Dynamic Resource Discovery for Sensor Networks*

Sameer Tilak[1], Kenneth Chiu[1], Nael B. Abu-Ghazaleh[1], and Tony Fountain[2]

[1] Computer Science Department, SUNY Binghamton
[2] San Diego Supercomputer Center, University of California at San Diego

Abstract. As sensor networks mature the current generation of sensor networks that are application-specific and exposed only to a limited set of users will give way to heterogeneous sensor networks that are used dynamically by users that need them. The available sensors are likely to be dynamic (e.g., due to mobility) and heterogeneous in terms of their capabilities and software elements. They may provide different types of services and allow different configurability and access. A critical component in realizing such a vision is *dynamic resource discovery*. In this paper, we develop a resource discovery protocol for sensor networks, outline some of the challenges involved, and explore solutions to some of the most important ones. Specifically, we first discuss the problem of what resources to track and at what granularity: in addition to the individual sensor capabilities, some resources and services are associated with sensor networks as a whole, or with regions within the network. We also consider the design of the resource discovery protocol, and the inherent tradeoff between interoperability and energy efficiency.

1 Introduction

The success and increasing deployment of Wireless Sensor Networks (WSNs) is driving towards a new generation that envisions commodity sensor networks providing standard services that can be composed by clients into a wide range of applications. These sensor networks then form a critical, yet generic interface between the physical and digital worlds, converting physical qualities into measurements that can be then harnessed for a wide-ranging spectrum of applications. The impact of such a development would be profound: no longer would sensor networks be specialized networks serving a limited set of users, but rather a basic infrastructure supporting and encouraging unanticipated functions and modalities of operation. Increasingly advanced and accessible applications will be enabled as sensor networks are shared dynamically and ubiquitously, allowing clients to unite disparate components on demand into *virtual sensor networks*. Some clients may even compose services spanning multiple sensor networks into a single end-to-end service for use by applications.

* This work was partially supported by NSF Awards SCI-0330568, CNS-0454298 and DBI-0446298, and Air Force grants FA8750-04-1-0211 and FA8750-05-1-0130.

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 785–796, 2005.

Despite the recent emergence of sensor networks as a field of study, already many sensor hardware platforms with a wide range of capabilities have emerged. In addition, there are large differences in the software elements (operating systems, networking protocols, data base systems, etc.) used for these platforms. We believe that this heterogeneity is inevitable, and, furthermore, we believe that an attempt to impose uniformity on this diversity would stifle experimentation and innovation. But without some commonality, interoperability is not possible. We thus believe in the adoption of minimal, extensible standards that can promote interoperability by supporting the discovery of formats and protocols.

A necessary precursor to interoperability is the ability to discover beneficial resources and the attributes of these resources required for interoperation. Resource discovery in sensor networks is challenging for a number of reasons. Since different sensor networks are deployed and managed by different organizations, they are heterogeneous in aspects such as protocols, architectures, security policies, and management policies. Also, the nature of resources (sensing and coverage characteristics, connectivity characteristics, available energy, computational and storage capabilities, protocols and software elements) and services (e.g., localization, synchronization, calibration) are significantly different from traditional distributed systems. Furthermore, the potential mobility of sensors and clients introduce unique challenges [17]. Finally, the embedded nature of sensors place a premium on energy efficient solutions. Thus, WSN resource discovery requires significantly different solutions from traditional naming and resource discovery in distributed systems [1, 10].

The resource discovery problem in sensor networks can be broken down into two components: (1) Determining what resources to track and at what granularity to track them. In the context of sensor networks this is a challenging and multifaceted problem; and (2) determining the resolution protocol that queries the sensor network for the resource information and transfers this information to queries in an energy efficient way. Note that the two components are similar to other distributed resolution systems (e.g., DNS' servers and resolvers [15]). However, sensor network operation introduces many characteristics that make this problem unique such as resource attributes that are associated with the network as a whole or regions within, as well as energy efficiency concerns.

