Ad hoc sensor network experimentation using the RCX by Mindstorms

Antonio E. Gonzalez-Velazquez
University College London
Electronic and Electric Engineering
Torrington Place,
London,WC1E 7JE,U.K.
ae.gonzalez@ucl.ac.uk

Lionel Sacks
University College London
Electronic and Electric Engineering
Torrington Place,
London,WC1E 7JE,U.K.
lsacks@ee.ucl.ac.uk

Ian W. Marshall
BTexact Technologies
B54 Rm143 Adastral Park
Martlesham Heath Suffolk IP5 3RE
BTExacT Technologies
ian.w.marshall@bt.com

Abstract

In this paper we present work in progress on sensor networks, specifically the testing of our hypotheses and improvements made to the platform we chose for prototyping: The RCX by Mindstorms. RCX is better known for its practical robotics characteristics and it has a wide set of running applications developed by the community; however, we identified some internal characteristics such as the micro operating environment and minimal resources, like computing power, battery among others, that made it a good prospect for being considered as a ready-node in an ad hoc network, a unique assignment for it.

Having an existent ready-node for sensor ad hoc networks prepared to behave as one is distinctive, and we take the opportunity to describe in this paper why the RCX can be advantageous and suitable for these activities. We will introduce a new ad hoc communication protocol with special characteristics that has been implemented and still is in the stage of experimentation and collection of data, running on a group of RCX over a modified version of LegOS using the infrared interface. The introduced Find-a-friend communication protocol, proposes a simple mechanism for establishing communication in an autonomous and self-configured group of nodes.

Introduction

Sensor networks have specific restraints and conditions that have attracted consistent research; although some of these characteristics have been extensively modelled, there is still no agreement on a definite structure or conditions. Most of the relevant research has proposed algorithms and methods for sensor networks assuming specific conditions, outside of those conditions the algorithms and methods cannot be easily extended, and consequently, comparing them is not an easy task. On the other hand, having these specific scenarios exposes a different approach to the traditional interest for finding the only-standard-agreed method.

The well-known "Robotic Invention System"(RIS) by Mindstorms has frequently attracted the attention of enthusiasts for its abilities to respond to external input; it comes with a graphical programming environment and software Developers Kit to use its functionality to programs running on the Windows platform. Robotics has been the main use for RIS, along with some people who have proposed and successfully employed upgraded programming environments such as TinyVM, NQC, legOS and lejOS.

The "programmable brick" as it is known by the community, is the heart-and-brain of RIS containing an 8-bit MCU with its related peripherals and companion circuitry; legOS and other environments base their efforts on the simplicity of the RCX design to implement "operating systems" that should drive this small specialized computer for their robotic purposes. The MCU is the HD8/3292, the smallest member of the 3297 Hitachi family of micro-controllers, within the H8/300 8-bit architecture. It internally holds 16KB in ROM and 512B in RAM, uses 8-bit addressing, contains 16KB masked ROM meanwhile the 32 KB RAM are accessed as external in the address map, it uses an Infrared interface for downloading and communication purposes.

In this paper, we start depicting the RCX as the physical corpus in an ad hoc network node. Given characteristics for nodes in an ad hoc network outlined in previous papers, we are going to address the issue of whether or not it is feasible to adapt the RCX to support an operating environment specifically for the task.

We will explain the specific implementation of an ad-hoc communication protocol for sensor networks using the RCX as the model node that could favourably extend features currently available for this platform and allow us to experiment and develop better understanding of some of the complexities in this field.

Related work

In 1997, Evans et al proposed a platform for embedded mobile networking in its Piconet Project [1]. Although it had different strategies for network transport, it addressed concerns for platform design and implementation details for updating software at nodes. There has been general recognition of the importance of the sensors as the simplest devices participating in an ad hoc network, and [2] is a good example where members of UC Berkeley outline their project named Smart Dust, in which they aimed to build simple nodes that were wireless sensors measuring 1 mm^3. They inserted a well-equipped and full-featured node called the "base station" to assist in several tasks in their proposals, reducing the load on the smaller nodes with some loss of flexibility for mobile behaviour of the so-called "motes". This project includes practical observations for actual limitations of executing security algorithms in suitable micro-controllers and micro-processors living inside the energy limited sensor nodes. They also provide specific proposals to overcome these limitations.

