2017 International Conference on Computational Science and Computational Intelligence

Distributed Algorithms on IoT Devices: Bully Leader Election

Mariano Méndez
dept. Computer Science
Facultad de Ingeniería
Universidad de Buenos Aires
CABA, Argentina
marianomendez@fi.uba.ar

Fernando G. Tinetti
III-LIDI, Fac. de Informática, UNLP
Comisión de Inv. Cient. de la Prov. de Bs. As.
La Plata, Argentina
fernando@info.unlp.edu.ar

Adrian M. Duran
dept. Computer Science
Facultad de Ingeniería
Universidad de Buenos Aires
CABA, Argentina
aduran@fi.uba.ar

Daniel A. Obon
dept. Computer Science
Facultad de Ingeniería
Universidad de Buenos Aires
CABA, Argentina
marianomendez@fi.uba.ar

Natalia G. Bartolome
dept. Computer Science
Facultad de Ingeniería
Universidad de Buenos Aires
CABA, Argentina
nbartolome@fi.uba.ar

***Abstract*—We present in this paper the implementation of a well-known coordination algorithm on specific and low-performance IoT (Internet of Thing) devices. The implementation of the bully algorithm for leader election is achieved in a two-stage process: a) an IoT independent implementation, made in a high level progamming language (Java, in particular), and b) the implementation on the IoT devices, where several limitations and characteristics have to be taken into account. We have used this algorithm for coordination of a set of small cars. Beyond the specific implementation, this work has two main underlying contributions: a) show that it is possible to take advantage of many algorithms and results already proven to be useful in the area of distributed computing, and b) show that IoT devices limitations (e.g. in computing power and storage) do not necessarily imply that useful algorithms cannot be used.**

***Index Terms*—Distributed System, Leader Election, Internet of Things**

I. Introduction

The term Internet of Things (IoT) was coined in 1999 by Kevin Ashton, as the title for the presentation on RFID technology delivered by the author at Procter & Gamble (P&G) [15] [1]. About twenty years later, IoT became one of the most prosperous fields in Computer and Electronic Science and techonology. As claimed by Atzori et al [2], the power behind IoT lies in the high impact it has had on several aspects of everyday life and in the behavior of its users. Gartner Inc. estimates that 8.4 billion connected things will be in use worldwide in 2017, "The consumer segment is the largest user of connected things with 5.2 billion units in 2017" [18]. If the current trends on IoT devices connected on the Internet persist, the number of connected devices is expected to grow to 30 billion by 2020 [21] [11]. As explained in [15], there are 100.000 billion potential objects to be connected to the Internet. In this new era, where "Sensors and actuators embedded in physical objects are linked through wired and wireless networks, often using the same Internet Protocol (IP) that connects the Internet" [13] one of the many fields of Computer Science that has become closely related with these trends is that of Distributed Systems. They have been studied along several years in Computer Science as well as Electronic Engineering, to the extent that they have become a science and technology field in themselves.

Distributed systems have been thoroughly studied on personal computers (PCs), and there exists abundant bibliography on Distributed Algorithm and implementation on PCs. A set of IoT devices can be considered a distributed system in itself. Thus, several well-known problems can be approached as in distributed systems, but IoT devices devices are by far less powerful than a common desktop computer/PCs. The specific adaptation of well-known approaches to distributed systems problems to IoT devices should be taken into account and specifically analyzed too.

The main objective of this research work is to implement a Leader Election Algorithm (classically built on distributed systems of full-blown computers), on IoT development boards in order to develop a task guided by a leader, in this case to traverse a land. IoT development boards are available for developers to "internetize" things or objects. We will use a STM32F205RGY6 120Mhz ARM Cortex M3 with 1MB flash, 128KB RAM and a Broadcom BCM43362 Wi-Fi chip that support 802.11b/g/n Wi-Fi as the hardware to implement a Leader Election Algorithm as described in classical bibliography [12]. Figure 1 shows the cars on which we have evaluated the implementation, and the same controlling hardware has been responsible for car control and movement coordination, using a set of six cars.

