BlueS: Query Processing in Ubiquitous Environments

Service Discovery vs. Information Exchange

Andre Peters
University of Rostock
Institute of computer science
Chair of databases and information systems
ap@informatik.uni-rostock.de
Advisor: Prof. Dr. Andreas Heuer

ABSTRACT
Assisting the user is the main goal of ubiquitous environments supported by dynamic ensembles: local agglomerations of smart appliances, whose composition is prone to frequent, unforeseeable, and substantial changes. Globally coherent behavior with respect to the users need can emerge from interaction of individual appliances. Therefore, its necessary to develop intelligent efficient indexing strategies for knowledge addressing and exchange in dynamic ensembles. Especially in scenarios like MuSAMA [13], a project dealing with Smart-Rooms, data access, query evaluation on mobile devices and retrieval techniques face problems such as resource constraints, power limitations and time delays in information exchange. At this point a solution to manage such problems needs to be devised. This paper introduces our work to find such a solution. Therefore, a framework BlueS is presented, which supports service discovery and data exchange in spontaneous linked environments, e.g. Bluetooth networks.

Categories and Subject Descriptors
C.2 [Computer-Communication Networks]: Distributed Systems; H.2 [Database Management]: Languages; H.4 [Information Systems Application]: Miscellaneous

Keywords
Dynamic Taxonomies, Distributed Information Retrieval, Service Discovery

1. INTRODUCTION
Work in MuSAMA (Multimodal Smart Appliance Ensembles for Mobile Application, [13]) supported by the DFG (German Research Foundation) concentrates on ubiquitous machine intelligence provided by dynamic ensembles. One part of it is query processing in spontaneously linked environments like MANETs.

Example: At [13] we focus on smart meeting rooms. Currently all devices like lights, blinds, projectors, light sensors or data sources are connected via wire. Services like lighting intensity and blinds up/down have to register on a central service bus. Now we research on mobile P2P architectures for dynamic behavior of individual appliances. Therefore, it is necessary to describe and to offer services in an ad hoc like manner. One possible approach is to add an extra mobile device (linutop: http://www.linutop.com) to each of the devices listed above. The extra device is able to manage the device it is docked to and its data. Using wireless technologies like Bluetooth we have to manage local services and data in background by indexing and index exchange.

Obviously we also have to support knowledge exchange in ubiquitous environments. The goal of smart rooms is to provide relevant services and information to the meeting participants (e.g., speakers, audiences). If we think about information (structured - sql, semistructured - documents, images, ...) this data can be provided by the devices or the audience itself. So we have to concentrate on data access in mobile ad hoc networks (MANETs) too.

MANETs are collections of mobile devices connected without any central point of organization. Usability of these devices from an application’s point of view is hardly discussed. Like [7], we believe mobile ad hoc networks are only
useful [...] if the members cooperate by efficiently searching and using services and resources offered by other devices.” Complex problems can be solved by aggregating services to an ensemble-like behavior. So searching for services that can help to solve such problems is one very important step. “The proliferation of these services along with the ever increasing use of diverse computational devices such as mobile phones or PDA’s entails the potential to provide users with ubiquitous access to all kinds of relevant and up to date information.” ([5]). Following the idea of data sources accessible via services, we have to concentrate on indexing support for service discovery, data replication/synchronization and data management in general now. Our objective is to develop mechanisms that will enable efficient adaptable processing of queries in ubiquitous environments. We think about queries as follows:

**Definition 1. Boolean Queries in IR**

In the theory of Information Retrieval a boolean query \( q \) is a conjunctive query in the form \( q_1 \land q_2 \land \ldots \land q_n \) with \( q_i \):

- \( q_i = t \): The subquery is a term representing a word.
- \( q_i = \neg t \): The subquery is a negative term.
- \( q_i = t_1 (t_2 \lor t_3 ) \land \ldots \land t_k (t_l \land t_m) \): The subquery is a disjunction of terms or negative terms.

**Definition 2. PSJ Queries in Databases**

In the theory of databases a PSJ query \( Q \) is a relational algebra expression containing only the operations project, select and join. Cartesian product is seen as a special case of a join. Any valid PSJ query \( Q \) can be transformed into a standard form consisting of a Cartesian product, followed by a selection, followed by a projection:

- \( Q = \pi_d \sigma_{C_0} (R_1 \times R_2 \times \ldots \times R_l) \) with
- \( A = \{ A_1, A_2, \ldots, A_i \} \) the attribute set
- \( R = \{ R_1, R_2, \ldots, R_j \} \) the relation set
- \( C \) the selection conditions

This paper introduces the ideas of an integrated approach to design and implement an information system, which addresses the problems mentioned above. On the one hand we have to look at humans querying devices (using their own mobile device) without any knowledge about formal query languages and on the other hand we also have to concentrate on automatic information exchange, e.g. exchange of compact data index structures as represented in section 2, without any human intervention.

