Research Repository UCD

Title	An extensible dashboard for sensor networks control and visualisation
Authors(s)	Ruzzelli, Antonio G., Dragone, Mauro, Jurdak, Raja, Muldoon, Conor, Barbirato, Alessio, O'Hare, G. M. P. (Greg M. P.)
Publication date	2009-02
Publication information	Ruzzelli, Antonio G., Mauro Dragone, Raja Jurdak, Conor Muldoon, Alessio Barbirato, and G. M. P. (Greg M. P.) O'Hare. "An Extensible Dashboard for Sensor Networks Control and Visualisation," 2009.
Conference details	Poster presentation at the 6th European Workshop on Sensor Networks (EWSN 2009), 11-13 February 2009, Cork
Item record/more information	http://hdl.handle.net/10197/1293

Downloaded 2024-04-17 06:41:05

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

Poster Abstract: An Extensible Dashboard for Sensor Networks Control and Visualisation

A.G. Ruzzelli, M. Dragone, R. Jurdak†, C. Muldoon, A. Barbirato, and G. M. P. O'Hare
CLARITY: The Centre for Sensor Web Technologies
University College Dublin Belfield, D4, Ireland
† CSIRO ICT Centre, Australia.

I. Introduction

A tool to dynamically reconfigure and monitor online the wellness of Wireless Sensor Networks (WSNs) is fundamental to developers both for debugging purposes and for gaining a deep understanding of inter-node communication. For the final users, the tool should provide a user-friendly interface for setting events and tuning application parameters. We present OctopusJB, an extensible JavaBeans-based dashboard to control the topology and behaviour of the network through a network map and interact with the network by reconfiguring and monitoring a number of parameters of application and radio, e.g frequency channel, nodes duty-cycle and monitoring energy consumption. The dashboard allows formulating and injecting composite queries into the network while data can be logged into a file for network analysis. A Network Chart plots live data for network analysis while a 2D Floor Plan allows node localization support.

OctopusJB [2] is open-source software developed specifically for TinyOS version 2 that provides a modular programming architecture. OctopusJB can be utilised as an effective debugging and assessment tool for new underlying algorithms and modules by simply adding them in the OctopusJB configuration. The standard configuration of the dashboard utilises the collection tree protocol (CTP) and the low power listening (LPL) access control defined in TinyOS 2.

Facilitating future extensions and code reuse conforming to certain standards is an important objective of OctopusJB that implements a JavaBeans component-based architecture to distinguish between a component-based infrastructure framework and a set of functional components. The architecture is based on the features associated to the BeanContext class. This eases much of the component composition while the abstract API enables different possible implementations of these features. This architecture enables new or existing components to be plugged-in without altering the remainder of the system. The objective is to provide a high degree of configurability and the creation of different versions of OctopusJB in order to satisfy specific deployment scenarios and application requirements.

OctopusJB greatly improves the state of the art of network state monitoring and network control. Currently, Mviz [1] is the standard tool for TinyOS v.2 that provides a basic visualization with no features for remote network control.

©*Ruzzelli, Dragone, Jurdak, Muldoon, O'Hare, 2009.*

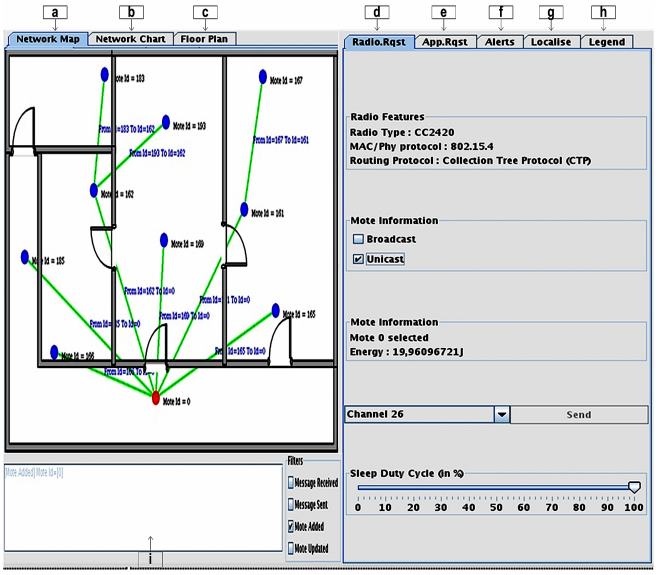

Fig. 1. A Screenshot of OctopusJB with connections and location of nodes.

II. Panels Highlights

The dashboard provides 3 main views namely Network Map *NP*, Network Chart *NC*, and Floor Plan *FP* as shown in Figure 1 tabs a,b, and c. On the right-hand side of the dashboard, a side menu contains support tools for the user to formulate application requests, tune radio parameters, set email and sms alerts, and localise nodes in the network. Incoming packets can be visualised live by the Consol panel at the bottom of the board as shown in Figure 1 tab i that provides also some checkboxes to filter the messages displayed. A Legend panel allows selecting more visualisation options on the main board and logging the data into a file.

We now illustrate the basic architecture and logical organization of OctopusJB, highlighting the basic functionalities of the Radio and the Application panels. Then, we detail some of the advanced functionalities of the Alert and Localise panels.

