Personalized Pocket Directories for Mobile Devices

Doron Cohen, Michael Herscovici, Yael Petruschka, Yoëlle S. Maarek, Aya Soffer
IBM Research Lab, Mount Carmel, Haifa 31905, Israel
{doronc,ayas}@il.ibm.com

Dave Newbold
Lotus Software Division of IBM
Westford Lab, Massachusetts, USA
dnewbold@us.ibm.com

ABSTRACT

In spite of the increase in the availability of mobile devices in the last few years, Web information is not yet as accessible from PDAs or WAP phones as it is from the desktop. In this paper, we propose a solution for supporting one of the most popular information discovery mechanisms, namely Web directory navigation, from mobile devices. Our proposed solution consists of caching enough information on the device itself in order to conduct most of the navigation actions locally (with subsecond response time) while intermittently communicating with the server to receive updates and additional data requested by the user. The cached information is captured in a "directory capsule". The directory capsule represents only the portion of the directory that is of interest to the user in a given context and is sufficiently rich and consistent to support the information needs of the user in disconnected mode. We define a novel subscription model specifically geared for Web directories and for the special needs of PDAs. This subscription model enables users to specify the parts of the directory that are of interest to them as well as the preferred granularity. We describe a mechanism for keeping the directory capsule in sync over time with the Web directory and user subscription requests. Finally, we present the Pocket Directory Browser for Palm powered computers that we have developed. The pocket directory can be used to define, view and manipulate the capsules that are stored on the Palm. We provide several usage examples of our system on the Open Directory Project, one of the largest and most popular Web directories.

Categories and Subject Descriptors

H.5 [**Information Systems**]: Information Interfaces and Presentation; K.8 [**Computing Milieux**]: Personal Computing

General Terms

Management, Human Factors

Keywords

Mobile search, Mobile devices, Personalization, Hierarchical browsers

1. INTRODUCTION

The dramatic increase in the use and availability of mobile devices such as cellular phones and Personal Digital Assistants (PDAs) in the last few years has resulted in the ability to access information anytime and anywhere. IDC [13], a leading provider of technology intelligence, forecasts that by the end of 2002 there will be more wireless subscribers capable of Internet access than wired Internet users. Yet, we are still far from the dream of having Web information as conveniently accessible from a handheld device as it is from our desktop. This is due to two main limiting factors:[1]

1. **Form factor:** Existing information discovery mechanisms for searching and browsing the Web are not well-suited to the limited screen real estate and input capabilities of handheld devices. These mechanisms need to be revisited in order to take into consideration the specific device and/or task at hand. Examples of research work in this direction have been presented at the last WWW conference and include [7, 12, 3, 1].

2. **Communication mode:** Mobile devices are typically connected through networks with low bandwidth and high latency. Both the slow response time and the still prohibitive prices of over-the-air connection make users reluctant to stay constantly connected.

In this paper, we propose a solution for supporting one of the most popular information discovery mechanisms, namely Web directory browsing, from mobile devices. There are two common modes of information discovery on the Internet: search and browse [6]. While search engines such as Google and Altavista are better at finding narrowly defined information, directories are better at presenting broad categories of information. The predominant general-purpose directories include Yahoo![30], the Open Directory Project (ODP) [24], and Looksmart[17]. The continuous growth of these directories is the best testimony of their popularity. The Open Directory, for example, consists of almost 3 million sites, more than 40,000 editors, and nearly half a million categories (as of October 2001). In addition to general-purpose directories there are numerous special purpose and regional directories. Some examples include: the Environmental Organization Web Directory [10], the UK directory [28], and the librarians' index to the Internet [16]. On the Intranet, and in the corporate world more specifically, various commercial products offer rich directories, which are often semi-automatically generated, in

[1]Other restrictions often cited are (1) limited storage capabilities and (2) limited CPU. Note however, that (1) is less of an issue these days with the ability to add memory sticks or microdrives to PDAs, and (2) has no real impact on the specific application at hand here, namely browsing.

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

order to organize corporate knowledge. Examples include the Lotus Discovery Server [15], Microsoft Sharepoint Portal Server [19] or Semio Taxonomy [26].

To date there is no convenient way to take advantage of Web directories (Intranet or Internet) from handheld devices. While it is possible to use a regular PDA Web browser to access them as any regular Web pages, limitations (1) and (2) cited above will typically make the interaction very tedious. Improving the browsing experience can be achieved in several ways. PowerBrowser [7], for instance, improves users' navigation within a Web site via an "accordion" display. Proteus [1] observes the behavior of Web users to automatically customize Web sites to mobile users. The WebView system [12] allows users to easily create simplified and personalized views of Web content. Users can record views in advance [2] and have them preprocessed on a server. The user can then get one click access to the desired content when working from their connected mobile device. WebViews represent only a small portion of the original contents and therefore reduce the amount of data that needs to be transferred to the client. Other examples of research work in this area are the Pythia [11] and Digestor [5] systems that also transform contents for devices with limited screen real-estate and limited network bandwidth.

While PowerBrowser provides an adequate solution (better than regular PDA Web browsers) for navigating within an arbitrary Web page, it would require too many over-the-air transactions to effectively use a Web directory: going down/up the tree multiple times, searching, etc. WebViews are one of the best solutions around for accessing frequently visited pages (including complex form-based services), yet it is not adequate for the specific task of directory browsing as users do not know in advance what their navigation path might be and therefore cannot prerecord a special purpose view.

There is a need for a specific solution that enables fast and convenient browsing of directories, with minimal over-the-air communication. Our proposed solution consists of caching enough information on the device itself in order to conduct most of the navigation actions locally, with subsecond response time, and possibly communicating with the server only for the last stage of viewing the desired content. This solution thus depends on having minimal storage capabilities on the mobile device, as well as regular over-the-wire (*i.e.*, typically free) connection to the Internet. These prerequisites are already met by most modern PDAs, be they PalmOS, Pocket PC, or Symbian based. Local storage capabilities are also starting to appear, as expected, in cellular telephones such as the Nokia 9210 Communicator [22] with its Symbian OS, the Motorola i85s [14], with its ability to download Java applications, running under J2ME, via a wire, and the Samsung SPH I300 Palm Powered Phone [25], etc.