In section 2 the state of the art is outlined. The major challenges for the approach to be coped with are described in Section 3.1. Section 3.2 introduces the actual approach including the information system architecture. Section 4 presents some first results, e.g. on indexing support for service discovery in Bluetooth environments. Finally, Section 5 gives a conclusion and identifies the problems which have to be taken care of within the next steps of ongoing work.

2. STATE OF THE ART

In general, query processing systems are tailored for specific types of queries on restricted data sets according to a particular semantic. However, answering special queries may require certain tasks. To illustrate this, consider a scenario of information service applications in Smart Labs. Among other features all available devices should be able to inform about their features and their stored data in real time. So standardized interfaces are needed to support such information application. Services are able to fulfill this task. As presented in [19] a service interface typically is described by \( \{ f, X, Y \} : f \) the name or identifier of the service, \( X \) the input parameters and \( Y \) the output parameters. In [12] additional parameters like \( \text{preconditions} \) and \( \text{postconditions} \) are given. Using these information, heterogeneous software and hardware resources may be discovered and integrated transparently. An illustration of service data providers can be found in [5].

Discovering services often takes a lot of time and produces a lot of network traffic. The Service-Discovery-Protocol (SDP), proposed by the Bluetooth Special Interest Group (SIG) [18], provides a simple discovery mechanism based on requesting service classes or service attributes successively from all devices in range. The Bluetooth specification declares a service discovery profile which supports Bluetooth enabled devices to search for services specified by a 128 (128B), 32 (32B) or 16 bits (16B) UUID. The standard size is 128 bits, but other aliases can be extended like:

- \( 128B = 16B \cdot 2^{96} + \text{Bluetooth Base UUID} \)
- \( 128B = 32B \cdot 2^{96} + \text{Bluetooth Base UUID} \)

At first available devices have to be found by inquiry. After that a P2P L2CAP Bluetooth link can be established. Finally, the SDP allows service discovery using a request/response scheme over the L2CAP transport protocol. However, resulting network structures comprise interconnected devices with increased energy consumption for connection maintenance. To overcome this increased energy consumption, a low layer Bluetooth improvement for service discovery can be implemented by service indexing as described in [14]. After all we are able to pre-filter services on inquiry.

The user of Smart Labs doesn’t know anything about formal query languages. He just want to know, what he can use and how it can be used. As (semi-) domain specific environments can be modeled with graph representations like ontologies or taxonomies, e.g. gen and gen product attributes used in bioinformatics [http://www.geneontology.org/], this semantic annotation can be used for query declaration. Current and recent database research deals with querying and manipulation of graphs which in the most cases boils down to the development of graph query languages [16, 17, 3]. In [3] a graph data model has been proposed, enhanced with features of semantic data models, as alternative to the relational model. [3] defines the query language GQL for its graph data model. [17] presents an object-oriented graph data model based on GQGL. Currently we focus on dynamic taxonomies for service browsing introduced by Sacco [16]. He describes his query process as follows:

Let \( T = (C, \leq, I, Q) \) be a taxonomy with
Every taxonomy also contains two special terms $\bot$ (the bottom-term) and $\top$ (the top-term) with:

- $\forall c \in C : c \leq \top$
- $\forall c \in C \setminus \{\top, \bot\} : \bot \lneq c$
- $\bot \leq \bot$ and $\bot \leq \top$
- and $\forall I : I(\bot) = \emptyset$

The semantic of $S(q)$ on a query $q$ is defined as:

$S(c) = \bigcup\{I(c') : c' \leq c\}$

$S(q \land q') = S(q) \cap S(q')$

$S(q \lor q') = S(q) \cup S(q')$

$S(\neg q) = Obj \setminus S(\neg q)$

The index $O_I$ of any object in relation to its interpretation $I$ can be formulated with:

$O_I(o) = \bigwedge\{c \in C : o \in I(c)\}$

If we now map services to taxonomies we are able to select interesting information by browsing. The initial user focus $F$ is the universe of all objects. The user can select one or more concepts of $T$ and zoom over it. The zoom operation changes the current state in two ways:

- All selected concepts $C_i$ are used to change $F$: $F = F \cap I(C_i)$.
- The taxonomy getting cut in order to represent just the new focus.

