who is visiting Boston. Upon her arrival, the traveler will stay at the Copley Plaza Hotel in downtown Boston. In advance of her trip, she charges her personal information manager (PIM), expecting (among other things) data that will help her reach her hotel from the airport.

The traveler can reach the Copley Plaza by either renting a car or taking a shuttle. The traveler has no preference about how she gets there, but wants her PIM to have been charged with the data required to make the trip by either means:

- If the traveler goes by rental car, she needs data from a rental car company describing its rates, location within Logan Airport, frequent flier miles policy, and so on. Having this data for competing rental car companies is useful, since it allows her to compare rates or seek alternatives if one of the companies has no available cars. However, she decides that having rental car data for three companies is enough, and data for more than this has no added value to her.
- If the traveler rents a car, she also needs directions to get to the Copley Plaza from Logan Airport. In fact, without these directions, rental car data has no value to the traveler.
- If the traveler is going to take a shuttle to her hotel, she needs a shuttle schedule for one of the shuttle companies that serves downtown Boston. Only one shuttle schedule is necessary. If she takes the shuttle, directions to her hotel may still be of use (in case the shuttle doesn't stop at the Copley Plaza directly, but instead at a nearby hotel), but the shuttle schedule has value regardless of whether she also has directions.

The business traveler profile will also specify interest in other data related to her travels, including restaurant listings, theatre reviews for current shows, and news items related to topics on the agenda for her meetings. But even the simplified profile described here is sufficient to demonstrate what might be considered good and bad data charges. Given the profile described above, it is clear that a good data charge will leave the data traveler either with data for one or more rental car companies and directions to the Copley Plaza hotel, or with a shuttle schedule, perhaps with directions to the hotel also. On the other hand, it is clear that a charge that leaves information about rental cars without directions to the hotel is undesirable.

As is evident from our example, a profile must be able to specify two things.

What are the interesting data objects to the user? We assume here that a *data object* is any object retrievable via an http or ftp request off the Internet. This includes files of varying formats (including text, html, xml, jpeg, gif, wav, and pdf) as well as objects and tables maintained in a database. In the case of the business traveler profile, interesting data objects might be Web pages for rental car companies (hereafter referred to as rental car *objects*), files of any kind with directions from Logan to Copley Plaza (direction objects), and shuttle schedules serving downtown Boston from Logan (shuttle objects). The set of interesting objects specified in a profile is that profile's *profile domain*.

What is the utility of each interesting data object? The utility of a data object indicates its relative worth when included in a data charge. A data object's worth may be independent of other data objects included in the data charge (as in direction objects which are always useful), or dependent on the presence or absence of other data objects in the data charge (as in rental car objects, whose worth depends on the presence of direction objects in the same data charge, and shuttle schedules, whose worth depends on the absence of either direction or rental car objects). A profile's *utility function* maps every potential data charge to an integer that denotes its value.

```
PROFILE BusinessTraveler
      DOMAIN:
     RC = rental car company web pages
     Dir = directions from Logan Airport to Copley Plaza
     Sh = shuttle schedules for shuttles serving Boston from Logan
  UTILITY:
     U (RC) = 0;
     U (Dir) = 1;
     U (Sh) = 2;
     U (RC with Sh) = 2;
     U (Dir with Sh) = 3;
     U (RC [1] with Dir) = 2;
     U (RC [2] with Dir) = 3;
     U (RC [i>2] with Dir) = 4;
     U (RC [0<i<3] with Dir with Sh) = 3;
     U (RC [i>2] with Dir with Sh) = 4
   END
```

Figure 2. *A business traveler's profile.*

How Should a Profile Be Expressed?

Figure 2 shows one possible way to express the business traveler profile introduced in the previous section. The profile in this figure has two sections: the `DOMAIN` section specifies the *profile domain*, and the `UTILITY` section specifies the *utility function*. In this section we consider issues in the design of profile languages, using the profile example of Fig. 2 as an example.

