Context-aware Web Services for Distributed Retrieval of Points of Interest*

Claudio Bettini Daniele Riboni
DICo, University of Milan, Italy
via Comelico 39, I-20135 Milan, Italy
{bettini,riboni}@dico.unimi.it

Abstract

Due to the widespread availability of accurate localization technologies, navigation systems are more and more present on mobile devices. These applications usually provide facilities for managing and searching points of interest. *However, currently available navigation software does not support sharing of points of interest, and the search facilities are quite primitive, being exclusively based on location and categories. In this paper we present novel algorithms for context-aware retrieval of distributed resources. These algorithms can be executed on any unstructured peer-to-peer network, and are based on the distributed evaluation of a scoring function that takes into account a wide set of context data. The algorithms preserve the correctness of the result set until a certain time-to-live, while reducing the exchange of data in the network. The proposed algorithms have been integrated into a Web service-based, peer-to-peer system for management and sharing of an extended form of points of interest.*

1 Introduction

The number of GPS enabled mobile devices is rapidly growing, and this or similar technologies for accurate localization will be eventually integrated in most mobile phones. Navigation systems are recognized as the killer application for the adoption of localization technologies on handheld devices. Today, most applications for navigation support provide facilities for managing and searching *points of interest*, i.e., physical locations corresponding to resources that can be interesting to users (e.g., hotels, museums, gas stations).

In current commercial products, the list of points of interest is provided with the navigation software and stored on the device. Hence, navigation systems generally do not provide facilities for sharing points of interest between communities of users; moreover, points of interest can be searched only on the basis of the user location, and of a category of interest. We believe that, in order to effectively satisfy the user demand, a search facility for points of interest should adopt much more sophisticated techniques. In particular, more accurate search results could be obtained if a much wider set of context data – including user activity, interests, time of day– would be considered.

In this paper we present novel techniques for context-aware retrieval of distributed resources. The proposed algorithms have been integrated into the *POIsmart* [4] peer-to-peer system for management and sharing of an extended form of points of interest. POIsmarts can be considered as the convergence between physical points of interest and virtual points of interest (Web resources). In addition to location information, POIsmarts can contain references to Web resources (e.g., a Web site). Moreover, in order to support sharing, POIsmarts can be tagged with metadata (keywords and categories). The POIsmart system makes use of Web services for the communication between the various entities. Our solution is based on the distributed evaluation of a scoring function that provides a relevance value between POIsmarts and the current user context. In order to obtain a comprehensive context description, our system is coupled with the *CARE* [1] middleware for context awareness. The scoring function is transformed into a *multi-feature* query (also known as *top-k* query) that is locally executed by the peer that receives the request, and propagated into the peer-to-peer network.

Generally speaking, multi-feature queries are used to calculate the degree of match of a particular object with respect to various features. Each returned tuple is associated with a score that expresses the degree of match with respect to the query. Multi-feature queries are widely adopted, e.g., for retrieving documents from multimedia databases. Recently, techniques for efficiently evaluating multi-feature queries in structured peer-to-peer systems have been proposed (e.g., [3, 10]). While these techniques can guarantee properties like correctness of results and optimality of data traffic, they rely upon a strong cooperation between peers,

*This work has been partially supported by Italian MUR (FIRB "WebMinds" project N. RBNE01WEJT_005).

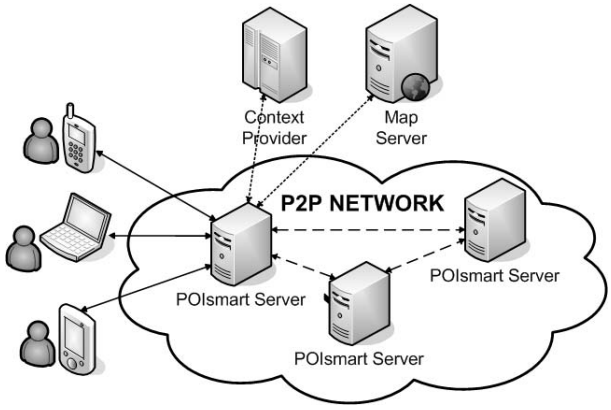

Figure 1. The POIsmart network

and possibly on the existence of super-peers in charge of maintaining indexes. On the contrary, in order to guarantee high flexibility, we have organized the POIsmart system as an unstructured peer-to-peer network. The main advantages of the algorithm for distributed multi-feature query evaluation we propose reside on the fact that: *i)* it is intended to work with any pure peer-to-peer network; *ii)* it reduces the exchange of data in the network, and *iii)* it guarantees the correctness of the result set until a certain time-to-live.