A great deal of research has been carried out in the area of sensor networks, giving the variety of immediate applications envisaged, as in [3-10] where the wide range of proposals can be easily identified, each with a specific focus and assumptions. What begins to emerge are efforts to take into consideration energy, location, complexity and security issues on widely dispersed but highly vulnerable nodes.

Why an Ad hoc network could be a good place for the RCX

We have explored ad hoc networks issues, and chosen to focus on simple devices as the physical holders for the participant nodes. Projects like SmartDust have already used 8-bit MCUs with single sensor inputs, and they have explained their reasons to support their choices.

The limitations for memory, CPU power and battery, have driven the conceptualisation of the nodes; finding a suitable platform with the right operating environment is not an easy task. Basically, there are two choices for the hardware platform, as well as for the operating environment: use an existent one or develop a new one.

The RCX possesses a unique combination of characteristics:

– The computation platform has proven suitable and reliable for task driven endeavours, although there are no metrics available at the moment.
– IR is reliable and functional for extending ad hoc communication
– There is a complete set of hardware drivers already available for their main inputs/outputs
– It is energy conscious and has a sense of mobility
– Amongst its operating environments, LegOS [11] is one of the most stable and the full code is open to public participation under the GNU licence.

Using RCX-LegOS[12] as the developing platform for ad hoc efforts looks promising; it could give the facility to quickly implement ideas from the notebook and try them out in the field field. We should find out if using the RCX-platform in this way could provide better conditions than developing a complete new one without having all the requirements in advance.

Given our interest in making the nodes self-configuring, self-upgradeable and autonomous, I think it is crucial that our platform is able to "survive" in the ad-hoc network with a minimal amount of dependency on user intervention. Code in RAM cannot be completely reliable given volatile characteristics in storage, faults and phases between changes of operating environment.

LegOS Network Protocol (LNP)

LNP is the LegOS communications module; it provides basic communication functionalities to programs through a simple set of calls. It takes advantage of the IR interface available and the basic communications skills built on the RCX.

Basic IR communication in the RCX

As in [13] describes that the RCX uses a 38kHz carrier, with sampling rate at 2400 bps, which makes each bit approx 417µs; the IR protocol associated with sending a "message" to the RCX corresponds to bit encoding is 2400 bps, NRZ, 1 start, 8 data, odd parity, 1 stop bit. A '0' is coded as a 417us pulse of 38kHz IR, a '1' bit is 417us of nothing to send.

Meanwhile for the packet level every packet has the form :
   0x55 0xFF 0x00 D1 ~D1 D2 ~D2 ... Dn ~Dn C ~C
where D1...Dn are the message data, and C = D1 + D2 + ... Dn.

The data for sending an IR message is F7 followed by the 8 bit "message". For example, the following is a packet sending the message "0x12" to the RCX:
   0x55 0xFF 0x00 0xF7 0x08 0x12 0xED 0x09 0xf6

Kekoa[14] writes:
"The scheme used to transmit data results in an equal number of zero bits and one bits, allowing a receiver to compensate for a constant signal bias (caused by

ambient light) simply by subtracting the average signal value. Note that the header also has an equal number of ones and zeros; this warms up the receiver before the real data arrives."

The basic use of this format is the interchange of opcodes between RCX and the IR tower connected to a PC; the opcodes activate specific actions and stored functions on each side, having some specific reply and others do not expect one.

Internals in LNP

Luis Villa mentions: "LNP has two messaging layers, the integrity layer and the logical layer. The integrity layer makes sure that packets get through uncorrupted, but they aren't directed anywhere in particular. The logical layer adds addressing on top of the integrity layer, so that packets can be directed to a specific port on a specific device."

The programmer has the opportunity to use pairs of functions for receiving and sending messages, with the difference of having an addressed port and RCX Id, or not addressing information. The latter could be considered to be similar to a broadcast message, given that it indicates to all the potential receivers that the message could be for everyone. One application should always be set up and ready to receive the appropriate messages in order to read its contents.