2 Dynamic Resource Discovery in Sensor Networks

An essential component in systems where foreign and heterogeneous elements interoperate is the ability to discover relevant information regarding available resources. A new client generally has no knowledge of what sensor resources exist around it. Moreover, properties such as protocols and formats that are relevant to interoperation must also be obtained. We call this problem *Dynamic Resource Discovery (DRD)*. This section overviews the challenges in DRD for sensor networks. The discussion is organized into the two major aspects of DRD: (1) Determining tracked attributes; and (2) Resource Discovery.

Determining Tracked Attribute Set: The decision on what resources, services and other system attributes are tracked to support interoperability and service discovery is an important one. The candidate attribute set for tracking may be classified along multiple axes. First, attributes can be classified into those that provide information regarding interoperability (e.g., protocols used or formats for messages), and those that describe the metastate of the system (resources or services provided). Standardization plays a role in determining what interoperability information to track (e.g., if sensors are standardized to be completely homogeneous, no interoperability information needs to be tracked). It is important to discriminate here between resource discovery and data collection. While both operations require collecting information from the sensors, DRD collects meta-information and not data. Although the resource discovery and data collection may be combined for efficiency in certain cases, it is likely that separating them will lead to a more effective and modular solution.

A different axis for classifying tracked attributes resource granularity. We classify attributes as simple (associated with single sensors) and complex (representing multiple sensors). Clearly, individual sensor resources are of interest. These include sensing resources (such as available sensors, their tolerances, and their coverage) as well as computational, storage and communication resources. Additional properties exist for the network as a whole (e.g., what routing protocols are used). Moreover, resources may be aggregated: instead of tracking and reporting sensor resources in detail, they may be summarized (e.g., coverage in an area instead of individual sensor location). Thus, the system must be able to associate and track resources at these different granularities.

A related issue arises due to the data-centric nature of sensor networks: they are embedded within the environment they are monitoring, and are mostly of interest only in terms of what information they can provide about the environment. Resources may be associated with regions in terms of network organization (e.g., resources or services in a cluster) or in terms of the environment being monitored (e.g., resources or services available in a room). The resource discovery protocol should provide the flexibility of tracking resources in terms of application-level or infrastructure-level organization. The choice of what attributes to track and at what granularity to track them is left to application developers and not discussed further in this paper.

Resource Discovery Protocol: Once the tracked attributed set is determined, the role of the resource discovery protocol is twofold: (1) Track the values of the attributes at selected points within the network; and (2) Respond to client queries. It is necessary to minimize the communication required for implementing DRD due to energy efficiency considerations. Energy efficiency also dictates being able to aggressively power down sensors when they are not being used; powered down sensors cannot receive or respond to queries.

In terms of tracking the selected attributes, a choice exists between pushing the attributes proactively to selected points in the network or pulling them in response to received queries. In distributed systems, generally pull is preferred to push when requests are infrequent relative to the data change rate, while

push is preferred when requests are frequent. In sensor networks, two additional factors favor at least partial push of resource information: (1) because sensors must operate on a low duty cycle, pulling the data causes large delays if the data is locally maintained at sensors which are currently sleeping; and (2) for complex attributes, the attributes must be summarized across multiple sensors; this requires pushing the data to points in the network where they can be combined.

Depending on the chosen approach, query resolution is implemented. In this component of the resource discovery protocol, the client query is forwarded to attribute repositories (locations in the network where attributes are collected) that are relevant to it. In a brute force unstructured approach, the queries may be flooded within the network. More efficient unstructured solutions may use probabilistic algorithms such as gossiping. Alternatively, if the resource space is structured such that the location of attributes relevant to a query are known then more efficient query forwarding can be used (e.g., name resolvers in DNS [15]). Further, caching may be used to optimize operation in either approach [20].

