The Architecture of a Generic Application Environment for Wireless Sensor Networks Accessible via Internet *

Ioannis Chatzigiannakis, Georgios Mylonas† and Sotiris Nikoletseas
Department of Computer Engineering and Informatics, University of Patras,
and Computer Technology Institute (CTI), 26500, Patras, Greece
{ichatz, mylonasg, nikole}@cti.gr

Abstract

In this paper we present the design of jWebDust, a generic and modular application environment for developing and managing applications based on wireless sensor networks that are accessible via the internet. Our software architecture provides a range of services that allow to create customized web-based applications with minimum implementation effort that are easy to administrate. We here present its open architecture, the most important design decisions, and discuss its distinct features and functionalities. jWebDust allows heterogeneous components to interoperate and the integrated management and control of multiple such networks by defining web-based mechanisms to visualize the network state, the results of queries, and a means to inject queries in the network.

1 Introduction

Wireless sensor networks are very large collections of small in size, low-power, low-cost sensor devices that collect and disseminate quite detailed information about the physical environment. Large numbers of sensors can be deployed in areas of interest and use self-organization and collaborative methods to form a sensor network. The flexibility, fault tolerance, high sensing fidelity, low-cost and rapid deployment characteristics of sensor networks help to create many new and exciting application areas.

This wide range of applications is based on the use of various sensor types (i.e. thermal, acoustic, magnetic, etc.) in order to monitor a wide variety of conditions (e.g. temperature, object tracking, humidity) and report them to a (fixed or mobile) control center. Thus, sensor networks can be used for important applications, including (a) military (like forces and equipment monitoring, battlefield surveillance, targeting, nuclear, biological and chemical attack detection), (b) environmental applications (such as fire detection, flood detection, precision agriculture), (c) health applications (like telemonitoring of human physiological data) and (d) home applications (e.g. smart environments). For an excellent survey of wireless sensor networks see [1].

A possible categorization of the applications for these networks is based on the notification strategy of the authorities, i.e. the way that the authorities are updated on the monitoring state. For example, in a museum, it is important to report only when emergency situations arise, whereas in habitat monitoring, continuous monitoring of the physical environment is required. Therefore, depending on the actual application, the following services are required: (i) Periodic Sensing (the sensor devices constantly monitor the physical environment and periodically report their measurements to a control center), (ii) Event driven (sensor devices monitor the environment and send reports only when certain events are realized) and (iii) Query based (sensor devices respond to queries made by a supervising control center).

In light of the above categories of applications and services, we present the design of jWebDust, a software environment that allows the implementation of customized applications for wireless sensor networks that can (i) provide a wide range of services, (ii) minimize the overall implementation effort and (iii) considerably reduce the needs for network administration. jWebDust is modular and extendable as the final user can select a set of features, modify some of them and provide new ones that best suit his needs; in this sense, jWebDust is able to deal with several kinds of applications (size, functionality, etc.).

jWebDust differentiates the system into two main groups: the networked sensor devices (from now on referred to as motes) that operate using TinyOS and the rest of the network (e.g. control centers, database server, etc.) that is

---

* This work has been partially supported by the IST Programme of the European Union under contract number IST-2005-15964 (AEOLUS), the PYTHAGORAS Programme under the European Social Fund (ESF) and Operational Program for Educational and Vocational Training II (EPEAEK II) and the ALGODES PENED Programme of GSRT under contract number 03ED568.

† Currently a PhD student.
all the necessary tools and operations to allow the implementation of a wide range of applications are few.

TinyDB [10] is an application that allows multiple concurrent queries, event-based queries and time synchronization through an extensible framework that supports adding new sensor types and event types. The central idea of TinyDB is to provide an SQL-like interface to the programmer that makes the sensor network look like a DBMS. A tree based routing scheme is used for multi-hop communication inside the network. In addition, TASK [9] is built on top of TinyDB in order to further simplify application deployment and development. In contrast to the database schema used in TinyDB, our approach is more detailed and can be easily extended to the final application’s functionality needs.

Mote-VIEW [8] is an application that provides tools to the user to visualize results from a sensor network, combined with a data logger that runs on the sensor network gateway. Readings arriving from the network are stored in a relational database. Motes in the network poll their sensors at a sampling rate specified by the user and send them to the gateway using a multi-hop protocol. The user can check readings from the motes’ sensors on the fly, see a visualization of the topology, produce graphs from selected motes’ readings, and check their status. ScatterViewer [12] is an application designed for the ScatterWeb sensor platform, with characteristics similar to MoteView, plus the useful capability of replacing the motes firmware over the air and the capability to run on an embedded platform.

Microsoft’s MSR Sense [11] is a software suite (under development) that helps users to collect, process, archive, and visualize data from a sensor network. These data come from applications running on the motes, that are independent of MSR Sense.

A preliminary version of this paper has appeared as a short paper in [6].

