

MobiShare: Sharing Context-Dependent Data and Services among Mobile Devices*

Efstratios Valavanis,¹ Christopher Ververidis,² Michalis Vazirgianis,¹ George C. Polyzos²

¹DB-net Research Group
²Mobile Multimedia Laboratory
Department of Informatics
Athens University of Economics and Business
Athens 104 34, Greece

{valavani, chris, mvazirg, polyzos}@aueb.gr
<http://www.db-net.aueb.gr>, <http://mm.aueb.gr>

Abstract

Rapid advances in wireless and mobile communications and computing technologies have enabled handy personal mobile devices to be used in everyday life. Such small mobile devices can now become not only information browsers and service clients, but also information and service providers, complementing or replacing fixed hosts connected to the wireline network. Such mobile resources can be very important for other moving users and mobile devices, creating significant opportunities for many interesting and novel applications. The MobiShare architecture outlined in this paper provides a scalable infrastructure for ubiquitous mobile access to dynamic information and mechanisms for publishing, discovering and accessing heterogeneous mobile resources in a large area, taking into account the context of both providers and requestors. Any wireless communication technology could be employed between a device and the system. In addition, the use of XML-related languages and protocols for describing and exchanging metadata gives the system a uniform, easily adaptable and platform-independent interface, allowing a variety of devices to use it. The overall approach is data-centric and service-oriented, implying that all the devices are treated both as potential providers or requestors of data wrapped as information m-services.

* This research is partially supported by EU IST project *DBGlobe* (IST-2001-32645) under the FET Proactive Initiative on Global Computing for the co-operation of autonomous and mobile entities in dynamic environments.

1. Introduction

The recent wave of innovations in computing, wireless communications, networking and electronics has embedded processing power, storage space and communication capabilities in electronic devices of everyday use (“information appliances”[3]), leading to the era of ubiquitous computing [1]. The main characteristics of this change is the relationship “one user–many computers” [2], as well as pervasiveness [5], invisibility [3] and mobility of computing nodes. For example, notebook computers with wireless Internet connectivity can carry mobile Web sites or Web-enabled databases, handheld devices such as PDAs or cellular phones with embedded cameras can capture and transmit multimedia content, computers with GPS receivers are embedded in vehicles providing location information and maps, smart rooms and buildings offer Internet connectivity and location-specific services to their users. In the vision of interconnecting diverse resources and their potential consumers under a common framework, many research challenges arise, such as *ubiquitous connectivity*, *heterogeneity of sources*, effective and precise resource discovery [15].

Context-awareness has also been elevated to an essential system feature in environments where objects are constantly moving and demonstrate unpredictable behavior and volatile characteristics. *Context* can be defined as the properties of a system that make it aware of its (user’s) state and surroundings and help it adapt its behavior accordingly [20]. Ideally systems should sense dynamic information in real-time from the user’s environment, such as position, orientation, temperature, lighting conditions, people’s identity, user’s emotional state and many more. Due to advances in relevant technologies [22] and the low cost of many sensor components, the main issues concentrate around providing a universal context representation, selecting the appropriate context for each type of application, and utilizing this information in order to add value.

The focus of our work is on identifying the key issues, developing appropriate techniques to address them and in the end on providing innovative solutions for handling information and resources that are not only requested, but also provided (shared) by moving nodes of relatively small computing power and storage space, interconnected via a basically unreliable wireless network. We support a service-oriented approach: data is made publicly available through services [7]. Services provide a well-defined and published interface that allows access to data in a universally understandable and customizable manner. Therefore, we use the concept of a *request* instead of a *query*, although the expected result is usually data. Services also have the advantage of being more easily handled by machines, allowing the existence of mechanisms that automatically categorize the resources and refine the requests semantically, while minimizing user intervention. Our notion of services extends the m-service class [22] since we exploit both user and source context assuming that all the mobility issues mentioned above apply to the sources as well.

MobiShare is proposing a middleware system and an infrastructure architecture acting as a communications network for this middleware, by offering ubiquitous connectivity to mobile hosts. The purpose of the middleware is to provide an enhanced resource publishing and discovery mechanism and a way to capture and use context for optimizing the functionality of the system and protecting the user from result overflow. Access to the system is provided through wireless access points, tied to local administration servers that manage the area of coverage of these access points. Whenever a device enters the range of an access point it connects to the corresponding administration server using wireless communication technologies (Figure 1). Should an authorized user want her device to become an active source, she has to submit a formalized description of the resources it provides. If the device is incapable of providing resources or its users do not wish to do so, it should at least be able

to submit requests in a predefined format. Every local administration server is responsible for maintaining a repository of devices and services available inside its area of authority. Furthermore, it provides discovery capability with semantic matching features, to assist the users locating the services that can fulfill their needs and performs context-based filtering on request results. Having located the appropriate service and the device that provides it, the requestor-device communicates directly with the source-device.