Interoperability favors a standardized protocol like ASCII-based XML that does not presume specialized protocols with potentially foreign elements generating the queries. However, efficiency dictates custom protocols that are compact. In the next two sections, we investigate alternatives to balance these requirements and show that it is possible to optimize the size of the exchange messages without sacrificing interoperability.

3 Architecture

To explore the feasibility of our vision of future sensor networks, we have developed a simulation-based prototype for DRD. We use an architecture where sensors self-organize to form clusters. A cluster is a collection of sensors that are associated/represented by a single *Cluster Head* (CH). For DRD, this organization serves the following purposes: (1) The CH represents a logical point for maintaining complex attributes; (2) The CH receives DRD queries and is able to respond to them, freeing the remaining sensors to be powered down in periods of inactivity; and (3) Finally, if hierarchical naming is desired, clusters can in turn themselves be clustered into bigger entities to provide more efficient query forwarding for structured attribute sets. We note that the use of clusters is common in sensor networks [11], mainly to allow data aggregation/reduction (similar to our complex attributes) and resource arbitration resulting in more scalable and energy efficient solutions.

In our algorithm, similar to GAF [30], cluster membership is determined geographically. The sensor field is divided into zones such that all sensors within a zone are in range with each other. Cluster selection is then localized to a zone such that a sensor only considers CH advertisements occurring in its zone; only one CH is selected per zone. We note that this approach requires either preconfiguration of the sensors or the presence of a location discovery mechanism (GPS cards or a distributed localization algorithm [3]). In sensor networks, localization is of fundamental importance as the physical context of the reporting

sensors must be known in order to interpret the data. We therefore argue that our assumption that sensors know their physical co-ordinates is realistic. We emphasize that cluster formation is orthogonal to the proposed resource discovery protocols and other cluster formation approaches can be used. The overall operation consists of the following phases:

1. Cluster head election: Cluster election identifies cluster heads, which serve as control points, data sinks and meta-data repositories for their cluster members. In this phase, sensors send Resource Description Format (RDF) [27] advertisements indicating their remaining energy to neighbors. Upon receiving all neighbor messages, the node with the highest remaining energy is elected as CH. Other selection criteria are possible.
2. Meta-data exchange: In this phase, nodes send their meta-data to the CH as an RDF message, where it is inserted into an SQL database using a simple mapping from RDF properties to table columns. For static attributes, this phase is combined with CH election, piggybacking meta-data on CH election messages. For dynamic attributes collection may be required more frequently. For example, if a video sensor changes its angle, zoom, or resolution such meta-data needs to be registered with each change.

   Up till now, we have focused on the discovery protocols (attribute tracking). However, meta-data is also needed to describe the data stream reported by a sensor. Interestingly, for some long running queries, the data reported by a sensor might change more dynamically. Consider a sensor that has temperature and audio transducers on board. A query may require the sensor to report its data when any of the temperature or audio measurements have interesting events. In such cases, the sensor needs to associate meta-data with its data stream so that the receiver can identify whether its a temperature, audio, or a combination of these streams.
3. Query generation: In this phase, new clients that need to ineract with the network pose queries to it. The query API is either standardized or negotiated via the resource discovery protocol. In our study, the query message is an SQL query sent as ASCII.
4. Query forwarding: In this phase, queries from clients are forwarded towards attribute repositories that can satisfy them. Depending on the application, there may be room for optimized query forwarding (e.g., for attributes that are strictly hierarchical, or when queries can benefit from the results of previous ones). For unstructured resources, multicast or epidemic algorithms are most efficient.
5. Query processing: A CH upon receiving a query, processes it and sends back the appropriate reply.
6. Resource/service usage: Based on the reply from the previous stage, S_1 (or a client) starts accessing the requested resources and services. For example, the new sensor might find its current physical coordinates using localization service provided by location anchors and upon negotiating data transfer protocol, it might start reporting its observations to the new CH.