"By combining these [...], ideas, we are able to provide a visual framework in which the user can select and combine appropriate concepts in the taxonomic tree, retrieve the corresponding set of documents, display the reduced taxonomy in order to obtain related concepts, and iterate until the result set is sufficiently small for manual inspection." ([16])

### 3. BLUES: SERVICE DISCOVERY VS. INFORMATION EXCHANGE

In this section we want to introduce our solution BlueS, a framework for Service Discovery and Information Exchange in spontaneous linked mobile environments.

#### 3.1 Challenges

With respect to an information system the MuSAMA project raises a number of challenges to be met in an adequate approach.

#### Efficient Retrieval

The fast amounts of data pertaining various domains appear to be converging towards a service-oriented environment. Data are available as services, meaning that they are accessible via standardized protocols. Discovering services often takes a lot of time and produces a lot of network traffic. The Service-Discovery-Protocol provides a simple discovery mechanism. The native Bluetooth SDP is inefficient due to the following reason: Service discovery is based on connection establishment between two or more nodes. In ad hoc environments the network topology changes continuously resulting in high energy consumption and a significant time delay due to connection establishments. To overcome this situation in Bluetooth environments with high node concentrations we try to improve the native SDP by service meta data indexing as represented in section 2.

#### Query Language

Based on the fact that data of (semi-) domain specific environments can be modeled with taxonomies we have to concentrate on query languages for semantic representations. With respect to the MuSAMA project the conceptual design of means for users, who have no notion of any kind of formal query languages, have to be taken into consideration. The dynamic taxonomy model is an excellent model to address these problems and propose both a visual paradigm for accessing complex information bases and an effective, complete and easy to use model. Intelligent user-centric access to complex information must be considered holistically, both from the data modeling and from the user interaction side.

#### Data Storage

Using mobile environments we have to concentrate on support for mobile databases. Since we are using graph representations for querying as described in section 2, a graph based storage approach is inevitable. Nevertheless we also have to store index informations and replica data. Therefore, the first main challenge is to devise a sophisticated combination of both semi-structured and graph techniques to create an efficient information system, in terms of storage and retrieval, enhanced by domain-specific features that meets the projects requirements.

#### 3.2 Information System Architecture

This section describes the ideas for an information system architecture, illustrated in Figure 2. This architecture is aligned to satisfy the needs of the three main goals nominated in Section 3.1. Currently there are four main layers:

1. Communication layer
2. Selection layer
3. Registration layer
4. Data layer
The requirements pointed out in section 3.1 are taken care of at layer 1 and 4. Layers 2 and 3 describe the basic application data model (3) and operations (2) which 1 and 4 are based upon. Therefore, they are represented first.

**Registration Layer**
Since data are modeled as services local- and global services have to register at the registration layer. Services are mapped to available semantic data representation like taxonomies via categorization, clustering algorithms or semi-automatic approaches. On the other hand, indexes are created, representing service meta data as service descriptions and service conditions, that can be used in the selection layer for information retrieval.

**Selection Layer**
The selection layer contains the dynamic part of the application data model, including basic operations for the data and graph part of the static model described above. Furthermore, a number of domain-specific functions belong to the operation layer as well. A minimal subset of all operations forms the query algebra. To support difference indexing and ranking strategies we provide plugin-based interfaces.

- **Semi-structured operations:**
  Since semantic models like taxonomies normally are represented as XML we have to look at basic operations for the semi-structured parts of the static model, such as retrieval and update operations as introduced in [6, 2]. There are diverse approaches for efficient XPATH support, depending on the storage model [4, 9, 11]. The basic retrieval operations include selection, extraction and restructuring. Selection identifies nodes from a sequence of XML fragments on the basis of their structure or content. In XQuery this basic operation is utilized by the WHERE clause in order to
determine, which nodes, currently iterated by the respective FLWOR expression, have to be selected. The extraction operation is applied to a sequence of nodes in order to retrieve sub-elements within the XML fragments. In XQuery this operation is employed in path expressions. The restructuring operation includes the creation of new XML nodes. These can be used to combine other nodes. The return clause in XQuery is the most common expression, which makes use of this operation.

**Graph operations:**
The operation layer also includes the basic operations and algorithms for graphs as newGraph, loadGraph or delete, select and matchSubGraph. Several functions can be applied to a graph to determine if it meets particular properties, regarding e.g. cycles. There are a number of modification operations to manipulate the properties of the graph’s constituents. This includes modification of weights, labels, and edge bonds. The modification operations are most useful in collaboration with the select functions. We also concentrate on graph ranking mechanism like [10] which can be used for information retrieval. Here, we look at shortest path, level of nodes and level of parents to determine ranking values between two nodes of a graph. This scenario is represented in section 4.

