

Semantic Information Retrieval in the COMPASS Location System

Frank Kargl, Günter Dannhäuser, Stefan Schlott, and Jürgen Nagler-Ihle

Ulm University, Media Informatics, Ulm, Germany

In our previous paper [1], we have presented the COMPASS location system that merges the results of different location sensors like GPS or WLAN signal positioning and thus provides a more accurate and available positioning solution.

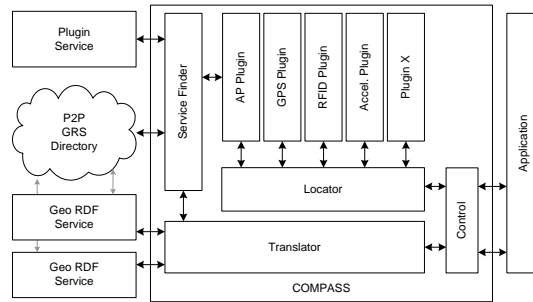


Fig. 1. COMPASS architecture

Figure 1 shows the overall architecture of COMPASS. The output of each sensor is a so called Probability Distribution Function (PDF) which is based on ideas presented in [2,3]. In our architecture, it is the job of the *Locator* to manage all the different plugins and to collect and combine the PDFs. Using the Control API, applications can retrieve the combined PDF and determine the position with the highest probability.

For many applications, a position given as geographic coordinates is not very useful. Instead, a semantic form of position representation is needed, e.g. in terms of city, building, room, etc. Given a specific geographic position discovered by the *Locator*, the *Translator* component is responsible for collecting this kind of information from the network.

In order to allow for a flexible and extensible way of describing locations and services semantically, we employ the *Resource Description Framework (RDF)* [4]. *Geo RDF Services (GRS)*, as depicted in 1, provide RDF facts about arbitrary geographical areas.

The area where a RDF fact is valid is given by geometric primitives like rectangles, circles or polygons¹. This area is stored as a *validIn* property with

¹ for simplicity reasons, our prototype implementation only supports rectangular areas, where the corners are given in WGS 84 coordinates

the RDF facts and is called a *fact validity area (FVA)*. We call the minimum bounding box (MBB) of all fact validity areas of a Geo RDF Service its *service coverage area (SCA)*, i.e. the area where the service might be able to provide facts for locations lying inside. Geo RDF Services are implemented as Web Service, providing a simple SOAP interface, where a client (the Translator component in COMPASS) connects, provides its WGS 84 coordinate as argument and receives as result a RDF/XML document containing all the facts that the service knows for this location.

In order to find information on its current location, a COMPASS client needs to find a suitable list of Geo RDF Servers, or more precisely a list of service URLs of these servers. This process is commonly called service discovery. In our special case we need to find all services where the SCA includes our current location. We therefor apply the term "location-based service discovery" (LBSD).

Here, we present work in progress toward a decentralised, scalable and self-organising mechanism for the location-based service discovery of Geo RDF Services.

A key component in our architecture is the *P2P GRS Directory*, where we employ a structured Peer-to-Peer (P2P) network based on a Distributed Hash Table (DHT) for distributed storage and retrieval of Geo RDF Service URIs. In contrast to the well known unstructured P2P systems like Napster or Gnutella, structured P2P systems do not flood requests throughout the network but organize the P2P nodes in a way that the (storage or retrieval) requests can be efficiently routed towards a destination. Data is stored under a unique key that is usually generated by hashing an identifier like the filename.

In our solution, we use a variant of a quadtree-algorithm [8], the MX-CIF quadtree [9], to partition the globes' coordinate space and map fact validity areas as well as geographical coordinates to DHT-keys. In other words, we distribute a quadtree index of spatial data on a P2P-network.

Quadtree-algorithms decompose a two-dimensional space by recursively subdividing it into four quadrants. Each quadrant corresponds to one of up to four child-nodes of the respective node in the tree data structure. Quadtrees can easily be extended to cover 3D-data and are then called Octrees.

For each of the FVAs it holds, a Geo RDF Service determines the partition section that completely includes the FVA. The coordinates of the central point of that partition section are hashed to a DHT-key using a uniform hash-function. To avoid load-balancing problems, we use a hash-function with a uniformly random distribution of keys (SHA-1 in the implementation). The Geo RDF Service completes its registration for the FVA in the P2P GRS Directory by issuing a put-operation with its Web Service URI as value to be stored under the acquired key.

As there is no explicit tree stored in the DHT, the GRS also sets routing-flags that let clients determine whether additional tree levels are available. Further tuning options are maximum and minimum storage height, as suggested by Tanin et al in [11,12]. In order to prevent too many results or too deep trees, we set

a minimum and maximum threshold for the tree-level where no data is stored above or below this level respectively.

