Extracting Information from Sensor Networks through SQL-like Interface

Rezső Nagy[*]
[*] Óbuda University Alba Regia University Center, Székesfehérvár, Hungary

nagy.rezso@arek.uni-obuda.hu

***Abstract*— Software issues of sensor networks are concerned for many years in the subject "Micro operating systems" for the students of Faculty of Informatics, on the branch of "Ambient Systems". Now we will develop this topic.**

First of all, we will survey more detailed the TinyOS operating system, widely used in wireless sensors.

Programming data collection from sensor networks can be a very hard task, because their network functioning is rather complicated. Furthermore the sensors dispose of extremely limited resources, so they cannot be programmed one by one on higher level programming languages.

Therefore in the above mentioned subject we will present TinyDB as well. TinyDB is a query processing system for extracting information from a network of TinyOS sensors, providing a simple, SQL-like interface for the application program.

I. Introduction

Sensor networks are cooperative networks built of intelligent sensors. The sensors perform a common task in a distributed way. Network connections are realized usually by radio channels with low bandwidth. The sensors forward the measured values or the data derived from these values towards a central node: the base station (or gateway node).

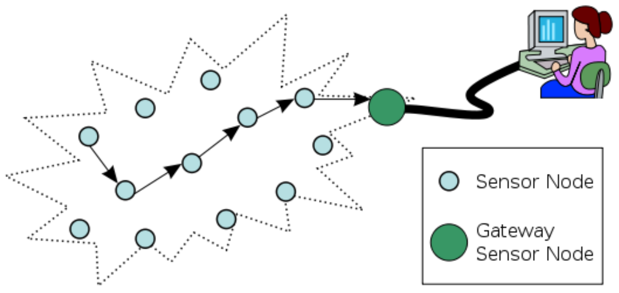

Figure 1. Scheme of a sensor network

The nodes of the sensor networks communicate with the base station usually not directly, but through each other. In the subject "Micro operating systems" we survey also some routing algorithms used in sensor networks. Their network functioning is rather complicated. Furthermore the sensors dispose of extremely limited resources, so they cannot be programmed one by one on higher level programming languages.

In this paper we deal with the development of the above mentioned subject. After surveying some problems of sensor networks, we present a technology which makes possible the database-like query of the data collected in the sensor network, using the high-level TinySQL language similar to standard SQL.

II. Some Problems of Sensor Networks

The most important problems that sensor networks raise are the following:

Limited resources: capacity of the power recource is one of the most important limit. Sensors work autonomously and power source usually cannot be replaced. Sensors have small size, low memory capacity (memory requires a lot of energy). They have low working and data processing rate.

The amount of energy is in close connection with the life-time of the network. Therefore sensor nodes spend in low-power (sleeping) state as much time as possible.

Concurrency: execution of parallel tasks inside the sensor (parameter measuring, data processing, communication with the nighbours). On the other hand, the sensor network itself works as a disrtibuted system, therefore testing and simulation becomes rather complicated.

Various architectural solutions: there are several solutions in both parameter processing and communication protocols. The operating system running on the sensor devices has to support the different architectural solution, while memory capacity and power resources are limited, so the operating system can not contain all the alternatives as a monolithic package.

Distributed and unreliable communication channels: sensors usually communicate through wireless channels with very small transmitter power. Transmission errors are relatively frequent. In many cases, the exact location of the sensors can not be known.

Vulnerable nodes: topology can change because a sensor breaks down, or simply beacase of running out of power. Sensors can be stolen, damaged of the weather or other environmental effects. If topology changes, new routing is necessary, this requires additional energy.

III. TinyOS

A significant part of sensor motes is operated by TinyOS operating system, it supports their programming. Its general characteristics are:

Scheduler:	limited two-level scheduling
	weak handling of threads and events
Components:	commands
	event handlers
	memory frames
	concurrent tasks

Limited memory model:

frames per components, common stack, no heap

TinyOS is not an operating system in the usual sense, but a programming framework. Using this framework we can implement application-specific operating systems for the different applications. These operating systems contain only the functions necessary for the given applications. They are built in the application itself.

The reason of this approach are the extremely limited hardware resources of the motes. An application of some tens of kbytes includes typically some hundreds of bytes belonging to the operating system.