Before considering issues concerning profile languages, we first illustrate how profile specifications determine the behavior of the profile manager. Figure 3 shows a profile, **P**, in the context of a data recharging system. **D**, **I**, and **C** denote progressively smaller subsets of data objects which are determined by **P** and the profile manager that processes **P**. **D** is the *profile domain*: the set of all data objects that are specified to be of interest within **P**'s domain specification. **I** is the *profile instantiation*, consisting of the data objects in **D** the profile manager is able to locate. **C** is the chosen *data charge*: the subset of objects in **I** that the profile manager chooses to deliver to the mobile device on the basis of **P**'s utility function specification. For the business traveler profile described in this section, **D** consists of three kinds of data objects: rental car objects, direction objects, and shuttle objects, and **C** is determined from the rental car, direction, and shuttle objects the profile manager is able to find (**I**), the utility function specification that establishes the relative worth of these data objects, and the resource restrictions (e.g., bandwidth, space available on charged device) evident at the time when charging occurs.

Expressing Profile Domains — A profile's domain specification determines the data objects in **D**: the objects of interest to the client whose device gets charged. In the `DOMAIN` section of Fig. 2, we have specified the business traveler's profile domain by identifying three disjoint *data classes*: sets of data objects with similar semantic content. For this profile, `RC` names a set of rental car objects, `Dir` names a set of direction objects, and `Sh` names a set of shuttle objects.

The data classes of Fig. 2 are defined informally. Any formal language for specifying data classes must satisfy the following properties:

- *Declarative*: It must allow data sets to be specified declaratively. Thus, a profile domain language has much in common with a database query language.
- *Class membership*: It must allow membership in a class to depend on content and metadata. Content-based criteria might demand that data objects contain certain **keywords** (e.g., rental car objects must contain the keywords "rates" and "reservations") or have particular **values** for particular fields (in the case of data objects that are structured, e.g., relational database tables). Metadata-based criteria might examine a data object's **format** (e.g., shuttle schedules should be text files), **structure** (as in the case of XML data, where structure might be specified with a DTD), source (e.g., rental car objects should be taken from official Web sites for one of the three largest rental car companies in the country), **creation date**, or **size**.
- *Dynamic domain*: It must allow class membership to be determined dynamically, allowing D to expand if the profile manager finds new objects between data charges, and allowing D to contract as a result of a data object becoming stale, or uninteresting because the client's context (e.g., the client's geographical location) has changed.

One way to identify objects of interest according to content is to use keywords, as in a Web search engine. Indeed, a search engine can be viewed as a primitive "data collector" that generates a profile instantiation (the list of links it finds) given a set of keywords. Keywords are declarative specifications and permit individual objects to be examined for class membership dynamically. However, metadata-based criteria for class membership cannot be expressed with keywords alone.

The fact that profile domains are data sets makes it possible to specify data classes with database queries. There are many advantages to expressing profile domains in this way, especially with regards to data collection where query processing technology could be applied. Queries also permit interesting data to be generated and not just collected "as is," as in a profile that integrates data found in different sources. This becomes especially useful as XML becomes more ubiquitous, since data restructuring and integration is made simpler by XML's use of semantically meaningful data tags.

On the other hand, most query languages do not permit data to be queried using metadata criteria as we require. For example, SQL queries filter data according to the content of the data only. XML-based query languages such as Quilt [2] support queries that filter data according to structure also, exploiting the self-describing flavor of XML. Query languages designed for heterogeneous databases such as SchemaSQL [3] also allow the structure of data (relations) to be queried. However, none of these query languages permit data to be identified by declaratively characterizing data sources (the query languages we know of demand that sources of data be explicitly named) or other metadata properties such as the age of the data. For a query language to be extended to specify a profile domain, it will be necessary to extend its predicate language to permit reasoning about metadata properties. Indeed, the standard distinction found in query definitions between data source (`FROM` clause) and data predicate (`WHERE` clause) would have to be blurred in this context, to permit more flexi-