With respect to the issue of managing points of interest, we must point out that the idea of bookmarking physical locations is not new [7], and in the last years it has been exploited for different means in the fields of ubiquitous and pervasive computing. For example, architectures for managing virtual notes that allow the user to attach comments, reminders, and multimedia resources to objects identified by a physical location have been presented in [9] and [6]. A key difference with our work is that these systems are mainly intended for personal use, and do not support distributed search. Moreover, they rely on location as the solely context parameter; hence, the search facilities they provide are based only on that data. On the contrary, having coupled our system with a framework for context-awareness, our search algorithm takes into account a wide range of context data.

The rest of this paper is structured as follows: Section 2 shows an overview of the POIsmart system architecture; Section 3 presents the algorithms for distributed retrieval of POIsmarts; Section 4 illustrates the current prototype implementation; and Section 5 concludes the paper.

2 Architecture

As anticipated in the introduction, the POIsmart system is based on a peer-to-peer network of POISMART SERVERS. An overview of the POIsmart network is shown in Figure 1. Users can access the POIsmart network from a wide range of devices (called CLIENT SYSTEMS in the rest of this paper), including desktop computers, laptops, and handheld devices such as smartphones and PDAs. Each POISMART SERVER, upon receiving a user search request, retrieves from an external CONTEXT PROVIDER the context data useful for service adaptation. Moreover, an external MAP SERVER is queried by the POISMART SERVER for obtaining maps that show the position of the returned POIsmarts, as well as information for navigation support. All the entities in the POIsmart network communicate by means of Web services.

POIsmart servers: The main component of the architecture is the POISMART SERVER, which is in charge of managing POIsmarts, and of searching POIsmarts in the peer-to-peer network on the basis of the user's context and explicit search keywords. The mechanism of search in the peer-to-peer network is presented in detail in Section 3. The POISMART SERVER is also in charge of retrieving context data from an external CONTEXT PROVIDER, and of requesting maps and other information for navigation support to an external MAP SERVER. POIsmarts returned to the CLIENT SYSTEM are represented in XML according to an extension of the XML Schema for points of interest presented in [4]. Typically, a POISMART SERVER can reside on a department server, an ISP, or on a personal server. The peer-to-peer network organization also allows a CLIENT SYSTEM to connect to an arbitrary POISMART SERVER in the network and to transfer and operate on its POIsmarts through it.

Client Systems: CLIENT SYSTEMS provide an interface for the user to browse her own POIsmart hierarchy, reorganize the hierarchy, add new POIsmarts, and search shared POIsmarts in the peer-to-peer network. CLIENT SYSTEMS are device-dependent, since the graphical interface and several features need to adapt differently to desktop computers, PDAs, cellular phones and other devices. Different features are implemented depending on the device capabilities. For example, the CLIENT SYSTEM on devices that have enough memory and that are not always online work using a local copy of the hierarchy and synchronize with the POISMART SERVER upon user request. On the contrary, on other devices only part of the POIsmart hierarchy is transferred to the client, and any change is immediately communicated to the POISMART SERVER.

Context provider: The CONTEXT PROVIDER is in charge of providing the POISMART SERVER with the context information that is used for retrieving in the peer-to-peer network those POIsmarts that best suit the current situation of the user. This data includes – among other things – the user's location, interests, current activity, and preferences regarding the service adaptation. The problem of integrating context data that are naturally distributed between different context sources is particularly challenging;

the POIsmart system takes advantage of the *CARE* [1] middleware for context awareness.

Map server: Once the POISMART SERVER has retrieved the list of POIsmarts to be returned to the user, the MAP SERVER is queried in order to obtain a map that shows the current position of the user, as well as the location of the POIsmarts. The MAP SERVER is also queried for supporting the navigation of the user, in the case she decides to reach the physical location of a particular POIsmart. In this case, the MAP SERVER provides a map showing the suggested route to follow, as well as textual directions. The route is chosen considering context data; e.g., in order to decide whether to provide a pedestrian or a car route, the system takes into account the current activity of the user (walking, driving, etc.).

