A PEER-TO-PEER ENVIRONMENT FOR MONITORING MULTIPLE WIRELESS SENSOR NETWORKS

A. Antoniou, I. Chatzigiannakis, A. Kinalis, G. Mylonas, S. Nikoletseas, A. Papageorgiou∗

Computer Engineering and Informatics Department, University of Patras, and Research Academic Computer Technology Institute, Rio, Patras, 26500, Greece

{antonioa, ichatz, kinalis, mylonasg, nikole}@cti.gr, papagap@ceid.upatras.gr

Abstract In this work we present the basic concepts in the architecture of a peer-to-peer environment for monitoring multiple wireless sensor networks, called ShareSense. ShareSense, which is currently under development, uses JXTA as a peer-to-peer substrate. We demonstrate its basic functionalities using a simple application scenario, which utilizes multiple disparate wireless sensor networks. This application scenario involves monitoring of such networks using a separate management environment and a custom application GUI, as well as using Google Earth as an additional user interface.

Keywords: Wireless sensor networks, peer-to-peer, JXTA, management, application, design, architecture

1. Introduction

Systems and applications that deal with data related to the physical world have existed since the very early days of computing. However, only recently systems capable of retrieving and delivering the physical data to great distances, with good precision, and without the need for human participation have become widely available, thanks to sensor networking techniques and the development of sensor devices that are small, cheap and energy-efficient and can collect, process and disseminate data efficiently. A vision has risen of monitoring the physical world at an unforeseen scale and applying it to many application domains, such as

∗Authors’ names are listed in alphabetical order
environmental monitoring, intelligent buildings, health care, road-traffic control, logistics, precision agriculture, etc.

Research related to Wireless Sensor Networks (WSN) ([1] is a good general source for the work and research issues to date) has focused mainly on what could be characterized as “the lower level” of such systems, i.e., sensor-device deployment techniques, sensor-device communication protocols, energy-efficient algorithms, etc. Still, the fact remains that most of these protocols and architectures assume single, isolated, wireless sensor networks, dedicated to the monitoring of a specific network field. In addition to traditional network models and architectures, a network model that deals with a lot of small interconnected WSNs (e.g., with the use of sensors integrated on mobile phones and every mobile phone functioning as a base station or with the interconnection of already existing WSNs) is also useful, apart from large WSNs with thousands of nodes and complex routing schemes.

Building software infrastructures for the interconnection of disparate sensor networks requires effort and careful design. This fact, along with the anticipation that the amount of WSN applications will increase dramatically, makes the development of relevant overlay software critical. This overlay software should handle issues that are common to relevant applications, and should also be general-purpose and reusable, simplifying the development of applications for sensor networks. Some of the challenges such a software platform is called to face are the following:

- handle WSN heterogeneity,
- handle mobility and intermittent connectivity,
- support scalable systems,
- simplify the development of web-based applications.

ShareSense, presented here, attempts to be such a general, extensible and easy to handle overlay software solution for WSN. ShareSense unifies many WSNs in a peer-to-peer fashion, with all the users of the system, and provides an API so that the user can issue queries to the network without having to worry about the underlying details. Literally, *sharesense* means the ability to project one’s own sensory impressions by telepathetic means into another’s mind, so we think it is a suitable name for our environment.

ShareSense utilizes JXTA [17] as a peer-to-peer substrate. In short, JXTA is a set of protocols and specifications for the development of peer-to-peer systems. There are many implementations of the JXTA platform available, each of which follows certain specifications, so that
applications built on any of them can intercommunicate. This offers the opportunity of OS-independent and language-independent communication between the peers. Also, it is important that we mention the micro edition of JXTA, which gives to smaller (and therefore resource-constrained) devices the opportunity to host JXTA peers.

ShareSense is currently under development, so the purpose of this demo basically is to show the main concepts that lie behind its architecture and to demonstrate part of its core functionality.

2. Related Work – Our Contribution

There has been a lot of research in WSN management environments the last few years. [2] and [3] are sources providing references to related solutions proposed thus far. Generally, there is a number of approaches quite similar to our own available, regarding the multitude of sensor networks and large-scale system size. However, not all of the software solutions presented thus far deal with exactly the same issues, i.e., their respective application scenarios and target domains differ.