The addressed port and RCX ID use one byte in the message transmitted using an agreed subnet mask, giving up to 255 potential different combinations of addresses and ports; an additional transmitted byte used as a checksum provides the opportunity for detecting simple errors in the transmission. Additionally to the previous functions, programmers could call functions relative for lnp_logical_write() that do not use the previous addressing scheme and allows different frame structure to be built. Although the routines for detecting collisions[15] for sending and receiving are limited, it gives primary detection for listening to IR echoes and they are effective for the common use the RCX usually has.
LNP was intended as a protocol to extend the communications facilities for robotic purposes, and allows the users to have the opportunity of having a means of collaboration from the program itself outside of the pre-built functions. Every RCX in LNP has the chance of receiving all the packets in sight, but it is not able to do anything else but receive them by a program.

Find-a-friend: an Ad hoc Protocol for Sensor Networks

Once we identified that RCX could be a suitable platform for executing real ad hoc functions, we needed to build a suitable communication mechanism for the members of the network. RCX characteristics, particularly the Infrared interface, along with our previous work gave the main elements for proposing Find-a-friend, as an ad hoc protocol.

In the proposal of Find-a-friend it has been considered that RCX's Infrared interface represents a directed transmission with no carrier sense, and some basic medium access should be implemented. When a specific node is not in the illumination cone of any sender, our specific could potentially spin or move itself through the space available until it has contact with the scope of another member of the network. Although is possible to increase the range of the transmission to cover longer distances, that does not guarantee the existence of a node and in close environments can easily induce more "infrared light-noise".

Thus Find-a-friend makes it essential for these nodes to find at least one "friend" to communicate with, and makes the vicinity area a more desirable space where the nodes can "live". Further optimisation to this vicinity could bring more benefits.

In order to keep most of the information available fairly current, and to produce information for nodes to evaluate and reconsider their decisions, we have produced a mechanism for unattached data spreading, similar to gossiping. A single node always has the ultimate decision whether to assess this data and evaluate actions, including its own, that could affect its participation in the network.

We have been building this protocol expecting to provide a simple and small set of function calls for accessing its functionalities, increasing the range of applications to be tested adding to the wide set already available from the LegOS and RCX community. We expect to develop better understanding of running sensor applications using ad hoc networks, and reduce the time to implement improvements and carrying experimentations.

The following scenario has been assumed:

- There are RCX freely available in a close environment, with limited "light noise"
- There may be access to one or more "super-nodes", with potential additional capacities and needs, able to share information about their findings.
- The nodes are able to collect sampling data for themselves with local parameters and policies.
- Nodes are happy to share information about their collection if some flow is established, whenever they belong to the same group.
- There is no explicit demand to establish one route from every node to every other,

therefore the routes are built by demand and kept current locally.

- The nodes push feed back to their neighbourhood for two main reasons: to spread information about their general success and state, and to give the opportunity to the original sources to accomplish end to end decisions to improve efficiency transmission as required.
- Long latency is expected.
- No specific topology is explicitly expected, and the number of members could start from a handful to a few dozen; a larger number of nodes has not yet received attention, although there is no indication that this could not be faced.
- Nodes are willing for cooperation adding source route information to a request packet

Main components

Hello/Heartbeat

This process should be permanently running, in a periodic/event basis. When the node starts operation, it is used for sending advice to the vicinity about its presence. The listeners do not have an enforced action to do, although in the best cases, they could add the presence of the node in this vicinity.
When running in a periodic basis, every node should find suitable to its own conditions, a period of time in which it has not transmitted data and wants to keep others informed about its presence and status. The data sent should comply with the Fellowship-Dataframe (FD) containing information potentially useful to other members. In this case this process is similar to the implementation of Hello or Heart beat similar to other proposals.
When the node has been transmitting in recent time, it doesn't need to send "Hello type" messages, given that the others are aware of its presence; the node might wish to send the FD as a piggyback of its transmissions, either as a broadcast or as an addressed message.
This process should be checking the Last-time-to-transmit (LTT) register to identify it it is necessary to broadcast a new FD.

Find-a-friend/find-fellowship

This process is very important for our proposal, given that it holds the task of finding a suitable fellow/fellowship to be attached to. If it succeeds, it will use the fellowship information available to arrange its conditions, such as battery power, range, direction, position, etc.
Fundamentally, finding a suitable fellowship to communicate with enables the node to know the conditions and reach their scope in terms of members and services. For the incoming node, it allows it to share more efficiently the shared transmission medium, for those already members of the fellowship, it can activate this process in order to extend or widen the actual scope of the fellowship using local resources they are willing to provide, eg, increase power range in order to help some messages to reach beyond the local vicinity.

