A Middleware-Based Approach for Heterogeneous Wireless Sensor Networks

F. GRAZIOSI, L. POMANTE, D. PACIFICO
Center of Excellence DEWS
University of L’Aquila
Piazza E. Pontieri, 67100 Monteluco di Roio (AQ)
ITALY

Abstract: - Wireless sensor networks (WSN) are networked embedded systems that use small size devices with some processing and short-range radio communication capabilities for monitoring and controlling systems and environments. However, languages and IDE (Integrated Development Environments) used to develop applications for WSN are often strictly related to the involved HW/SW platform leading to complex activities, especially when dealing with heterogeneous WSN composed of nodes with different hw features, operating systems, communication standards, etc...

A middleware (MW) is a SW abstraction layer between operating systems and applications that simplify operations, enable heterogeneity and masks the basic HW/SW layers (of remote or local devices). Finally, a middleware concretely helps the designer to develop applications, and this is especially true in the heterogeneous WSN world. So, this work presents a middleware-based approach for WSN that copes with radio heterogeneity, i.e. it is able to transport data across heterogeneous networks (i.e. based on different communication standard) while offering to the designer an homogenous API (Application Programming Interface) for the applications development.

Key-Words: - Wireless Sensor Networks, Middleware, Heterogeneous Networks

1 Introduction

Wireless sensor networks represent one of the latest research areas about the (networked) embedded systems. These networks are constituted of small devices designed to interact the environment and to communicate to each other without cables. The deployment of these networks in the target environment should be maximally simplified. The great variety of potential applications of wireless sensor networks promotes their use in several areas [1][2][3]:

- **Environment** - There are a lot of applications that are dedicated to the monitoring of terrestrial animals, birds, marine species and for the environmental monitoring and precision agriculture;

- **Military** - Wireless sensor networks could be used by military control systems, military communication systems, elaboration systems, processing, intelligence, surveillance, recognition and identification;

- **Health** - Some example of application are: interfaces for the disabled, integrated monitoring of patients, remote monitoring of personal physiological data, administering drugs in hospitals;

- **Domestic Use** - The sensor nodes can be put into appliances, such as microwave ovens, refrigerators; these devices can interact with each other and with an external network, via Internet or satellite networks and allow the user to easily remotely control appliances activity;

- **Commercial Application** - Other possible applications include: virtual keyboards; behavior analysis of material under mechanical stress; robots control, etc…

However, despite to the increasing proliferation of WSN applications, languages, tools and IDE (Integrated development Environments) used to develop them are still very similar to the ones used in the classical embedded world, i.e. they are too often strictly related to the involved HW/SW platform leading the designer to complex design and development activities especially when dealing with heterogeneous WSN (i.e. composed
of nodes with different HW features, microcontrollers, operating systems, communication standards, etc…). Moreover, it is worth noting that WSN typically act as embedded distributed systems so inheriting part of the design issues involved in the distributed systems domain! Fortunately, and particularly in the recent research panorama, something is changing and the password of such a change is Middleware.

In such a context, this work presents a middleware-based approach for a WSN that copes with radio heterogeneity, i.e. it is able to transport data across heterogeneous networks (based on different communication standard) while offering to the designer an homogenous API for the applications development. By means of such a MW is so possible to transparently use different type of nodes in the network (including a proper gateway node when needed).

This feature could be useful, for example, when it is needed to cover large regions with different propagation characteristics (Figure 1). It could be impossible to find a radio interface suitable for all the situations while with different types of nodes we could be able to use a proper radio interface according to specific link properties.

The next Section discusses some issues related to MW focusing on WSN. Section 3 describes the work that has been taken as a starting point for the development of the presented MW while Section 4 describes with some details the proposed extensions. Finally, Section 5 shows a simple application of the proposed MW while Section 6 draws out some conclusive considerations about the presented work and the related ongoing activities.

2 Middleware for WSN

Design and management of distributed systems can be an hard task for developers because of their very heterogeneous and dynamic structure. A middleware approach could significantly simplify this task and it has been followed very often in the past. The main features of a middleware has been generally [4][5] the following:

- **Marshalling/Unmarshalling**: data should be formatted to ensure hardware and software independence from hosts;
- **Data Representation and Codification**: the middleware protocol should define all type of data that the applications can manage;
- **Remote Invocation**: modern middleware should allow remote processing and method invocation;
- **OSI protocol compliant**: encoding and data preparation should not interfere with middleware protocol;
- **Location transparency**: the application should be able to communicate with all the devices composing the distributed systems by using the middleware capabilities;
- **Quality of services**: middleware is also suited to offer QoS to the application layer.

Another interesting example (Figure 2) is the case of the deployment of a very dense WSN where the concurrent use of different radio technologies could be the only way to avoid medium access conflicts and expensive retransmissions.
However, focusing on WSN the scenario is quite different. In fact, although middleware is well established research area in distributed computing systems, WSN poses new challenges because the traditional middleware techniques cannot be applied directly to WSN. In fact, most distributed system middlewares \cite{7}\cite{8} aim to hide the context, but WSN-based applications should usually be context-aware. Moreover, although many mobile computing middleware \cite{9}\cite{10} supports context-awareness, their major concern is how to satisfy the interests of single nodes. In contrast, WSN-based systems are data centric, reflecting the application goals. Also, WSN often use attribute-based addressing \cite{7}-\cite{13} rather than relying on network-wide unique node addresses. Again, data aggregation in intermediate nodes of the forwarding path is desirable in WSN, but no such kind of support is provided in traditional middleware. Finally, WSN requires the middleware to be lightweight for implementation in sensor nodes with limited processing and energy resources. In recent years, lot of work has been done on WSN middleware and several survey papers can be found in literature (e.g. \cite{14}-\cite{20}).