4 Architecture Discussion

DRD involves several aspects, including resource description, resource registration, resource query, and message encoding. The presented architecture describes the overall interaction of these aspects, but is not tied to the particulars of how resources are described, how queries are formulated, or how information is encoded. These can all be done in a number of ways, each with different trade-offs. Resource discovery in sensor networks can draw from a number of efforts in distributed systems and web services. In this section, we provide further discussion of some of the important facets of DRD.

Resource description: A number of possibilities exist for resource description. We could of course simply use a completely ad hoc resource model. This has the advantage that we can customize it to be very compact, and tailored for DRD needs. The disadvantage, though, is that it would be incompatible with other resource descriptions. This means that it would not be able to leverage other associated software and standards. Since we could not see any significant advantage to creating our own resource descriptions, we used the RDF model developed by the W3C. We anticipate that as our research matures, we will actually develop an ontology of resources. This will allow us to apply description logic to resources. As we develop an ontology, we will naturally move from RDF to OWL. This is aided by the fact that OWL actually uses RDF. We are currently not using the standard RDF/XML syntax, but we anticipate that we will as our work matures. We have addressed size concerns by using binary XML, which we describe below.

Query formulation: When a client needs to query the CH for the available resources, it needs to formulate the query in some language. We are currently map the RDF description to a table, and thus formulate the query as an ASCII SQL string. This has the advantage of using a common, simple language for queries, but cannot handle more complex, structured resource queries. Query formulation for resource descriptions and ontologies is currently an active research and development area. A number of RDF and OWL query languages exist, such as OWL-QL [8], RDQL [18], and RQL [14]. Further investigation is necessary to determine which approaches will work best for sensor networks.

Formats and encodings: A number of formats can be used to encode the messages used in the various phases. Since communication requires large energy expenditures, a compact format has the advantage of conserving energy. For example, the resources of a node may be represented as: (`41 1000 0 500 1000`). This data being 5 integers, takes only 20 bytes, but to interpret these numbers correctly, the client must know that the first number is the ID, the second number is the precision, the third number is the minimum of the range, the fourth number is the maximum of the range, and the last number is the capacity. Such detail is error-prone, and the format may lack flexibility and interoperability. Changes to the format will be hard to detect, may cause strange failures or hard-to-find bugs.