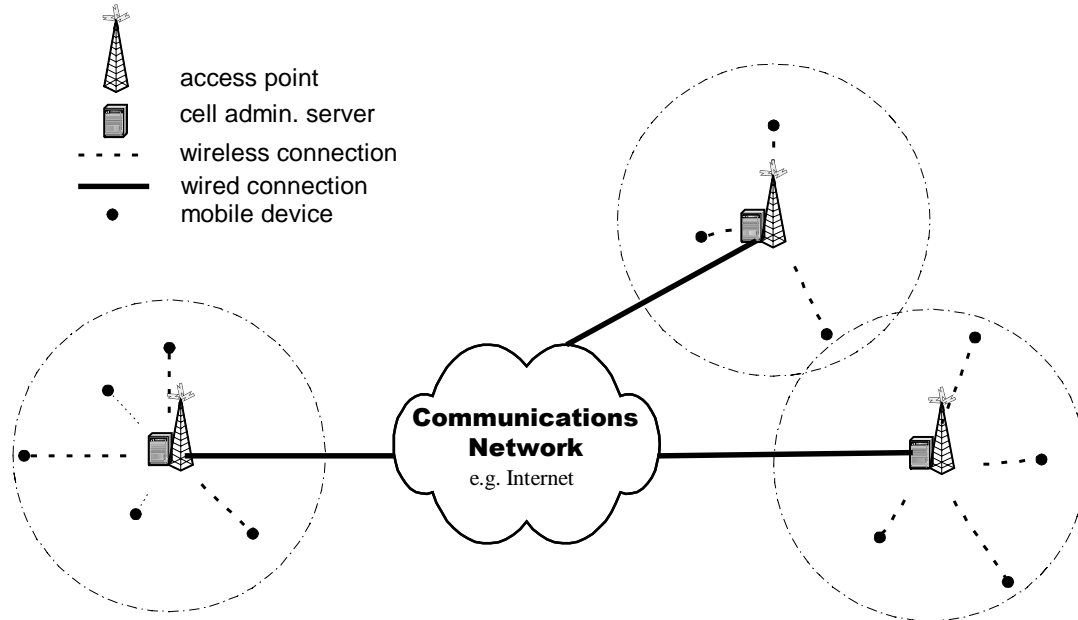


Figure 1: Device connectivity

The remainder of this paper is organized as follows: Section 2 gives an overview of some related research activities, in Section 3 we discuss system design decisions, in Section 4 we describe important aspects of the system's functionality (service discovery), Section 5 is dedicated to the prototype implementation, in Section 6 we outline our future plans and in Section 7 we provide our conclusions.

2. Related work

There are many areas relevant to MobiShare and therefore numerous research efforts deal with several of the aforementioned issues. We summarize the most interesting ones with respect to our objectives next.

Mobile Web research focuses mostly on mobile browsers (i.e., clients only) that request data and services hosted on fixed locations, dealing with the problems of security, quality of service and content delivery in various formats [4], [21]. In particular, *MobileIQ* [21] is a distributed proxy-based system that offers mobility management, personalization and asynchronous operation to mobile hosts. *MobileIQ* is dealing with the problems of ubiquitous access and host disconnections. Regarding context-awareness it tries to reduce the waste of wireless bandwidth and constrain information overloading by predicting and aggregating on the wired side the Web content that the mobile user would probably ask for. Personal preferences is a valuable type of context however it is questionable whether the method of performing clustering on collaborative filtering predictions is effective in such an

environment, since it assumes that the request results are rated by the users and that there is a big enough group of ratings for each possible user in order to produce results. Furthermore, there is no direct support for mobility of sources and the result of requests is assumed to be static Web content rather than services thus not dealing with heterogeneity of sources.

At the architectural level, the *DBGlobe* approach (as described in [18]) is considering the use of chained hierarchies of directories, where unsatisfied requests are forwarded to a higher geographical authority and then sequentially along a chain of global-scale authorities. Pure peer-to-peer architectures do not consider authorities of any kind, instead they consider all participating devices as equal and requests are forwarded on a neighbor-to-neighbor basis [23]. We make use of regional authorities, which index local resources and can respond to requests about their area autonomously. Wider-scale requests are propagated on a neighbor-to-neighbor basis.

Recent efforts in the area of mobile services are the *Centaurus* framework [10], the *Konark* middleware [11] and *XMIDDLE* [12]. *Centaurus* provides a uniform infrastructure for a multitude of devices to use heterogeneous services and is targeted to serving the needs of smart rooms. The client connects to the nearest CS (*Centaurus* System) and the CS is responsible for maintaining a list of available services and executing them on behalf of the client. *Centaurus* does not provide service discovery features, since smart rooms usually offer few services that can be presented on a list. The CS cannot communicate with any adjacent CS to retrieve services from other smart rooms. Finally, mobility is supported but the position or the orientation of the device are not actually used. The use of infrared for communications in the first version of the prototype is also restrictive (line-of-sight is needed and the range is short). *Konark* is a middleware for service discovery and delivery of device independent services in ad-hoc networks. This implies that *Konark* assumes totally (or at least partially) ad-hoc networks without fixed nodes that have additional capabilities, functionality or offer more trust. No infrastructure or centralized server is used. Every device maintains a tree-based structure (written in XML) as registry of the hierarchy of services to facilitate service management. The tree is uniform and present across all devices implementing the *Konark* Architecture. The obvious advantage of this approach is the capability of devices to communicate with each other without the need of a central system. However, this would demand either a simplistic taxonomy of services or an overwhelming computational and storage-space overhead for small devices, like cellular phones. Moreover, it would demand a bandwidth overhead to keep the service list up-to-date. *XMIDDLE* is a middleware that handles disconnections, low/expensive bandwidth, scarce resources and addresses replication and synchronization problems and other mobility related issues (location-aware services). Like *Konark*, *XMIDDLE* is targeted to pure ad-hoc networks. No discovery services are set-up and all hosts have the same capabilities.

3. System design

We will provide an overview of *MobiShare* and discuss design decisions concerning the space model, the infrastructure, the essential system components, their role and the device and service location mechanisms.

3.1 Space Model and Infrastructure

In *MobiShare* geographic two-dimensional space is divided into cells. A *communication cell* is actually the area covered by a wireless network access point (equivalent to a base station in cellular communications networks) and ideally looks like a disk (see Figure 1). Every device (mobile or fixed) is connected to exactly one access point at any given time.