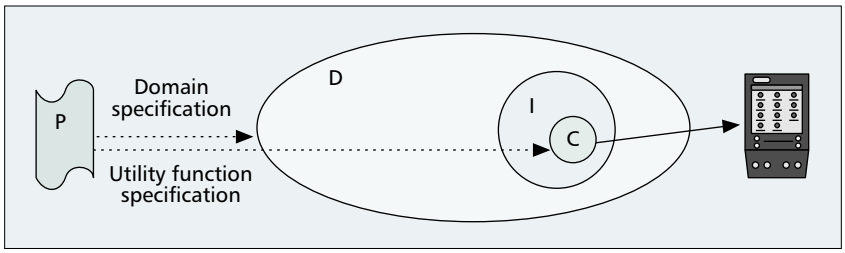

Figure 3. *A profile working in a data recharging system.*

bility in identifying desirable data sources. Appropriate processing techniques for evaluating such queries (for the purpose of data gathering) remain a topic for further research.

To support the dynamic domain requirement, queries used to specify profile domains must be persistent. A persistent query is like a view; it defines a result that evolves over time. Views are handled by database systems in two ways: they are either reevaluated when they are referenced (e.g., when a query is posed in terms of a view), or materialized and updated when the data sources they depend on are modified. Of these approaches, the latter is likely to be more appropriate in a data recharging scenario given that data collection should occur at least partially offline (i.e., between data charge sessions) to make best use of the time available during a data charge session. On the other hand, the view maintenance problem for materialized views is made more complicated because profiles need not explicitly name their data sources.

Expressing Utility Functions — A profile's utility function specification specifies the relative worth of data objects specified in its profile domain. Any language used to specify utility functions must satisfy the following requirements:

- *Conciseness*: It must permit complex utility functions to be expressed concisely; not requiring, for example, space that is exponential in the number of data classes.
- *Comparability*: It must indicate which members of a set of data objects have the most value to the client. For example, a utility function for a business traveler might indicate that a shuttle schedule object is more valuable than a rental car object.
- *Dependent Utility*: It must allow the worth of data objects to depend on the *presence* or *absence* of other data objects in the same data charge. For example, the business traveler profile should have a utility function that specifies the worth of a rental car object to be dependent on the presence of a direction object in the same data charge, and the worth of a shuttle object to be dependent on the absence of either direction or rental car objects in the same data charge.

One way to express a utility function is in terms of a function that maps data charges to integer values. We illustrate this in the `UTILITY` section of Fig. 2, where 10 equations are used to specify U: a function that maps subsets of the business traveler profile domain to integer values. Equations in the utility function specification are of the form

`U (<Class>) =` *i*

such that `<Class>` specifies a class of subsets of the profile domain (i.e., a class of potential data charges) which have the same value, *i*. To characterize data classes, we reuse the data class names (`RC`, `Dir`, and `Sh`) introduced in the `DOMAIN` section of the profile. But in this section of the profile, these names are used to describe properties of data charges, as we demonstrate with three example equations:

- The class `RC` characterizes a data charge that includes one or more rental car objects and no direction objects or shuttle objects. Therefore, the equation
  `U (RC) = 0`
  indicates that data charges that can be characterized in this way have value 0. This says that rental car information has no value when unaccompanied by directions to the traveler's hotel.
- The equation
  `U (RC [1] with Dir) = 2`
  says that any data charge that includes exactly one rental car object (`RC [1]`), one or more direction objects (`Dir`), and no shuttle objects has value 2.
- The equation
  `U (RC [0 < i < 3] with Dir with Sh) = 3`
  says that any data charge that includes one or two rental car objects (`RC [0 < i < 3]`), one or more direction objects (`Dir`), and one or more shuttle objects (`Sh`) has value 3.

The advantage of expressing the utility function in the manner shown in Fig. 2 is that it captures the comparison and dependent utility requirements for a utility function specification. One can specify the value of an individual data object of some data class, `D`, simply by defining an equation of the form

`U (D) = i`

if the value of one object in class `D` is the same as the value of several objects of class `D`, or

`U (D [i]) = f (i)`

(for some function `f`) if every object in this class gives added value when present in a data charge. To indicate that the value of an object is dependent on the presence of another object in the same charge, one simply associates with the data charge that contains objects from both data classes a value larger than the sum of values associated with the data charges that contain these objects individually. For example, the value of a data charge that includes one rental car object, at least one directions object, and no shuttle objects (`U (RC [1] with Dir`) is larger than the sum of values of rental car objects (`U (RC)`) and directions objects (`U (Dir)`) considered separately.