Formats such as XML, along with standards such as SOAP [26] and WSDL [6], offer the advantage of having a large community of developers and researchers working on a common, well-known language. By using these, we can leverage the standards and available expertise. The disadvantage of these formats, however, is that they are verbose. The verbosity incurs large energy costs during communication. An XML document typically contains many identical strings referring to tags, namespace prefixes, and namespace URIs. Every XML element also includes a redundant end tag. The following XML document contains the same data as the custom message above, but requires 191 bytes.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<index version=''1.0''>
  <id>41</id>
  <precision>100</precision>
  <minrange>0</minrange>
  <maxrange>500</maxrange>
  <capacity>1000</capacity>
</index>
```

One alternative that reduces the energy costs, but retains many of the advantages, is to use alternative encodings of XML. XML is specified as a textual format for structured data. The specification implicitly suggests an abstract data model for tree structured data consisting of parent nodes and child nodes, with their associated attribute, content, and other information. This model does not inherently preclude an efficient encoding, and thus alternate serializations of that are much more concise than textual XML. These alternative serializations, are commonly known as "binary" XML, since they often utilize the full 8-bit range of the constituent bytes. Since binary XML is logically the same as textual XML, a single API can work for both. Also, all standards based on XML work with binary XML, since most XML standards reference the XML abstract model known as XML Infoset.

We have tested two versions of a binary XML. The first version eliminates the redundant end tag, and replaces ASCII numbers with their two's complement representations. The second version achieves further compactness by separating much of the static meta-data about the message from the data of the message itself. This is achieved by using a separate XML Schema of the message, and combining the Schema with a compact representation of the data to reconstitute the complete XML document. This is similar to ideas in PBIO [7] and DFDL [5].

Interpretation of these compact messages requires an associated schema. This association can occur through a number of mechanisms. One technique is to simply include a schema URI in the beginning of the message. This overhead may be too high for short messages, however, since the URI is typically a relatively long string. Another technique is to assume that some previous information has been exchanged, and then associated with the particular communication channel corresponding to the compact message. For our prototype, we have simply hard-coded this association. We note that this use of the schema to interpret a compact representation of the message can also be applied to textual XML.

5 Experimental Study

In this section we first describe the design of our simulation framework and then demonstrate its utility using a simple prototype simulation study. We hope to extend this framework as we develop our VSN subsystem to complement implementation on real sensors. The evaluation testbed is completely based on open source software components; either already available in public domain or built in house. The integration of these typical software components ranging from an embedded database, network simulator, and XML parsers provides an accurate development and evaluation environment that can reduce the development barriers for a broad spectrum of applications. It can also assist researchers in evaluating their protocols. We now describe the individual components of the framework.

- *SQLlite database.* SQLite is a self-contained, embeddable, zero-configuration SQL database engine [19]. We decided to choose SQLite because it implements most of SQL92 and requires zero-configuration – no setup or administration is needed. This property is useful because the large scale and autonomous nature we target, prohibits manual configuration and administration. It stores the complete database in a single disk file. This is important given the resource constraints – having multiple database files on disk and memory may not possible for some embedded sensor filesystems. Most important to our purposes, it has a small code footprint.
- *Libxml2.* It is the XML C parser [24] and toolkit developed for the Gnome project. We are also exploring various other XML parsers such as TinyXml [22], XPP [29], Xerces [28] and would like to evaluate their performance and power characteristics.
- *Binary xml parser.* We used an experimental binary XML implementation developed in-house, which is discussed in Section 4.
- ns-2: It is an open source discrete event network simulator [16]. In our simulations we used a CSMA based MAC layer and our power analysis is based on the energy model pre-built in ns-2.

This framework does not imply that there exists heterogeneity in terms of software elements. Specifically, we do not expect individual software components (SQLite, libxml etc.) to be installed on all the sensors. However, we expect the system components to follow appropriate standards and expose standardized interfaces. For example, a mote might run an instance of TinyDB [21] database, whereas a PASTA node might run an instance of SQLite. However, since both these database technologies follow standards and expose SQL interface, they fit in within our system design notion. To summarize, we take a technology agnostic view and our system design is based on open standards and standardized interfaces. Above mentioned software components just provice a concrete basis to build our simulation framework, and not as an end in itself.