Actually a device can possibly be in a location where cells overlap. A device can perform a handover to a new cell, for example if an access point with stronger signal is available. The Cell Administration Server (CAS) within the cell keeps track of the devices that enter or leave the cell's boundaries (and connect to it), locates their position and keeps information (metadata) describing their status and the resources they offer or services they provide. It is possible for a (physical) CAS to manage more than one cell. That is convenient for pragmatic reasons, assuming capacity of the CAS can handle the maximum number of devices expected in all administered cells. Grouping multiple communication cells into an area managed by a single CAS forms an Administration Cell. For reasons of simplicity we will assume in the remainder of this paper that each CAS manages only one communication cell.

Unlike ad-hoc network approaches, *MobiShare* assumes a certain infrastructure. The general concept of the architecture shares some common features with cellular networks (e.g. space models of area coverage). This choice reflects our intention to effectively support mobility and build a prototype that could support most of the existing wireless technologies. We decided to avoid a pure ad-hoc network approach, where all mobile nodes have equal capabilities and functionality and form spontaneous networks [17]. It was considered more efficient and closer to the real world to design an infrastructure consisting of more powerful computers that perform tasks such as service discovery and indexing, while the mobile devices simply provide the actual service execution and issue requests.

3.2 Mobile devices

The *mobile device* that interacts with the system could be any autonomous, portable, electronic device capable of connecting to the access point and communicating with the CAS of a cell. Another characteristic property of such a device is that it can submit requests or produce data, wrap it as a service, communicate with an administration server and function as a source. Each mobile device has some built-in, global identifier (i.e. IMEI on GSM mobile phones) associated with user identification during registration. In the first stage of this research work we assume a one-to-one relationship between users and devices. Devices can be of fixed location (desktop computer, sensor controller) or mobile (PDA, notebook, cell phone). The mobile device offers a certain functionality that makes use of the following hardware components: *Digital Compass* (for capturing orientation), *GPS receiver* (for capturing positional information), *wireless communication interface* (i.e. WLAN card). Software components include a *request definition tool*, a *service definition tool*, an *application server* (only for devices able to function as sources) and a set of viewers for documents, *images*, *media*, etc. We use documents conforming to the CC/PP standard [6] to store the static profile of each device (containing the device capabilities) and a dynamic profile in RDF[25] to describe dynamic properties (location, orientation, connection bandwidth, session information).

3.3 Cell administration servers

Each CAS is an autonomous centralized system that manages a cell and the devices that are online in the cell. Each cell can potentially support large numbers of mobile devices moving inside its area and acting as sources or requestors. The CAS functionalities include: assigning addresses and identifiers to these devices (i.e. IP addresses by a DHCP server module), performing authentication and access control, handling requests, publishing the services offered by these devices, hosting the service description files (e.g. WSDL files), maintaining a list of the addresses of other cell administration servers, etc.

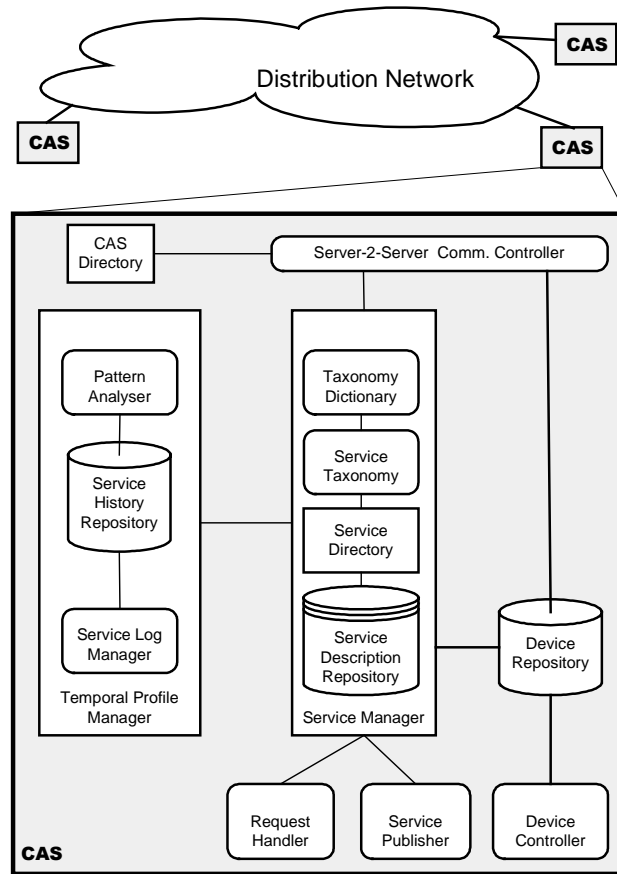


Figure 2: Cell Administration Server (CAS)

The CASes are interconnected through a communications network, e.g. the Internet or a WAN. Although they can function autonomously they are also aware of their neighbors (that manage geographically adjacent cells) and cooperate to increase the range of requests. The possible data flows between the CASes are: (a) extension of requests to neighboring CASes, (b) forwarding the list of neighbors (c) request/deliver device location information (d) exchanging service description files (proactively and on-demand).

A CAS consists of the following modules (see Figure 2):

- A global *service taxonomy* of service categories, specific to the context domain (i.e. travel and tourism) having a hierarchical structure (starting from a universal concept). A *taxonomy dictionary* contains all the terms in the taxonomy, ordered lexicographically to allow fast search and matching.
- *Service directory* that lists all the services offered by devices in the cell.
- *Service description repository* of the local services.
- *CAS directory*, containing addresses of other CAS.
- *Device repository* containing the list of devices in the cell and their profiles, and
- *Temporal profile manager* for storing the history of device appearances, discovering patterns and estimating probabilities of next appearance to assist proactive behavior.

