

Enabling Wireless Context Sources

Gilman Tolle

University of California, Berkeley
Ubiquitous Computing Course Project
get@cs.berkeley.edu

CONTRIBUTION AND BENEFIT

Presents suggestions to enable context frameworks to effectively utilize sensor networks for context gathering. Can be used to extend current context frameworks to support complex and loosely connected sensor types.

ABSTRACT

Wireless sensor networks are sufficiently different from the types of sensors and direct human input systems previously considered for use in context management frameworks. New techniques may be needed to effectively use them for data gathering. Clouds of wireless sensing nodes provide opportunities for complex querying of properties from individual nodes and collections of nodes defined at the time of the query. The high cost of communicating with sensor networks suggests a need for in-network processing of context data. The connectivity of sensor networks changes frequently and unpredictably. This paper delineates new challenges created by these requirements and provides suggestions for future work on context frameworks.

INTRODUCTION

Context management frameworks are a class of infrastructure intended to support access to data about implicit user actions and changing environmental parameters. Examples of context include “identity, location, environmental state, activities, etc.” [4] Physical contextual data is generally gathered directly from sensors embedded in the environment, and virtual context is obtained by interpreting this data and merging it with information gathered from other digital sources. Context frameworks are intended to facilitate this sort of data access and manipulation for prototyping and development of context-aware systems.

Wireless sensor networks represent a novel tool for gathering information about the physical world. A sensor network is constructed as a cooperating collection of small general-purpose computing and communication nodes. Each node possesses a battery power source, a processor, a stored program, a small amount of RAM, and a radio for communication. In addition, each node may be outfitted with one or more physical sensors. Current sensor network installations have focused on collecting light, temperature, vibration, and other primitive data, but future uses of sensor networks will surely include RFID sensing, motion detection, entity tracking, and visual image processing.

Wireless sensor networks are intended to be easy to deploy, relatively self-configuring, and inexpensive.

Current context management frameworks have been used to gather data from RFID sensors and motion detectors, but these have generally been directly wired to a host computer. Enabling context frameworks to effectively utilize wireless sensor networks should expand the range of data that can be sensed and the environments that can be instrumented, and provide a valuable new source of context that should allow researchers to build more context-sensitive applications.

DATA-GATHERING FRAMEWORKS

Context management can be viewed as a multi-user distributed database problem. Multiple sources of information exist within a space, and these sources are often located on multiple systems connected by a network. In addition, because this context data can be changing constantly as time passes, context management is really a streaming distributed database problem. [10] Context frameworks attempt to present a state of the world that can be easily queried by context sinks and modified by context sources, but this worldview must necessarily be sampled from the actual flow of implicit data that is possible to extract. This highlights the distinction between the two necessary tasks for a context management system: data gathering and data representation.

Context data has heretofore been presented as semi-structured and entity-centric instead of relational. Each piece of context data is viewed as a statement about a particular entity. This is intended to enable a more natural mapping between primitive data items like names and the spaces and users these data items are referencing. Some context management work has taken this idea of entities and statements further, and included the ability to make statements relating multiple entities which can themselves have statements made about them.

In order to make statements that relate entities, a context framework needs a clear concept of the unique name of an entity. If entities can be located at multiple network locations and have the ability to independently present information, then a unique name must be included in statements if one entity is intended to be able to present information about another.

To allow for data presentation that crosses personal trust and control boundaries, each piece of context should have a home server corresponding to a particular entity. If we enable one entity to present information about other entities, possibly entities in different trust domains, a further need for unique naming arises.

Looking at the data gathering mission of context frameworks, access methods have traditionally been divided into *queries* and *subscriptions*. For a query to act as a blocking call and response operation, a snapshot must be taken of the world state under query. If this is impossible, or too costly, or cannot be done within a reasonable time, then a query cannot be a simple call and response. This becomes increasingly evident when a query must access more than just cached data on a server. Accessing data from a mote cloud requires propagating a request across a mesh of slow links, and waiting as results trickle back to the source. To effectively query sensor networks, all queries should be treated as subscriptions.

We now turn to an examination of existing context frameworks in light of these requirements.

Context Toolkit

The Context Toolkit, developed by Dey, Abowd, and Salber, is a conceptual framework and Java toolkit intended to enable the inclusion of contextual data into prototype applications. [3] The fundamental abstraction provided by the CTK is the *widget*. A widget is a network-addressable representation of context data gathered by a single particular source. The widget is a metaphorical piece of hardware, combining both data gathering and data presentation. Queries over contextual data are directed at these physical sources of context data.