Using this process, the node is able to update its Fellowship table (FT) and being able to take decisions around more reliable nodes, and their conditions and experiences they are having in the vicinity. Every node should handle this information for its own purposes.

Route-Request (RR)/Route-Request-Reply (RRR)

This implementation is similar to other common and widely available dynamic routing protocols. The purpose of transmitting a message carrying this flag is to request information about acknowledging a path where the source could reach the final destination. Given this definition, the kind of answers it is possible to obtain are not restrained to direct communication of the respondent, they also include potential routes that the respondent has been receiving depending on his own conditions of freshness. The freshness could be modified for the respondent if it evaluates there are not conditions where it can be trusted, given time, contrary indications, and similar.

For our specific scope, it is more likely to receive this kind of hard routes prior to establishing communication with the Tower, collector of samplings and potential generator of control requests to nodes. Having specific routes to a very common target reduces overhead on finding dynamic routes for every node, and particularly it opens the opportunity to build a trended "flow" of data to the collectors of information or other strong source of data requests.

Not many applications in this environment should require a FR, and those should be ready to afford costs associated to keeping a FR current; there are other elements that could help to decrease or increase the premiums of having FR implemented, like mobility, CPU consumption and nodes likeliness for going to sleep for saving power.

Time to Live

This implementation still requires revision from a wider perspective. So far it has been useful for controlling the number of requests flowing in the actual network. However, it is envisaged that further information from bigger simulations or deployments could be helpful to distinguish the relevance of this implementation.
Its original purpose is to avoid having messages running over very long periods of time in the network and preventing broadcasts from flooding far outside the vicinity. We plan to develop new tools to improve control of long-lived packets.

Transmission Window

The transmission window is the mechanism used to receive and transmit messages prior to reaching the medium. In this way, Window Array (WA) should always be available with information about messages being transmitted and received. Similar to the Sliding Window implementation in TCP/IP suite, but there should be modifications to the size and use in this environment, given the longer latency for each message and the lack of requirements for a tight handshake between sender and receiver, in some cases the sender could not have a known receiver prior to the transmission.

The sliding window implementation could be very helpful in the cases where some route RR has been issued, and some RRR is in the way. If a specific application requires better knowledge of the reception at the other end, it is recommendable to implement a fixed route first, and then use this route as long as it is possible to receive further FD that helps it to understand the effectiveness of the approach. Sending duplicate data should not be discarded.

Given that sensor nodes have reduced resources, how they select the moment for next transmission (TNT) and the amount of data they can keep before they start discarding data are important decisions. Collecting more precise results about suitable Total travelled time (TTT) for each packet could help to develop better understanding of this problem; it is expected that this number increases in proportion with the size of the network among other characteristics.

For this kind of working environment, I am proposing that the transmission between sender and transmitter should keep as little state information as possible, and trying to have more understanding with the members of its vicinity about environmental conditions; in cases where the node requires a more reliable circuit setting a Fixed route represents an appropriate alternative.

Fellowship-Dataframe(FD)

This data structure is fundamental in the design of this proposal. This dynamic self-contained structure provides the main mechanism for spread "vicinity awareness". With its vicinity scope, spontaneous distribution and uncommitted data, the nodes receive and transmit this structure for sharing its status with the rest of the vicinity.

After receiving this structure, a node could use it to assist its awareness about its environment and potentially its own performance according to its neighbours' perspective. At least a couple of them have been implemented, describing the basic node transmission status (collisions, successes, succeeded forwards, total packets transmitted, and known direct neighbours) and its hardware status (program version, energy and running time since last upload).

The assumptions to model this structure and send it, either periodically or event based are:

- Once the vicinity where the node is running has been identified, reducing the number of collisions and increasing the performance for data transmission represent the next challenge. It is envisaged that the nodes could implement some method for coordination of efforts, and have a closer anticipation of potential transmitting nodes.
- Given the former, the nodes have the choice of either:
  - Listening to the medium and locally evaluate the moment for trying its Time for next transmission (TNT):
    - Indicating its new presence in the neighbourhood
    - If enough resources, and probably active in transmission, broadcast its Local-Fellowship-Dataframe (LFD)
    - Send regular data with or without FD
  - Listening to the medium and update its LFD
  - Listening to the medium and given its own resources, broadcast its LFD.
  - Avoiding listening to the medium and save resources (may be going to sleep-mode and saving energy) and may be calculating its time for awakening.