On the other hand, the utility function specification of Fig. 2 is not concise. The number of equations required to express the values of all potential data charges is exponential in the number of data classes (or more, if the value of a data charge depends not on the presence or absence of objects of each data class, but also on the number of objects that satisfy each data charge). The example language in this article is just meant to be illustrative of the kinds of things we need to be able to express. We have recently been investigating alternative approaches based on declarative languages such as database query languages. In such an approach, we express utilities independent of how they will be used. We essentially define utility as a function that assigns a value to any subset of the domains.

Further Research Issues

Advanced Profiling Issues

Profiles are what drive data recharging. As discussed in the previous section, the language with which we express profiles is a key component of any such system. The expressive power of that language is what provides us with opportunity. The internal representation of profiles written in that language is what facilitates our attempts to optimize and plan their use. There is, of course, a tension between the expressive power of a profile language and the ability to process it efficiently. A major thrust of our language design attempts to balance these conflicting goals.

We see the data recharging problem as requiring additional capability from a profile language in order to realize the full potential of our vision. In particular, a profile language must include facilities to specify resolution and context:

- *Resolution*. When resources such as bandwidth or memory on the mobile device are limited, it might be desirable to transmit a lower-resolution version of an object as long as that version still has some value. By sending less data, we preserve resources for other objects (possibly at lower resolution as well). Planning the appropriate resolution to send as well as the order in which to send items is the job of the data recharging infrastructure.
  What constitutes a legitimate lower-resolution object is dependent on the object's type. For audio and video, lower resolutions are obvious. For things like documents or books

(say represented in XML), a lower-resolution version might omit components. For example, a lower-resolution book might be the book without its figures. The profile language would need to specify which combinations of components make sense as well as how these combinations of components should be valued.

- *Context*. As mobile users conduct business, they find themselves in different states in space and time. We call a representation of these states the *context*. Profiles for data recharging produce different charges depending on a user's current context. A profile language must incorporate context in order to adjust the domain D and the value of charges accordingly.

  A context can be defined by the user's location, PIM-based calendar, and/or current task/subtask. For the latter, the task might be defined by a series of smaller steps in the form of a formal workflow diagram. The state of the task within that workflow would affect the data included in the charge. For example, if a user is in Washington on her way to a meeting with the President and has just completed a meeting with the Mexican ambassador, the charge might include recent documents on proposed revisions to NAFTA.

Efficient Profile Processing

A data recharging system must be capable of handling a very large number of users in the context of huge amounts of data. Finding the absolute best recharge for all users at all times is a daunting (likely exponentially complex) task. Thus, one of the biggest challenges facing any real implementation involves developing *profile optimization* (analogous to query optimization) techniques that mitigate the computational burden of the problem. It should be noted that since profiles specified by users are a best guess at where their future interests will lie, an optimal solution is likely not necessary. A good approximation will serve our purposes quite well.

When a data recharge is needed for a mobile user, we need a way to create this recharge without having to generate and test the powerset of all objects in the profile domain **D**. Here, we need ways to add objects to the charge using some kind of hill-climbing technique. Our experience to date suggests that the internal representation of profiles will have a significant impact on how expensive this is. We have begun to look at the problem of how to efficiently generate a charge and have explored greedy algorithms as well as the Precedence Constrained Knapsack Problem (PCKP) as a solution.

Furthermore, we need strategies for processing multiple user profiles together. If there is commonality in some group of profiles, it makes sense to process the common part once and share the result.

Data Management Issues

Data recharging provides data to a mobile device at any location in the network. Moreover, a fully deployed data recharging system must be able to efficiently scale to millions of users. We believe this can be achieved by providing data management capability within the data recharging system. To this end, profiles can serve a second purpose of guiding the management of data prior to its delivery. Thus, there must be a sophisticated data management and delivery infrastructure to support this.