The concurrency features available in the device firmware

978-1-5386-2652-8/17 $31.00 © 2017 IEEE
DOI 10.1109/CSCI.2017.235

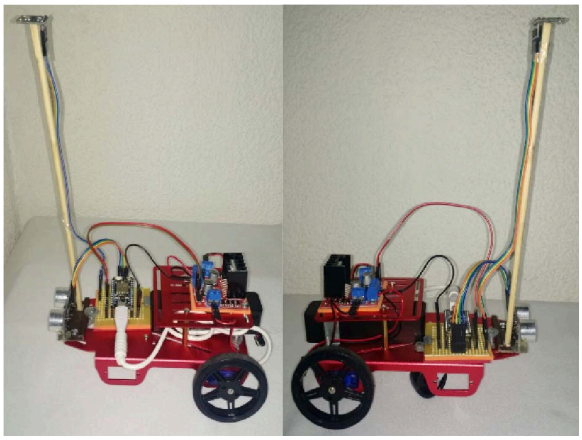

Fig. 1. Cars to be Coordinated

based on FreeRTOS operating Systems allowed the implementation with multi-threading programming. Thus, each IoT controller will implement the election algorithm as well as other real-time tasks, such as car movement, communication, etc.

The aim of this research work is centered on the foundation for distributed systems comprised by IoT development boards that can interact with one another to perform the task of rescuing survivors in the event of a natural catastrophe or when searching for missing people. Infrastructure and human life struck by natural catastrophes involves losses worth millions and causes deep public concern. Hurricanes (i.e. 2017 Irma), earthquakes (2017, Chiapas Mexico, 8.2 Richter scale) are some examples of natural disasters that countries have had to face. A possible contribution of technology can be the combination of IoT and Distributed System as they are of great use to assist survivors of such disasters. In addition, this application can be used to pinpoint the location of missing people in landscapes that are difficult to access by human beings.

The structure of this paper is as follows. The next section describes the related work on the research topic. In Section III the hardware platform and components in used have been described. Section IV presents a thorough description of how the Bully algorithm is implemented. Finally, Section V presents conclusions and future work.

II. Related Work

There are several works related to IoT coordination and networks of nodes (many of them in the context of sensor networks) used to cover a destination area, such as:

1) IoT Coordination:

- The coordination of nodes in the Internet of Things [4]: a method is proposed for synchronization of IoT nodes. The proposed method is divided in two main activities: a) Controller or "well-known area" activity, where a coordinator handle a group of IoT nodes, and b) Cloud processing or "unknown area" activity, where the cloud is used for finding and communicate IoT nodes.
- A Revised BROGO (Branch Optima to Global Optimum) Algorithm for Leader Election in Wireless Sensor and IoT Networks [5]: the BROGO Algorithm is taken as a departure point/basis, in which a spanning tree is generated. Each leaf of the spanning tree sends a message to the root tree (through its corresponding tree branch) in order to identify the leader. Then, the root choses the global leader from the information received through the tree branches. It is possible that the tree root fails before or during the selection process. In order to avoid the failure of the whole selection process, the spanning tree root node is dynamically selected. The requirements set for the spanning tree root node selection are a) it has to be non-failing node, and b) it is the node with the minimum identification.
- A Networking Perspective on Self-Organizing Intersection Management [16]: the inter car communication such as Wi-Fi and 3G/4G is used in order to avoid possible car collisions. Several possible analysis range from enhancing traffic performance to handling critical security issues. The proposed "Virtual Traffic Light" (VTL) system is an enhancement of the current physical light system. The VTL underlying strategy is to handle cars clusters taking advantage of inter car communication, and the main objective is the reduction of urban area vehicular traffic.

2) Node Networks Used to Cover a Destination Area:

- Semi-Stochastic Topology Control with Application to Mobile Robot Team of Leader-Following Formation [9]: a team of robots following leader is highly effective when an underground area has to be covered during an emergency. This paper proposes a semi stochastic topology, a combination of control and movement of the robot team. Experiments show that the performance if optimized by controlling the topology as the robots approach the destination points.
- Several projects in which a set of autonomous drone for real-time operation in disaster areas [19]. Drones may explore and process audio and video, and a team of operators would access to the drone reports via a web site. Thus, the drones provide information for decision support of rescue missions.