In the simplest scenario, when a device moves from cell_A to cell_B, all the descriptions of the services the device provides should also move from CAS_A to CAS_B (unless the specified advertisement policy prohibits it, see below). Service descriptions remain in CAS_A as well,

with an indication that the source is *off* the cell. The pattern analyzer estimates the probability of device appearance in cell_A based on historical data. As the device moves across cells there is always one CAS responsible for maintaining the master copy of a service description (home cell CAS). This CAS is also keeping a list of other CAS (description holders) where this description has been propagated. The device is aware of the identifier and the address of the CAS that holds the master copy. Each new CAS receiving the service and holding the description document (either because the device moved in the cell or proactively) has to notify the initial CAS. If the user of the device updates the service description, the CAS where this update took place takes ownership of the master copy, retrieves the list of description holders and notifies them.

Maintaining metadata for every service everywhere (full-replication architecture [23]) would result in unnecessary waste of resources (storage space and bandwidth) to keep them up-to-date. Therefore, we use a specific *advertisement policy*. When publishing the service the user has to: 1) declare an initial geographic area where he wishes the service to be available (user-defined policy) and 2) specify whether the service is fixed (i.e. an on-site service giving information about a monument) or mobile. In the case of a mobile service (i.e. a picture sharing service on the phone of a tourist) the user must decide if the service availability area should be extended to the areas where the device moves (mobility-based policy) or to the areas where it is requested (request-based policy). This advertisement profile of a service is part of the service description and the CAS of the cells that publish the service are aware of it, since it determines whether they should proactively push the description to their neighbors.

3.4 Locating devices and services

The device location mechanism is part of a CAS and determines the address of the CAS of the cell where a specific device is currently connected. Every mobile device is registered in one cell, which acts as a *home cell* for the device. The home cell's CAS is responsible of keeping track of the device's current location. The device is aware of the home cell and reports it to every new cell it enters. The cell is then responsible to report the presence of the device to the home cell. Once a device enters a cell it reports: its presence, the previous cell where it comes from, the home cell where it is registered and receives a network address, which is valid for the session in the current cell. The network address of the device is logged in the device repository of the cell's CAS. The device also reports the services it hosts, the CAS servers that hold the master copies and retransmits their descriptions if necessary, i.e. when their copies are stale. The CAS registers the device's services in the service directory and requests their descriptions from the holder of the master copy.

If a service hosted on the specific device is requested, the CAS searches through its device repository and returns the current network address of the host device and the service description document. In case of failure to locate the device the server will forward the request to its neighbors--who might forward it further--until the device is discovered, a hop counter is exhausted or the request is explicitly cancelled. The final response will contain the address of the device in the cell it currently is connected to. To make this mechanism more efficient the CAS keeps an extra attribute for each device in the device repository holding the previous and next the cell it moved to (pointer-forwarding technique). Whenever a device arrives in a cell, the cell's CAS notifies the CAS of the previous cell about the presence of the device. This optimizes the process of locating a device. The CAS simply has to follow the chain of hosting servers (beginning from the device's home CAS which in most cases will be informed about the latest position of the device) to locate the device (see fig. 3). Only in the

case of failure the CAS would start broadcasting the request to the neighboring cells. The overall process is significantly faster and saves the network from unnecessary messages [8].

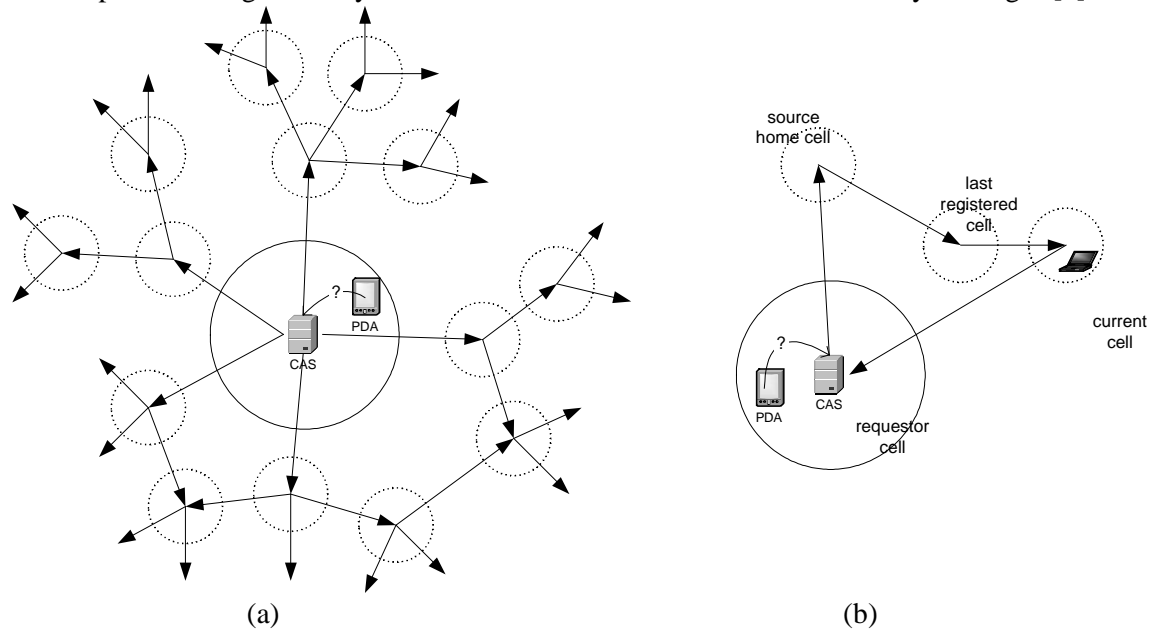


Figure 3: Illustration of (a) device request broadcasting vs. (b) pointer forwarding

4. Service discovery

Service discovery refers to a mechanism that allows users to locate services on-demand and in reasonable time [13]. It is crucial for a service discovery mechanism to retrieve all the available resources and reduce or eliminate the results that do not fit the user's needs, increasing the *recall* and the *precision* [15] of the answer. The results are further refined by context (i.e. location of the source and the requestor) to avoid sending excessive results back to the device.