For example, profiles can provide an opportunity to plan the flow of data in the network in order to optimize anticipated recharging events. Intuitively speaking, the data ought to follow users in their travels. While this may be impractical for all data for all users, it might be possible for popular data to move to a location that is "closest" (in terms of latency) to the highest number of interested users.

While profiles form the basis for data recharging decisions, they can also be used for making data management decisions within the data recharging infrastructure. Data management decisions include caching, prestaging, indexing, clustering, replication, and precomputation. The mode of delivery of data related to a charge would also be included as a kind of data management policy. Data delivery can be push-based or pull-based; it can be unicast or multicast. We expect that this work would contribute substantially to the development of a data recharging infrastructure.

Two-Way Data Flow

Up to this point, the data recharging discussion has focused largely on scenarios in which data recharges flow from the fixed data grid to mobile devices. In many applications, however, data will be updated or collected at the mobile device. Data that originates in the mobile device will often need to flow back into the network for use by other applications. This reverse information flow introduces a further dimension to the design of the data recharging infrastructure.

While current PDA devices support such bidirectional synchronization, the task is more difficult in the data recharging framework due to the expressiveness of the profile description language and data model. Data cached on the mobile device is essentially a materialized view of data abstracted from sites across the Internet. Thus, one problem that must be addressed is the mapping of updates back to base data. A second challenge is the maintenance of data consistency in the presence of mobile updates. Either pessimistic (i.e., avoidance-based) or optimistic (i.e., resolution-based) approaches to consistency could be used. While there are difficulties in supporting such techniques in the general case, protocols could be developed that are tailored to the specific sharing characteristics of many recharging-based applications. Such protocols will necessarily trade off strict consistency for availability in a way that is most appropriate for each type of application.

Related Work

User profiles form the basis of many types of information delivery systems, ranging from information filtering applications to the personalization of content on the World Wide Web. Data recharging profiles are unique in that they combine a powerful language for specifying predicates over data items with detailed specifications of the users' preferences, priorities, and requirements. Still, there has been significant work in user profile modeling and management on which we can draw.

File hoarding for mobile, disconnected operation has been studied in the Coda system [4] as well as the SEER system [5]. In these efforts, hoarding profiles specify user interest in files, and this information is used to prefetch files in anticipation of a disconnection event. A CODA profile is a list of file or directory path specifiers with additional annotations. An annotation, for example, can state that a directory and all its children are of interest. Such a path specification can also be assigned a priority, which matches our notion of utility. Our notion of a profile includes a more declarative specification of the data of interest and also expands the idea of a priority to include more application-level complexities such as utility dependencies. There has been some work [6] on the specification of hoarding profiles that takes the semantics of data into account. The hoards in this work are defined as partitioned views on a set of base relations without any notion of utility.

User profiles for Web-based applications (MyYahoo, PointCast [7], etc.) are typically fairly simple, allowing the user to specify particular categories of information they are interested in receiving. Such categories are often referred to

as "channels," thereby emphasizing the similarity of the service to that of broadcast media. This approach to building information dissemination systems is generally known as the "publish/subscribe" model [8]. Publish/subscribe systems tend to use a "walled garden" approach, in which the universe of data that can be delivered to the user is restricted to specific content sites. Most systems allow simple, channel-specific predicates to be applied to the data on the channels selected by the user, for example, to specify particular companies (for stock prices), cities (for weather), or teams (for sports scores) of interest to the user. The Grand Central Station system [9], developed at IBM Almaden, provides a more general form of predicates over its channels, and is therefore closer to our notion of the domain specification portion of a profile.

User profiles for text-based data have been extensively investigated in the context of information filtering and selective dissemination of information research [10]. The systems in these areas use techniques from the information retrieval (IR) world for filtering unstructured text-based documents [11]. In general, IR profile systems use either a Boolean model or a similarity-based model. In the Boolean model a user profile is constructed by combining keywords with Boolean operators (e.g., And, Or, Not), and an "exact match" semantics is used — a document either satisfies the predicate or not. Similarity-based models use a "fuzzy match" semantics in which the profiles and documents are assigned a similarity value (typically based on a vector space model [12]) . A document whose similarity to a profile is above a certain threshold is said to match that profile. The Stanford Information Filtering Tool (SIFT) [13] is a well-known content-based text filtering system for Internet news articles. With the advent of XML, filtering of Web documents based on structure as well as content has become more feasible. The XFilter system [14] is a recent example of such a filtering system.