III. The Hardware

A set of six autonomous small robots (cars, such as that in Figure 1) have been built. The central processing and control unit is an ARM Cortex M3 processor based on an IoT development board has been selected, the Particle Photon. This development board combines a high speed Wifi chip with the ARM processor. Specifications are listed below (https://docs.particle.io/datasheets/photon-datasheet/) :

- The Microcontroller:
  - STM32F205 120 Mhz ARM Cortex M3

  - 1 Mb of Flash Memory
  - 128 Kb of RAM
  - 18 inputOutput GPIO, See Table I
- The WiFi chip:
  - Cypress BCM43362
  - WLAN Standards:IEEE 802 11bgn
  - Antenna Port: Single Antenna
  - Frequency Band:2.412GHz – 2.462GHz
  - Sub Channels:1 – 11
  - Modulation: DSSS, CCK, OFDM, BPSK, QPSK, 16QAM, 64QAM

Other features described by the board seller are: Open Source Design, Real-Time Operating System (FreeRTOS), Soft AP (Access Point) setup, FCC, CE and IC certified. Programming is mostly made using a web interface or an internet connection to the seller company.

Peripheral Type	Qty	Input(I) / Output(O)
Digital	18	I/O
Analog (ADC)	8	I
Analog (DAC)	2	O
SPI	2	I/O
I2S	1	I/O
I2C	1	I/O
CAN	1	I/O
USB	1	I/O
PWM	9[3]	O

TABLE I
BOARD PERIPHERALS

A. The processor

The processor family STM32F20x is based on the high-performance ARM Cortex-M3 32-bit RISC core operating at a frequency of up to 120 MHz (STM32F205x August 2016, datasheet, STMicroelectronics). This processor has been designed to provide high performance and low power consumption. It has been released by ARM in 2006. The main focus of the processor design has been the 32-bit embedded processors market, and some of its applications are [20]:

- Low-cost microcontrollers
- Automotive
- Data communications
- Industrial control
- Consumer products

The ARM Cortex M3 is a fifteen general register processor with dual mode operation (Privileged mode, User mode). The processor is a 3-stage Pipeline Core Based on Harvard Architecture (thus maximizing memory usage) and the core executes the Thumb-2 instruction set. Some other features of this processor are: a Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing, multiple high-performance bus interfaces, a low-cost debug solution, and optional Memory Protection Unit (MPU). An abbreviated scheme of the STM32F205 can be seen in Figure 2 (at the right of the figure),

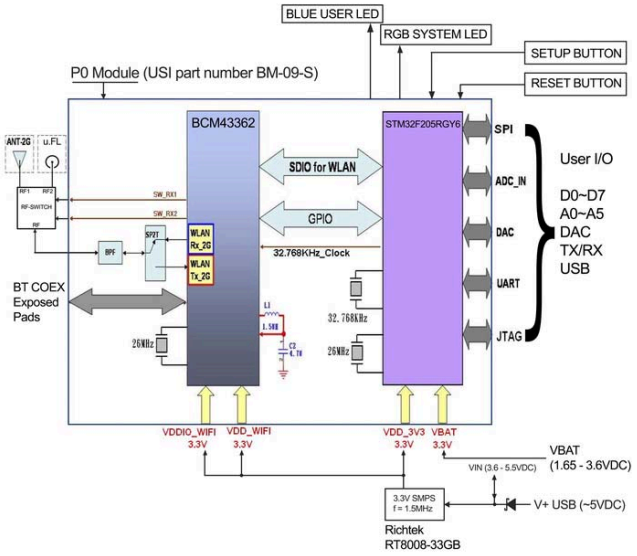

Fig. 2. Particle Photon Scheme

along with its connection to the WiFi module, the BCM43362 (at the left of the figure).

Other minor hardware components have been used to build the mobile units such as digital compass HMC5883L, and H-bridge whose main component is a L298N chip, a proximity sensor HC-SR04, and finally a robot chassis kit (including all the mechanical parts, such as motors, wheels, etc.).

IV. THE SOFTWARE

The software design and implementation process of the Bully Algorithm has been carried out in a two-stage process:

- A first stage implementation in a high-level programming language, independently of IoT devices.
- A seconds stage on the IoT devices themselves, taking into account all the details and restrictions involved in the IoT devices.