4.1 Service request

There are several types of requests that can be supported by the system:

- (i) local requests, issued in the requestor's current cell,
- (ii) remote-directed, directed to a specific cell other than the current,
- (iii) geographically clustered, issued in multiple cells using spatial constraints,
- (iv) geographically dispersed, issued in all the cells, and
- (v) nearest neighbor requests, issued in the local cell and if the result is not satisfactory expanded to all the adjacent cells one hop away [19].

The simplest scenario of a local request is that of a mobile user (a tourist with a mobile device wandering around a city with a deployed *MobiShare* infrastructure) who wishes to submit a specific request to the system (i.e. wants to locate services that allow him to find taxis inside his cell).

4.1.1 Request process. The tourist issues a request with the initial search string "book taxi". The string is sent to the CAS of the user's current cell. A request handler is initiated for the specific request and stores the metadata of the request (time and location of requestor). The request handler sends the search string to the taxonomy module and receives a set of service

identifiers in return. These services semantically match the request initiated by the user. The set of answers is forwarded to the user, who selects one or more. The selected services are then returned to the service handler, which in turn request the service directory and the device directory to find which service instances are available in the cell and which devices offer them. The device then receives a list of devices refined by location, and their IP addresses, the user select the most appropriate one and invokes it.

In case the user is not satisfied with the results the search can be extended to adjacent cells (nearest neighbor request). This implies that the local request handler retrieves the list of neighboring CAS, re-issues the request and initiates child request handlers on these servers. These request handlers collect the results in their area and return them to the “parent” request handler process.

4.1.2 Semantic matching. The user does not know in advance which services are available at the moment of request, or the necessary interfaces to interact with a service. Therefore the first step in the request process (see above) is to understand what the user is looking for and offer him a set of alternative services that provide this kind of output. This would not be possible by simply using the standardized service description language WSDL [14] and the UDDI [24] mechanism for publishing/discovering service descriptions. WSDL descriptions do not contain semantic information and the search would perform exact matching of words such as “taxi” and “reservation” in the name and short description of the service. Therefore, we provide the device with a request definition tool that assists the user to describe the request in a formal manner assisted by the static global taxonomy of services accessed through the local CAS.

The initial set of words sent by the user $\{w_1, \dots, w_k\}$, is matched in the dictionary of terms (thesaurus) to find exact lexicographical matches. The dictionary works as an index to the taxonomy, to avoid traversing the entire taxonomy for every request. For a given set of words, there are potentially more than one matches in the taxonomy (i.e. due to homonyms), therefore the query set is mapped to different paths in the taxonomy. Using an appropriate distance measure [16], we select the part of the taxonomy where the set of words appear closer to each other. Thus, we locate the best possible conceptual context for the user request, with respect to our taxonomy.

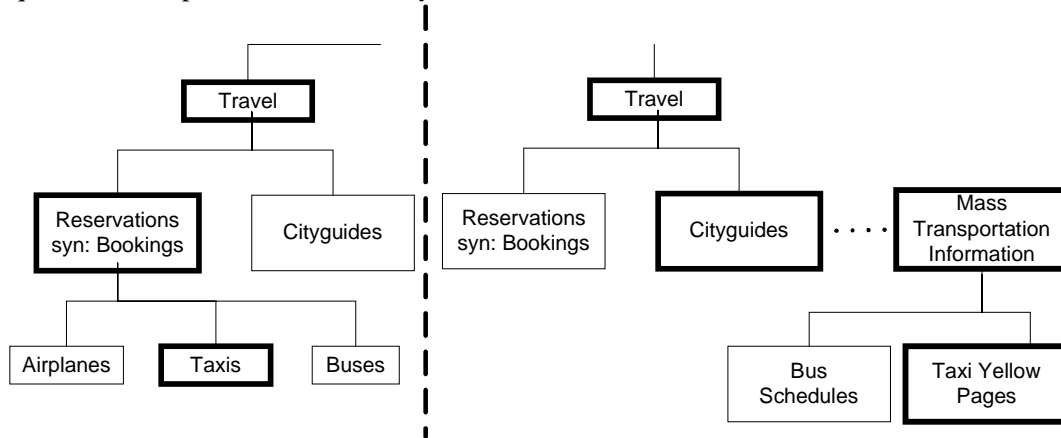


Figure 4: Taxonomy paths

Assume that the user is issuing a request for “travel, book, taxi”. One appearance of these words in the taxonomy is related to the path:

Travel → Reservation (synonym: bookings) → **Taxis**

and the other to the path:

Books → Travel → Cityguides ^{relevant-to} **→Mass transp. information→ Taxi yellow pages**

The matching mechanism will prefer the first appearance as more relevant (see Figure. 4). Having chosen one path in the taxonomy we traverse the categorization down to the leaves and select all the services that are categorized there. In some cases there might be a big set of services that fulfill these requirements. In that case we exclude the services that originate from devices that are not on-line in that cell at the moment or the ones that should not be expected to enter the cell soon.

This taxonomy is universal and static for our system and stored in RDF format. A service can possibly belong to more than one category. On top of all service classes (or concepts or categories) lies the universal concept (i.e. travel and tourism).

4.1.3 Context-awareness effect in requests. Capturing, storing and using context is necessary in order to minimize *user distraction* when using the system, to reduce information overflow and bandwidth waste when the response reaches the mobile requestor and to assist the proactive behavior of the system.

There are two types of context that we are currently considering: location of requestor and source and mobility parameters of the requestor. Whenever a user issues a request both the location and the orientation of her device are appended to the actual request. As described above the current cell of the requestor is the default target of the request, unless she explicitly extends the request. This mechanism assumes locality of requests –which is expectable in the application domains we are considering- without however restricting broader requests such as in smart-room systems [10]. If the user is located at the border of the cell or moving fast towards it, the CAS automatically forwards the request to the appropriate CAS that manages the adjacent area, thus demonstrating the proactive character of the system.