At some level in the system, commonality must exist. From this common level, clients would then bootstrap and configure themselves to work with the relevant sensors. For the lowest levels, we believe that RDF is the appropriate

Table 1. Energy consumption study

Protocol	Data size (bits)	Transmit Energy (J)	Receive Energy (J)
custom	160	0.000033	0.000038
ASCII xml	1448	0.000302	0.000342
binary xml	160	0.000033	0.000038

choice. Above that, however, we believe that the issue is an open question which we will address in future work. We also believe that connectivity to the grid and web services will be important. Such interoperability can greatly increase the effectiveness and utility of sensor networks by allowing them to be plugged into wide area cyberinfrastructures.

To demonstrate the utility of the proposed framework, we conducted prototype study to evaluate the energy efficiency of message encoding format alternatives. For modeling energy model realistically, we used a 802.15.4 style low power radio: the bandwidth is 250kbps, and the transmit, receive and idle power are 0.0522, 0.0591 and 0.00006 Watts respectively. The simulation area was set to $350 \times 350\ m^2$ with 50 sensor nodes. Each zone was $70 \times 70\ m^2$ (total 25 zones). Radio transmission range was set to 100 m to ensure that all sensors within a zone are in range with each other.

Table 1 shows the mean energy consumed per sensor per message for various message encoding formats. As expected, we see that using ASCII based XML format, both the transmission and reception energy consumed is almost 9 times higher than that of custom encoding approach. On the other hand the use of a compact binary XML encoding format (with predetermined, separated XML schema) consumes same amount of energy as that of a custom encoding format. Completely customized protocols are most energy efficient but offer very little flexibility and interoperability. Whereas the textual XML represents the other end of the spectrum. Using XML has a number of attractive benefits, but the verbosity incurs large energy costs during communication (almost 9 times higher than the custom based formats). However, the use of binary XML is a compromise between textual XML and custom formats. The results helped us to validate our simulation framework.

6 Related Work

In an open distributed system, the problem of naming and locating resources is challenging and has been well studied in literature [1, 10]. Several protocols for service discovery have been proposed [13, 23, 25]. Moreover, in mobile environments, the effect of mobility has been also considered (e.g., [4, 17]).

Jini [13] supports service discovery via service registration and lookup. While Jini works well for its intended applications, we believe that it is not ideal for realizing DRD for sensor networks. First, Jini is Java-based, and this permeates its design. In theory, one could bind its wire formats and protocols to other languages, but the result would likely be awkward. For example, Jini's service description model is based on Java's rules for class derivation. A service query

matches a service if it is a supertype or same type as the service. This categorically precludes the use of a number of promising research areas for wide-scale interoperability and mediation, such as ontologies, RDF query languages [18, 14], description logics [2], and semantic mediation [9]. Furthermore, its communication model is based on Java RMI and it relies heavily on passing Java objects across network. This can impose high energy costs for small objects, which will be typical for DRD. The required changes to Jini to make it energy efficient would render it incompatible with the original specification and other technologies built with Jini.

Universal Plug and Play (UPnP) architecture [23] also aims for zero-configuration. However, it has been designed for a different context. In UPnP, components are divided into control points and devices. Each device has a device service. When a device comes on-line, it broadcasts itself to control points on the network. Once a control point has discovered a device, it then takes control of the device and sents control messages to the device's service. UPnP is for a network to discover devices, rather than for a client to discover resources, and does not include any notion of clustering. It is also designed for resource rich devices and its complex architecture is not easy to deploy on resource constrained sensors.

Additionally, many existing resource discovery schemes including Jini [13], UPnP [23], and SDS [4] assume an underlying IP based networking infrastructure. They also rely on support for multicast and assume presence of a reliable transport layer protocol such as TCP. However, the communication models as well as protocol stack architecture for sensor networks is still evolving and it is not clear whether the final model will support an IP based communication infrastructure or employ a complex machinery such as TCP. Therefore we believe that the existing technologies are not directly applicable to sensor networks.

Approaches developed in the context of ubiquitous computing [1] and mobile computing [4] share some of our design goals in terms of ubiquitous interoperation and energy efficiency respectively. However, they do not take advantage of other properties of sensor devices and applications (such as data-centric operation, and different levels and modes of granularity) to realize their systems.