For the first stage, i.e. the implementation of the Bully election Algorithm [7] by using a high level programming language we selected the Java programming language. Each node in the system is identified with an exclusive name, in this case the device MAC (Media Access Control) address has been selected as the unique identifier. The basic idea behind the Bully Algorithm is to use such name to impose the nodes with smaller identifier to adopt the node with the highest unique identifier as the leader or coordinator [7] [17]. When each device starts, it assumes itself to be the coordinator and then relays its Id to the other network nodes by sending a multi-cast message (i.e. the same as the one used in the Spanning Tree Protocols to select the root bridge [10]). When the message is received by a node, it compares the newly received id, id_{new}, with the id of the current coordinator, $id_{Coordinator}$. If the new id_{new} is smaller than the $id_{Coordinator}$, then the sender of the id is now considered the new coordinator $id_{Coordinator} = id_{new}$. Otherwise, if the receiver of this message is the current

Coordinator, it sends a new message with its own id in order to let the other nodes know that it is still the coordinator. For this scenario, the device MAC address serves as id and at the same time as the node priority. Multi-cast messages have been selected in order to guarantee that any new node which joins the network can do so without restrictions.

The second stage of the algorithm implementation has been carried out and tested on a hardware-dependent platform. The selected device imposes a programming language which is a subset of C++ programming language. The IoT development board provides a cloud IDE (Integrated Development Environment) and it can also be used on a text editor plus some plug-ins. Due to the fact that the implementation is highly coupled with the hardware, a Kernel summary has been made. The ARM Cortex-M3 is a multitasking microcontroller with a single core which implies that only one thread or task can be executed at a time. There are two kinds of real time requirements, hard real time requirements which implies that there is a time deadline after which an absolute system failure will be reached; or a soft real time requirement which does not imply an absolute system failure [14] [8]. FreeRTOS, the real time kernel running on the IoT working platform, can run many tasks concurrently, scheduling them according to the priority assigned by the programmer. Tasks related with hard real time requirements will be assigned higher priorities, and tasks related with soft requirements have lower ones [6] [14]. The version of FreeTOS ported for the Cortex-M3 provides or implements several usefull features [3]:

- Pre-emptive or co-operative operation
- Very flexible task priority assignment
- Queues
- Binary semaphores
- Counting semaphores
- Mutexes
- Tick hook functions
- Idle hook functions
- Stack overflow checking
- Trace hook macros

The bully algorithm implementation has been designed to be run on the previously described IoT platform, taking into account the multi-threading features of the microcontroller. The main thread will include three child tasks:

- The leader election main thread: This is the main thread or the core of the leader election algorithm. By Following the process described in [7] the leader election process will start by setting itself as a new leader; each device will perform it after booting. After that, three child threads will be created: ReceiverBully, KeepAliveSender, and LeaderLivenessVerifier.
- Receiver bully child thread: This thread will be listening for the messages belonging to the leader election process. When a Bully message is received, the thread checks the message device id and its priority. If the received priority is considered better than the stored one, then the id of the current leader will be replaced by the id included in the received message. If the received priority is lower, the current thread will consider itself the new leader, by broadcasting a message with its id and its priority. If the received priority is lower than the stored one, and the thread is not the leader, the message is discarded and the thread will wait for a message sent by the leader.
- Leader liveness verifier: Due to the fact that the implemented algorithm works essentially by using timers, when a new leader is elected or a message of the current leader is received, each thread on each device will initialize a timer. If the timer goes off without a received message of the current leader, then the device will assume that the leader have failed and it is necessary to elect a new leader by starting the algorithm again. This thread is in charge of checking the timer and start the leader's election algorithm again in case the conditions require so.
- Keep alive sender: Once a leader has been selected, the network of devices reach a stable state where no election messages are sent. In order to keep this state as long as possible the selected leader must regularly broadcast keep alive messages, to let the other devices know that he is still alive and coordinating the work performed by the devices.

Due to the characteristics of the hardware, each device possesses two RGB leds. In order to identify which device has been elected as the leader, the two leds will be turned on, while the other devices will only have one led light on. Once the leader device has been selected, it will become responsible for coordinating the other devices in order to achieve a cooperative task. In this research work, the leader's objective defines the direction to be followed by each device, in order to cover as much area as possible. To achieve this task, the leader device has a list with the network addresses (IP) of the cars it "manages" and the directions each car is supposed to follow. Such directions are obtained by a magnetic compass installed in each device and they are sent by using multi-cast messages along with the IP device. The decision of using multi-cast messages is based on the fact that each device can get the information of the others and in the event of failure by the leader, the new leader counts on all the information on the rest of the devices without the need to request it from other devices.

V. Conclusions and Future Work

Distributed coordination algorithms have been widely used and implemented on distributed systems based on personal computers. Nowadays, with the advent of the Internet of Things and its major breakthrough in our daily activities, distributed algorithms based on such devices are gaining momentum.