Our notion of user profiles is also related to the notion of *continuous queries* (CQs) as studied in the database community. Continuous queries are standing queries that allow users to be informed when updates of interest occur in a database. Early work on CQ for relational databases was done by Terry *et al.* [15]. More recently, several CQ systems have been proposed for information delivery on the Internet. OpenCQ [16] employs an SQL-like query language and runs on top of a distributed information mediation system that integrates heterogeneous data sources. The NiagaraCQ system [17], allows the specification of standing queries using an XML-based query language. The C3 project [18] provides a service that allows users to subscribe to changes in semi-structured information sources.

Finally, data recharging will make heavy use of techniques to generate and deliver data at multiple resolutions. Multiresolution delivery of media has long been advocated for mobile environments (e.g., [19]). We plan to extend these techniques to deal with structured and semi-structured documents as well as to aggregated data delivered from databases.

Conclusions

As mobile devices become smaller and more pervasive, consumers of computing technology will be able to perform their tasks using many different platforms throughout their working day. We see the ability to supply these devices with the data they need in a manner as simple as the way they recharge their battery power as being crucial to realizing the full potential of device-based computing.

Data recharging is fundamentally dependent on the ability to express rich information requirements in the form of a profile. We have discussed some of the requirements of a language for expressing these profiles. Like a query language, a profile language should be declarative; however, unlike a query, a profile does not specify a single correct answer. The profile is simply a specification of items of interest and their relative values. Moreover, unlike a query, a profile can generate an "answer" that is dependent on context and the available resources in the environment.

Data recharging is a fertile area for further research. We believe that the study of profiles and their role in advanced applications like data recharging is a key area for the application of data management ideas and techniques. If this work is ultimately successful, it will allow for useful mobile and pervasive computing to take place without as much user intervention as is required today.

We are presently exploring the infrastructure that is required to build real data recharging systems. This includes the study of appropriate algorithms (e.g., caching and prefetching), the identification and construction of common services (e.g., data caches), designing appropriate APIs (e.g., negotiation protocols from device to wired machine), and building a small but realistic application (e.g., Mobile CoursePak).

Acknowledgments

This work was supported in part by the National Science Foundation under NSF Grant number IIS00-86057. Additional support was provided by NSF grant IRI96-32629, DARPA grant #N66001-99-2-8913 and by contributions from IBM, Microsoft, Sun Microsystems, and Siemens. We would like to thank Danny Tom and Matt Denny of UC Berkeley, Eddie Galvez of Brandeis University, and Don Carney, Greg Seidman, Nesime Tatbul, and Ying Xing of Brown University for their valuable contributions to the Data Recharging project.