**Communication Layer**
The communication layer addresses the second main challenge outlined. Currently we concentrate on dynamic taxonomies for browsing concepts of multidimensional classifications: items (services) in the extension are classified under several concepts. It serves as an interface between the internal operations and the user respectively, who has no previous knowledge of any formal query language. Now we research on SQL-like support using dynamic taxonomies, e.g. using tableau-queries. If we are able to create a taxonomy view of data tables (R) automatically, data tuples can be mapped on it. By selecting nodes we are able to select our free variables. Also, we are able to restrict our nodes (n) with constraints like \(n = x; x \in \text{dom}(R)\). For a full description of tableau-queries see [1]. Finally, the communication layer offers as interface for service discovery in Bluetooth environments. Here, we are able to search for services by UUID’s as described in [18].

**Data Layer**
The preferred storage architecture is hybrid. Data-centric, plain data are stored relationally, together with their metadata, whenever possible. Thus, the information system can benefit from the advantages of well proven relational techniques. Currently we research on storage of semi-structured content using mobile databases. Obviously there is some support for XML data structures, e.g. as respective data types, to provide efficient XQuery evaluation. The use of indexes derived from the domain-specific structural part of the application data model will expedite access and retrieval for those operations based upon the domain-specific data structures. Using mobile databases we have to concentrate on resource management at all. Policies can be one solution to do. The domain knowledge represented in the domain specific part of the data representation layer determines which matrices and graph path indexes are necessary, have to be kept in a linking cache ubiquitously and which can be located out of core.

4. **INITIAL WORK**
This section represents some result we already worked on.

4.1 **Time Efficient Service Discovery using Bluetooth**
In order to retrieve services in an efficient way we had a look at the Bluetooth service discovery procedure as presented in section 2 [14]. An inquiry process is used to find neighbor-devices. It is initiated by broadcasting an ID packet. All receivers respond with an FHS packet (Figure 3) containing, among other values, the senders Bluetooth address and the 24 bits class of device value.

![Figure 3: FHS packet](image)

Although the CoD field is defined, it is never used in real applications nor in Bluetooth SDP. Using the free Linux Bluetooth stack BlueZ [15] and its tools the CoD field can be changed for own purposes. Using this approach local services can be indexed by their mapping representation in the CoD field. On device inquiry each remote device can select devices which probably advertise services the remote device is looking for. In this case, SDP is only necessary for the selected devices and not separately on each node. In order to verify the optimization achieved by our approach we measured the performance of the native SDP versus our approach in an experiment. Therefore, we used ten USB Bluetooth modules connected to Linux workstations (server/client). We measured the time for device inquiry \(T_{\text{inq}}\) and service discovery on selected devices \(T_{\text{search}}\) for a full service discovery process (device connecting, SDP request transmission, SDP server request processing, SDP response transmission). In a two day test we observed that the device inquiry time \(T_{\text{inq}}\) is always about 7-12 s. It varies around about 5 s due to moving Bluetooth peers during our test (each test lasted about one hour). The results are presented at [14]. The native SDP requires successive connection establishments to all devices within the environment. Thus, the full service discovery time for \(s\) devices will be the sum of the inquiry time \(T_{\text{inq}}\) and native SDP time \(T_{\text{search}}\):

\[
T = T_{\text{inq}} + \sum_{i=1}^{s-1} T_{\text{search}}
\]

The gain of time \(T_G\) by the CoD based discovery model compared with the native SDP will be:

\[
T_G = T_{\text{inq}} + \sum_{i=1}^{s-1} T_{\text{search}} - (T_{\text{inq}} + \sum_{i=1}^{a} T_{\text{search}})
\]

\[= \sum_{i=1}^{s-1-a} T_{\text{search}}\]

As we have seen, with this approach we can get a gain of time of about 60%.
4.2 Using Dynamic Taxonomies on Mobile Devices

We concentrate on dynamic taxonomies as query language for mobile devices. Therefore, we developed a framework called BlueS_mobile, which supports data loading (represented by services), data mapping and zooming in an adaptive way. Here we are able to implement and to integrate different mapping and zooming operations at run time.

Normally information services are mapped to taxonomies by categorization or conceptual clustering. Currently we support three kinds of mapping functions:

• manual mapping:
  Services are mapped manual by the user who has any knowledge of the given domain. Therefore the user can select a given service and edit it’s mapping representation.