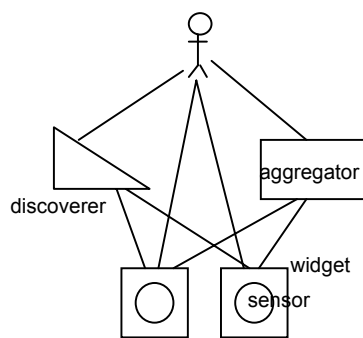


Figure 1: Architecture of the Context Toolkit

Widgets register the names of the data items they may present with a central discovery server, intended to be unique within the local subnet. Context sinks must first query the discovery server for widgets providing the sort of context data they intend to access, and then query the widgets directly. Queries are a Boolean combination of predicates over attribute/value pairs presented by the individual widget.

The toolkit also includes *interpreters* and *servers*. Interpreters are stateless transformers of context data, and may be accessed similarly to simple widgets. The server is intended to allow aggregation of context data and query through a single point, but has been mainly unused in applications thus far.

The CTK does support subscriptions to a single widget, by allowing subscriptions to the entire collection of context provided by a widget. The application can also apply filters to subscribe to a subset of the available context. These are similar to queries. Network messages are sent when new data is gathered by the widget's sensor.

Context Fabric

The Context Fabric is a next-generation context toolkit currently under development by Jason Hong. One underlying motivation of the CF is to separate the act of gathering from the act of representation. [7]

The Context Fabric's fundamental abstraction is the InfoSpace. Each InfoSpace is intended to present data about a particular entity, and this data may be acquired by multiple sensors. The CF uses the *context tuple* to represent a statement about an entity, and it is these tuples which are the objects of and return values from queries. [8] Each tuple represents a named link between an entity and a piece of primitive data. A tuple contains information about sources for the data in the tuple and names of the entities described in the tuple, but these concepts are not yet clearly defined. The InfoSpace containing the tuple is implied to be the subject of the tuple's data.

In addition, a tuple can contain a link to another InfoSpace. This can be used both as a statement relating two entities by relating their InfoSpaces, and as a method of discovering other context data through a decentralized hyperlink model. Tuples also contain some information about when the piece of information was obtained, and how long it should be considered "live".

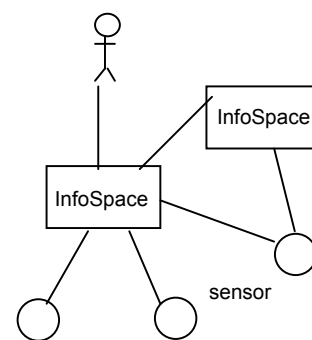


Figure 2: Architecture of the Context Fabric

A single tuple effectively contains two different types of information: statements and meta-statements. Statements are the basic triple of (implied) subject entity, the name of

the relationship, and the primitive data or InfoSpace link acting as the object. Meta-statements include the source of the statement, the time the statement was created, and the access permissions present on the statement. This distinction will be useful in supporting a richer aggregation model.

Queries are directed to a single InfoSpace, and are also composed of Boolean attribute-value predicates. The current implementation of the CF uses the surface XML syntax containing the context tuples, as opposed to a more abstract query language over the statements themselves.

Liquid

Liquid, a project by Jeffery Heer *et al*, is intended to move beyond the single-destination queries of the Context Fabric. Liquid queries can follow links between InfoSpaces using entity type paths (e.g. location.occupants.name), and Liquid servers are responsible for moving queries and result tuples along the path. [6] This approach is limited to processing queries in the order provided by the links, and a fuller aggregation system could allow more complex queries with less network traffic.

TinyDB

The current state of the art in querying sensor networks is the TinyDB system. TinyDB uses a single connection point between the mote cloud and a network, and presents a conceptually infinite table of relations between a mote identifier, a sampling epoch, and data values gathered by the individual motes. TinyDB can be queried through a SQL-like language, extended with information about sampling intervals. Queries enter the network and single relations representing results stream back as they are collected. The richness of SQL allows queries to apply predefined functions like average or maximum over data values of the same type. The TinyDB program running on each mote builds a tree structure, and actually computes the result of the function as the data streams out of the network. This drastically saves energy and communication costs. [9]

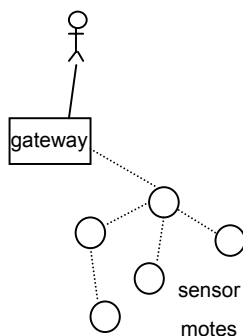


Figure 3: Architecture of TinyDB

Because TinyDB is focused specifically on sensor networks, queries cannot be used to merge sensor network data with data gathered from other sources. The ability to

make this fusion will be a benefit of enabling context frameworks to directly query sensor networks. In addition, users of TinyDB cannot cooperatively access one cloud from multiple points. Much of the rest of the paper will focus on applying the ideas presented by TinyDB, the current best-of-breed sensor network query system, to the wider field of context frameworks.

