Resource Footprint of a Manycast Protocol Implementation on Multiple Mobile Platforms

Ekhiotz Jon Vergara*, Simin Nadjm-Tehrani*, Mikael Asplund* and Urko Zurutuza†
*Dept. of Computer and Information Science, Linköping University, Sweden
{ekhiotz.vergara,simin.nadjm-tehrani,mikael.asplund}@liu.se
†Dept. of Computer and Information Science, Mondragon University, Spain
uzurutuza@mondragon.edu

***Abstract*—Wireless communication is becoming the dominant form of communication and ad hoc wireless connections are posed to play a role in disaster area networks. However, research efforts on wireless ad hoc communication protocols do not pay enough attention to measurable and reproducible indications of the mobile footprint including power consumption. Protocols and applications are initially designed and studied in a simulation environment and are hard to test in in-field experiments. In this work we report a multi-platform implementation of Random-Walk Gossip, a manycast protocol designed for message dissemination in disaster areas. Our work is focused in studying the resource footprint and its impact on performance on commercially available devices. We show both how different aspects of the protocol contributes to the footprint and how this in turn affects the performance. The methodologies used here can be applied to other protocols and applications, aiding in future optimisations.**

***Keywords*-resource footprint; energy; mobile ad hoc networks; disaster area networks; multi-platform;**

I. Introduction

Wireless communication is becoming the dominant form of communication and taken for granted in both urban and rural areas. For example, in countries like USA or Ghana there are more mobile subscribers than telephone main lines [1]. However, just as the wired networks, wireless communication is dependent on existence of an infrastructure, i.e. cellular technology or access points for WiFi in combination with a wired Internet core. When a disaster strikes, this infrastructure is typically rendered useless (e.g., wiped out or severely overloaded). Thus, any infrastructure-less mode of communication is worth exploring as a possible solution in such scenarios.

Rescue professionals typically rely on special equipment like satellite phones or tactical radios, not available for volunteers or victims located at a disaster area. As these equipments are scarce and expensive, new approaches to complement information exchange are proposed. Mobile ad hoc networks (MANET) are envisioned to allow communication [2]–[4] without any infrastructure. However, due to node mobility and sparse topology, connectivity disruptions can occur leading to intermittently connected mobile ad hoc networks (IC-MANET).

Mobile communication with handheld devices, and IC-MANET communication in disaster scenarios in particular, are naturally focused on limited resources. This would appear to make a case for extensive studies of the resource consumption characteristics of protocols and applications and their impact on performance aspects such as response time. However, we find that neither the massive volume of research on communication protocols nor the emerging focus on mobile applications pays enough attention to measurable and reproducible indications of the mobile footprint. Most protocols are initially designed and studied in a simulation environment, and at best, their resource consumption is based on rough indications of the aggregated signalling overhead.

This paper is a first attempt at studying the resource footprint of a protocol that has been proposed as an energy-efficient dissemination protocol in IC-MANET. First, the protocol had to be implemented on a range of wireless devices to study their comparative resource footprints. Then we studied the basic resource characteristics of the protocol as well as their interdependencies and implications on performance. We believe that the methods in this study can act as a base for further systematic evaluations of resource footprint and performance for optimisations of mobile applications and protocols.

The contribution of this paper is two-fold: (1) a multi-platform implementation of the existing manycast protocol Random-Walk Gossip (RWG) designed for disaster areas [5] and (2) the evaluation and analysis of the resource footprint in terms of energy consumption, CPU usage and memory consumption. This analysis presents resource footprint interdependencies as well as the impact on performance in terms of response time.

The rest of this paper is structured as follows. Section II briefly reviews the fundamental features of the Random-Walk Gossip protocol. Section III covers the architecture and design of the protocol implementation. Section IV presents the evaluation of the protocol resource footprint and performance on various hardware platforms. Finally, the related work and conclusions are presented in Section V and Section VI respectively.

II. Random-Walk Gossip

Random-Walk Gossip [5] is an efficient manycast protocol designed for message dissemination in IC-MANET. This section describes the basics of the operation of the protocol, explaining its characteristics and terminology.

The protocol is designed to deal with the challenges that a disaster area scenario presents. Since RWG does not need any pre-existing infrastructure, the nodes can exchange messages without having a priori knowledge of the network topology. The goal of the protocol is to disseminate the message to k nodes, where k is decided by the sender. When the message has been received by k or more nodes, the message will be k-delivered and nodes will stop sending it. RWG uses a store-carry-forward like mechanism in order to disseminate the messages overcoming network partitions.

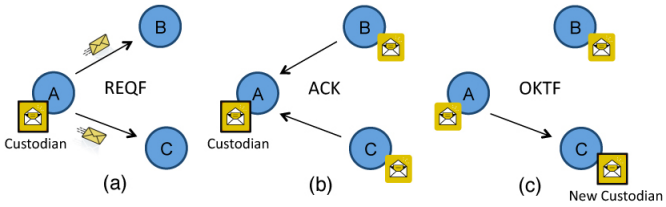

Figure 1: Basic packet exchange in Random-Walk Gossip.