Although such a research area is actually very active, there are not works that currently copes with the radio heterogeneity issue. Therefore, this work presents a middleware-based approach for WSN able to manage such an heterogeneity, i.e. it is able to route data packets across heterogeneous networks, using different communication standards, while offering to the designer an homogenous API for the applications development. The presented approach is based on an existing transport layer for WSN called Stream System (created in the context of the MaD-Wise project \cite{6}) that, for its native characteristics (described with some detail in the following), could be seen, for our purposes, as a sort of MW for WSN.

3 Stream System

Typical applications of WSNs periodically request data from nodes to monitor real phenomena. These information are sent to a sink node that is usually connected to a personal computer by different interfaces (e.g. cable, radio link, satellite). Stream system \cite{23} proposes an abstraction of collection and transmission data at the application layer, this means that there is an unidirectional stream channel by which it is possible to transmit information.

Stream System offers three types of streams (Figure 3): sensor stream, local stream e remote stream. Sensor stream is an abstraction that acquires data from transducers, these can be only read by the application because writing operation is done only from the associated transducers. Local stream is a local data channel, writing and reading operations occur on node in which it was born. Remote stream needs of two different nodes cooperation. In this case the writing operations are done by only one node (write-end) while the other node is in the reading mode (read-end).

Local stream allows internal node communication between processes. Remote streams seem like sockets that permit operation from different processes also on different devices. An application can identify a stream by its descriptor id that is generated at the end of a request. In a remote stream, data structures are in the Stream System module on the read node, so writing on the remote stream means to cross the network module that delivers the message to destination node in which the information are stored in the stream stack.

To implement a remote stream, Stream system needs a network module that could provide a connection oriented service. Stream System requests to the network module to establish an unidirectional communication channel towards the remote sensor. Such a consideration is the one that gave us the idea to consider the Stream System as a sort of MW: developing a SW module that implements the Stream System interface while properly using the platform dependent HW/SW services (i.e. the network module and the other local OS services) such a layer could be used to provide the stream concept to the application layer masking different heterogeneousities comprehensive of the radio technology one. The hardware and
software needed to exploit in such a way the features of the Stream System are described in the following.

4 Stream System as a MW for WSN

Figure 4 shows how the Stream System Module is related with system and the other modules. Since the focus of the work is on radio heterogeneity, it has been considered only a Stream System implemented on TinyOs/nesC [21]: to adapt the Stream System to different HW/SW platform it is needed to port its module while keeping application-level compatibility.

Then, the Stream System Module has been associated to the TinyOS network module that provides a connection service between two nodes in a multi-hop network. In order to afford the radio heterogeneity topic, limiting other portability issues, two different nodes of the MICA family [22] have been used, both based on TinyOS but with different communication standard: IEEE 802.15.4 for MicaZ and a proprietary one for Mica2 [22].

5 Validation

5.1 The gateway node

As described before, it has been needed to develop a gateway node composed of a MicaZ and a Mica2 connected via UART. Figure 5 shows the involved PCB with a detail of the connector used to physically connect the nodes to the PCB itself. An additional serial output (the UART has been adapted to RS-232 by means of the IC SP3232E [24]) has been added to allow the gateway to act also as sink node and for debug purposes. The power could be provided by means of the nodes batteries or via an USB port (adapted with the IC XC62FP [25]) setting a proper jumper and powering off the nodes.

5.2 Test application

In order to test the whole system it has been used a Stream System based version of two example applications provided with TinyOS: CntToLedsAndSS (derived from CntToLedsAndRfm) and SSToLeds (derived from RfmToLed). The system is composed of a simple heterogeneous network with a MicaZ node on which CntToLedsAndSS is running and a Mica2 node on which SSToLeds is running (but it is straightforward to exchange the applications on the nodes).

As shown in Figure 6 the X node (MicaZ) sends packets that contain a data structure with an integer by means of the middleware layer. These information came to the Y node (Mica2), that displays the received value on the leds, after they were passed through the gateway node.

During the start phase, source node opens a remote stream (using gateway and destination nodes addresses) versus the gateway node that, in
turns, opens an UART stream between internal nodes in order to forward the received information to the Y node. The Mica2 node on the gateway board, obtains from internal packet data structure the destination address and then opens a remote stream versus the target node. After the remote stream from gateway and Mica2 node has been opened, the packet flow form node X to Y in transparent mode respect to the gateway node and to the applications.

![Diagram](image)

**Figure 6. Sample application**

6 Conclusion
In this work we have presented a suitable middleware-based approach for heterogeneous wireless sensor network. It permits a common abstraction to exchange information between different types of network nodes that uses different radio interfaces. Starting from the Stream System we have developed and implemented a system that permit data transmission trough heterogeneous networks. By means of simple applications it was possible to test all the features. In this way we have been able to transmit packets over the network independently from the source. A new, simple hardware platform has allowed the connection between nodes with different hardware architecture and has extended the gateway role also to a sink role.

The current efforts are devoted to introduce multi-hop features in the Stream System, to port the Stream System of different HW/SW platform, and to develop and engineered version of the gateway node.

References:


Science and Engineering, Michigan State University, 2005.


[21] www.tinyos.net

[22] www.xbow.com