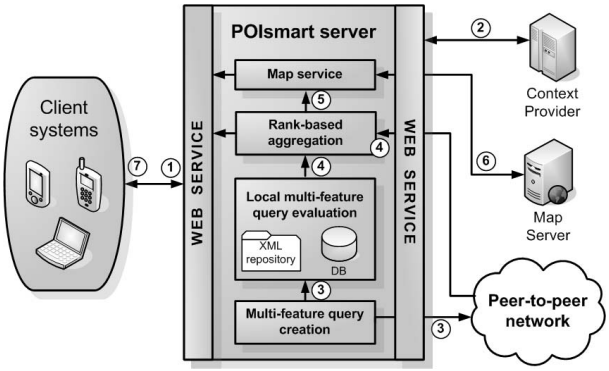

Figure 2. Distributed search in the POIsmart network

3 Context-aware POIsmart retrieval

In currently available navigation systems, the user generally can search for points of interest only by explicitly specifying a category she is interested in (e.g., hotel, gas station, etc.). The system responds to such a query with a list of points of interest belonging to the chosen category; points of interest are ordered according to their distance from the user. We believe that much more sophisticated techniques should be adopted for better adapting similar services to the demands and needs of mobile users. In particular, the quality of results can be improved by considering a much wider set of context data describing the current situation of the user. Moreover, a mechanism for automatically deriving the current preferences and needs of mobile users is of paramount importance. As a matter of fact, to input data in a handheld device is a tedious task, which often interferes with the current activity (e.g., driving).

In order to address these issues, our choice has been to couple the POIsmart service with the *CARE* CONTEXT PROVIDER, in order to obtain a exhaustive description of context, and to dynamically derive adaptation parameters at the time of request. POIsmarts are selected and ordered on the basis of a scoring function that provides a relevance value for the POIsmart with respect to the current user context and preferences. The scoring function is mapped into a multi-feature query that is locally executed by the peer that receives the request, and propagated into the peer-to-peer network. In this section we present the adopted algorithms and protocols.

Data flow upon a search request Figure 2 shows the data flow upon a search request. In step 1, the user submits a query to the POISMART SERVER; the query consists in a (possibly empty) set of keywords. In order to adapt the service, the POISMART SERVER queries the external CONTEXT PROVIDER for retrieving context data such as user's location, interests, preferences (including the number k of POIsmarts to be returned), and activity (step 2). These data are used to build a multi-feature query, that is locally executed by the POISMART SERVER, and propagated into the peer-to-peer network (step 3). The list of POIsmarts to be returned is calculated by merging the POIsmarts obtained by the peer-to-peer network with the ones obtained by the local repository; the top k POIsmarts are selected on the basis of their scores (step 4). Then (step 5), the coordinates of the selected POIsmarts are communicated to the MAP SERVICE module, which queries the external MAP SERVER for obtaining a map showing the location of the POIsmarts as well as the current location of the user (step 6). Finally (step 7), the list of POIsmarts and the map are returned to the CLIENT SYSTEM.

Scoring function As anticipated before, POIsmarts are selected and ordered on the basis of a scoring function $S(p, u)$ that calculates a relevance value for the POIsmart with respect to the current context and preferences of the user. The score depends not only on location and on a (possibly empty) set of keywords, but also on a wide set of context data, including user's interests, current activity, time of day, device capabilities, network conditions. The scoring function is formally defined in Figure 3.

The global scoring function $S(p, u)$ depends on the relative scoring functions of location, keywords, and interests. The relative weights of these functions $(\alpha_1, \alpha_2, \alpha_3)$ are dynamically set by the CONTEXT PROVIDER on the basis of context. For instance, a user could prefer to give more importance to POIsmarts in her proximity when she is walking, while giving more importance to POIsmarts that fit her interests when she is moving by car. In the former case, the relative weight of the location scoring function will be higher then the one of the interests scoring function; it will

Location score: $S_L(p,u) = \max\{(1 - \frac{d_{p,u}}{r_u}), 0\}$

Keywords score: $S_K(p,u) = \begin{cases} 1 \text{ if } K_u = \{\} \\ \\ \prod_{i=s}^{t} w_{p,k_i} \text{ elsewhere} \end{cases}$

Interests score:

$w_{p,i_j,u} = \begin{cases} w_{i_j,u} \text{ if } i_j \in C_p \\ \\ 0 \text{ elsewhere} \end{cases}$