The basics of the message exchange are shown in Fig. 1. When a node starts disseminating a certain message, it sends a *Request forwarding* (REQF) as depicted in Fig. 1(a), which is the only packet type carrying the data of the application. A node issuing a REQF is called *custodian* and is in charge of spreading the message in the network. In Fig. 1(b), nodes B and C store a copy of the message and reply by sending an *Acknowledgement* (ACK), whereby node A becomes aware of which nodes are in its vicinity. Node A randomly chooses one of the nodes among the nodes that have sent an ACK and sends an *OK to forward* (OKTF) as in Fig. 1(c). The intended receiver of the OKTF (node C) will be the new *custodian* of the message, starting a new random-walk by sending a REQF. The other nodes will silently keep an inactive copy of the message.

The header's structure of RWG is the same for all the packets. Each message is identified by the concatenation of a sequence number and the source address of the sender. The *informed* field is a bit-vector which indicates the number of nodes that are informed about a certain message. A standard hash function is used to map a node to a bit, thus a node knows whether a message has been received by a certain node or not. A message will be deleted if it is k-delivered or if its maximum time to live has expired. There are also some further optimisations in addition to the active spreading phase, such as a rejuvenation mechanism to make sure that a message does not get stuck in a dead end during the random walk, as well as the fact that a node will send a message every once in a while to avoid the network becoming permanently silent.

III. Design and Architecture

The implementation relies on the IEEE 802.11 technology in ad hoc. Bluetooth, as an alternative, would oblige the nodes to establish a connection before sending any packet, restrict the number of connected nodes to 10, and not allow sending a broadcast message to all the neighbours [6].

The implementation was developed using Portable Operating System Interface (POSIX). Since in Symbian OS only a subset of POSIX libraries was available, the development of the protocol was focused on that subset. This implies that it also works on the other platforms with few adaptations. However, this also entailed overcoming the limitations of the Open C plug-in, e.g. the logic of the algorithm is distributed in 5 threads instead of 2 of the original design due to the impossibility of using signals.

Fig. 2 describes the architecture which is briefly explained below. Using this architecture we have implemented three versions of the protocol that were used in the analysis of Section IV.

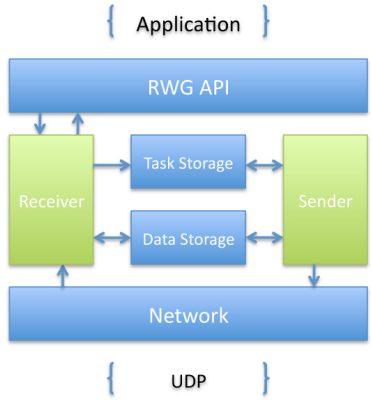

Figure 2: Architecture overview.

DataStorage: every node has to keep a copy of the original message. For that purpose, every message is stored in an object (*REQFobj*) that provides functions to access the needed information and perform operations of the protocol. *Data Storage* contains the *REQFobj* that are still alive. The ID and time to live of each message is used to sort and store them efficiently, providing logarithmic access time and avoiding duplicated messages. The *Data Storage* consists of a map and a multimap of the Standard Template Library.

TaskStorage: the RWG protocol works in an event-based manner, scheduling procedures that have to be executed at certain times. This behavior is implemented by the *Task Storage*, which contains the procedures in objects called *Task*. The *Task Storage* consists of two multimaps of the Standard Template Library.

Table I: Devices used to perform the tests.

	CPU	Operating System
Nokia N97	ARM 11 @ 434 MHz	Symbian OS v9.4
Nokia 5800	ARM 11 @ 369 MHz	Symbian OS v9.4
MacBook	Core 2 Duo @ 2.4 GHz	Mac OS X 10.5.8
Dell Latitude E5400	Core 2 Duo @ 2.54 GHz	Ubuntu 9.10
Asus EeePC 901	Atom N270 @ 1.6 GHz	Ubuntu 9.10

Network: this module is a wrapper of network functionalities, containing UDP sockets to create the connection to the default interface. It retrieves the MAC address of the node and provides the functionalities of sending and receiving broadcast packets.

RWG API: it provides the application functions to interact with the protocol. The application and the protocol exchange different types of messages using two named pipes in order to achieve full duplex communication.

Receiver: the receiver threads take care of the new messages arriving from the application and network. If a new message arrives from the network it is delivered to the application.

Sender: the sender threads are in charge of sending the responses. They also ensure that the network will not become permanently silent by sending a message when there has not been any packet exchange for a while.

IV. Analysis of resource footprint

The first implementation of RWG was developed and evaluated in NS-3 [5]. However, in the simulation environment nodes have unlimited resources in terms of CPU and energy. Radio range and transmission power are parameters of the nodes, which are based on assumptions that make simulation results differ from field tests [7]. This section describes how the resource footprint was studied in real devices.