2 The Architecture of jWebDust

jWebDust is designed on a component-based architecture with several and diverse design goals as a guide that emphasizes on autonomy, reliability, and availability. At a high-level, the components of jWebDust are organized, using the N-tier application model, as follows: (i) the Sensor Tier that consists of one or more wireless sensor networks deployed to areas of interest, (ii) the Control Tier that corresponds to the control centers where the wireless sensor networks report the realization of events, (iii) the Data Tier responsible for storing the information extracted from the wireless sensor network(s), (iv) the Middle Tier that is responsible for processing the data to generate statistics and other meaningful information and (v) the Presentation Tier that interfaces the information with the final user in an easy way based on the capabilities of the user’s machine. The
five tiers that make up jWebDust are shown in Fig. 2.

The Sensor Tier. Naturally, the sensor tier is the foundation of any application based on wireless sensor networks. The motes are usually scattered in the area of interest and form one or more sensor networks. Each of these scattered motes has the capability to collect data and route data back to the control center and the end users. Data are routed to the control center by a multi-hop architecture and then the control center communicates with the other tiers of the system possibly via the internet. Each mote will execute software based on the TinyOS operating system, use a mote discovery protocol for reporting its sensing and processing characteristics to the control center and a query dissemination protocol for distributing queries in the sensor network and disseminating the data matching these queries back to the control center (see Sec. 3).

The Control Tier. The Control Tier consists of the control centers of each sensor network. Control centers are responsible for the gathering of all the readings coming from the sensor networks and the forwarding of the queries from the data tier to the motes; in other words, they act as gateways between the sensor tier and the data tier. On hardware level, a typical control center consists of one mote connected to a desktop PC or laptop that keeps a network connection to a database server. The mote attached to the control tier is necessary to communicate with the sensor network. Alternatively, an embedded platform can be used. One important feature of the control tier is the ability to operate even when there is no connection to the data tier for sustained periods of time. The importance of this feature is outlined by the fact that control centers themselves may have a wireless connection to the data tier and uninterrupted communication is not guaranteed. During such a disconnected period, any new queries made will not be forwarded to the sensor network (since the control center cannot be informed about these queries), while sensor readings received will not be sent to the data tier but will be stored locally. As soon as the control center establishes a connection with the data tier, all locally stored readings are forwarded to the data tier and new queries are forwarded to the network.

Another feature of jWebDust is the support of multiple sensor networks in a way that they are seen as a single virtual sensor network. For each sensor network a unique ID is given to its respective control center. This sensor network ID helps to distinguish one mote from another, when they have the same mote ID but belong to different sensor networks, thus making it possible to manage different networks in a unified way.

The Data Tier. The components at the data tier are based on the relational database system and the functionality and methods provided are related to the services required by the middle tier and the control tier. Our database schema consists of ten tables that can be organized in three categories:

(i) Mote related tables. The information related to the hardware characteristics of the motes that comprise the wireless sensor networks are organized in this category with the use of five tables. The database scheme supports heterogeneous sensor networks, i.e. networks that consist of different kinds of motes (e.g. Telos, Micaz motes), which may have different kinds of sensors attached to them (e.g. light, humidity, etc.).

(ii) Query related tables. This group of tables holds information regarding the queries that are made on the network that may possibly address multiple sensors and/or multiple sensor types (see Sec. 3).

(iii) Sensor readings table. All the information received from the sensor networks that match a specific query is stored in this table. Each record represents a reading coming from a specific mote and a single sensor.

The Middle Tier. The Middle tier is comprised by all the components that make up the jWebDust logic and are responsible for delivering structures and data to the presentation tier. The components of the middle tier can be considered as applications that run on a server without a face (also referred to as servlets). These components have autotelic functionality that is executed independently in order to process certain data structures and produce specified output. In this extent, components contribute to a simpler distribution of workload and allow easy development of the presentation tier services. In jWebDust the components of the presentation tier and the middle tier can operate simultaneously and independently. Components are formulated based on the data structures available and the operations required by each entity. For example, the management of queries is handled by a single component and the methods implemented perform actions such as createQuery, deleteQuery, updateQuery, etc. All these methods accept parameters given by the presentation tier. The logic
of the creation of a new query (i.e. check if a similar query already exists or if the mote provides the sensors specified in the query) is standardized through a single component. Such components are also referred to as the executants. The executants provide a specified interface to the other components; thus if modifications to jWebDust logic are carried out, the implementation of this interface can be modified. As long as the interface remains unaffected, components that use the executants will not notice any change and will not require any further change.

The Presentation Tier. The presentation tier is basically the user interface; it is the layer most available to the end users, responsible for the collection of input and presentation of the information collected from the sensor network. The components that make up the presentation tier are based on different technologies. The terms thin client and rich client are used to describe the capabilities of the presentation tier given the available resources. We characterize as rich clients the components of the presentation tier that are rich of functionality. The support of thinner clients will ensure the interoperability of the system through the various available machine architectures. The end users are allowed to choose among the available client solution in order to maximize the available resources. The open architecture of jWebDust system allows new components to be introduced at later stages that will cope with these issues.