In this paper, a classic election algorithm (the Bully Algorithm), has been implemented on a distributed system based on IoT device nodes. A detailed description of the selected hardware has been presented. Based on such selection, a distributed leader election algorithm in a wifi private network has been built. Such implementation has been tried out using

real robots built with the described hardware. Additionally, the multithreading features of the freeRTOS operating system supported by the hardware have been tested.

Future research work will encompass:

- The development of a more robust robot platform.
- The study of coordinated movement such as squadron movement coordinated by a leader device.
- The development of such concept on a based in Long-Term Evolution (LTE) network.
- The application on a real catastrophe scenario.

ACKNOWLEDGMENT

The physical support and the technical contributions of 'Computer Science Department of the Engineering Faculty of the University Of Buenos Aires' is highly appreciable. Without their care, it would have been impossible to reach the goal. We are also very thankful that this project has been developed with the support of Grupo de Investigación en Ciencias Informáticas (REINVENT) http://www.fi.uba.ar/es/node/537 at the Universidad de Buenos Aires.

REFERENCES

[1] Kevin Ashton. That 'internet of things' thing. *RFiD Journal*, 22(7), 2011.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] Richard Barry. Using the freertos real time kernel-arm cortex-m3 edition, 2010.

[4] Gang Liu Ben Zhang and Bi Hu. The coordination of nodes in the internet of things. *2010 International Conference on Information, Networking and Automation (ICINA)*, 2:V2–299–V2–302, 2010.

[5] Ahcène Bounceur, Madani Bezoui, Reinhardt Euler, Farid Lalem, and Massinissa Lounis. A Revised BROGO Algorithm for Leader Election in Wireless Sensor and IoT Networks. In *IEEE Sensors 2017*, IEEE Sensors 2017, Glasgow, United Kingdom, October 2017.

[6] Maryline Chetto. *Real-time Systems Scheduling*. ISTE. Wiley-ISTE, 1 edition, 2014.

[7] Hector Garcia-Molina. Elections in a distributed computing system. *IEEE transactions on Computers*, (1):48–59, 1982.

[8] Amitava Gupta, Anil Kumar Chandra, and Peter Luksch. *Real-Time and Distributed Real-Time Systems: Theory and Applications*. CRC Press, 2016.

[9] Yanjun Hu, Lei Zhang, Li Gao, and Enjie Ding. *Semi-Stochastic Topology Control with Application to Mobile Robot Team of Leader-Following Formation*, pages 125–134. Springer Berlin Heidelberg, 2015.

[10] Sivalasya Kasu, Larry Hash, John Marsh, Ronny Bull, et al. *Spanning Tree Protocol*. PhD thesis, 2015.

[11] Arun Kejariwal, Sanjeev Kulkarni, and Karthik Ramasamy. Real time analytics: algorithms and systems. *Proceedings of the VLDB Endowment*, 8(12):2040–2041, 2015.

[12] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[13] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, and Alex Marrs. *Disruptive technologies: Advances that will transform life, business, and the global economy*, volume 180. McKinsey Global Institute San Francisco, CA, 2013.

[14] Giorgio Buttazzo (auth.) Sanjoy Baruah, Marko Bertogna. *Multiprocessor Scheduling for Real-Time Systems*. Embedded Systems. Springer International Publishing, 1 edition, 2015.

[15] Gérald Santucci. The internet of things: Between the revolution of the internet and the metamorphosis of objects. *Vision and Challenges for Realising the Internet of Things*, pages 11–24, 2010.

[16] Christoph Sommer, Florian Hagenauer, and Falko Dressler. A Networking Perspective on Self-Organizing Intersection Management. In *IEEE World Forum on Internet of Things (WF-IoT 2014)*, pages 230–234, Seoul, March 2014. IEEE.

[17] Andrew S Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[18] Rob van der Meulen. Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016. *Gartner, Inc., https://www.gartner.com/newsroom/id/3598917*, 2017.

[19] E. Yanmaz, M. Quaritsch, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter. *Communication and Coordination for Drone Networks*, chapter 17, pages 77–91. Springer Cham, 2017.

[20] Joseph Yiu. *The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors*. Newnes, 2013.

[21] Matt Zwolenski, Lee Weatherill, et al. The digital universe: Rich data and the increasing value of the internet of things. *Australian Journal of Telecommunications and the Digital Economy*, 2(3):47, 2014.