Our proposed approach is consistent with the "intermittent connection" communication model. In this model, users synchronize their device via a cheap, reliable and fast connection (*e.g.*, cradle, wire, infra-red) on a regular basis so as to cache locally as much information as possible for working offline. Many PDA users seem to prefer this model over the constant connection model implicitly adopted by the related work mentioned before. It is supported by several popular applications including MapQuest [18] and AvantGo [4]. MapQuest uses a simple model of downloading preprocessed travel and direction information from the Web to the PDA. The actual data transfer to the PDA occurs during the next synchronization operation. AvantGo uses a slightly more sophisticated model: it allows PDA users to view Web content in disconnected mode by subscribing to channels, which represent collections of Web pages. Channel content is updated on the PDA at synchronization time according to the channel definitions maintained on the AvantGo server, and according to requests that where made by the PDA user in disconnected mode. The set of pages that make up a channel is defined by a root page and a set of neighboring pages up to some fixed depth.

Our Approach

We suggest to follow a similar approach, but instead of using channels based on topological proximity, devise a subscription model that will adapt to the specific nature of Web directories and comply with the services they provide. In order to make this solution effective, one key requirement is to identify the "right" amount of information that has to be stored on the local device. This "right" amount of information has then to be isolated, made usable in a stand-alone manner and eventually copied to the PDA at synchronization time.

The method that we describe in this paper encapsulates this "right information" into what we call a "directory capsule". The directory capsule represents only the portion of the directory that is of interest to the user in a given context and is sufficiently rich and consistent to support the information needs of the user in disconnected mode.

The authors have described elsewhere [3] how arbitrary units of knowledge can be preprocessed ahead of time, encapsulated in a "knowledge agent base" on the server, and then finally downloaded or beamed to a mobile device for local operations. While following the same spirit, the work presented here is fundamentally different as it focuses on the very specialized task of directory browsing rather than general purpose information gathering. The task at hand imposes specific constraints on the structure of the data that needs to be cached on the device. Namely, the local data needs to mirror the existing structure of the Web directory, and the local application needs to mimic the directory browsing service in such as way that the user experience is the same on the desktop as on the mobile device.

This highly structured "directory capsule" represents a surrogate of the original Web directory, and is the only piece of information accessed by the local application, that we call, in the following, the "Pocket Directory Browser".

In addition, the local data needs to be always the "right" one from the user's viewpoint. Keeping track of the user's preferences in the context of directory browsing is more complex than in the previous cited work on "knowledge agents" (where the user had only to specify his needs as free-text queries like in a search service). The user's object of interest might involve categories, documents, or more generally any of the various objects available in a Web directory. In order to ensure that the user's needs are fulfilled, we introduce here a novel subscription model (see Section 3.2) for users to easily express their interest while browsing the local directory.

The definition of directory capsules along with the dedicated Pocket Directory Browser holds the promise of bringing the convenience of Web directories to handheld devices.

The highlights of our approach include:

- An architecture that supports intermittently connected devices and which incurs minimal delays in terms of the user.
- A novel subscription model geared for Web directories and targeted to PDAs.
- A sophisticated mechanism to ensure that the directory capsule is consistent with the directory over time and to propagate user's requests to the directory server and back.
- A fully featured pocket directory browser application for defining, viewing and manipulating directory capsules that are stored on the PDA.

The rest of this paper is organized as follows. Section 2 introduces our approach and the concept of capsules. Section 3 formally defines the core model of our approach, including a generic directory model and the subscription model needed to generate capsules. Section 4 describes how the system maintains a consistent and fresh directory capsule. Section 5 presents an embodiment of our approach in a working system, provides some performance figure, and shows examples on one of the most popular Web directories, the Open Directory Project. Section 6 concludes and summarizes the contributions of our work. Related work is discussed as relevant throughout the paper.

2. FROM WEB DIRECTORIES TO LOCAL DIRECTORY "CAPSULES"

Several factors need to be accounted for in the task of capturing Web directory data into local directory capsules. One important characteristic of these capsules is that they are "dynamic replicas" rather than static snapshots of the directory, although their replication mode is intermittent and mediated rather than continuous. More precisely, the directory capsule is generated by a *mediator* - a software component that is responsible for accessing the Web directory on the Internet, encapsulating the right amount of information, conforming to the subscription requests and always maintaining consistency both ways between the Web directory and the capsule.

As depicted in Figure 1 the mediator component interacts with the capsule and the Web directory via two operations:

- **Synchronization** - the PDA and the machine hosting the mediator communicate through a synchronization (or sync for short) operation, during which data and requests can be transferred in both directions over the wire.
- **Load** - The mediator machine and the directory server communicate in one direction, with the mediator machine downloading data from the server via a "load" operation.

The sync and load operations differ in terms of their timing as well as in terms of associated workload. While the sync process is initiated by the user wishing to refresh the directory on the PDA, the load process can be initiated either manually by the user, or at some prescheduled time. In terms of associated workload, the load process is much

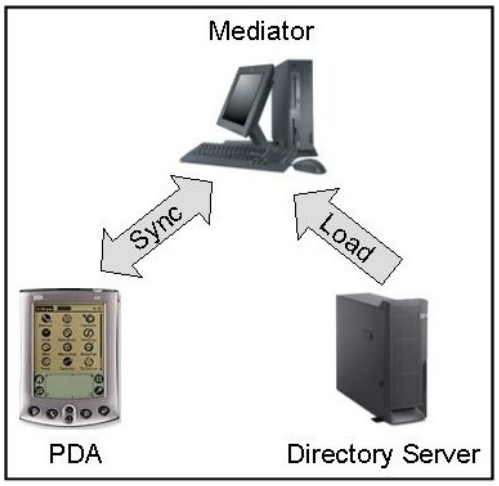

Figure 1: High Level View of Pocket Directory System

"heavier" since it needs to communicate with the directory server. Typically, the load process should be performed in advance rather then when the user explicitly requests updates via the sync process.

The basic mobile device considered here is a standard PDA such as a Palm or Visor device, with decent storage capabilities. Our approach can be however generalized to other devices including cellular phones with storage and synchronization capabilities. The mobile device connects via a wire/cradle or infra-red to a connected machine to which the user has access. This machine that we call "mediator" has access to the Internet and can reach any available directory server. Note that the exact placement of the mediator component is flexible and depends on the particular system configuration. For example, if the PDA is a Palm device, then the mediator machine is typically the desktop used to backup the device; the sync process is the Hotsync process initiated when the user presses the Hotsync button; the sync functionality is implemented within a Palm Hotsync manager conduit. In another setting, the mediator could be placed on a central Web server, which manages user subscriptions. Our model can be adapted to any of these settings as required. However, for the sake of simplicity – and since it is still the most typical configuration – we consider in the following the case where the mediator machine is the user's desktop.