Most of these environments follow and N-tier architecture (e.g. [5], [6]), while others, more recent ones, extend this concept by using a peer-to-peer logic, to variable degrees, while their orientation is also different. IrisNet [9] was probably the first proposal for dealing with very scale sensor networks. CarTel [11] emphasizes more on coping with mobility at high speeds, while MetroSense [8] focuses on node intercommunication techniques, and Skylark [13] aims to integrate sensor networking in the mobile phone world. P2PBridge [12] and Agimone [10] focus more on the concept of inter-WSN communication and interoperability. Hourglass [16] is another peer-to-peer environment, which has a number of goals that are most similar to the goals of ShareSense. There is also the OGC Sensor Web Enablement [14], which does not use peer-to-peer techniques but aims to supporting global-scale sensor applications.

Regarding their presentation layer, some of these environments use custom application GUIs, others are web-based, and others use tools like Virtual Earth [15] or Google Maps to present a more geographically-centered interface, or use a combination of these methods.

Our contribution: We present an environment for monitoring multiple wireless sensor networks which uses a peer-to-peer approach for communicating between a number of disparate such networks. We use JXTA as our peer-to-peer substrate, and build on its functionality to provide users with an environment that takes care of connectivity and interoperability issues. Currently, we use jWebdust as a platform for the management of the underlying WSNs and build on its features to support our system
with real sensor network data. We also use a Google Earth-based inter-
face, apart from a custom GUI, to visualize results coming from the
peers participating in the system, which is a feature that sets it apart
from most other similar environments.

3. The Architecture of ShareSense

In this section, we provide a brief description of the architecture we
use in ShareSense, giving insight to its overall functionality, the types of
peers participating in the system and the functionalities associated with
each such peer type.

Overall functionality: Concerning the peers participating in the sys-
tem, for the moment it should be kept in mind that each WSN base-
station (gateway) is a peer and each user is a peer, too. The main goal
is to treat the underlying WSNs as a single, unified network, to which
users can issue queries without having to define details related to the
network structure, topology and architecture.

Each application based on ShareSense will be able to issue queries
whenever it wishes and in many different ways. These are the applica-
tion queries. ShareSense analyzes the application query and splits it to
smaller queries that are sent to the appropriate gateways. The query
analysis and splitting procedure takes place on the users side so that
no useless data is sent through the network. Only queries that can be
answered are sent to the gateways that can answer them. This central
functionality of the overlay is shown between the dashed lines of fig. 1.

ShareSense Peers: There are three different peer types and each of
them has different software from the other peer types:

- **Routing Peers** perform the functionality of the JXTA rendezvous
  and relay peers and they track the connection status of the others.

- **Gateway Peers** are usually hosted on the base-stations of the dis-
  parate WSNs. They could also be hosted on workstations that
  have remote access to the information stored in the base-station.
  The base-stations have knowledge of all the details of the sensors
  they control and they can communicate with them.

- **User Peers** are hosted on the machines of the users of applications
  that are based on ShareSense i.e., wherever the application is in-
  stalled. Still, the application software should not be confused with
  the users ShareSense software. They co-operate but are separate.

Routing Peers: The routing peers run simple software and should be
hosted on machines with IP addresses that are known to the other peers.
They are programmed to act as JXTA rendezvous and relays. This means that they are responsible for routing JXTA messages etc., but this is hidden by the JXTA overlay and the interested reader should resort to the JXTA bibliography in order to learn how this is implemented. Apart from that, the ShareSense routing peers keep track of the connection status of the other peers.

**Gateway Peers**: Each of these peers should maintain a periodically updated description of the WSN it controls. These are XML-based descriptions that contain all the information that the user needs in order to be able to issue queries and handle the data in an appropriate manner. For example, info about the locations and the capabilities of the sensors are contained. In addition, there is info about the supported data types, so that heterogeneity is supported. The format of a gateway’s advertisement can be seen in figure 2.

