PhoneSensing - Smartphone-based Application for Distributed Sensing in Human Networks

Nadir Türkman

Under supervision of Prof. Doutor Rui M. Rocha

IST - Taguspark, Porto Salvo, Portugal

May 13, 2012

Abstract

The evolution and proliferation of modern smart-phones has opened a door to a new type of Wireless Sensor Networks (WSN). While traditional WSNs consist of fixed sensor nodes, smart-phone based ones have a large mobility factor.

Modern smart-phones also possess a respectable amount of sensors which allied with their mobility and processing power are able to produce a large-scale Opportunistic Sensor Network (OSN). The sensors also provide developers with the possibility to create applications that improve lifestyle by collecting useful contextual information from the user's lifestyle. We propose to explore both the opportunistic and social potential by creating an application that uses an underlying OSN to improve the user's day to day personal and social experience.

To explore the opportunistic potential we propose a decentralized resource sharing protocol to apply to the underlying OSN. It will provide the possibility for mobile nodes to exchange resources and supply the necessary framework for the social sensing application.

Keywords: Smart-phone, Sensor, Sharing, P2P, Bluetooth, PhoneSensing

1 Introduction

WSN consist of sparsely distributed autonomous sensors that in cooperation monitor the surroundings they're adjacent to. WSNs have been used on a myriad of scientific fields on all sorts of applications but have struggled with inherent limitations commonly associated with the lack of resources that characterize the majority of sensor nodes. Such

limitations exist due to the fact that nodes that compose these networks are usually resource limited devices, with low autonomy and also low power and low bandwidth radio interfaces [1]. These limitations prevent large-scale deployments of the aforementioned networks and as such they are very useful in research contexts on closed-door laboratories and small scale experimentations. When ported to real case scenarios that require large deployments they become expensive and very high maintenance costs but it doesn't mean that they have not been used and that they haven't been extremely useful. They have been crucial on all sorts of fields ranging from domestic to professional and or industrial use.

Meanwhile modern wireless telecommunications have had an observable improvement regarding the used mobile devices. They started off with big cell phones that had just one simple function, and that was to make and receive calls. Later the possibility to send and receive text messages was added. As the time went on, phones morphed into Personal Digital Assistant (PDA) that incorporated more memory and better CPUs. This yielded a panoply of new applications that improved their usability. Since evolution never stops, eventually smart-phones appeared, and today's smart-phones are a long way from the behemoths that their ancestors where.

One of the particularities of modern smart-phones is the inclusion of an array of sensors that were at first intended exclusively for intended purposes so that some functionalities can be made possible [2]. A proximity sensor exists so that modern touch screens don't get activated when the user puts the phone close to the face while making a call. Orientation sensors and accelerometers are used to rotate the phone's screen every time the user rotates the phone itself but ended up also being used on other applications such as games. Light sensors we're added so that the phone could smartly manage battery consumption by deeming the lights when they were not necessary. Magnetometers and digital compasses were also throw into the mix. The camera has been a part of modern phones for a while now, and the microphone has been on phones since their birth.

With the inclusion of these sensors, smart-phones have become powerful mobile devices and it has not gone unnoticed. There has already been quite a bit of researching into the possibility of using these devices to deploy large scale WSNs [3] [4]. Since smart-phones are managed by their users they're usually kept charged and on for the majority of the day and besides packing quite a punch performance wise there are more than 400 million[1] of them worldwide.

Traditional WSNs are thought out before being

[1] http://gigaom.com/2010/03/24/mobile-milestone-data-surpasses-voice-traffic/?utm_source=gigaom&utm_medium=navigation

deployed and their structures are a byproduct of the project's motivations. In contrast, a WSN composed of smart-phones has no predefined structure, it's a sort of mutating, anarchic network where humans dictate how it morphs by moving about with their smart-phones. Smart-phone based WSNs have a large mobility factor when compared with the traditional WSNs composed by fixed nodes. Most of the work done on this field, but not all [5] [6], has been directed to the human component, mostly the social aspect of it. Social networks have been a growing industry and there is already been a modicum amount of research into the possibility of enrichment of the social networking experience [7] [8] [9] by introducing the capacities of modern smart-phones and WSN composed by them. This is commonly referred to as social sensing.