The mobile user can also specify a *request radius* around his position, along with the request. The request handler on the CAS will refine the results of the request by filtering out the services outside the defined area. This is important in case the cell covers a large area and there exist too many services that semantically match the request. The CAS will reduce the result to include only services located on devices around the user, before returning it.

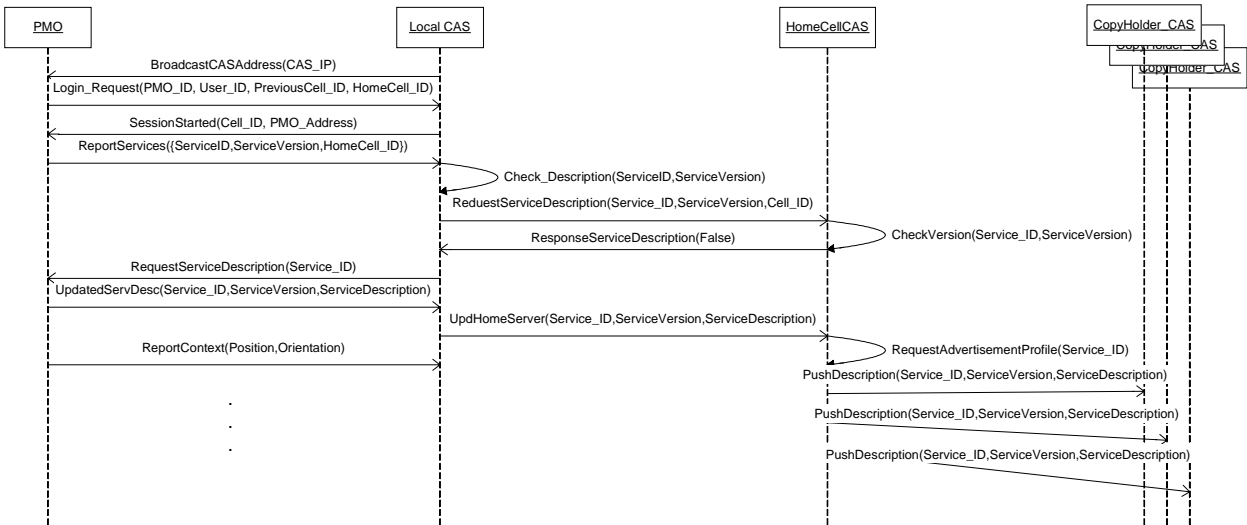


Figure 5: Sequence diagram of a service description submission scenario

4.2 Service description submission

Assume that a specific mobile user wishes to publish a new service (i.e. a cab driver is accepting reservations on his mobile phone or a sports fan sharing pictures on his PDA). A service definition tool guides the user through the process of classifying his newly published service in the fixed taxonomy of service descriptions. The user has to define: the service name, a short description of the service that includes important keywords that help classifying the service, the position in the taxonomy and a description of the service's interface (input and output parameters). The approach has to be compliant to existing technologies and able to incorporate existing services. Therefore, a WSDL service description is used to define the user interface and the semantic based information is stored in RDF format. Finally, the user has to select an advertisement profile (see Subsection 3.2) for the new service choosing between user-defined, mobility-based and request-based advertisement policy.

In Figure 5 we provide a sequence diagram showing the typical message exchange between the participating system components in a scenario where a mobile device providing a service enters a new cell. We assume that the mobile device has altered the service description while it was disconnected. The current CAS contacts the CAS that holds the service description master copy (see Subsection 3.3), verifies the version conflict, requests retransmission of the service description and the holder of the master copy redistributes the new version.

5. Implementation status

A prototype of the *MobiShare* architecture, consisting of the various software modules and a hardware infrastructure, has been developed and is currently under further refinement.