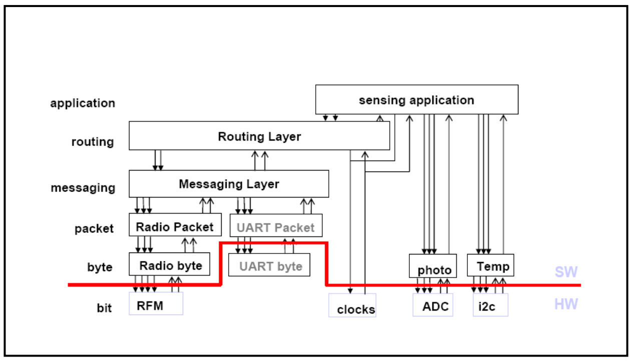

Figure 2. Internal architecture of TinyOS

TinyOS treats the embedded application as a network of software components, separated by strict interfaces. The “network” of components is created in compiling time. Connections cannot be modified later, the system does not support dynamic reconfiguration in running time.

Components contain a specific service set. Services are specified by interfaces. Actually, TinyOS consists of a set of reentrant system components and a task scheduler.

The application connects the components by a “wiring specification”. This wiring specification is independent of the implementation of components. It determines the whole component set used by the application.

A component has two interface classes: interfaces of the provided and of the used services.

Interfaces are bidirectional: they include commands and events as well.

Command is a function implemented by the provider of the interface, and event is implemented by the user of the interface.

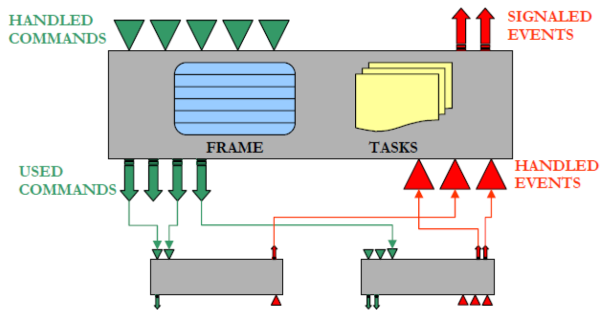

Figure 3. Internal communication of a software component in TinyOS

In a sensor network events can happen at any time. They can have influence on the current computations and other operations. Some of these events can be time-critical, therefore TinyOS starts a handler immediately in the form of “active message”.

Name of the interface	Task of the interface
ADC	Hardware interface of the sensor
Clock	Hardware clock
EEPROMRead/Write	Read and write of EEPROM
HardwareId	Access to hardware identifier
I2C	Interface to I2C bus
Leds	LEDs (red/yellow/green)
MAC	Radio Media Access Control sublayer
Mic	Microphone interface
Pot	Hardware potentiometer
Random	Random number generator
ReceiveMsg	Receive Active Message
SendMsg	Send Active Message
StdControl	Initialization, start and stop components
Time	Get current time
TinySec	Lightweight encryption/decryption
WatchDog	Watchdog timer control

Figure 4. Core interfaces provided by TinyOS

Scheduling of the tasks is not preemptive, that is the tasks run until their completion. Various not preemptive scheduling algorithms are usable. The standard TinyOS scheduler works according to the FCFS (First Come First Served) strategy, but other strategies were implemented as well e.g. EDF (Earliest Deadline First) algorithm which is suitable for real-time tasks.

Thus, tasks cannot interrupt each other, but they can be interrupted by interrupt handlers connected with commands and events. In this respect we distinguish synchronous and asynchronous codes.

Synchronous Code (SC): reachable only from tasks.

Asynchronous Code (AC): reachable from at least one interrupt handler.

Network communication system of TinyOS works by the principle of Active Messages (AM). It transmits small (36-byte) packets provided with a 1-byte handler identifier.

AM implements an unreliable, single-hop datagram protocol. It provides a unified interface to wired and wireless communication. Wired communication can work using the built-in serial port. This can be important mainly for base stations. Controlling computer is connected to base stations usually by wire. The base station is a permanently working node, therefore it is advisable to use a wired connection, because we can also supplying power through the network cable.

IV. TinyDB

TinyDB is a query processing system. Using it, we can extract information from such sensor networks, where motes are operated by TinyOS operating system. This task can be programmed without writing embedded programs for the sensors. Instead of this, we only have to install the TinyOS components serving TinyDB to the motes. TinyDB will provide data collected from the motes in the form of a data base.