What we have here is, in theory, the largest WSN waiting to be explored like any other WSN. In [10] this is called People-Centric Sensing since sensors are not deployed in a traditional form, rather than on trees or buildings, humans become the focus area of sensing. Consequently applications thought out for this network will also be aimed at the human kind with the goal of improving life quality on a day to day basis by providing useful information that would otherwise go unnoticed.

One other aspect that should also be mentioned is opportunistic sensing. Such concept is already very useful on typical WSN and can also be used on the context of people centric sensor networks. In [10] opportunistic sensing is regarded as the act of pervasively gathering information, that is, the phone will sense the environment without any necessary input from the user. For the remainder of this document, we will regard opportunistic sensing as defined in [11], where it defines opportunistic sensing has the act of tasking an application request for a given sensor in a remote device within a preferred time frame. This translates to the possibility of a less capable device tasking another device, that has the desired sensor, in the surrounding area to collect data from a sensor that would otherwise be unavailable.

We propose a solution that explores the aforementioned possibilities of a smart-phone-based sensor network. This shall be done by creating a framework that exploits opportunistic connections with other passing devices. This should help improve sensing accuracy and in some cases, more than improve since these opportunistic connections might lead to resource sharing. A solution that overcomes inherent limitations of less capable devices. In order to test this framework we also propose a possible application that will use the aforementioned framework to provide the user's with an improved social experience.In order for that to be possible the devices must be able to process and evaluate sensor data in order for it to be possible to infer contextual information

The remainder of this extended abstract is organized into three main sections. Section 2 provides an overview of the implemented architecture, both for the framework as for the social sensing application. Section 3 details the testing process during the resource sharing protocol development and also the validation process of a case scenario. Last section, section 4, summarizes and analyzes the obtained results drawing some final conclusions.

2 Architecture

The are two different approaches for the resource sharing protocol, centralized or decentralized. In [12] [13] the authors propose a centralized solution and in [14] they propose a hybrid solution but with the main focus on a decentralized resource sharing protocol for OSNs. We approach resource sharing with a purely decentralized solution.

Figure 1 is an overview of the implemented architecture. It shows the two main interest groups. The decentralized communication between mobile devices and the centralized communication used for logging and result analysis.

They communicate amongst each other using bluetooth and with the centralized Transmission Control Protocol (TCP) server by way of Wi-Fi. Inter-Device communications are done with Bluetooth because it is the only resource available that allowed ad-hoc connections. Due to driver restrictions, the smart-phones used during the development don't yield Wi-Fi ad-hoc connections.

Also, using bluetooth left the Wi-Fi radio interface free for the centralized communications with the TCP server. It would have been very complex to schedule the use of the WiFi for both protocol communications and for logging purposes.

The application will be the tool used to prove that the concept works. The idea behind it is a simple application that has various configured profiles. Each profile is suited to a different situation, and the sensors gather information process it, and decide wether the surroundings are suitable for the selected profile. This pertains to the social sensing part of the solution.
When a device is less capable than others, and doesn't have a required sensor it would then use the underlying OSN composed of other devices running the same application, to acquired said resource.

The application uses a framework that consists of functions that implement the OSN's communication protocols. The framework can be used as a base for any other that would find it useful to use the resource sharing protocol to enhance it's capabilities.

Figure 2 shows a global view of the architecture's stack structure and its communication paths between modules. The Application Interface and the

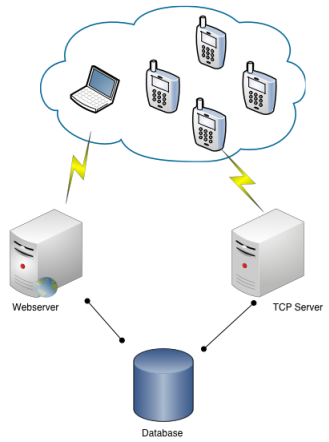

Figure 1: Architecture Overview

Application Core are relative to the social sensing application. All other modules refer to the application's support framework.

The Application Core Module (ACM) sits in between the user interface and framework modules. It has the job of orchestrating all the applications background operations. These background operations include handling results from the Sensor Request Manager Module (SRMM), deciding what's the process next in the algorithm, user interface flux and settings management.