The directory and the subscription models which are the central concepts in defining local capsules are defined more formally in the following sections.

3. DEFINING THE LOCAL DIRECTORY CAPSULE

As mentioned above, it is crucial to ensure that the "right" information, as perceived by the user, is extracted from the Web directory in order to produce a capsule. Two main challenges are involved. First, defining what data the capsule should contain. Second, defining the refresh patterns in terms of scheduling and granularity. In light of these challenges, it is apparent that a personal subscription mechanism suited for Web directories is required.

The idea of subscribing to Web content is not new. Nguyen et al. [21] describe a system for query subscription in an XML Webhouse. Their system is part of a larger XML warehouse system *Xyleme* [29]. Xyleme continuously crawls the Web, using some strategies to prioritize page refresh activities, stores Web XML data in an XML database, and classifies documents. Two subscription modes are supported – *monitoring*, where query results are monitored for changes, and *update*, where queries are fired repeatedly based on user defined schedules.

There are several commercial Web subscription utilities. These include *Mind-It* [20], *Northern Light* [23] and *AvantGo* [4]. Mind-it allows registered users to subscribe to several Web pages. Each subscription can be configured separately, specifying how often and which changes should trigger a notification. Northern Light allows users to subscribe to queries, but not to specific Web pages and emails notifications to them. AvantGo allows PDA users to view Web content in disconnected mode by subscribing to channels, which represent collections of Web pages. While AvantGo does not provide notifications, it does enable users to subscribe to Web data and receive regular updates. Channel content is updated on the PDA at synchronization time according to the channel definitions maintained on the AvantGo server, and according to requests that where made by the PDA user in disconnected mode. The set of pages that make up a channel is defined by a root page and a set of neighboring pages up to some fixed depth. The AvantGo channel model is very well suited for sets of pages that are regularly visited and are topologically close to a given root page.

None of the existing systems, however, is tailored to the specific needs of directories. For instance, AvantGo does not allow to define a channel as a path in the directory tree which is a must in our context. We propose here to devise a subscription model that is tailored to the hierarchical structure of the directory model. It needs to support subscriptions for the various item types managed by the directory. Most importantly, the subscription model must take into account that its target is a PDA, and as such, special concerns must be addressed. Specifically, users must be able to easily: subscribe to interesting items, be aware of potential costs in terms PDA space, find new and updated items (on the PDA), execute local searches on the data residing on the PDA.

In the remainder of this section, we first define a generic model for describing directories. Based on this model, we define the subscription model which enables users to specify the parts of the directory and the individual items that are of interest to them, as well as the desired information granularity. The derived directory subset along with the information required to support the user's information needs in disconnected mode is encapsulated to yield the directory capsule.

3.1 Directory Model

Web directories consist of a hierarchy of categories, where each category may contain documents as well as other categories. The hierarchy often has a single root, the top most category, typically called the *Top* category. This category usually contains subcategories and no documents. Documents can be associated with one or more authors. Similarly each category can be associated with one or more persons, who are considered authorities in the area corresponding to this category.

Web directories usually support some form of search, in addition to navigation and browsing. A user can typically submit queries to the entire directory or to a particular category. The search results consist of the categories and documents that answer the query best. Search results are similar in form and contents as categories, and can be viewed as "virtual categories" generated upon demand. To pursue the analogy with categories, we can assign people to query results, the same way that an editor is associated with categories. For instance, editors whose profiles match the query, or editors of categories returned by the query could be associated with the query results entity. We see therefore that while the objects or entities involved in defining a directory are well known, one can define various relationships between them.

Consequently, we propose to define directories according to the classical Entity Relationship model [9] where the entities are: predefined categories, documents, people, and query results, all involved in a set of predefined relationships.

More formally, the directory information DI can be defined as

$$DI = (E, R)$$

where

- $E = \{E_1, \ldots, E_n\}, n \geq 1$ is the set of entities, and
- $R \subseteq \{R_{ij} \mid 1 \leq i, j \leq n\}$ is the set of (directional) relationships between entities E_i and E_j.

For example, the entities represented in the Open Directory Project (ODP) are as follows:

- E_1 – OD categories
- E_2 – Web pages or "OD sites" in the ODP terminology
- E_3 – Persons
- E_4 – Query results (virtual categories)

and the relationships between these entities, from the user's viewpoint, can be represented by the matrix shown in Table 1.

Note that the relationship matrix is nonsymmetrical. Indeed, in ODP, R_{12} holds (categories may contain sites) but R_{21} does not, sites do not contain categories. Note also that relationships might exist while not being obviously exposed to the user for navigation purposes. For instance, the relationship "site belongs to a category", R'_{21}, is not listed in the matrix above (which consequently misses the E_2 row) because it is not exposed to the user when navigating the categories. This relationship presumably exists in the underlying ODP data structures as is apparent by the search results view where each site listed is typically followed by its containing directory.

The relationships listed in Table 1 are deliberately not exhaustive. More relations may be added or removed according to the amount of information that the ODP designers want to expose to users, as well as according to the features that are offered to them.

