A Simple Architecture for Delivering Context Information to Mobile Users

Sigrid Steinholt Bygdås, Pål S. Malm and Tore Urnes
Telenor Research and Development
P.B. 83, N-2027 Kjeller, Norway
+47 63 84 84 00
sigrid-steinholt.bygdas@telenor.com

Mobile devices running context-aware applications are in need of data sources delivering context information. We envision that context information data sources will be offered by third-party providers, and will likely be distributed across the Internet. Context information providers will prefer to use small and simple databases that do not require complex data modeling, configuration, and administration.

Context-aware applications require that database systems be able to run application-specific queries, often involving tailored operators. Unfortunately, it is impractical to manually deploy the tailored operators of each application on the necessary remote data sources. A solution to this problem is to have applications dynamically move query operator code to the remote database sites, as described in [1].

In this paper we briefly describe an architecture for delivering context information to wireless mobile devices from data sources on the Internet. The architecture is based on code mobility and the conceptually simple tuple space technology.

Context-aware information retrieval

In [2] Brown proposes a model for context-aware information retrieval. In this model, each piece of context information, i.e. a piece of information augmented with context data, is represented by a set of fields, each field being a name/value pair. Based on this model, a rule for matching the present context with pieces of context information can be defined. A matching rule defines the fields that are to be compared with the present context and the conditions for a field value match. For example, check all fields with name *temperature* and *location*, and return a temperature match if the value of the temperature field is the same as the present temperature, and a location match if the present location is closer than 100 m from the value of the location field. Matching rules may change dynamically, for example if a user removes a temperature sensor the application should no longer take temperature into account when retrieving information.

Tuple space technology

Tuple space technology [3] provides a global communication buffer for distributed applications, ensuring loose coupling between information sender and receiver. In a tuple space, information is organized in tuples. A tuple is a vector of typed values called fields. Storage of tuples in a space is persistent and the operations performed on the tuples are transaction secure. The search for information in a tuple space is made associative by using templates. A template is similar to a tuple except that some of the fields may not have values. A tuple matches a template if they have the same number of fields and each

corresponding field has the same type and value (or just type for template fields without values). A template that looks like this: [27.09.2000, string, “conference”] matches the following tuple [27.09.2000, “Bristol”, “conference”].

In our view, tuple space technology offers a natural (and simple) way of implementing Brown’s information model. This is due to the fact that a set of name/value fields can be represented by a vector of type/value fields (a tuple) where the field names are values of type string, like [“temperature”, 17.5]. Tuple space technology does not, however, provide sufficient flexibility to realize the matching rule semantics. What is needed is an extended tuple space implementation that facilitates dynamic customization of the tuple space matching algorithm. Recently, researchers at IBM provided such an extended tuple space system called T Spaces [4]. Based on Java and its support for code mobility, T Spaces offers dynamic uploading of application-specific query operators (matching rules). T Spaces also offers traditional database features while retaining the simple conceptual model of tuple spaces.

Proposed architecture

We propose an architecture for distributed context aware applications based on code mobility and tuple space technologies. Under this architecture, context information data sources are implemented using T Spaces and made available on the Internet. Context-aware applications running on wireless mobile devices dynamically install their query operators at the appropriate data sources (for now, discovery of available information sources is realized in a heuristic fashion). For low bandwidth situations we recommend deploying triggering engines and filters on proxy servers at the edge of the wired network, hence minimizing traffic over the wireless link.

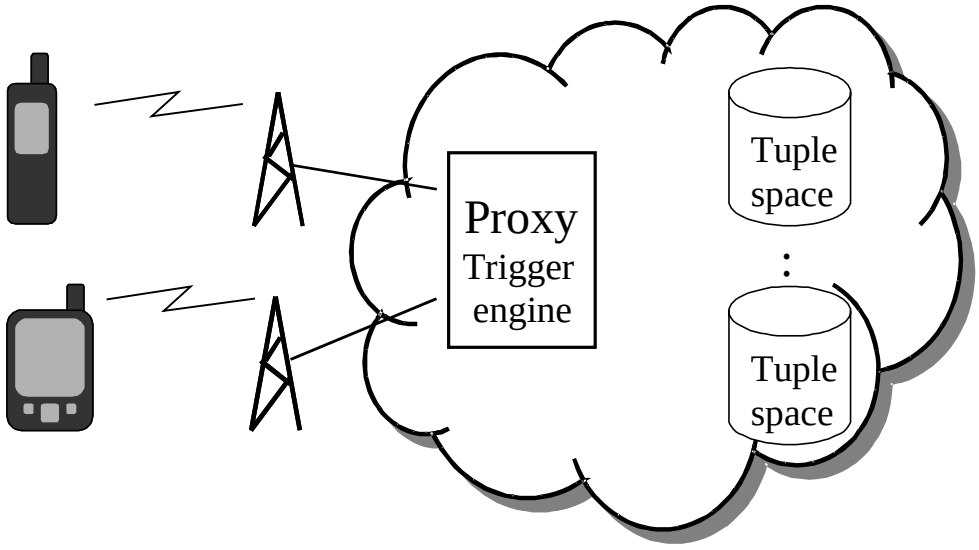

Figure: A simplified illustration of the architecture.

To test the architecture, we implemented a simple context-aware application. Context information in the form of notes augmented with location information, are stored in several T Spaces. Applications decide at run time which T Spaces to use. We use location data obtained from GPS sensors[1] as the present context. The application runs on Windows-based laptop computers and on EPOC-based PDAs from Psion. Triggering engines use the present context to provide users with the notes that satisfy matching rules. Through integration with a context-aware map application [5], users may also point on a map to see if there is a note in or around a location, without physically going there.

Concluding remarks

We have proposed and demonstrated a simple architecture for delivering context information to wireless mobile devices from data sources on the Internet. The two main features of the architecture are the use of code mobility and tuple space technologies. This makes it simple for both developers of context-aware applications and context information providers to use this architecture.

Acknowledgements

We would like to thank Martin Valland for implementing an early prototype of the triggering engine. Ole Martin Bakke ported the test application from Java 2 to Java 1.1.4 on the EPOC operating system. Finally, Øystein Myhre helped solve many problems related to developing Java applications on small wireless devices.

References

[1] M.R. Martinéz and N. Roussopoulos. MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources. ACM SIGMOD RECORD 2000, No. 5, 213-224, May 2000.

[2] P.J. Brown. Triggering Information by Context. *Personal Technologies*, 2(1):1-9, September 1998. (http://www.cs.ukc.ac.uk/pubs/1998/591/index.html)

[3] D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80-112, January 1985.

[4] P. Wyckoff et al. T Spaces. *IBM Systems Journal*, 37(3):454-474, 1998. (http://www.research.ibm.com/journal/sj/373/wyckoff.html).

[5] T. Urnes et al. Building Distributed Context-Aware Applications. *Personal Technologies*, 5(1), to appear, January 2001.

[1] We also support differential GPS sensors and have implemented an architecture where the differential signals are shipped to sensor-enabled devices using TCP/IP over the GSM network, eliminating the need to carry a separate antenna for receiving those signals.