The rest of the blocks are responsible for implementing the resource sharing protocol and compose the OSNs framework that the ACM will use to enhance it's user experience. The framework implements a resource sharing protocol that we envisioned. The selected protocol was chosen as a result of a series of tests ran on the ONE simulator [15] agains two other protocols.

The goal of the protocol is to in the end receive data from a sensor that isn't available locally at the device. Hence, a request reply handshake must exist. The flux of sending a sensor request and receiving a sensor reply is unavoidable. All three protocols share that phase of the message exchange. All three don't resend requests or replies if the message was successfully delivered. Messages are only resent if they failed to reach the destination. They all have a Time To Live (TTL) value associated, this is so if a request reaches the destination and the destination goes out of range for a long time, the reply isn't sent when / if they connect in the future. None of the protocols use ACK or NACK messages. The sender is always blind, it never knows if the messages where successfully sent.

The one that got the best results is called Multiple Active Request (MAR) protocol.The MAR protocol does not skip the handshake process. When a device requires a sensor it does not have he sends a Capability Response Message (CRESM) to all neighboring devices. It then waits for a CRESM.

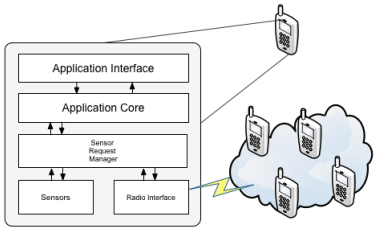

Figure 2: General view of the proposed architecture

Upon receiving one it checks if the device has the desired sensor. If this is the case, then a Sensor Request Message (SREQM) is issued to that device. It keeps issuing SREQM for every CRESM received.

In the case the remote device does not have the desired sensor then that message is ignored, and the device is flagged as not having the required sensor. Other CRESM will be accepted.

The protocol converges when the device receives the Sensor Response Message (SRESM). After that, all other messages referring to that request are ignored.

In order to keep exchanged messages small the protocol uses RDL. RDL is a unified description language for network resources. It is able to describe a resource both qualitatively as quantitatively [16].

The RDL API exposes two formats to describe the resources, XML and KLV. The KLV format is very lightweight, as a matter of fact due to it's size it can consume as little as 14 times less than the counterpart format [16].

3 Tests and Validation

The tests are divided up into two parts. The first, are the simulator tests that we the base for the protocol choice. In order to study complex scenarios simulator tests were performed in the ONE simulator. The main goal in the protocol testing is to ascertain which of the three is the best all-rounder. It is not required for the protocol to be the best in all tests, but to be the one that suits the best in a vast majority of situations. The tests were prepared so that they represented most of the situations in daily use.

The second, is the case study test. The main goal behind the case study tests is to analyze how the chosen protocol performs after it's implementation compared to the theoretical results ran in the simulator.

3.1 Protocols

On all test scenarios only one of the protocol was able to perform better on every request time completion test, and that was the MultipleActiveRequest. It was outperformed by both other protocols when it came to the number of messages

created and total bytes transferred. The SensorOnlyRequest performed very well on both those tests, coming first on all test scenarios. It didn't perform so well in the tests where the nodes weren't stationary, it was the only protocol to have request timeouts.

The decision on which protocol to chose would always be between the SensorRequestOnly and MultipleActiveRequest, since the SingleActiveRequest didn't out perform any protocol. As mentioned at the beginning of this chapter, the main choosing criteria is the request completion times, and hence the chosen protocol is the MultipleActiveRequest. On another network where time wasn't so critical, the SensorRequestOnly would be the better choice. Apart from the timeouts when mobility was a factor, it would still be a very good choice. Convergence times when compared to the MultipleActiveRequest weren't so different to make it an obvious decision.

3.2 Case Study

The scenario reflects the underlining idea that the PhoneSensing represents. It represents an social sensing application that would help the user augment their activities by being aware for the surroundings ate letting the user know when they are not optima.

For example, would the user be at a coffee place working on he's dissertation the profile would be setup so that it would trigger an event when the lighting and/or conditions were not optimal. If such an event was triggered it would try to find a suitable place nearby using the GPS. If the GPS was not available it would use the sensor sharing framework to request it from any neighboring devices using the same applications. Had the application have connectivity with social networks it could also post that the user was about to change the location where he was studying.