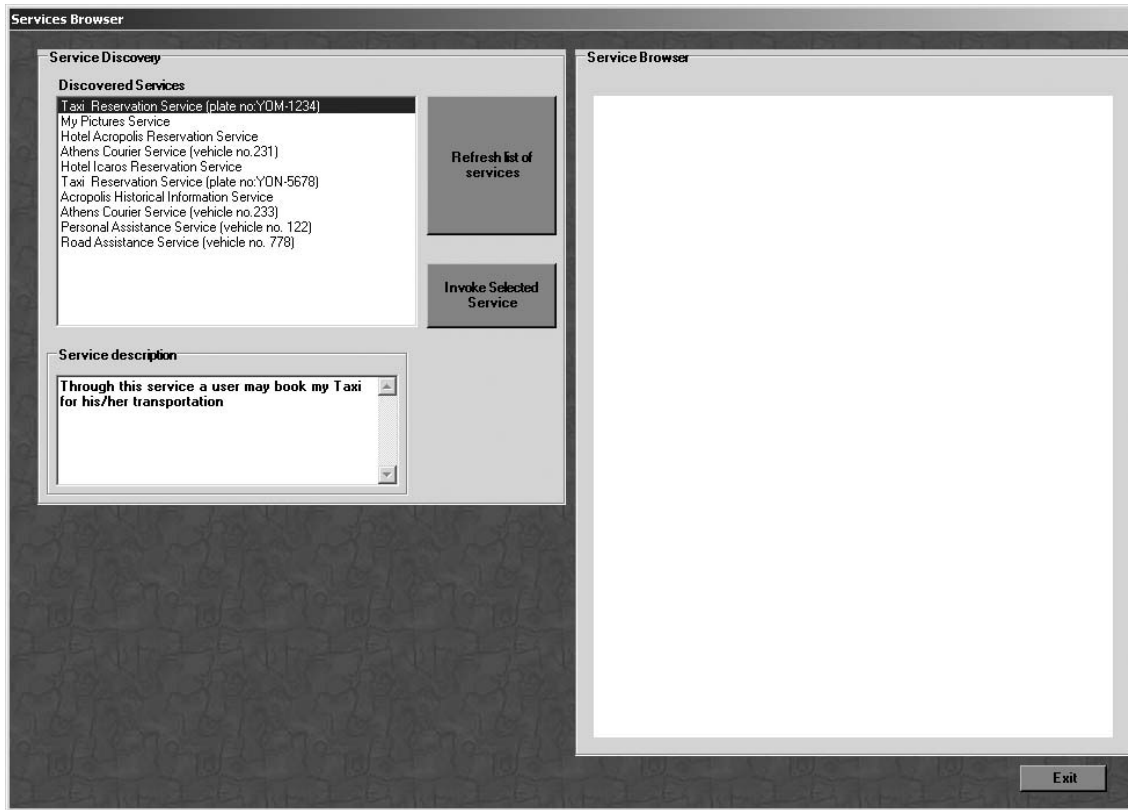


Figure 6: Request results and viewer

We have chosen the IEEE 802.11b WLAN standard for the wireless communication layer and notebook computers equipped with WLAN cards as mobile devices. Of course, any other wireless communication technology could be used instead. WLAN technology was chosen because it offers low cost (since it operates in unlicensed frequency bands), reasonable range (about 1 Km diameter in open space and coverage of many rooms or offices on one floor in a building), high bandwidth (up to 11Mb/s for IEEE 802.11b and up to 54 Mb/s for IEEE 802.11a interfaces) and does not have line-of-sight limitations. We used WLAN access points to setup cells covering different areas and the cells are managed by Microsoft Windows 2000 servers running the *MobiShare* CAS module.

The mobile device software and the CAS modules for the prototype were developed using C# for *.NETTM*. The main reason for this choice was that the *.NETTM* framework offers support for most of the state-of-the-art standards related to network programming and ensures interoperability and platform independence. The developed modules can potentially be installed on any desktop or notebook PC (a Linux version of the *.NETTM* framework is also available), as well as on many handheld devices. We based our metadata description and information exchange on standards such as XML, RDF, WSDL, CC/PP and SOAP so that parts of the system can be developed using other programming environments (e.g., Java).

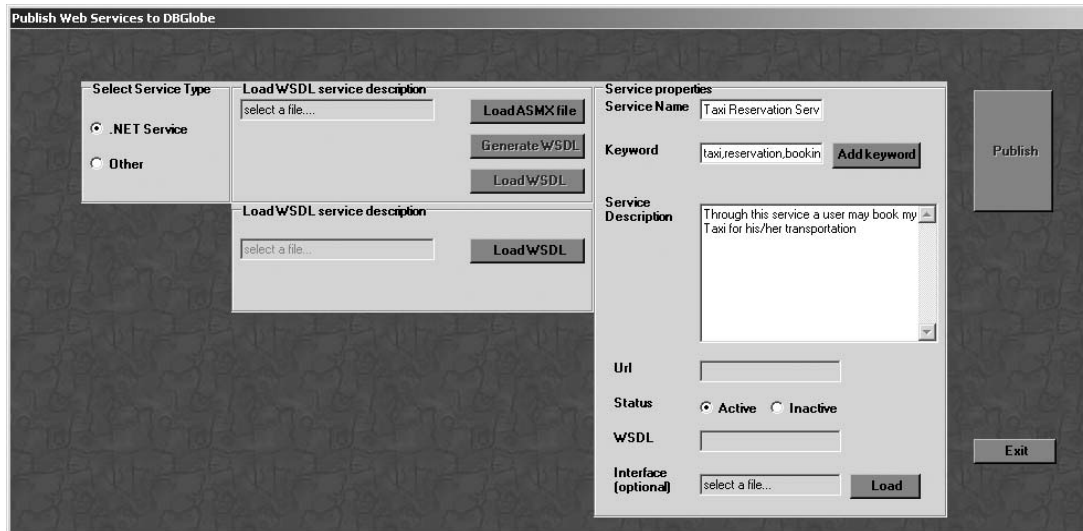


Figure 7: Service Definition Tool

The CAS modules that have been implemented are the Device Repository, the Device Controller, the Service Publisher, the Service Request Handler, the Service Description Repository and the Service Directory. The mobile device modules that have already been implemented are the Request Definition Tool (see Figure 6), the Application Server (actually, a lightweight Web server), the Viewer (in the form of an embedded Internet browser) and the Service Definition Tool (see Figure 7).

The development of a prototype system allows us to investigate a number of key system parameters and we believe it will permit us to refine our architecture and extend the functionality of the middleware we are developing. For example, we have been able to actually measure the time t_{reg} needed for a mobile device to detect that it has entered a cell and adapt to the new environment (get a network address and the address of the CAS). This time is about 2 seconds with line-of-sight between the mobile device and the CAS and seems to not be affected by the number of mobile devices. A second metric we determined is the time between a service publication request issued by a mobile device to its local CAS and the activation of this service in the cell t_{sub} . This time is 4-6 seconds for simple services with description files of 15 Kbytes.

6. Future work

Our efforts are currently focused on improving and extending the prototype and evaluating the architecture based on experimental results. Future plans include the extension of the framework's role to include centralized service execution, i.e., the CAS could also act as a source proxy, instead of simply suggesting to the user services that fulfill his needs. This will help dealing with disconnected resources, low-bandwidth connections and increased popularity of certain resources leading to high contention. Scalability and performance for such a system can only realistically be tested through simulation and we are working in this direction.

In the area of semantic service discovery we are planning to introduce semantics for the service input/output parameters (in addition to service semantics) that should be taken into account for answering requests.