Other Related Work

The Directed Diffusion system is a less centralized sensor network query system supports name-value access to data within the network from multiple points outside of and even within the network itself. [5] DD does not currently include in-network computation of aggregation functions or access to data that contains information from more than a single node.

The Cougar Sensor Network Query System [11] is similar to TinyDB with in-network computation of aggregation functions, and is tightly focused on querying a sensor network as a distributed database.

NEW DESIGN SUGGESTIONS

The main distinction between a mote cloud and a collection of separate sensor platforms is the existence of intra-cloud communication. One cloud of motes containing physical sensors can present many virtual sensors, created by selection and aggregation predicates chosen at query time. These virtual sensors can be simple aggregations or complex functions, and can be defined differently with each query. The SensorML data representation language is intended to support virtual sensors as well, but these virtual sensors must generally be predefined.

Flexible Aggregation support

In current context frameworks, statements about an entity can only be obtained by communicating directly with that entity. A mote cloud is both a single entity and a collection of entities, and thus for optimal processing of queries within the network, queries submitted to the “mote cloud” entity should be able to return results about individual mote entities. These may be entities directly representing physical motes, or they may be virtual entities defined by the query. Regardless of the exact meaning of the entities, future framework should support this sort of aggregation.

To enable this, individual pieces of context data should have a clearly defined notion of the name of the entity about which they are making a statement. If this name is something like a URL, and contains information about network access, then aggregation of data from multiple entities at different network locations becomes possible. If multiple mote entities and multiple external entities can both be used in aggregation, then the query processor can further optimize queries by combining these sources of information.

For example, if a mote cloud entity has access to location information about each mote that each individual mote does not directly contain, then the query processor can merge

this information with a location predicate in the query and direct only certain motes to respond with their data. If the cloud gateway entity can translate this separate contextual data about location into something that can be stored within the motes themselves, the query entered into the cloud does not even have to contain a list of valid motes. This can save both time and energy for complex queries.

The need for aggregation is not only limited to optimizing query processing. Examples of virtual entities, like collections of privacy settings presenting different faces of a single user, could easily be imagined for more traditional context data. Must these virtual entities necessarily be accessed from their own separately addressable database? Giving one entity the ability to present information about other entities enables a much richer representation of the world.

Finally, individual wireless motes can be connected and disconnected frequently. If the context client is required to directly address individual motes, then the need to manage connection state is more frequent and places a greater burden on the client when a mote cloud is being queried.

Enhanced Query Language

As described earlier, functions can be computed over collections of sensor data. Clients currently have the ability to express a selection query for this data and then compute the function after the source data has been acquired.

Communication within sensor networks is very expensive. If the client desires the result of a function that takes less space than the data required to calculate the function, then only the result should be extracted from the network. Sensor network nodes can be arranged into an aggregation structure and can each compute partial values such that the communication costs diminish as the query results leave the network. In fact, such drastic energy savings can be obtained from in-network computation of aggregation functions that all queries for data which will eventually be combined should certainly be processed within the sensor network.

The variant of SQL used in TinyDB can express named predefined functions over single elements of relations. Future context framework query languages should also be able to express functions directly within the query so they can be computed by the query processor itself. Defining these functions may require a complex decomposition into partial computation programs, so even if queries cannot actually define new functions at runtime, they should still be able to apply functions that have previously been programmed into the sensor network.

Variability over Time

Not only must queries account for changes in the data being queried, but when querying sensor networks, queries must consider the time and energy cost of transporting the data itself. Because information must be propagated through a slow multi-hop network, query results can only arrive

piecemeal. Some may not arrive at all. If the query expects a global snapshot of the data, then the client must be prepared to wait for an arbitrarily long time. This necessitates the inclusion of the query length into the query and the ability to accept partial results over time.

If the results of earlier queries can be cached, then the query language should also contain some way to express how old the cached data is allowed to be before a new sample from the environment must be taken. In order to evaluate the results of this caching and make new estimations, query results must include the time that they were sampled, and if necessary the time they were retrieved from the cache.

One approach to the caching problem would be to proactively sample the whole network at a rate decided by the mote cloud entity. This is by far the most wasteful way to collect information. In general, data should not be gathered until a query explicitly requests it. If the query is intended to force the taking of samples by the sensor network nodes, then query language should also be able to express a sampling period.

Not only should query languages be able to express temporal information about the data gathering process, but clients may also need to filter the data by temporal predicates. In current context frameworks, a subscription query can be given to an RFID sensor, and when a new tag appears, the state of the world has obviously changed and results will be sent immediately back to the subscriber. When the sensor is sampling a continuous value, like a temperature or a changing location, then the query should be able to contain predicates over time that indicate when enough of a change has occurred to justify informing the subscriber.