References

[1] M. Altinel *et al.*, "DBIS Toolkit: Adaptable Middleware for Large Scale Data Delivery," *Proc. ACM SIGMOD Conf.*, Philadelphia, PA, June 1999.
[2] D. Chamberlin, J. Robie, and D. Florescu, "Quilt: An XML Query Language for Heterogeneous Data Sources," *Proc. WebDB 2000 Wksp.*, Dallas, TX, June 2000.
[3] F. Gingras *et al.*, "Langauges for Multi-database Interoperability," *Proc. ACM SIGMOD Conf.*, Tucson, AZ, May 1997.
[4] J. Kistler and M. Satyanarayanan, "Disconnected Operation in the CODA File System," *ACM Trans. Comp. Sys.*, vol. 6, no. 1, Feb. 1992, pp. 1–25.
[5] G. Kuenning, "The Design of the SEER Predictive Caching System," *Proc. Wksp. Mobile Comp. Sys. and Apps.*, Santa Cruz, CA, Dec. 1994.
[6] B. R. Badrinath and S. H. Phatak, "On Clustering in Database Servers for Supporting Mobile Clients," *Cluster Computing*, Jan. 1998, pp. 149–59.
[7] S. Ramakrishnan and V. Dayal, "The PointCast Network," *Proc. ACM SIGMOD Conf.*, Seattle, WA, May 1998.
[8] B. Oki *et al.*, "The Information Bus – An Architecture for Extensible Distributed Systems," *Proc. 14th SOSP*, Ashville, NC, Dec. 1993.
[9] Q. Lu, M. Eichstaedt, and D. Ford, "Efficient Profile Matching for Large Scale Webcasting," *7th Int'l. WWW Conf.*, Brisbane, Australia, Apr. 1998
[10] P. W. Foltz and S. T. Dumais, "Personalized Information Delivery: An Analysis of Information Filtering Methods," *Commun. ACM*, vol. 35, no. 12, Dec. 1992, pp. 51–60.
[11] N. J. Belkin and B. W. Croft, "Information Filtering and Information Retrieval: Two Sides of the Same Coin?," *Commun. ACM*, vol. 35, no. 12, Dec. 1992, pp. 29–38.
[12] G. Salton, C. S. Yang, and A. Wong, "A Vector Space Model for Information Retrieval," *Commun. ACM*, vol. 18, 1975.
[13] T. W. Yan and H. Garcia-Molina: "The SIFT Information Dissemination System," *ACM Trans. Database Sys.*, vol. 24, no. 4 1999, pp. 529–65.
[14] M. Altinel, and M. J. Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information," *Proc. VLDB Conf.*, Cairo, Egypt, Sept. 2000.
[15] D. B. Terry et al., "Continuous Queries over Append-only Databases," *Proc. ACM SIGMOD Conf.*, June 1992, pp. 321–30.
[16] L. Liu, C. Pu, and W. Tang, "Continual Queries for Internet Scale Event-Driven Information Delivery," *Special Issue on Web Technologies, IEEE Trans. Data Eng.*, Jan. 1999.
[17] J. Chen *et al.*, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," *Proc. ACM SIGMOD Conf.*, Dallas, TX, June 2000.
[18] S. Chawathe, S. Abiteboul, and J. Widom., "Representing and Querying Changes in Semistructured Data," *Proc. Int'l. Conf. Data Eng.*,

Orlando, FL, Feb. 1998.
[19] R. H. Katz, "Adaptation and Mobility in Wireless Information Systems," *IEEE Pers. Commun.*, vol. 1, no. 1, 1st qtr., 1994, pp. 6–17.

Biographies

MITCH CHERNIACK (mfc@cs.brandeis.edu) has been an assistant professor at Brandeis University since January 1999. Before that, he completed his Ph.D. at Brown University. His research interests include formal methods (especially as applied to database development), web databases and programming languages. He was a recipient of an NSF CAREER grant in 2000, for his work on developing correct query optimizers using formal methods.

MICHAEL FRANKLIN (franklin@cs.berkeley.edu) is an associate professor of computer science at the University of California, Berkeley. His research focuses on the architecture and performance of distributed databases and information systems. Previously, he was at the University of Maryland, College Park, where he led the development of the DIMSUM flexible query processing architecture and was a co-developer of the Broadcast Disks data dissemination paradigm. He is an Editor of *ACM Transactions on Database Systems* and is Program Chair for the 2002 ACM SIGMOD Conference. He received the NSF CAREER award in 1995.

STAN ZDONIK (sbz@cs.brown.edu) is currently a professor of computer science at Brown University. He received his Ph.D. from the Massachusetts Institute of Technology in 1983. He did extensive work on the topic of object-oriented databases and was co-developer of Broadcast Disks. He leads the Network Data Management Group at Brown and is the co-principal investigator of the NSF-sponsored Data Centers project. He is an Associate Editor for *ACM Computing Surveys*, the *International Journal of Distributed and Parallel Databases*, and the *Journal of Intelligent Information Systems*. His research interests include network data management, mobile database systems, profile processing, and database query processing.