An alternative way to describe the directory information DI is as a directed labeled graph, where nodes are entities and labeled edges are the relationships (one relationship per

E_i x E_j/ R_{ij}	E_1 **category**	E_2 **site**	E_3 **person**
E_1 **category**	R_{11} "contain": category contains category	R_{12} "contain": category contains site	R_{13} "editor": category's editor is a person
	R'_{11} "parented": category parented by category		
E_3 **person**	R_{31} "edit": person edits category		
	R'_{31} "bookmark": person bookmarks category		
E_4 **query results**	R_{41} "contain": results contain category	R_{42} "contain": results contain site	

Table 1: Some relationships between entities in ODP

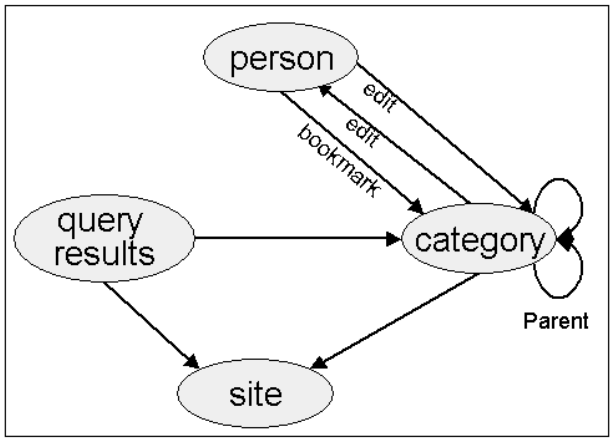

Figure 2: Open Directory Project Model

label) between the entities. A *DI* graph for the Open Directory Project is shown in Figure 2. Edges without labels are containment edges. The recursive containment edge from the *Category* entity to itself reflects the hierarchical nature of the directory: categories can contain other categories. The functionality of the directory is also described by these edges. Specifically, browsing the directory is achieved by traversing the category to category edges until reaching the desired one, and then traversing to the people or documents it contains. The *DI* graph can be further decorated to define the subscription model, as discussed next.

3.2 Subscription Model

The subscription model is key to ensuring that the "right" information, as perceived by the user, is extracted from the Web directory in order to produce a capsule. In the context of directories, subscriptions are a mechanism for users to indicate the directory entities in which they are interested. The *content* of these entities will be extracted and stored in the directory capsule. Furthermore, this content will be synchronized with the directory server in order to keep it fresh. The question is, what constitutes the content of an entity. For example, in the context of the Open Directory Project, the content of a document entity could be defined as its description only. Alternatively, the content could include the corresponding Web site as well. Similarly, for categories the content could include all of its sites, people, and subcategories, or only a subset of these.

The definition of the content will have significant implications in terms of the size of the local replica and hence on the time it takes to refresh this replica. We therefore define two granularities of content for each data item: *summary* and *details*. Summaries are stored for each entity included in the capsule, while details need only be stored for a subset of these. The directory subscription should thus include the definition of the entity summary and the entity details for each entity in the system. These definitions govern the behavior of the application, as we now demonstrate. Table 2 gives the definitions of "summary" and "details" for the ODP entities.

Another definition for categories could be as follows:

- **Category summary** - name, description, and summaries of contained categories and of its parent category.

- **Category details** - summaries of editors and of contained sites.

Although this is a rather small change, it makes a very significant difference. Using the first definition, the content of unsubscribed categories is not stored on the PDA. On the other hand, according to the second definition, assuming (reasonably) that the PDA contains the summary of the home category, the PDA (recursively) contains the summary of each and every category. Such a definition is clearly not adequate for very large directories such as the ODP, yet it may be suitable and convenient for other, smaller, corporate and/or domain specific directories.

The hierarchical structure of directory categories is often called a *taxonomy*. The taxonomy consists of category information, but it does not include other data items such as people or documents. The two definitions suggested above differ in that the second definition stores the entire taxonomy on the PDA while the first one does not.

When the taxonomy is too large to be stored in its entirety on the PDA, as in the case of the ODP, a more flexible solution may be desired. In particular, for each subscribed category we would like to store a few additional neighboring categories, rather than just immediate neighbors, yet not store all of its sub/parent categories. Furthermore, in some cases, we may wish to vary the granularity per entity. For example, store category details but only summaries of sites. In order to support this flexibility, we can formulate the definition of the *entity content* and define it in an inductive manner.

E_i **x item**	**summary**	**details**
E_1 **category**	category name and description	summaries of editors, of contained sites, of contained and direct parent categories
E_2 **site**	site title, description, and URL	the complete site page
E_3 **person**	person name and profile	email, summaries of categories edited by person, and bookmarks
E_4 **query results**	the query that produced these results	summaries of returned sites and categories

Table 2: Summary and details for ODP entities

Let $N(x)$ be the set of immediately reachable neighbors of x. For each neighbor y of x, i.e., $y \in N(x)$, let $s(x,y)$ be a Boolean value indicating whether the summary of y should be included when including the summary of x. Similarly, let $d(x,y)$ be a Boolean value indicating whether the details of y should be included when including the details of x.

Finally, let $S_0(x)$ and $D_0(x)$ denote the summary and detail information associated with node x, respectively. We define summary and details recursively as follows:

$$S(x,k) \doteq \begin{cases} S_0(x) & k = 0 \\ S_0(x) \bigcup_{y \in N(x) \wedge s(x,y)=T} S(y, k-1) & k > 0 \end{cases}$$

$$D(x,k) \doteq \begin{cases} S_0(x) \cup D_0(x) & k = 0 \\ D_0(x) \bigcup_{y \in N(x) \wedge d(x,y)=T} S(y, k-1)) & k > 0 \end{cases}$$

Based on these definitions, we can complete the definition of the *content* of entity x: Let $k_s(x)$ be the requested summary depth for a subscribed entity x, and $k_d(x)$ be the requested details depth for a subscribed entity x, then the content of x is defined as

$$C(x) \doteq C(x, k_s(x), k_d(x)) \doteq S(x, k_s(x)) \cup D(x, k_d(x))$$

A few points to note. If we use the definition in its precise form, then the summary/details of some nodes may be inserted more than once since the directory graph may include cycles. This is not strictly a problem since we define the summary/details as a set. An implementation of this model should keep track of the visited nodes as well as the associated depths in order to handle cycles correctly. Another point to note is that sometimes we may wish to actually include the summary of certain entities along the entire graph (*e.g.*, when we wish to include the entire directory in the capsule). Using the above definition, we would need to specify a very large summary level, k, in this case. We can, however, use the convention that if k is not specified, then we include the entire graph. In other words

$$S(x) \doteq S_0(x) \bigcup_{y \in N(x) \wedge s(x,y)=T} S(y)$$

Again, in terms of implementation the application must keep track of the visited nodes.

The subscription model for a directory with model DI can be defined based on these definitions by decorating DI as follows:

- For each node x, specify the summary and details subscription depth $k_s(x)$ and $k_d(x)$, respectively.
- For each node x, specify the fields to include in the summary and details, respectively.
- For each edge (x,y), specify the Boolean values $s(x,y)$ and $d(x,y)$, indicating whether the summary/detail of the entity being pointed to should be included.