The peer’s functionality is, in short, as follows:

- Configures its parameters in the JXTA network.
- Starts the JXTA platform and goes on only when a connection to any routing peer is present.
Figure 2. Example of a gateway peer’s capabilities advertisement

- Finds or creates and then joins the Group of the “global” sensor network, publishing its specific peer information as well as the XML description of capabilities available in the WSN(s) it controls.
- Registers the appropriate handler that will handle any queries received.
- Sleeps, periodically searching for appropriate messages, which will trigger the start of a service, normally in order to serve a query.

**User Peers:** The main functionality concerning the configuration and the connection of the peer to the network is similar to this of a gateway. What makes the difference here is the existence of the Query Processor and the Data Type Adapter and the API provided to the application programmer. What the Query Processor does is the query analysis, splitting and submission, described in section 3.1. In short, the user peer:

- Configures its parameters in the JXTA network.
- Starts the JXTA platform and goes on only when a connection to any routing peer is present.
A Peer-to-Peer Environment for Monitoring Multiple WSNs

- Finds and joins the Group of the global sensor network.
- Finds the descriptions of all of the gateways and extracts from them all the necessary info. The info is stored so that the Query Processor can later perform the analysis, splitting and submission of the queries correctly and efficiently.
- Registers the appropriate handler that will handle the submission of queries.
- Sleeps, waiting for its API to be called by the application (or by the Query Processor on demand of the application).

Its API is closely related to the two main services offered (see next section).

4. Services Provided by ShareSense

Although ShareSense’s API can be extended, modified and used in many ways, the main services offered are two: the Resource Discovery Service and the Query Handling Service. The latter can be divided in two subservices, one handling single data and the other handling binary data. The services should be defined independently from the peers as they are not totally implemented on a single peer type and their correct functionality relies on the co-operation of the overlay software of all peers.

The Resource Discovery Service: This service is based on the mutually-agreed-upon description schema that was presented earlier. Since each WSN is described by a valid description as such, the rest is easy. Each user “discovers” such published descriptions before proceeding to issue any queries and stores the necessary info. As an enhancement to the ShareSense overlay, this info could be stored in databases of appropriate schemas, so that they can support larger amount of info and are stored permanently. Still, it is important that this info is frequently refreshed, so this justifies the current decision, i.e., that the resources should be discovered each time an application starts. In addition, the API of the Service gives the user the ability to refresh the information easily any time it is considered necessary, with a single method call. The Query Handling Service is based on the Resource Discovery Service in the sense that it uses the info about the resources in order to generate and submit queries. So, the latter should have done its work at least once before any use of the Query Handling Service is possible.

The Query Handling Service: Thanks to the info about the network structure, the overlay software can efficiently split the application
queries and communicate with the appropriate peers in order to receive responses. Still remains to be clarified how the communication is implemented and which is the role of each of the two subservices of the Query Handling Service.

Among others, the users overlay software has also knowledge of the data type that the response to any of its queries will contain. So, it divides the queries into two categories: those that will be answered with single data and those that will be answered with binary data. This decision has two main reasons. Firstly, binary data is often larger (images etc.) and should be handled in a different manner. Secondly, JXTA offers an efficient way to exchange single data by means of a core service that uses XML-based messages and this feature should be utilized when possible. The implementations of the two subservices have significant differences.

The binary data subservice is based on the JXTA Pipe Service and it has two main advantages:

- It is the only solution when the amount of data to be exchanged is significantly large.
- It is the best approach in order to handle intermittent connectivity because the queries remain on the network for as long as we wish and the interested party can find them and answer them whenever its connection is restored.

Both of these features justify the decision to use this subservice for binary data. The single data subservice is based on the JXTA Resolver Service and it has the following advantages:

- Its implementation is faster and more efficient as the interested peers dont have to discover the queries but they are instantly informed about them, as opposed to the JXTA Pipe Service, where a peer should periodically poll its Routing Peers to discover pending requests for connection and data transfers.
- It can be extended in order to support queries with multiple receivers, which could be an advantage for systems with replicated data.