COMPASS Clients discover relevant GRS for a position by traversing the distributed quadtree starting from its root node, or rather from the node rooting the corresponding subtree, if a minimum storage level is set. This is done in the *Service Finder* module, which provides the Translator with the URIs of the identified GRS. For each level, the Service Finder determines the partition section that includes the sought-after coordinate, hashes the central point of it to a DHT-key using the uniform hash function and queries the P2P GRS Directory for that string. As result, the Service Finder receives URI-strings of relevant GRS and/or values containing routing information. Traversal of the distributed quadtree stops, when no matching routing-flags are received or the maximum storage level is reached.

The Translator component tries to contact every unique GRS URI discovered by the Service Finder using the SOAP interface mentioned earlier in this paper. The RDF/XML documents delivered by the GRS are then merged into a single RDF model containing all the RDF facts available on the network for the clients actual position. One big advantage of using RDF is that this basic RDF model could be easily refined and using additional facts that other public RDF database services in the semantic web may provide.

We have implemented the described system as a prototype using OpenDHT [6], a free, publicly accessible DHT-*service*. OpenDHT is a P2P network consisting of about 260 PlanetLab-hosts [7] running "Bamboo" [5]. Bamboo is an open-source implementation of a DHT based on a ring-structure similar to Chord or Pastry. Each OpenDHT node acts as a gateway exposing a minimalistic put/get-interface to clients which can issue operations using Remote Procedure Calls. OpenDHT-entries expire after some time, so Geo RDF Service registrations need to be updated regularly. This prevents stale entries from GRS that become unavailable. As opposed to implementing our own DHT-system or adapting an existing one, the usage of a DHT service like OpenDHT frees us from the need to deploy and maintain our own DHT infrastructure as well as it promotes a strictly layered approach of our architecture. A second advantage is that it lowers resource requirements for COMPASS-clients as they do not take part in message routing and data storage of the DHT.

A number of preliminary tests showed that our approach is usable in practice. At the same time they pointed out that the OpenDHT service sometimes performs very poorly. Although most get-requests were answered within one second, the distribution of response times showed a long tail with single response times above 10s. As the information of GRS is not expected to be very dynamic, the median values we obtained in our test cases would be feasible for COMPASS. The main problem lays with "stragglers" – slow nodes in the OpenDHT-network – which can introduce delays of dozens of seconds to the registration or lookup-process of GRS. The OpenDHT authors are aware of this problem and are working on solutions [10].

Our next steps will be to test and enhance the performance further. Another idea is to implement some kind of fact cache. At the moment, facts are retrieved freshly on each request. But as facts are also labeled with properties that express the area where they are valid in, this information could be used to temporarily cache the RDF statements in the translator and add them to future queries as long as the client stays in the valid area. Finally, the RDF vocabulary should be enhanced to cover the requirements of as many application scenarios as possible. As this also involves standardization, this should not be done by a single institution, but as a community effort.

References

1. Kargl, F., Bernauer, A.: The compass location system. In: Location- And Context Awareness, First International Workshop, LoCA 2005. Number 3479 in LNCS (2005)
2. Angermann, M., Kammann, J., Robertson, P., Steingaß, A., Strang, T.: Software representation for heterogeneous location data sources within a probabilistic framework. In: International Symposium on Location Based Services for Cellular Users, Locellus (2001) 107–118
3. Wendlandt, K., Ouhmich, A., Angermann, M., Robertson, P.: Implementation of soft location on mobile devices. In: International Symposium on Indoor Localisation and Position Finding, InLoc 2002, Bonn, Germany (2002)
4. W3C: The resource description framework (rdf). <http://www.w3.org/RDF/> (2005)
5. Rhea, S., Geels, D., Roscoe, T., Kubiawicz, J.: Handling churn in a dht. In: Proceedings of the USENIX Annual Technical Conference. (2004)
6. Rhea, S., Godfrey, B., Karp, B., Kubiawicz, J., Ratnasamy, S., Shenker, S., Stoica, I., Yu, H.: Opendht: A public dht service and its uses. In: Proceedings of ACM SIGCOMM 2005. (2005)
7. Bavier, A., Bowman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., and Mike Wawrzoniak, T.S.: Operating system support for planetary-scale services. In: Proceedings of the First Symposium on Network Systems Design and Implementation (NSDI). (2004)
8. Finkel, R., Bentley, J.: Quad-trees: a data structure for retrieval on composite keys. *Acta Informatica* **4** (1974) 1–9
9. Kedem, G.: The Quad-CIF tree: A Data Structure for Hierarchical On-line Algorithms. In: Proceedings of the Nineteenth Design Automation Conference, Las Vegas (1982) 352–357
10. Rhea, S., Chun, B.G., Kubiawicz, J., Shenker, S.: Fixing the embarrassing slowness of opendht on planetlab. In: Proceedings of USENIX WORLDS 2005. (2005)
11. Tanin, E., Harwood, A., Samet, H., Nutanong, S., Truong, M.T.: A serverless 3d world. In: GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems, New York, NY, USA, ACM Press (2004) 157–165
12. Tanin, E., Harwood, A., Samet, H.: A distributed quadtree index for peer-to-peer settings. In: ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05), Washington, DC, USA, IEEE Computer Society (2005) 254–255