• mapping by edit distance:
  Normally concepts of tree taxonomies are represented with concept descriptors like names. We are able to map services to the given concepts using their service descriptions and service names. Therefore the similarity of concept names and service names (service descriptions) is calculated by the Levenshtein distance [8]. This mapping is very fuzzy represented by a fuzzy weight $\in [0 \ldots 1]$.

• mapping by service distance:
  In case of already mapped services we can use this assignment as a training set for new mapping operations, so a new service get linked to concepts where the distance between the mapped service and the new service is minimal. Therefore it’s necessary to define such distance function between two services. Currently we are using the edit distance of service names and service descriptions as the service distance function.

Figures 4 and 5 show a brief overview of our mobile solution.

4.3 Service Ranking

The cutting algorithm used in dynamic taxonomies is very strict. Only objects indexed by selected nodes are represented as the result. Nevertheless, the mapping algorithms are not as strict. Obviously this mapping is fuzzy, e.g. using approaches like Levenshtein distance [8] to map a service (represented by its meta data description) to concepts of $T$ represented by the concept label leads to fuzzy mapping values, which can be stored as mapping weights. On cutting these values have to be considered. On the other hand, the basic cutting approach represents all result unsorted. There is given no ranking value. Fuzzy weights can be used to determine such ranking values. So we research on mechanisms for cutting, which are not as strict, e.g. looking at nodes near our selection. To do so, similarity values on graphs can be used as represented in [10]. Li compares two nodes (or concepts of $T$) $C_x, C_y$:

$$\text{sim}(C_x, C_y) = e^{-\alpha \phi(C_x, C_y)} \cdot e^{\beta \delta(C_x, C_y)} - e^{-\beta \delta(C_x, C_y)} + e^{-\beta \delta(C_x, C_y)}$$

with:

• $\phi(C_x, C_y)$: shortest path between $C_x$ and $C_y$

• $\delta(C_x, C_y)$: level of a node $C_i$, with $C_i$ is parent of $C_x$ and $C_y$. If there is a node $C_j$, which is also a parent of $C_x$ and $C_y$, then: $\text{level}(C_i) \geq \text{level}(C_j)$.

• $\alpha$ and $\beta$ are used as scale factors to represent human understanding of this similarity algorithm.
Obviously, this mechanism is such complex, we are using our own similarity value (transparent graph) with an scale factor $\alpha$ for human understanding.

$$\text{sim}(c_x, c_y) = \alpha \cdot \frac{1}{\phi(c_x, c_y)} \cdot \delta(c_x, c_y) \cdot \frac{\max_l}{\max_l, \alpha = 2.0}$$

Figure 6 compares this two solutions.

![Comparison of similarity mechanisms based on graphs](image)

As you can see $\phi$ and $\delta$ are also used, but there is a new value $\max_l$. By using the ratio $\frac{\max_l}{\max_l}$ the level of nodes getting more important. As on a node path of $\phi \geq 6$ the level of the parent node really influences the ranking value. Additionally, this ratio can be interpreted in different ways. We are using $\max_l$ representing the maximum level of $T$. If we want to interpret leave nodes in a special way, $\max_l$ can be interpreted as the maximum level of any leave of $C_x$ or $C_y$:

$$\max_l = \max(\text{depth}(C_x), \text{depth}(C_y))$$

$$\text{depth}(C) = \max(\text{level}(C'), C' \leq C)$$

5. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a framework for query processing on mobile devices in ubiquitous environments involving classic and service data sources. Based on challenges of efficient data and service retrieval, query languages for mobile devices and data storage techniques for (semi-) structured data we presented our initial approach. XML and related techniques as well as recent research have originated diverse instruments to address particular problems within the integrated design approach of an information system for the MuSAMA project.

We are currently working on a definition of a query approach for sql-stored data using dynamic taxonomy techniques. Future work includes the development of adaptive algorithms for each layer represented in our framework Blue$\xi$.

Further research concentrates on evaluation mechanisms to deal with the flexibility of query processing in ad hoc scenarios, e.g. using semantic caching approaches for data indexing on mobile devices.

6. ACKNOWLEDGEMENTS

The work of the author is supported by a grant of the German National Research Foundation (DFG), GRK 1424.

7. REFERENCES


graph query language and its query processing. In
ICDE '99: Proceedings of the 15th International
Conference on Data Engineering, page 572,

[18] B. SIG. In specification of the bluetooth system. In

[19] U. Srivastava, K. Munagala, J. Widom, and
R. Motwani. Query optimization over web services. In
VLDB '06: Proceedings of the 32nd international
conference on Very large data bases, pages 355–366.