5. Testbed Application and Presentation

Our testbed consists of a number of disparate wireless sensor networks, some of which are located in the campus of the University of Patras, Greece, and a few others located in the premises of a number
of other European research teams (actual testbeds will be complete by the time the demo will take place). Regarding the sensor network nodes used, we use a variety of nodes including mica, mica2, mica2, TelosB sensor network nodes, along with a variety of respective attached sensor boards. This variety of hardware used in the demo supports our claims for heterogeneity throughout our system. The testbed, depending on the nodes and sensor boards used, supports taking measurements from light, temperature, humidity, and other sensors.

For the purpose of supplying ShareSense with real sensor network data, we use jWebDust. jWebDust [5] allows the integrated management and control of multiple wireless sensor networks and also defines web-based mechanisms to visualize the network state, the results of queries, and the means to inject queries in a wireless sensor network. It defines a multi-tier architecture to provide rich functionality at different levels. On the lower tiers (sensor, control and data tiers), jWebDust uses the TinyOS operating system to handle communication issues on the wireless sensor network, defines services for controlling the managed WSNs and provides data persistence by storing gathered information in a database. Except for the sensor tier, jWebDust is written in Java. We use a number of jWebDust installations to manage each of the WSNs that comprise the demo’s testbed.

ShareSense “sits” on top of the jWebDust platform and uses the services defined in lower tiers in order to construct a list of the sensor capabilities and extract data from the underlying WSNs. Using jWebDust’s API it directly pumps data from the data tier and exports this data in an XML based form, as described in earlier sections. We have built a simple Java application, which uses the ShareSense client component to connect to the peer-to-peer network that disseminates the sensor information. The application provides to the end user a GUI to pose customizable queries and to display the results. The queries are oriented towards geographical criteria thus allowing independence of the underlying network structure. A screenshot of this application can be seen in figure 3.

Besides the actual sensed data, geolocation information that correlates that data to the geographical position of the deployed sensors as well as 3D models of the sensor network deploy areas are exported. Sensor network data in most cases is strongly related to geolocation information, so it makes sense to use an application that is based in such information to present results from the sensor network testbed. For this purpose, we chose to use Google Earth, which provides advanced visualization capabilities. Each peer of the system can provide along with its geolocation data a three-dimensional description of its surroundings (in the
Figure 3. A view of the test-bed applications Query Wizard

form of KML files). These descriptions are processed by a special peer paired with an application server, which gathers sensor data produced by the system peers and produces a file containing data in a form which Google Earth understands. The user can define what kind of sensor data she wishes through a simple interface. Finally, simply by pointing the GoogleEarth client to the appropriate URL, the end user can access a rich visualization of the collected data and of the deployment area. In figure 4 a screenshot of a simple representation of a number of offices along with their respective temperature and light sensor readings can be seen.

6. Conclusions and Future Work

ShareSense provides a peer-to-peer environment to work with multiple wireless (or even wired) sensor networks, and uses JXTA to implement peer-to-peer communication. Its architecture allows desirable features, such as easy integration with existing lower-level (single or multiple) WSN management solutions, independence regarding operating system and hardware, handling of intermittent connectivity, scales well, etc. We presented here a small application scenario to demonstrate the basic concepts in its architecture and functionality. Moreover, apart from using a more traditional application GUI, we use a Google Earth-based interface to demonstrate the potential of our approach.

Future work on ShareSense, among others, includes the following:
Enhancements and extensions to the software (e.g., testing and comparing possible peer communication solutions and selecting the most reliable and effective ones, experimenting with more networks and nodes, extending the API, etc.).

Combining our software with various lower-level overlays.

Deployment of actual applications based on the software developed.

Acknowledgements

This work has been partially supported by the IST Programme of the European Union under contract number IST-2005-15964 (AEOLUS). Also, by the Programme PENED under contract number 03ED568, co-funded 75% by European Union – European Social Fund (ESF), 25% by Greek Government – Ministry of Development – General Secretariat of Research and Technology (GSRT), and by Private Sector, under Measure 8.3 of O.P. Competitiveness – 3rd Community Support Framework (CSF).

References