$S_I(p,u) = \begin{cases} \frac{\sum_{i \in I} w_{p,i,u}}{\sum_{i \in I} w_{i,u}} \text{ if } I \neq \{\} \\ \\ 1 \text{ elsewhere} \end{cases}$

Global score:

$S(p,u) = \begin{cases} \alpha_1 S_L(p,u) \cdot \alpha_2 S_K(p,u) + \alpha_3 S_I(p,u) \\ \qquad \text{if } S_L(p,u) > 0 \text{ and } S_K(p,u) > 0 \\ \\ 0 \text{ elsewhere} \end{cases}$

Symbols:

(x_u, y_u): coordinates of the user u

(x_p, y_p): coordinates of the POIsmart p

r_u: search radius

$K_u = \{k_s, \ldots, k_t\}$: search keywords

$0 \leq w_{p,k_i} \leq 1$: relevance of p with respect to k_i

$I = \{i_1, \ldots, i_m\}$: set of interests

$0 \leq w_{i_j,u} \leq 1$: weight of interest i_j according to user u

C: set of POIsmart categories

C_p: set of categories the POIsmart p belongs to

$d_{p,u}$: distance between POIsmart p and user u

$w_{p,i_j,u}$: weight of the interest i_j with respect to POIsmart p and user u

$\alpha_1, \alpha_2, \alpha_3$ $(0 \leq \alpha_i \leq 1)$: relative weights of location score, keywords score, and interests score, respectively

Figure 3. Scoring function

be lower in the latter case.

Besides the locations of user and POIsmart, the location scoring function $S_L(p,u)$ depends on a *search radius* r_u. The radius r_u is determined by the CONTEXT PROVIDER analyzing context data such as the current activity of the user. For instance, the search radius for a user that is walking is smaller than the one for a user that is moving by car or by public transportation. The codomain of $S_L(p,u)$ is $[0,1]$; the score decreases linearly with the distance between the POIsmart and the user. POIsmarts not falling into the search radius are given score 0.

In order to improve the search results, POIsmarts can be associated with keywords and categories. We call $w_{p,k}$ the relevance of a POIsmart p with respect to a keyword k, $0 \leq w_{p,k} \leq 1$. If p is not associated to k, then $w_{p,k} = 0$. The keywords scoring function $S_K(p,u)$ is calculated as the product of $w_{p,k}$ for each search keyword k specified by the user. In the case the user does not specify keywords, the function $S_K(p,u)$ does not contribute to the global score. Note that POIsmarts not associated with any specified keywords are given weight 0.

Moreover, POIsmarts categories are matched with the user interests for augmenting the score of those POIsmarts that fit the demands of the user. Interests can be explicitly declared, or derived by the CONTEXT PROVIDER on the basis of context. For instance, if a user is traveling outside her hometown at noon, the CONTEXT PROVIDER can derive an interest for restaurants. A relevance value $w_{i,u}$ $(0 \leq w_{i,u} \leq 1)$ specifies the relevance of the interest i with respect to the user u. The interests scoring function $S_I(p,u)$ is calculated as the (normalized) sum of the $w_{i_j,u}$ relevance values of the user interests that match the POIsmart categories.

The global scoring function $S(p,u)$ is calculated as the product of the scores of location and keywords (weighted by α_1 and α_2, respectively), plus the score (weighted by α_3) of interests. POIsmarts either falling outside the search radius, or not associated with any explicit keywords, are given weight 0.

Optimized multi-feature query execution in the peer-to-peer network In order to guarantee high flexibility, the POIsmart peer-to-peer system is organized as an unstructured overlay network. Unstructured peer-to-peer systems like Gnutella [11]) do not have control about the precise topology of the overlay network. This means that the network is not aware of nodes joins and departures, and no assumption can be made regarding the type of content that each node is storing. For these reasons, queries are generally performed by flooding the network until a given time-to-live. The success of file sharing systems has shown the limitations of these protocols with regard to the problems of scalability and reliability. In particular, the routing protocol in a fully distributed architecture as Gnutella leads to inefficient use of bandwidth, due to the replication of messages. Furthermore, these protocols do not guarantee that queries are correctly answered.