For example, a query might make reference to the value stored at the last time the world sampled in a query like: “return the location of each mote where the temperature at time t differs from the temperature at time $t-1$ by 3.”

Depending on how much information is stored at the nodes themselves, more complex temporal functions like a moving average may be eligible for inclusion in a predicate. A query could request data only when “the temperature rises above the exponentially weighted moving average of past temperatures.”

Streaming database researchers are attempting to create a well-defined semantics for queries of this sort over temporal streams, and this work should be integrated into future context frameworks as well as future sensor network query systems. [1]

FUTURE WORK

To examine the worth of these suggestions, an existing context framework should be modified and then evaluated by testing it on the population of programmers intended to use it. [4] In addition, new context frameworks may be developed with stronger focus on the representation of

changing knowledge. One such example is presented in below.

RDFSspaces

In the Hewlett-Packard Cooltown project [2], the researchers chose to take advantage of existing Web standards like XML to make the toolkit more widely usable. The Semantic Web group has been developing a framework for representing complex metadata about documents, called RDF, and this standard may be useful in context management.

RDF is intended to express named relationships between named entities. This sort of triple can be equated with an attribute/value pair when the first entity is implied and the second entity is a primitive value. Because RDF uses URLs as the unique identifiers for entities, statements can be made about more complex network addressable data objects and other entities. If these URLs are also used as actual network hosts that contain statements about the entity represented by the URL, then RDF queries can follow these links for distributed hyperlink-style data access.

In addition, RDF statements themselves can be given unique identifiers. This gives the framework the ability to support meta-statements, statements about particular statements. These statements could include when the statement was created, representing sampling time, how often the statement is intended to be valid, the URL of the original source of the statement if it is presented by an aggregator, or a privacy policy indicating who is allowed to access the statement.

A URL representing an RDFSspace could be used to query and change the statements present within that space, much as the Context Fabric does now. Individual RDF statements could be streamed back as result tuples, and made available in other RDFSspaces for aggregation.

Researchers are currently developing query languages to collect data from the graph formed by RDF relationships. The developers of these languages should also consider the strengths of sensor networks and should consider include aggregation functions and temporal operators within queries. A temporally-aware distributed RDF query language could form the basis of a powerful new context management framework.

CONCLUSION

The unique requirements of wireless sensor networks when compared with powered standalone wired sensors create many new opportunities for gathering of more complex context data and optimization of the gathering methods. Future context management frameworks should include support for named subjects and aggregation to effectively present a sensor network cloud and its component sensors. Query languages should be extended to include function application to enable in-network query processing. Finally, contextual queries should be able to express temporal

predicates and clearly delineate how and when streaming data is to be gathered. When sensor networks are more commonly installed as a stationary source of context data for use by multiple clients, these improved query strategies should lead to longer life and more effective utilization of the network.

REFERENCES

1. A. Arasu, S. Babu and J. Widom. "An Abstract Semantics and Concrete Language for Continuous Queries over Streams and Relations." Technical Report, November 2002.
2. Philippe Debatty and Deborah Caswell. "Uniform Web Presence Architecture for People, Places, and Things." Hewlett-Packard Laboratories, Palo Alto, CA, USA.
3. Dey, A.K., Salber, D. Abowd, G.D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, *Human-Computer Interaction (HCI) Journal*, Vol. 16 (2-4), 2001, pp. 97-166.
4. W. Keith Edwards, Victoria Bellotti, Anind K. Dey, and Mark W. Newman. "Stuck in the Middle: The Challenges of User-Centered Design and Evaluation for Infrastructure." *Proc CHI 2003*. Ft. Lauderdale, FL, USA.
5. Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. "Next Century Challenges: Scalable Coordination in Sensor Networks." *Proc Mobicom 1999*.
6. Jeffrey Heer, Alan Newberger, Chris Beckmann, and Jason I. Hong. "liquid: Context-Aware Distributed Queries." *Proc Ubicomp, 2003*.
7. Jason I. Hong and James A. Landay, "An Infrastructure Approach to Context-Aware Computing." In *Human-Computer Interaction, 2001*, Vol. 16.
8. Jason I. Hong. "The Context Fabric: An Infrastructure for Context-Aware Computing." *Proc CHI 2002*.
9. Samuel Madden. *The Design and Evaluation of a Query Processing Architecture for Sensor Networks*. Ph.D. Thesis. UC Berkeley. Fall, 2003.
10. R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varna. "Query Processing, Resource Management, and Approximation and in a Data Stream Management System." *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, pages 245-256, Pacific Grove, California, January 2003.
11. Yong Yao, J. E. Gehrke. "The Cougar Approach to In-Network Query Processing in Sensor Networks." *Sigmod Record*, Volume 31, Number 3 September 2002.