We illustrate the use of this subscription model in Figure 3 by decorating the ODP directory depicted in Figure 2. The model we use follows the first of the two definitions suggested above for the ODP model (*i.e*, the summary includes basic fields and the details include one level of neighbors).

Each node is decorated with the subscription information (connected by a dotted line in the figure). The left side includes the summary depth and the summary fields, while the right side includes the details depth and fields. Additionally, each edge is decorated with two Boolean values. The left value indicates whether to include the summary of the pointed entity as part of the summary of the pointing entity. Similarly, the right value indicates this for the details.

Defining subscriptions in this manner provides tremendous flexibility. For example, if the desired behavior is that subscribing to a category will bring the summary of all of its ancestor categories and descendant categories up to a distance of 5 "generations", all that is required is to set $k_s(category) = 5$ (rather than 0), and to set *s(category, category)* to T (rather than F).

Note that this model does not allow inclusion of neighbor details along with the details of an entity, since the assumption is that this would make the capsule grow too much. The model can, however, be extended to allow this by modifying the edge labels to specify whether summaries or details should be included along with the entity details.

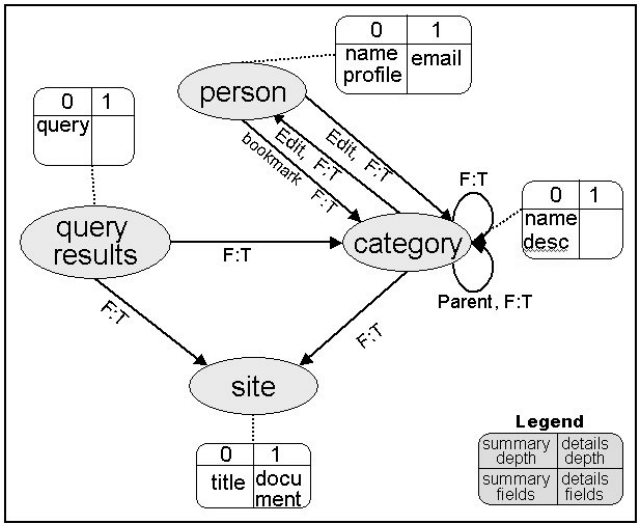

Figure 3: Open Directory Project – Subscription Model

4. MAINTAINING A CONSISTENT AND FRESH DIRECTORY CAPSULE

The key requirement for the local capsule is to remain up to date and consistent with the parent information in the Web directory. This task is handled by the mediator component, and must be conducted as fast as possible so as not affect the duration of the synchronization stage too significantly, at least from the user's viewpoint.

The capsule data can be divided into two disjoint parts – *capsule definition* and *capsule content*. The capsule definition (or *definition* for short) consists of the user subscription requests, while the capsule content (or *content*) is made of data loaded from the Web directory in order to fulfill these requests.

Inconsistency at a given point in time may occur in two cases:

- **User induced inconsistency**
  The user added or deleted subscriptions since the last sync operation. The capsule content should be updated accordingly.
- **Directory induced inconsistency**
  The directory has evolved since the last sync operation – entities and relationships between them might have changed. Both capsule content and capsule definition should be updated appropriately.

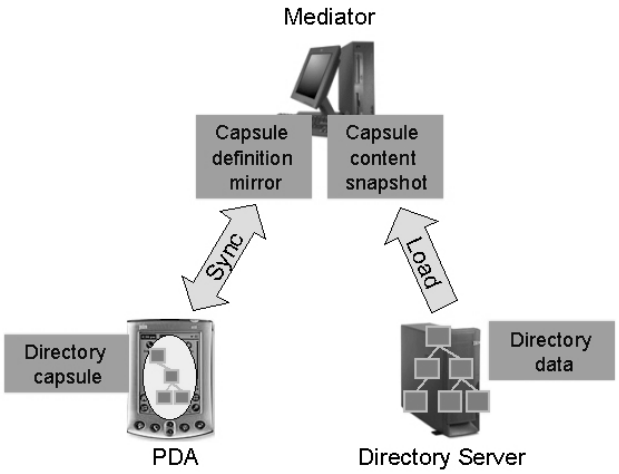

Figure 4: Data sets used for consistency maintenance

Both types of inconsistencies are resolved by the mediator component, using two data sets, as illustrated in Figure 4: a *mirror* of the capsule definition as downloaded from the PDA during the last sync, and a *snapshot* of the capsule content that was generated according to the definition mirror, and updated during the last connection with the directory server.

With these data structures defined, the task of the mediator can be defined as a sequence of the following two independent operations:

- **During sync**
  A new definition mirror is obtained from the device. Unsubscribed items are deleted from the content. Definition and content are stored on the device.
- **During load**
  The capsule content snapshot is updated from the directory according to the definition mirror. Newly subscribed items are retrieved, subscribed items that were modified on the directory are updated.

There must be a mechanism to ensure that the load and the sync operations do not occur concurrently as this could result in an inconsistent state. Special care must be taken if the directory model allows for deletion of entities. In this case, deleted items must be removed from the content snapshot during the load operation, and the capsule definition must be modified and stored on the device during sync operation, because items that were subscribed might not exist anymore on the directory.

For the purpose of efficiency, an additional mirror, of the capsule content that was stored on the device in the most recent sync, can also be maintained by the mediator. The sync process becomes more efficient, since (1) computing which items are new or updated can be achieved during sync by comparing the capsule content snapshot to the device content mirror, without having to download the complete capsule content from the device, and (2) only the updated or new items need to be stored on the device, thus largely reducing the sync time.

5. SAMPLE APPLICATION - THE POCKET DIRECTORY

Our approach has been embodied into a full system, the "Pocket Directory", that consists of a PalmOS client, fully implemented in C (for performance reasons) and a mediator component fully implemented in Java. The typical user performs a one time install of the client, the Pocket Directory Browser, on his PalmOS device, and then downloads and updates capsules via the client interface. In our case where the mediator machine is the user's desktop, the mediator component can be augmented with an interface, which gives more control to the user. The mediator interface allows the user for instance to trigger the load operation at will, so as to prepare the capsule content snapshot in advance (out of scheduled times) and speed up the next sync. For the sake of the generality though, we will not describe the mediator interface, since it is not crucial to the operation of the system and will be exclusively used by advanced users. We will restrict ourselves to the Pocket Directory Browser on the PDA which is described below.