Every node has three choices for sending its packet, either broadcasting or addressing a message to a specific member of the fellowship and using a previously established Fixed Route (FR). A node could evaluate the appropriate next hope node (NHN) to whom it should address the next message, given its own LFD and the understanding it can estimate from the vicinity, based on elements like known neighbours and their performance, their recent forward-success history and their perspective.

This proposal has specific interest in demonstrating how the loose control of routing and high collaboration in the neighbourhood could be useful on specific ad hoc sensor networks.

Further work

We are in the process of implementing a method to evaluate with more accuracy how our proposal is working, how each node sees, records and reports its performance is one of the main issues under evaluation. We are looking forward to evaluating the gathered results and making further adjustments.

We will need to do further analysis and tests to provide a stable API to offer the proposed functionality be used by programs running under the LegOS environment, and at the same time evaluate how the operating environment in LegOS is suitable for this new set of challenges.

We still need to consider when simulation could be brought to this project; we would like to implement its corresponding simulation model for adding elements different to those available in the implementation stage.

Conclusion

We have presented a specific experience for ad hoc sensor networking using the RCX device as a ready node for experimentation; we believe that this approach will bring even more benefits for our ad hoc networking interests once a reliable and stable communication protocol is in place. Find-a-friend proposes a simple decision mechanism for cpu constrained devices to establish medium to constant communication with specific interest and reliance in the vicinity of the node, drawing a specific scenario for data flowing through the network, leaving opportunities for further improvement and for considering further capabilities.

Reference List

1. F. Bennett et al., *IEEE-Personal-Communications.vol.4, no.5; Oct.1997; p.8-15* no.5; Oct. 1997; p.8-155, -155 (1997).

2. J. Hill et al., *Acm Sigplan Notices* 35, 93-104 (2000).

3. R. A. Burne et al., *Proceedings-of-the-SPIE --The-International-Society-for-Optical-Engineering.vol.3713; 1999; p.238-48* of-the (1999).

4. S. Arnon, *Electronics-Letters.vol.36, no.2; 20 Jan.2000; p.186-7* no.2; 20 Jan. 2000; p.186-77, -77 (2000).

5. K. Sohrabi, J. Gao, V. Ailawadhi, G. J. Pottie, *IEEE-Personal-Communications.vol.7, no.5; Oct.2000; p.16-27* no.5; Oct. 2000; p.16-277, -277 (2000).

6. E. Shih, B. H. Calhoun, H. C. Seong, A. P. Chandrakasan, *Proceedings IEEE Computer Society Workshop on VLSI 2001.Emerging Technologies for VLSI Systems.IEEE Computer Society, Los Alamitos, CA, USA; 2001; xiv+177 pp.p.16-21* Los Alamitos, CA, USA-211 (2001).

7. W. W. Manges, *Sensors.vol.17, no.5; May 2000; p.72-7* no.5; May 2000; p.72-77, -77 (2000).

8. S. S. Iyengar, *Computer-Science-and-Informatics.vol.21, no.1; July 1991; p.29-41* no.1; July 1991; p.29-411, -411 (1991).

9. R. Min et al., *VLSI Design 2001.Fourteenth International Conference on VLSI Design.IEEE Comput.Soc, Los Alamitos, CA, USA; 2001; xxxvii+541 pp.p.205-10* Los Alamitos, CA, USA-100 (2001).

10. M. Fondl and L. Linse, *Sensors.vol.17, no.12; Dec.2000; p.29-31* no.12; Dec. 2000; p.29-311, -311 (2000).

11. Noga, M. L. "Designing the legOS multitasking operating system". Dr.Dobb's Journal . 1-11-0099.

12. S. Nielsson, "Introduction to the LegOS kernel" 2000).http://news.lugnet.com/jump.cgi?http://legOS.sourceforge.net/docs/kerneldoc.ps

13. Villa, Luis. LegOS HOWTO. LegOS WWW . 1-10-2000.
http://legos.sourceforge.net/HOWTO/

14. Kekoa Proudfoot. **RCX Internals**. Kekoa Proudfoot web site . 1-1-1998.
http://graphics.stanford.edu/~kekoa/rcx/

15. LegOS news group. LEGO® MINDSTORMS(tm) Internals. LegOS WWW . 23-5-2002.
http://www.crynwr.com/lego-robotics/