The most important restriction imposed by our current communication technology is the fact that a single interface cannot simultaneously work in both “ad-hoc” and “infrastructure” modes. Consequently, we cannot use the prototype to investigate mechanisms that allow devices to automatically look for nearby devices when a CAS is not in range. We are considering the use of multiple wireless protocols to test fallback techniques that support ad-hoc system operation in isolated groups of devices.

Security and privacy issues were not a top priority this first implementation. Thus our prototype is exposed to various risks. For example, plaintext password authentication of users makes the framework vulnerable to unauthorized access. The absence of a safe security protocol for the communications facilitates man-in-the-middle attacks between the device and the CAS. These problems and security issues in general will be addressed in the future.

7. Conclusions

This paper presented *MobiShare*, a middleware platform that supports publishing, advertising and semantic discovery of mobile resources encapsulated in services. The key distinguishing feature of our approach is the focus on services provided by heterogeneous mobile devices (rather than the traditional approach of mobile devices being just clients). We discussed issues related to ubiquitous connectivity for mobile devices, context and location determination, request propagation and result delivery, and disconnected operation. We advocated, taking advantage of context to refine requests and reduce flooding for requests and results. We have also briefly described our prototype that we implemented in order to evaluate this framework and obtain meaningful parameters and insight for further work in this area.

MobiShare offers ubiquitous access and effectively supports integration of mobile heterogeneous resources, thus offering new capabilities to a wide range of application domains. Through this framework consumers and businesses can take advantage of their data and services located solely on mobile nodes. Consumer-to-consumer applications that depend on location such as tourist information sharing, location-dependent games, and business-to-consumer applications such as transportation reservations, can incorporate user resources on-the-fly and redistribute them to other mobile users avoiding the bottlenecks of centralized architectures.

Acknowledgements

We would like to thank K. Nørvgå and C. Doukeridis for their helpful discussions on various topics related to this paper.

References

- [1] M. Weiser, “Ubiquitous Computing,” *IEEE Computer*, October 1993.
- [2] M. Weiser, J.S. Brown, “The Coming Age of Calm Technology,” in *Beyond Calculation: The Next Fifty Years of Computing*, P J. Denning, R. M. Metcalfe, Eds., Copernicus, 1998.
- [3] D. Norman, *The Invisible Computer*, MIT Press, 1998.
- [4] A.C. Snoeren, H. Balakrishnan, “An End-to-End Approach to Host Mobility,” *Proc. 6th International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

- [5] M.Satyanarayanan, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, August 2001.
- [6] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, M. H. Butler, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies," W3C Working Draft, 2002, <http://www.w3.org/TR/CCPP-struct-vocab/>.
- [7] Web Services Activity, <http://www.w3.org/2002/ws/>.
- [8] I. Akyildiz, J. McNair, J. S. M. Ho, H. Uzunalioglu and W. Wang, "Mobility Management in Next Generation Wireless Systems," *Proceedings of IEEE*, Vol. 87, No. 8, August 1999.
- [9] M. Paolucci, T. Kawamura, T. R. Payne and K. Sycara, "Importing the Semantic Web in UDDI," in Proceedings of Web Service, E-business and Semantic Web Workshop (to appear).
- [10] L. Kagal and V. Korolev and H. Chen and A. Joshi and T. Finin, "Centaurus: A framework for intelligent services in a mobile environment," in Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC), April 2001.
- [11] S. Helal, N. Desai and V. Verma, "Konark--A Service Discovery and Delivery Protocol for Ad-hoc Networks," http://www.harris.cise.ufl.edu/projects/publications/konark_paper.pdf
- [12] C. Mascolo, L. Capra, W. Emmerich, "An XML based Middleware for Peer-to-Peer Computing," 1st IEEE International Conference of Peer-to-Peer Computing, August 2001.
- [13] S.A.McIlraith, T.C.Son, H.Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, 16(2):46-53, (Special Issue on the Semantic Web) March/April, 2001.
- [14] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," W3C Note 15 March 2001, <http://www.w3.org/TR/wsdl>.
- [15] A. Bernstein and M. Klein, "Towards High-Precision Service Retrieval," International Semantic Web Conference (ISWC 2002), June 9-12, 2002.
- [16] P. Resnik, "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity and Natural Language," *Journal of Artificial Intelligence Research*, 11:95-130, 1999.
- [17] C. Ververidis, S. Valavanis, M. Vazirgiannis, G.C. Polyzos, "An Architecture for Sharing, Discovering and Accessing Mobile Data and Services: Location and Mobility Issues," LOBSTER workshop, Myconos, Greece, 2002.
- [18] A. Karakasidis and E. Pitoura, "DBGlobe: A Data-Centric Approach to Global Computing," Proc. 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02), July 02 - 05, 2002.
- [19] Jianliang Xu, Dik Lun Lee, "Querying Location-dependent Data in Wireless Cellular Environment," WAP Forum--W3C Workshop on Position Dependent Information Services, February 15-16, 2000.
- [20] M. Satyanarayanan, "Challenges in Implementing a Context-Aware System," *IEEE Pervasive Computing* (Context-Aware Computing), July-September 2002.
- [21] P. Chandrasekaran, A. Joshi "MobileIQ : A Framework for Mobile Information Access," Third International Conference on Mobile Data Management, January 08-11, Singapore, 2002.
- [22] C.S. Jensen, "Research Challenges in Location-Enabled M-Services," Third International Conference on Mobile Data Management, Singapore, January 08-11, 2002.

- [23] B. Yang, H. Garcia-Molina, "Comparing Hybrid Peer-To-Peer Systems," Proceedings of the 27th VLDB Conference, Rome, Italy, 2001.
- [24] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI", IEEE Internet Computing, vol.6, no.2, March 2002.
- [25] '*Resource Description Framework (RDF) Model and Syntax Specification*', W3C Recommendation, February 1999, <http://www.w3.org/TR/REC-rdf-syntax/>.