The test scenario we present is the most basic possible. It's composed of two devices running the phone sensing application. One has a GPS sensor available to share, and the other has a studying profile available with a light threshold of 90%, so the lightest dimming of the lights will trigger an event to change location. That will require a GPS, which the device doesn't have.

Figure 3 shows the duration of each process since the device requested the GPS sensor so that he could chose a new location to go to. It took 27s from the discovery process until the device receiving the SRESM. The first important thing to point out, is that each Discovery Process will take exactly 10s, to return all the available devices. This time doesn't include the time it takes to cycle through every device and identify the ones that actually run the PhoneSensing application. At a first glance it takes much, much longer than the simulator to handle a single request. In the first test scenario described in the previous section, the longest request took 0.09s, and this was with other devices requesting at the exact same time. So the first

ID	ID App Req	Sensor Request	Device	Phase	Date
1	qcg4ut8ll5k0pabpv52vja1aeu	GPS	a5.alpha	DISCOVERY_START	29-04-2012 08:09:31
2	qcg4ut8ll5k0pabpv52vja1aeu	GPS	a5.alpha	DISCOVERY_END	29-04-2012 08:09:41
3	qcg4ut8ll5k0pabpv52vja1aeu	GPS	a5.alpha	PROTOCOL_START	29-04-2012 08:09:41
4	qcg4ut8ll5k0pabpv52vja1aeu	GPS	a5.alpha	PROTOCOL_END	29-04-2012 08:09:58

Figure 3: Protocol Execution Overview

ID	Log Type	Log Msg	Device	Date
3	DEBUG	Starting MultipleActiveRequest protocol	a5.alpha	29-04-2012 08:09:31
4	DEBUG	Starting Discovery Process	a5.alpha	29-04-2012 08:09:31
5	DEBUG	Discovery Process Finished	a5.alpha	29-04-2012 08:09:41
...	...	...	...	...
8	DEBUG	Sending Capability Request Message (CREQM) to a5.beta	a5.alpha	29-04-2012 08:09:57
9	DEBUG	Received CREQM from a5.alpha	a5.beta	29-04-2012 08:09:58
10	DEBUG	Sending CRESM to a5.alpha	a5.beta	29-04-2012 08:09:58
11	DEBUG	Received CRESM from a5.beta	a5.alpha	29-04-2012 08:09:58
...	...	...	...	...
18	DEBUG	Sending SREQM to a5.beta	a5.alpha	29-04-2012 08:09:58
19	DEBUG	Received SREQM from a5.alpha	a5.beta	29-04-2012 08:09:58
24	DEBUG	Received SRESM from a5.beta	a5.alpha	29-04-2012 08:09:58

Table 1: Scenario 1 Debug Table

conclusion that can be taken is that the simulator is far from yielding realist results. But that will be analyzed a bit more later.

Table 1 is an extract of the debug output generated by the devices during the test scenario. Has already mentioned, the process from the discovery until the SRESM took 27s. During the simulator tests the time logged in didn't count with the discovery process, because there wasn't any thing of the sort, the devices always knew at all times the surrounding devices. In this real world scenario, even excluding the discovery process, the message exchange takes much longer. In the simulator the longest request delay was 0.09 seconds. Other requests about halved that time.

Look at the table 1 another process that seems to take along time is the filtering process. Since the discovery process finishes until the first request is sent takes about 16s. This filtering process was one of the first challenges mentioned during the implementation process. This time will vary depending on how many bluetooth devices are around. During this test two more devices that weren't running the PhoneSensing application were returned in the discovery process. So adding up the discovery process and filtering process take about 96% of the duration of the process.

A big drawback to this is that during this 26s the device is unaccessible to answer to other requests because when it's in Discovering mode the bluetooth doesn't accept any other requests. During the filtering process it will be establishing connections with other devices. So in a high density node

situation devices would struggle to establish any connection at all due to the out time that each of them has to endure during the discovering process and filtering process.

4 Conclusions

This project proposed a decentralized solution for sensor sharing on a OSN. A decentralized approach allows for large-scale OSNs that don't require any extra infrastructural deployment. The motivation behind this is that by simply installing software on a mobile device, e.g. smart-phone, they become capable of sharing resources amongst them. With such a framework it is possible to deploy Social Sensing applications on mobile devices without them having to have a minimum number of required sensors. To prove that the concept works it was proposed to develop a Social Sensing application.