5.1 Pocket Directory Browser

The key features of the Pocket Directory Browser are:

- **Navigation/Viewing**
  The Pocket Directory Browser provides a complete hyper-link system for navigating between entities. It gives easy access to the three main entities (categories, documents, people) via a smart 3-tab view. The query subscriptions can be accessed by a special menu and then browsed via the same 3-tab view. The navigation user's experience is described in the next section. Navigation is further facilitated via a rich history mechanism (allowing users to navigate back and forth between views), and shortcut buttons for frequently used views. Special attention is paid to the viewing process with dedicated document and person summary views,

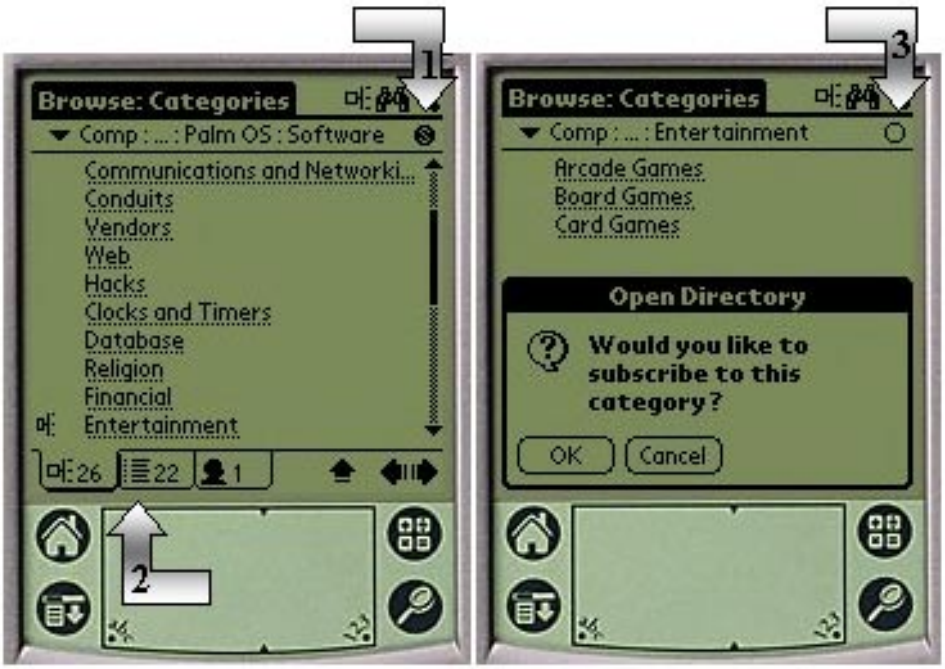

Figure 5: Subscribing to a category

as well as full HTML document display for cached documents.

- **Local search**
  Fast local search is supported by a full-text information retrieval for PalmOS, namely "Pirate Search", developed by the authors and described elsewhere [8]. Pirate decouples between the actual indexing of documents, which is performed on the desktop, and the retrieval which is done in disconnected mode on the PDA, after the inverted index is copied over at sync time. This model is consistent with the Pocket Directory approach, as the indexing component can be added to the mediator, and the retrieval component integrated into the Pocket Directory Browser to support local search. The key features of the search are demonstrated in the next section and include: subsecond response time, free-text querying with stemming, scope selection (between categories, people and documents separately), ranked results, highlight of query terms in candidates, etc. In addition, query results are shown in an ordinary category view (as follow up to the virtual category concept) and can be browsed via the same 3-tab mechanism described above.

- **Category/Query subscription**
  Users can subscribe/unsuscribe to categories and queries via a simple click mechanism as exemplified in the next section. In addition, navigation to subscribed categories (actual or virtual as results of subscribed queries) is facilitated by a special view showing them as a flat list upon demand (by clicking on the top right wrench icon in the interface).

- **New items**
  One additional special view (that is invoked by clicking on the same wrench icon) gives access to all new items, *i.e.*, new or updated categories, documents and people, again shown as a virtual category under the 3-tab view.

- **Palm feature integration**
  One important requirement for most PalmOS applications, is the smooth integration with the native PalmOS features and applications. The Pocket Directory Browser compiles to this need by allowing users to export persons' summaries to the PalmOS address book, and document summaries to the PalmOS Memo application. Documents (full or summaries) can also be beamed to another Palm as a memo. Finally, summary fields can be selected and copied to the Palm clipboard. One can envision even further integration in more advanced devices: turning email addresses or phone numbers in people summaries into hotlinks in order to allow for the following operations: Click on an address to invoke the local email application, on a phone number for speed dial in devices that have telephony capabilities.

5.2 Some Examples

We give below some examples of interactions with the Pocket Directory Browser in disconnected mode, after it has replicated some specific capsules from the ODP as required by the user. In our first example, the user has subscribed to a specific category "PalmOS Software". Immediately after the first sync is performed, the user can browse the contents of the subscribed category as shown in the left pane of Figure 5. The fact that the category in question is subscribed is indicated by the small "s" icon (follow arrow #1). The contents of the category is organized according to our model, and classified into 3 tabs (follow arrow #2). The first tab, which is selected by default, is the category tab, and there