3 Sensor Network Services and Protocols

Mote Discovery Service. One of jWebDust’s features is that every mote is able to register itself and its distinct features to the system, giving a clear view of the network and removing the need to initiate a mote discovery process periodically. The mote discovery service provides the user with a detailed view of the properties of each mote inside the (possibly heterogeneous) sensor network. Other existing applications do not provide such detailed information and thus do not cope well with the case of having motes with different sensors attached to them. As an example, consider the case where we want to have motes collecting temperature readings on the one side of the sensor network and on the other side motes with a different kind of sensor board collecting humidity readings. The use of different kinds of sensor boards could also be justified in terms of cost or because we want to take advantage of older equipment. The operation of the discovery service relies on the correct programming of the motes: when installing the application firmware on the mote, the actual hardware characteristics are passed as parameters to the service. Based on this information, the discovery and registration of every mote in the network is achieved using a simple protocol.

When a mote is powered on, and after its startup and initialization phase completes, it sends out to the control center a short message containing its ID, type and a list of available sensors. By sending this message, the mote registers itself in jWebDust’s data tier along with an amount of useful information when the control center receives the message.

Sensor Network Communication Protocol. Communication in wireless sensor networks presents many distinctive characteristics, which render the conventional and most ad-hoc routing approaches inadequate for use in these networks. Many routing schemes have been proposed specifically for sensor networks [3, 5], however only a few of them have been implemented in TinyOS. Routing protocols for TinyOS can be divided into two categories, based on the number of possible routing destinations.

We plan on using a communication protocol based on the multihop tree-based routing protocol included in the TinyOS distribution, called MultiHopRouter, with an extension that aims towards reducing the energy dissipation of the motes. More specifically, we propose an extension that uses a mechanism for varying the transmission range [2] on top of MultiHopRouter. The implementation of such a mechanism is based on the ability of TinyOS to adjust the transmission range of the motes.

The main idea is to periodically check whether a satisfactory number of nearby motes is active. This is done by periodically broadcasting “hello” messages from the control center and flooding the network. If motes maintain network connectivity, this message will reach all the motes of the network, at a certain time point. The concept of a varying transmission range protocol (VTRP) and a study of ways of modifying the transmission range are described in [2].

Sensor Query and Data Dissemination Protocol. The queries supported in jWebDust are categorized using two criteria; (i) the motes they are targeted to, and, (ii) the way information regarding the queries are reported to the control center. The first category includes mote-specific and attribute-based queries, while the second category includes periodic sensing, event driven and query based queries. These two categories are overlapping and the actual sensor queries are combinations of these categories.

In the case of a Mote-based query, we are interested in targeting specific motes by using their TID; the protocol will send one packet per each mote included in the query. On the other hand, attribute-based queries are not targeted to specific sensors, but to the whole sensor network; the protocol will flood the network with the query and only motes matching the specific attribute will respond to the query.

Periodic-update queries pose another time constraint along with start and stop time constraints, regarding the time of reporting to the sink. They also specify an interval time period between successive reports. Event-driven queries use the notion of events in order to define the time to report back to the sink.

Examples of possible queries to a sensor network in-
clude “give me the temperature and humidity readings from motes where light reading is over 200 starting from 10:15 till 13:30” and “give me the light readings from mote 4 when temperature reaches $30^\circ$C”. Thus, we have four types of possible queries, (i) attribute-based periodic update, (ii) attribute-based event-driven, (iii) mote-specific periodic update, and (iv) mote-specific event-driven queries.

Finally, the protocol supports the possibility of “canceling” a query that is currently active. The user can send a special cancel-query message that contains one or more IDs referring to the queries that are to be cancelled. Note that if the query that needs to be cancelled is mote-based, a message is sent directly to the mote involved; if it is attribute-based the message needs to flood the network.

4 Concluding Remarks

We have presented the architecture and the design of the main components of jWebDust, and discussed several distinct features along with their implementation and functionality. The main strengths of jWebDust are:

1. Provides an environment to package and manage the plethora of lower level protocols with minimum implementation and administration effort.

2. Allows to implement new functionality that can be easily integrated with the rest of the architecture at all levels of the hierarchy (i.e. sensor level, control centers level, database level, user-interface level) to best suit the application needs.

3. Supports multiple sensor networks (i.e. physically separated) with multiple/separate control centers and allows to handle them as a single virtual sensor network, even if more than one sensor share the same ID.

4. Handles sensor networks comprised of devices with heterogeneous characteristics. Real world sensor networks will rarely be homogeneous.

5. Supports Disconnected/Mobile Control Centers by taking special care of long disconnections of the control centers, and the sensor networks attached to these centers, from the upper parts of the architecture (i.e. database, web, etc.), so that information is not lost.

6. Many users can simultaneously query, monitor, and visualize the execution of the wireless sensor network through a Web-based user interface.

References