To achieve the proposed goals we started by creating a resource sharing protocol that would fit the project specifications. The protocol focused on fast convergence times to ensure that the protocol would work in scenarios where nodes were mobile and on small messages to ensure that the medium would not become saturated. Three different protocol's were thought up and put through various test scenarios using the ONE Simulator in order to select the one that guaranteed a high success rate in shorter time frames. To achieve small message sizes the protocol uses RDL to describe the available resources upon a device being queried. RDL is also used when requesting a sensor.
To test the protocol the Social Sensing application for smart-phones was developed. The message exchange was done using the smart-phones bluetooth radio interface in a P2P fashion guarantying the decentralized architecture requirement. The application worked as a proof of concept and also made it possible to test the protocol and compare the results with the previously ran simulations.

In the end the protocol was successfully implemented and the original idea of a decentralized OSN was maintained. It was shown that it is possible for such a concept to work on a real world situation, and that it does help support a social sensing application, or any other application that would benefit from resource sharing. The messages exchanged were small enough to be sent in acceptable times frames through the bluetooth interface. The simulation times diverged from the ones observed during the real case scenario tests, but not due to bad implementation but due to inherent limitations to the hardware in hand.

References

[1] G. Pottie, "Wireless sensor networks," in *Information Theory Workshop, 1998*, 1998, pp. 139 –140.

[2] H. e. a. N. Lane, E. Miluzzo, "A survey of mobile phone sensing," in *IEEE Communications Magazine*, 2010, pp. 140–150.

[3] B. P. Pál, "Sensorplanet: An open global research framework for mobile device centric wireless sensor networks," in *New York*, 2007, pp. 2006–2008.

[4] E. Paulos, R. J. Honicky, U. C. Berkeley, and E. Goodman, "Sensing atmosphere," in *Interface*, 2007, pp. 9–11.

[5] N. D. Lane, D. Lymberopoulos, F. Zhao, and A. T. Campbell, "Hapori: context-based local search for mobile phones using community behavioral modeling and similarity," in *Proceedings of the 12th ACM international conference on Ubiquitous computing*, ser. Ubicomp '10. New York, NY, USA: ACM, 2010, pp. 109–118.

[6] D. Peebles, H. Lu, N. D. Lane, T. Choudhury, and A. T. Campbell, "Community-Guided Learning : Exploiting Mobile Sensor Users to Model Human Behavior," *Computer*, 2008.

[7] A. T. Campbell, S. B. Eisenman, K. Fodor, N. D. Lane, H. Lu, E. Miluzzo, M. Musolesi, R. A. Peterson, and X. Zheng, "Transforming the social networking experience with sensing presence from mobile phones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ser. SenSys '08, New York, NY, USA, 2008, pp. 367–368.

[8] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 337–350.

[9] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*, ser. MobiSys '09. New York, NY, USA: ACM, 2009, pp. 165–178.

[10] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn, "The rise of people-centric sensing," *IEEE Internet Computing*, vol. 12, pp. 12–21, 2008.

[11] S. Eisenman, N. Lane, and A. Campbell, "Techniques for improving opportunistic sensor networking performance," in *Distributed Computing in Sensor Systems*, ser. Lecture Notes in Computer Science, S. Nikoletseas, B. Chlebus, D. Johnson, and B. Krishnamachari, Eds. Springer Berlin / Heidelberg, 2008, vol. 5067, pp. 157–175.

[12] S. Eisenman, H. Lu, and A. Campbell, "Halo: Managing node rendezvous in opportunistic sensor networks," in *Distributed Computing in Sensor Systems*, ser. Lecture Notes in Computer Science, R. Rajaraman, T. Moscibroda, A. Dunkels, and A. Scaglione, Eds. Springer Berlin / Heidelberg, 2010, vol. 6131, pp. 273–287.

[13] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "Bikenet: A mobile sensing system for cyclist experience mapping," *ACM Trans. Sen. Netw.*, vol. 6, pp. 6:1–6:39, 2010.

[14] S. B. Eisenman, "People-Centric Mobile Sensing Networks," Ph.D. dissertation, Columbia University, 2008.

[15] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[16] A. C.Santos, L. D. Pedrosa, and R. M. Rocha, "RDL: A unified description language for network resources," *RTCM seminar*, March 2010.