Recently, Stann and Heidemann proposed resource discovery optimizations for sensor networks [20]. In this work, a homogeneous sensor network is assumed (at least, in terms of administration and software). In addition, resource discovery is integrated with the directed diffusion model which is used for data collection [12]; queries are forwarded towards data sources simulataneously discovering and reserving required resources in a position to report required data. The target of this work is to optimize this flooding process by taking advantage of historical queries for similar resources. The scope of our work is wider (not just query forwarding) and is not tied to the directed diffusion model.

7 Concluding Remarks

Resource discovery is an important first step towards enabling interoperability of sensor networks, and eventually seamless integration among them (what we

call *Virtual Sensor Networks*). In this paper, we define the resource discovery problem in sensor networks and outline the challenges involved in it. Sensor networks are unique in several respects that influence resource discovery and necessitate specialized solutions. More specifically, their data-centric embedded nature, relatively poor resources, the emphasis on energy efficiency and the lack of existing standards combine to render traditional resource discovery approaches ineffective.

The paper first explores the space of the resource discovery problem in sensor networks. Resource discovery was organized into two complementary components: the decision on what attributes of the system to track; and the design of energy efficient and available resource discovery protocol. We outlined the challenges involved in each of the subsections. In addition, we demonstrated a solution to one of the challenges (the balance between interoperability and efficiency in the resource discovery protocol) and showed that its possible to improve efficiency while maintaining interoperability.

We are also exploring the concept of Virtual Sensor Networks in the context of second generation sensor networks. More specifically, we are studying various challenges associated with them. To that end, we would like to propose a component-based, modular, efficient service-oriented architecture.

References

1. W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Symposium on Operating Systems Principles*, pages 186–201, 1999.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider.
3. N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, Oct. 2000.
4. S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An architecture for a secure service discovery service. In *Mobile Computing and Networking*, pages 24–35, 1999.
5. Data Format Description Language project web page. `http://forge.gridforum.org/projects/dfdl-wg/`, 2004.
6. E. Christensen et. al. Web Services Description Language (WSDL) 1.1. `http://www.w3.org/TR/wsdl`, 2001.
7. G. Eisenhauer and L. K. Daley. Fast heterogenous binary data interchange. In *Proceedings of the Heterogeneous Computing Workshop (HCW2000)*, 2000.
8. R. Fikes, P. Hayes, and I. Horrocks. OWL-QL: A language for deductive query answering on the semantic web. `ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-03-14.pdf.gz`, 2003. KSL Technical Report 03-14.
9. A. Gupta et al. Registering Scientific Information Sources for Semantic Mediation. In *21st International Conference on Conceptual Modeling*, 2002.
10. M. Harcol-Balter, P. Leighton, and D. Lewin. Resource discovery in distributed networks. In *Proc. of ACM PODS 1999*, pages 229–237, 1999.
11. W. Heinzelman. *Application-Specific Protocol Architectures for Wireless Networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

12. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. 6th ACM International Conference on Mobile Computing and Networking (Mobicom'00)*, Aug. 2000.
13. Sun microsystems. jini technology architectural overview (white paper). `http://www.sun.com/software/jini/whitepapers/architecture.html`.
14. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. Rql: a declarative query language for rdf. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 592–603, New York, NY, USA, 2002. ACM Press.
15. P. Mockapetris and K. J. Dunlap. Development of the domain name system. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 123–133, New York, NY, USA, 1988. ACM Press.
16. Network Simulator. http://isi.edu/nsnam/ns.
17. C. Perkins and H. Harjono. Resource discovery protocol for mobile computing. *Mobile Networks Journal*, 1(4):447–455, 1996.
18. A. Seaborne. Rdql - a query language for rdf. `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`, 2004.
19. Sqlite. `http://sqlite.org/`.
20. F. Stann and J. Heidemann. BARD: Bayesian-assisted resource discovery in sensor networks. In *Proceedings of the IEEE Infocom*, 2005.
21. Tinydb: In-network query processing in tinyos. `http://telegraph.cs.berkeley.edu/tinydb/doc/index.html`.
22. Tinyxml. `http://sourceforge.net/projects/tinyxml/`.
23. Universal plug and play device architecture. `http://www.upnp.org/download/UPnPDA10_20000613.htm`.
24. D. Veillard. Libxml2 project web page. `http://xmlsoft.org/`, 2004.
25. J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Service location protocol, 1997.
26. W3C. Simple Object Access Protocol (SOAP) 1.1. `http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`, 2000.
27. W3C. Resource description framework (rdf). `http://www.w3.org/RDF/`, 2004.
28. Xerces: Xml parsers. `http://xml.apache.org/#xerces`.
29. Xml pull parser. `http://www.extreme.indiana.edu/xgws/xsoap/xpp/`.
30. Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM Press, 2001.