For writing query applications running on the computer TinyDB provides a simple Java API and the TinySQL query language. A simple graphical interface is also available for building queries and displaying the results.

A. Some Features of TinyDB

Metadata management (catalogue of the kinds of reading occuring in the sensor network).

High level queries (the declarative query language used by TinyDB makes possible that we only refer to the requested data, without giving the method of obtaining them). By this means the program will be independent of the sensor types as well.

Management of the network topology (TinyDB manages the underlying radio network: it tracks neighbours, maintains routing tables, ensures the efficient and reliable delivery of the data to the user).

Running multiple queries (multiple queries can be run simultaneously on the same set of motes, eventually having different sample rates and accessing different sensors).

Scalability (new motes can be placed into the network; to these motes we only have to install TinyOS with the TinyDB components, the motes send tasks to each other in network messages automatically).

TinyDB system can be divided to two subsystems: the sensor network software and the Java-based client interface. In the subject "Micro operating systems" we will survey the components of the two subsystems.

The sensor network software is the “heart” of TinyDB, although most of the users never modify it. It runs on each mote of the network, its main parts are as follows:

Sensor catalogue and schema manager: records the features of the different sensors and the readable data types. The network can be heterogenous by the type of devices and can be able to report different properties, so the characteristic of the list can be different for the different sensors.

Query processor: it fetches the values of local attributes from the above mentioned catalogue, receives the read data from the sensors of neighboring nodes, combines and aggregates the values together, filters out undesired data and transmits the obtained values

Memory manager: it is a small, handle-based dynamic memory manager, through which TinyDB extends the possibilities of TinyOS.

Network topology manager: TinyDB manages network connections for the sake of efficient routing of data and query sub-results.

Client interface is Java-based. It runs on a PC, connected to the network through a base station. It makes possible the SQL-based query of the data collected on the network’s motes. It consists of Java classes and applications. Its most important classes are as follows:

Network interface class: allows applications to send queries to the network and to receive answers from it.

Classes to build and transmit queries.

A class to receive and parsing answers.

Class to extract information about the attributes and abilities of devices.

A graphical user interface for building queries.

A graphical user interface for displaying diagrams and tables representing individual sensor results.

A graphical user interface to visualize dynamic network topologies.

A demo application consisting of a few objects, which realizes an interface working with queries over a sensor network.

B. TinySQL in TinyDB

The TinySQL query programming language is based upon the standard SQL. There are a few limitations in its possibilities as compared with standard SQL, and it contains some sensor- specific features.

TinyDB uses the TinySQL query language in the database-like management of sensor networks. The current state of the network in a given moment is stored in a table. This table will be actualized periodically.

Lines in the table represent the individual nodes, and columns represent the measured values and certain properties of the nodes. Querying of the current state can be done according to the usual SQL syntax. Earlier states can be stored by creating so-called saving points.

General form of TinySQL queries:

```
SELECT select-list
[FROM sensors]
WHERE where-clause
[ GROUP BY gb-list
[HAVING having-list]
[TRIGGER ACTION command-name [(param)]]
[EPOCH DURATION integer]
```

Usage of SELECT, WHERE, GROUP BY and HAVING is very similar to the standard SQL. Arithmetic expressions are allowed. If using optional GROUP BY, optional HAVING is also applicable.

C. Power Management and Time Synchronization

When running queries longer than a certain time (default value is 4 seconds) TinyDB activates power management and time synchronization function. This means, that each sensors will be active in exactly the same 4 seconds in every sampling period. In this time slice all results should arrive to the base station.

This synchronization and power management makes possible a long lifetime for the nodes of the sensor network.

For example, let's suppose that a mote requires 100 μA of current in sleeping state and 12 mA in active state, its batteries can supply 2400 mAh of energy, and active state lasts 4 seconds. If the sample period is 30 seconds, then the probable lifetime of the mote will be 60 days. Increasing sample period to 120 seconds, probable lifetime will increase to 200 days.

Some comments:

As we have seen, optimal choice of sample period is extremely important.

The base station (with the mote identifier 0) never sleeps. As we mentioned earlier, it is practical to supply it via cable, otherwise we should change batteries in every 2-3 days. It is not a problem, because location of the base station is always known, and connects directly to the controlling computer.