For overcoming part of these problems, we have modified the basic Gnutella protocol in order to improve message routing. The mechanism for discovering other peers in the network has been presented in [4]. In this section we present a novel protocol for executing multi-feature queries in a fully decentralized peer-to-peer network. The algorithm pseudo-code is shown in Algorithm 1. The main advantages of this algorithm are *(a)* a mechanism for reducing the replication of the same search message to the same peer, and *(b)* an algorithm for reducing the exchange of data in the network, which guarantees the delivery of the correct result set

Algorithm 1 Peer-to-peer search algorithm

Procedure *search-POIsmarts(user u, keywords K)*
1) context = CONTEXT PROVIDER.get-context-data(u);
2) q = create-query(context, K);
3) S = {0};
4) R = {};
5) (S, P) = search(q, S, R, x);
Return(P).

Procedure *search(query q, scores S, peer-set R, int TTL)*
1) *search top k POIsmarts having scores* ($s'_1, \ldots, s'_k$);
2) *determine max j such that* $s'_j > s_k, \ldots, s'_1 > s_{k-j+1}$;
3) P = *POIsmarts corresponding to scores* ($s'_1, \ldots, s'_j$);
4) S = update(S, ($s'_1, \ldots, s'_j$));
5) **if** TTL > 0 **then**
6) (S', P') = forward-query(q, S, R, TTL-1);
7) S = update(S, S');
8) P = update(P, P');
9) **end if**
Return(S, P).

Procedure *forward-query(query q, scores S, peer-set R, int TTL)*
1) $R' = \{\text{known-peers}\} \setminus R$;
2) **for all** $r \in R'$ **do**
3) (S', P') = r.search(q, S, {{known-peers} $\cup$ R}, TTL);
4) S = update(S, S');
5) P = update(P, P');
6) **end for**
Return(S, P).

in response to search queries, given a certain time-to-live.

When a specific peer p_1 receives a search request from a CLIENT SYSTEM (procedure *search-POIsmarts* in Algorithm 1), it queries the CONTEXT PROVIDER for retrieving context data and dynamic preferences (including the number k of POIsmarts to be returned) in order to build the multi-feature query q. The query is locally executed (procedure *search* in Algorithm 1), thus obtaining a ranked list of the top k POIsmarts. Then, p_1 forwards the query to all of its known peers, together with the scores S of the top k POIsmarts, the set R of known peers, and the time-to-live (procedure *forward-query* in Algorithm 1).

Since the procedure *forward-query* does not guarantee that the same search request is not received multiple times, the query contains an ID, which is checked by any peer in order to avoid the re-evaluation of the same query. When a known peer p_2 receives the search request from p_1, it locally evaluates the query q, selecting only the top j POIsmarts ($j \leq k$) whose scores are higher then the j lowest scores of S. In this way, POIsmarts that are not candidate to enter the top k list are recognized at each peer, and not sent to the requester. This operation allows the network to avoid the communication of unnecessary data, still preserving the correctness of the distributed multi-feature query evaluation.

The peer p_2 updates S with the new scores, and forwards the query to its known peers that are not in the set R of peers that have already been contacted before, or that are known to the requester. Before forwarding the query, the set R is updated by p_2 with the set of its known peers. The same mechanism of query forwarding is repeated recursively in the peer-to-peer network until the time-to-live expires. Each peer replies to the search request with a list of POIsmarts together with their scores. Once the peer p_1 receives the replies by all the contacted peers (or until a timeout), it updates the POIsmarts list, and sends it to the CLIENT SYSTEM.

4 The current system prototype

We have developed a prototype of the POIsmart system that implements the features explained in the previous sections. Our architecture adopts Web services for client/server and server/server communication. Server side, Web services simplify the communication of software modules implemented using different software platforms (e.g., the POISMART SERVER and the MAP SERVER). With regard to the communication between the CLIENT SYSTEM and the POISMART SERVER, the choice of Web services is mainly motivated by the need for device independence: users should be able to access their POIsmarts management system from many different devices. Many of these devices need an ad-hoc client software for the standalone client application. We currently consider Web services to be the best and simplest solution for the interaction between applications running on different environments.

The POISMART SERVER has been developed in Java, and implements the algorithms for POIsmarts scoring and distributed search presented in Section 3. Context data retrieved from the CONTEXT PROVIDER is represented in the extended CC/PP [8] language illustrated in [2]. This formalism allows to represent not only raw context data (e.g., device capabilities and network conditions), but also complex context data such as the user activity. Since a generally accepted standard language for representing multi-feature queries is not yet available, we express multi-feature queries in an ad-hoc language. Queries are executed on top of the MySql DBMS by a dedicated software module. The POISMART SERVER also includes various facilities for managing and searching private POIsmarts, as illustrated in [4].