Figure 6: Viewing people information

Figure 7: Issuing a free-text query

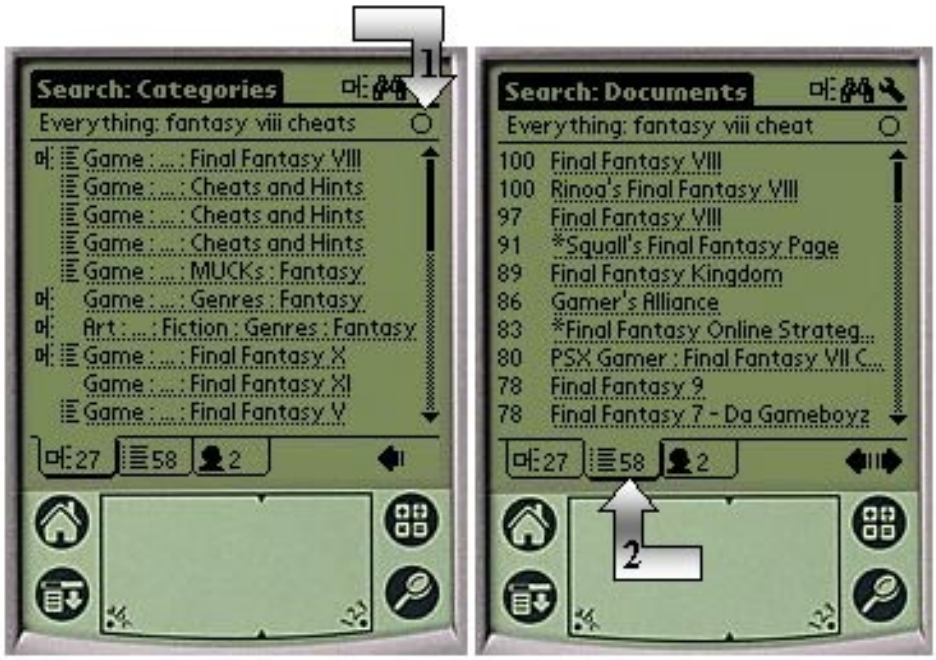

Figure 8: Organized results

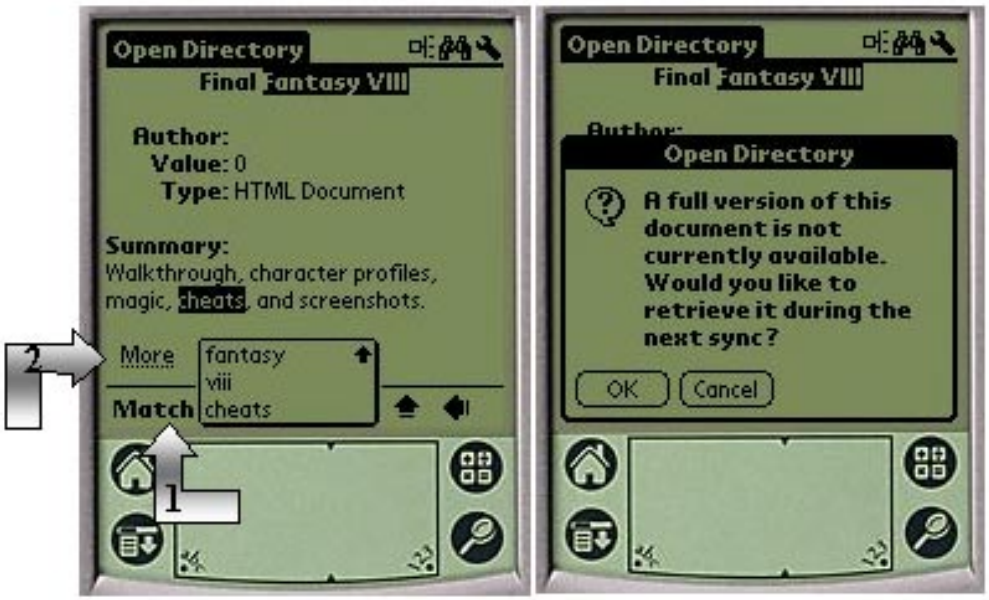

Figure 9: Viewing document results - Ask for more

Capsule	# subscribed queries	# subscribed categories	# category summaries	# document summaries	# person summaries	total size
Final fantasy	1	20	384	833	2	182 Kb
Seafood recipes	1	7	321	543	1	114 Kb
Palm Software	0	1	111	23	1	23 Kb
Stephen King	1	7	195	147	2	49 Kb
Survivor show	3	1	76	80	1	23 Kb

Table 3: Capsule data

are indeed, as indicated in the tab label, 26 subcategories in the subscribed category. The other tabs indicate that there are also 22 documents available and 1 person expert in the domain. Assume now that the user decides to browse deeper in the category tree, and explore the "Entertainment" subtree (last category in the left pane of the same figure). Even before clicking, it is apparent that this particular category has descendants, since it is decorated by a small "subtree" icon on the left. By clicking on it, the category is entered and 3 subcategories are seen (right pane of the figure). That category seems interesting to the user who decides at this point to subscribe to it, by clicking on the subscription circle (follow arrow #3) which is at this moment empty. A popup appears, and if the user confirms this action, at the next sync operation, the sub tree will get populated.

One interesting feature of our application is the people information. Since ODP is generated by people, it is important to know who these people are and what they recommended. In our example of a PalmOS software capsule, if one clicks on the people tab for PalmOS, 1 editor is returned, as shown in Figure 6, left pane. The person in question is named "tim", by clicking on it, one gets more information about him as shown in the right pane of the figure. The people information is even more crucial in Intranet directories, especially for large corporations, since these people are then considered experts in a given domain who can help their colleagues. This feature is for instance a key feature of the Lotus Discovery Server mentioned earlier, where these "affinities" between people and categories or documents are automatically identified, and therefore a greater number of experts is exposed to the public.

When a capsule becomes too large, simply because its associated contents on the Web directory is rich, browsing step by step becomes tedious, the user typically prefers searching the contents of the capsule by issuing a free-text query. We consider the following example here of a "Final fantasy" category, which is a very rich and active one on ODP. In order to search the capsule, the user clicks on the binocular icon of the browser toolbar (see arrow #1 in Figure 7, left pane). The popup that then appears allows the user to issue a free-text query against the capsule. It is possible to reduce the scope of the query to documents, people or category, by clicking on the combo box indicated by arrow #2 in the right pane of the same figure.

Once the user has clicked on the "Go" button of the Search dialog, the results shown in Figure 8 are returned by our Pirate local search component in subsecond respond time. The results, can be viewed as a virtual category as explained before, and as such can be browsed in exactly the same way, via the categories (27 category matches as per the tab label), documents (58 document matches) and people (2 person matches) tabs at the bottom. Note that if the user has a regular interest in finding the "cheats" for the Final Fantasy VIII game, he can subscribe to the query by clicking on the subscription circle (arrow #1) in the same way he would subscribe to a category. In our particular case, the user prefers to explore the document matches as in a regular search engine, by clicking on the document tab (see arrow #2). The documents are ranked by relevance, labeled with their score (normalized between 0 and 100). The first candidate has a very high score, and the user decides to view it by clicking on it.