The "waking period" can be changed. Its default value is 4 seconds. Decreasing it is not recommended. By increasing it, probaly a larger proportion of the network messages will arrive successfully.

V. SUMMARY

In this paper we treated the development of the subject "Micro operating systems". We surveyed the idea, characteristics and operation of the sensors and wireless sensor networks.

We presented the TinyOS operating system, which is used on a lot of motes.

We described a way of sensor network programming, when one can collect the values perceived by the sensors with queries written in TinySQL language using the TinyDB data base management system, builded on the special components of TinyOS.

ACKNOWLEDGMENT

Thanks to my former colleague Andrásné Kovács, to Sebestyén Dóra (Unicomp), to Zóra Barta and Zsolt Viniczai (Seacon).

REFERENCES

[1] DHOUTAUT D., « Etude du standard IEEE 802.11 dans le cadre des réseaux ad hoc : de la simulation à l'expérimentation », PhD thesis, CITI Laboratory, INSA de Lyon, december 2003.

[2] George F. Riley; "The Georgia Tech Network Simulator"; Proceedings of the ACM SIGCOMM 2003 Workshops, August 2003.

[3] Esteban Egea-Lopez, Javier Vales-Alonso, Alejandro Martinez-Sala, Pablo Pavon-Marino, Joan Garcia-Haro ; "Simulation Scalability Issues in Wireless Sensor Networks"; IEEE Communications Magazine, July 2006.

[4] Jamal N. Al-Karaki, Ahmed E. Kamal; "Routing techniques in Wireless Sensor networks: A survey"; IEEE Wireless Communications, December 2004.

[5] Fabrice Theoleyre, Fabrice Valois; "A virtual structure for Hybrid networks"; Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Atlanta, Georgia USA, 2004

[6] Fabrice Theoleyre, Fabrice Valois; "Virtual Structure Routing in Ad hoc Networks"; Proceedings of the IEEE International Conference on Communications (ICC), Seoul, Korea, 2005

[7] Wendi Rabiner Heinzelman, Anantha Chandrakasan, Hari Balakrishnan; "Energy- Efficient Communication protocol for Wireless Microsensor Networks"; Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000

[8] Nathalie Mitton, Anthony Busson, Eric Fleury;"Analysis of the Self-Organization in Wireless Multi-Hops Networks"; Rapport de recherche, RR-5328, INRIA, October 2004.

[9] Tim Daniel Hollerung;"The Cluster-Based Routing Protocol"; Project group 'Mobile Ad-hoc Networks Based on Wireless LAN', winter semester 2003/2004.

[10] Hannes Frey and Ivan Stojmenovic; " On Delivery Guarantees of Face and Combined Greedy-Face Routing Algorithms in Ad Hoc and Sensor Networks", in Proceedings of the Twelfth ACM Annual International Conference on Mobile Computing and Networking (MOBICOM), Los Angeles, CA, USA , 2006.

[11] Dr. Paul J.M. Havinga: Cooperating objects in Wireless Sensor networks

[12] Ph. Levis, S. Madden, J. Polastre, r. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler: TinyOS: An Operating System for Sensor Networks

[13] Vidács Attila: Szenzorhálózatok, BMEVIMM9082, 2005 http://hsnlab.tmit.bme.hu/~vidacs

[14] S. Madden, J. Hellerstein, W. Hong: TinyDB: In-Network Query Processing in TinyOS

[15] http://volgy.com/pubs/BIR_SensorNetworks.pdf

[16] A. Kovács, R. Nagy: Szenzorhálózatok alkalmazása

[17] Levendovszky János: Új algoritmusok a vezetéknélküli szenzoriális kommunikációhoz (www.hacusa.org/press/Levendovszky.pdf*)*

[18] Vass Dorottya, Vidács Attila: Energiahatékony kommunikáció szenzorhálózatokban

[19] www.cs.berkeley.edu/~culler/cs252-s02/slides/lec08-wireless.ppt

[20] www.artistembedded.org/docs/Events/9-ARTIST-DATE06-Havinga.pdf

[21] Vidács Attila: Szenzorhálózatok, hálózati réteg, 2007; http://hsnlab.tmit.bme.hu/~vidacs/education/vimm9082/2007/070322.pdf