A. The Basic Functionalities

The basic functionalities are provided by the panels of Radio Requests (*RR*), Application Requests (*AR*), and Legend accessible through the tabs d,e and h in Figure 1, respectively. At network setup, the *NM*, in Figure 1, is populated with nodes

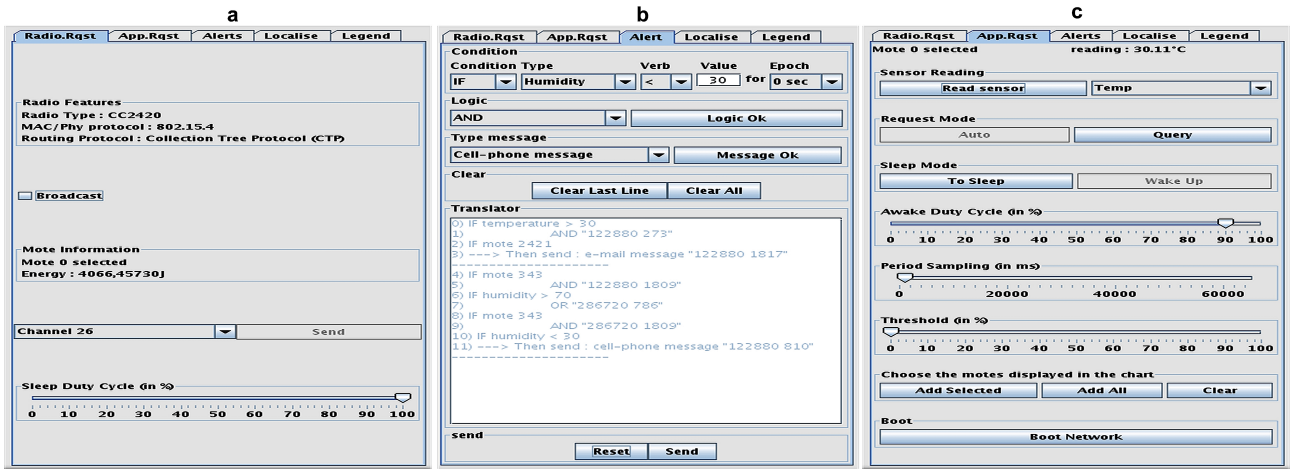

Fig. 2. A Screenshot of the Radio Requests, Alerts, and Application Requests panels in OctopusJB.

and routing connections from nodes to the gateway. Figure 2a presents the *RR* panel more in detail. The panel provides a "*sleep duty cycle (in %)*" slider for changing the node duty cycle. Dragging the slider to the wanted value and releasing the mouse generates a request packet from the gateway to the network. The request packet can be broadcast to the entire network or by selecting individual nodes on the *NM* and unicasting to them. The panel also lists the available radio channels to allow switching to a new frequency channel of operation. This will generate a new request which is broadcast by the gateway and must be acknowledged by all nodes. The *RR* panel provides estimation of the energy spent by each node since start-up. This is calculated by timestamping radio, CPU and sensor activities. Nodes calculates the energy according to provided data-sheet parameters then transmit the value to the gateway at a very low frequency interval. From the *AR* panel shown in Figure 2c, the user can select sensed values to be displayed live in the *NC*. From this panel, the user can select either time-driven or event-driven data collection from nodes. Selecting the Request Mode "*Auto*" activates transmission at a fixed interval while a "*threshold*" slider regulates a basic event-driven data reports. Setting a threshold to 0 is equivalent to activating nodes in auto mode. The *FP*, selected through the tab in Figure 1c, displays a floor map of the physical location of the network. *NM*, *NC* and *FP* can also be displayed in overlapping modalities. Figure 1 shows the *NM/FP* view. Initially, nodes are located randomly on the board, however, the combined view allows to run an interactive localisation algorithm, see Section II-B. Finally, a Legend panel allows the setting and the visualisation of further network parameters.

B. The Advanced Functionalities

The Advanced Functionalities include (1) the Alert panel for injecting composite queries into the network and send alerts to the users; (2) the Localise panel provides support for the nodes localisation.

The Alert panel, in Figure 2b, provides a user-friendly system to compose sentence-like queries. The panel consists of 3 blocks: the "*Query Sentence*", the "*Logic*" and the "*Alert Type*". Each query sentence consists of a *subject*, a *verb*, a *value*, and an *epoch*. The subject allows for the selection of packet fields like nodeID, sensor reading or link quality. A list of fields is generated automatically by the Message Interface Generator (MIG) provided in TinyOS. The *verb* represents the verifying condition while the *epoch* denotes the duration of the event before the sentence is validated. The *Logic* interlaces sentences to formulate composite queries such as "*if a node is within certain xy coordinate and temperature is greater than z then report*". Prior to setting an alert, the correctness of a sentence is evaluated on the screen through the "*Translator*" toolbox. At present, OctopusJB supports the composition of a maximum of 10-sentence query, converting each sentence into an efficient 40-bit data representation. The composite query is injected into the network when the "*Message Ok*" button is pressed. The GUI sets a countdown timer and waits for notification from the nodes before alerting the user. Nodes are provided with an NesC interpreter module to decode the sentence and activate the enquired mode. Queries also have an expiration time set on the node-side.

Finally, the dashboard provides support for interactive node localization (*ILS*) and comparison. By the Localise Panel, in Figure 1 tab g, the user can first upload an XML-based map onto the board containing coordinates, shape and size of the area network. Following, the user can drag and drop nodes of known location (anchors) in the correct position on the *FP*. Releasing the mouse generates a packet transmitted to the anchors or broadcasted to the network. This allows nodes to run their localization algorithm. After completion, nodes transmit their estimated location to the GUI that enables localizing them on the *FP*. If the accuracy obtained is not sufficient, the user may decide to repeat the process by dragging a further anchor to the correct position on the *FP*. Nodes' placements can also be saved to avoid running the algorithm every time the network is rebooted.

References

[1] Mviz http://www.tinyos.net/tinyos-2.x/apps/MViz/
[2] The Octopus Dashboard http://www.csi.ucd.ie/content/octopus-dashboard-sensor-networks-visual-control