Figure 9 shows the first candidate selected by the user, and automatically highlights the query terms that appear in the document as shown in the left pane. Note that by clicking on the "Match" combo box (arrow #1), the user can select whether to highlight all or just one specific term. By default, a summary of the full document is shown, by clicking on the "More" hotlink (arrow #2), the user gets the popup shown on the right pane of the figure and is warned that the full document was not replicated in the initial capsule. The user can indicate that the want to retrieve the document in its entirety at the next sync, by clicking on the OK button.

One desired feature, in case the PDA has wireless connection, is not to wait for the next sync for single document viewing but to retrieve it over the air, possibly truncating it if it is too large to guarantee a decent response time. The key advantage of using the local capsule in this case is that the over-the-air connection, if it occurs at all, is reserved to the very last stage when the adequate document has already been identified.

5.3 Performance

Table 3 shows data about some of the capsules that were used for testing the Pocket Directory system with ODP.

On top of the sizes shown in the table, we need to take into account the size of the full Web pages that are stored in the Pocket Directory's document cache. The Pocket Directory Browser application does not show embedded images, so the size of a cached full page would, on a rough estimate, revolve around 15 Kb per page, uncompressed.

Full text indices of categories, documents and people comprise about 30 percent of the total size of the data.

6. CONCLUSIONS

In this paper, we have introduced a model for browsing Web directories from mobile devices that have minimal amount of local storage capabilities. Our model follows the paradigm of intermittent connection, *i.e.,* are synchronized on a regular basis with a desktop or a server connected to the Internet. We have embodied our approach in a product-level system, Pocket LDS, which will be available early next year. Pocket LDS allows Lotus Discovery Server (LDS) customers to access their corporate knowledge from Palm powered de-

vices. The usefulness of this model for Web directories in general has been demonstrated in this paper by applying it to the Open Directory Project site.

We believe that this intermittent connection model is here to stay, whatever the ultimate mobile device turns out to be, whether it is a faster Palm VII, a Pocket PC, or a mobile phone integrated with a PDA. Indeed, even with the advent of 3G networks, the over-the-air response time will not be as reliable (some places simply prevent people from being connected such as airplanes) and as fast as on local devices. Even if mobile phones will soon open the doors of the Internet to populations who do not have access to desktops with which they would synchronize, we believe that the intermittent connection model will still be of value as long as "cradle stations" or "infra-red kiosks" are made available at convenient places like airports, gas stations, malls, etc. We also envision that CPU and local storage capabilities will keep increasing[2], and consequently users will be encouraged to store more and more information locally for reference. As more and more information is made available, disconnected information discovery mechanisms, such as the one presented here, will be more valuable.

The innovation of this work is not only on porting a given information discovery mechanism, namely directory browsing, to a specific mobile platform but also in defining a mechanism for personalizing and caching an organization or Web directory. Granted that the PDAs will be essentially limitless in a few years in terms of massive storage, CPU and better displays, we believe that Pocket Directories will continue to provide the essential mediation between the user and cognitive ontologies or worldviews and even a World Wide Directory (should one make its appearance).

7. REFERENCES

[1] C. Anderson, P. Domingos, and D. Weld. Personalizing Web sites for mobile users. In *Proceedings of the WWW10 Conference*, Hong-Kong, May 2001.

[2] V. Anupam, J. Freire, B. Kumar, and D. Lieuwen. Automating web navigation with the WebVCR. In *Proceedings of the WWW09 Conference*, Amsterdam, The Netherlands, May 2000.

[3] Y. Aridor, D. Carmel, R. Lempel, Y. S. Maarek, and A. Soffer. Knowledge encapsulation for focused search on pervasive devices. In *Proceedings of the WWW10 Conference*, Hong-Kong, May 2001.

[4] AvantGo, http://www.avantgo.com.

[5] T. W. Bickmore and B. Schilit. Digestor: Device-independent access to the Word Wide Web. In *Proceedings of the WWW6 Conference*, Santa-Clara, CA, 1997.

[6] C. M. Bowman, P. B. Danzig, U. Manber, and M. F. Schwartz. Scalable internet: Resource discovery. *Communications of the ACM*, 37(8), August 1994.

[7] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Seeing the whole in parts: Text summarization for web browsing on handheld devices. In *Proceedings of the WWW10 Conference*, Hong-Kong, May 2001.

[8] D. Carmel, D. Cohen, M. Herscovici, and Y. S. Maarek. Pirate - an information retrieval system for the Palm platform. In *Proceedings of ISCOL'01*. Haifa, Israel, February 2001.

[9] P. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[10] Environmental Web Directory, http://www.webdirectory.com.

[11] A. Fox and E. Brewer. Reducing WWW latency and bandwidth requirements by real-time distillation. In *Proceedings of WWW5*, Paris, 1996.

[12] J. Freire, B. Kumar, and D. Lieuwen. WebViews: Accessing personalized web content and services. In *Proceedings of the WWW10 Conference*, Hong-Kong, May 2001.

[13] IDC, http://www.idc.com.

[14] i85s, Motorola Cellular Phone, http://www.motorola.com/LMPS/iDEN/products/i85s/i85s.html.

[15] Lotus Discovery Server, http://www.lotus.com/discoveryserver.

[16] Librarians Internet Index, http://www.lii.org.

[17] Looksmart. http://www.looksmart.com.

[18] MapQuest, http://www.mapquest.com.

[19] Microsoft Sharepoint Portal Server. http://www.microsoft.com/sharepoint/portalserver.asp.

[20] MindIt, http://mindit.netmind.com.

[21] B. Nguyen, S. Abiteboul, G. Cobena, and L. Mignet. Query subscription in an XML webhouse. In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, 2000. http://citeseer.nj.nec.com/nguyen00query.html.

[22] Nokia 9210 Communicator. http://www.nokia.com/phones/9210/.

[23] Northern Light, http://www.northernlight.com.

[24] The Open Directory Project (AKA dmoz). http://dmoz.org.

[25] Samsung SPH I300 Palm Powered Phone. http://www.samsungusa.com/wireless.

[26] Semio taxonomy product. http://www.semio.com/products/semiotaxonomy.html.

[27] TRGPro, now called Handera, http://www.handera.com.

[28] UK Directory, http://www.ukdirectory.co.uk.

[29] Xyleme, http://www.xyleme.com.

[30] Yahoo! http://www.yahoo.com.

[2]Already today, the Handera Palm Powered device [27], for instance, has an expansion slot that supports CompactFlash^{tm} cards. The slot can be used to provide up to 1GB of additional storage.