Users can access the POIsmart system from a wide range of devices. At the time of writing, we have developed CLIENT SYSTEMS for desktop and laptop computers, PDAs, and smartphones, for the .Net Compact Framework and for Java-enabled devices (see [4] for further details).

As anticipated before, context data are provided to the

(a) Locations of user and POIsmarts (b) Route to a point of interest

Figure 4. Map services

POISMART SERVER by the CONTEXT PROVIDER module of the *CARE* middleware for context-awareness. The middleware adopts sophisticated techniques – based on logic programming and ontologies – for aggregating distributed context data, reasoning with them, and resolving possible conflicts. A description of the technical solutions adopted in *CARE* is outside the scope of this paper; details can be found in [5] and [2].

In the current implementation, the MAP SERVER is a .Net Web service based on Microsoft MapPoint. At each search request, the user receives a map showing her current position, and the position of the top k POIsmarts. The positions are represented by icons (see Figure 4(a)). In the case the user decides to reach one particular point of interest, the MAP SERVER is queried in order to generate a map showing the route to follow (see Figure 4(b)), as well as textual directions. Size and resolution of maps are chosen on the basis of context data such as available bandwidth and device capabilities.

5 Conclusions and future work

In this paper we have presented novel algorithms for context-aware retrieval of distributed resources. The proposed algorithms have been integrated into the POIsmart peer-to-peer system for management and sharing of an extended form of points of interest.

This work can be extended in various directions. A promising line of research consists in the investigation of techniques for improving the efficiency of the retrieval algorithms, by means of the adoption of Semantic Web technologies for annotating peers with semantic information. This feature could be exploited by the retrieval algorithm for identifying those peers that are more likely to own the desired resources. Interesting extensions of the POIsmart system consist in the support of proactive services (like proximity services), and in the integration of CLIENT SYSTEMS with client-side navigation software.

References

[1] A. Agostini, C. Bettini, N. Cesa-Bianchi, D. Maggiorini, D. Riboni, M. Ruberl, C. Sala, and D. Vitali. Towards Highly Adaptive Services for Mobile Computing. In *Proc. of IFIP TC8 Work. Conf. on Mobile Information Systems (MOBIS)*, pages 121–134. Springer, 2004.

[2] A. Agostini, C. Bettini, and D. Riboni. Loosely Coupling Ontological Reasoning with an Efficient Middleware for Context-awareness. In *Proc. of the 2nd Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, pages 175–182. IEEE Computer Society, 2005.

[3] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks. In *Proc. of the 21st Int. Conf. on Data Engineering (ICDE 2005)*, pages 174–185. IEEE Computer Society, 2005.

[4] C. Bettini, N. Cesa-Bianchi, and D. Riboni. A Distributed Architecture for Management and Retrieval of Extended Points of Interest. In *Proc. of the 1st Int. Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet*, pages 266–272. IEEE Computer Society, 2005.

[5] C. Bettini and D. Riboni. Profile Aggregation and Policy Evaluation for Adaptive Internet Services. In *Proc. of The 1st Annual Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous)*, pages 290–298. IEEE Computer Society, 2004.

[6] G. Borriello, W. Brunette, M. Hall, C. Hartung, and C. Tangney. Reminding About Tagged Objects Using Passive RFIDs. In *Ubicomp*, pages 36–53. Springer, 2004.

[7] P. J. Brown. The Electronic Post-it Note: a Metaphor for Mobile Computing Applications. In *IEE Colloquium on Mobile Computing and its Applications*, 1995.

[8] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recomm., W3C, January 2004. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/.

[9] A. Leonhardi, U. Kubach, K. Rothermel, and A. Fritz. Virtual Information Towers - A Metaphor for Intuitive, Location-Aware Information Access in a Mobile Environment. In *3rd Int. Symp. on Wearable Computers (ISWC 1999)*, pages 15–20. IEEE Computer Society, 1999.

[10] S. Michel, P. Triantafillou, and G. Weikum. MINERVA$_\infty$: A Scalable Efficient Peer-to-Peer Search Engine. In *Proc. of Middleware 2005, ACM/IFIP/USENIX, 6th Int. Middleware Conf.*, volume 3790 of *LNCS*, pages 60–81. Springer, 2005.

[11] Sourceforge. The Annotated Gnutella Protocol Specification v0.4. Tech. Rep., Gnutella Protocol Development.