Before proceeding with the results of the experiments, we will first describe the hardware platforms and the data collection tools. Table I presents the devices used to perform the experiments, ranging from smartphones to laptops with different CPU, memory and operating systems.

Wireshark and KisMac were used to analyse the packet exchanges of the protocol and the WLAN networks respectively. Nokia Energy Profiler (NEP) is a stand-alone test and measurement application for Series 60 3rd Edition and later devices that was used to gather CPU and energy related data in Symbian. The Activity Monitor of Mac OS X provided the CPU load of the MacBook.

The following tools were developed for gathering and analysing the data: *RWGClient* is a command line client application used to isolate the performance with respect to communication layers only. *RWGChat* is a chat user application used in order to test the protocol from the point of view of the end user. It was developed using the Qt framework, offering portability between the three platforms. The implementation of the protocol daemon creates a message log in each node, gathering the relevant data for each message.

Regarding the settings of the protocol, the k value was fixed to 10 in order to avoid the messages being k-delivered and deleted during the experiments (which were performed in test beds with fewer devices). For all the experiments except for the message deletion, the time to live was fixed to 120 seconds. This value was big enough to avoid the deletion of the messages during the experiments. The length of the informed vector was set to 2 bytes for all the tests.

A. CPU load

High CPU load leads to high energy consumption. The implementation of the protocol is designed to economise on CPU usage. We performed the tests using NEP in a Nokia N97 and the Activity Monitor in a MacBook, studying the CPU load in idle state and while sending data at different rates. Since the Activity Monitor provides the CPU load per process, the data was gathered monitoring the RWG process. NEP provides the percentage of the CPU load of the whole system every 0.25 seconds with integer precision. *RWGClient* was running on top of RWG in both the Nokia N97 and in the MacBook. For each data rate and device, the number of messages sent was the one needed to keep a constant transmission during one minute (e.g., 60 messages for 1 message per second) and the maximum CPU load was recorded. The experiment was performed in office environment on campus. The results are described below:

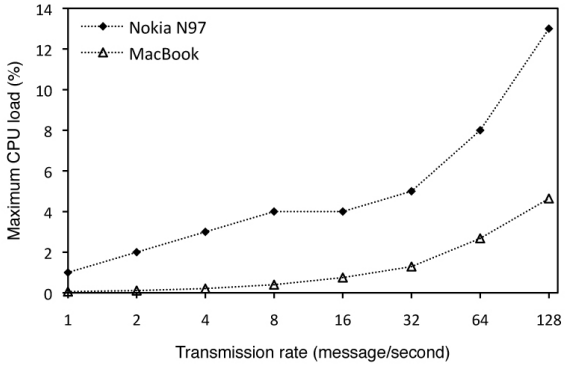

Figure 3: Maximum CPU load due to sending for different transmission rates in logarithmic scale.

MacBook: the CPU load in idle state is 0%. However, every once in a while a thread is woken up sending a stored message if there is any. This is part of the RWG mechanism to avoid the network becoming permanently silent and it is reflected in the results, increasing the CPU usage to 0.01% when the buffer is empty and to 0.03% when it contains messages. Fig. 3 shows the maximum CPU load for each

sending rate. The CPU load is increased to 4.64% when 128 messages per second were sent.

Nokia N97: the CPU load in idle state is between 0 and 1% [1]. Fig. 3 shows that even when the sending rate is 16 messages per second the CPU usage is close to an acceptable 4%. All data was gathered with the screen switched off to avoid the increase in CPU load due to rendering content on the screen.

As the sending rate increases, the protocol performs more operations, so the increase in CPU load is logical. As expected, the CPU load in the Nokia N97 is higher than in the MacBook due to CPU characteristics. However, given that the x-axis is in logarithmic scale the increase is still modest showing good scalability properties.

Note that increasing the number of neighbouring nodes increases the number of ACKs that the sender has to process, and therefore its CPU load. This increase was measured on the Nokia N97 for the case of 4 messages per second. The maximum CPU load of the sender with only one receiver was used as baseline. The average CPU load increase from the baseline is 3% and 7% when there are 2 and 3 neighbouring nodes respectively.

B. Timing measurements

In this section the relationship between the resource footprint and the performance in terms of response time of different devices running RWG is measured and studied.

All the experiments were performed sending messages from a MacBook running RWG and some other device was placed within radio range of the MacBook as the receiving node. The experiments were performed outdoors on the university campus using the same physical setup and the measured device was placed at 10 metres from the MacBook. The experiments were performed measuring one device at a time. The round-trip time (from the sending of a REQF to the reception of the ACK) was measured using Wireshark on the MacBook.

The first experiment measures the average response time of different devices running RWG. The test was performed using both the *RWGClient* and the *RWGChat* in every device. The transmission rate was 1 message per second and 100 messages were sent. The average response time is shown in Fig. 4.

One can note that smartphones are slower, and that the Nokia N97 is a little bit faster than the Nokia 5800 XpressMusic. The difference is believed to be due to the processor type. The response time of RWG in all the devices is slower using the *RWGChat* application. The third experiment studies this in detail (see below). Surprisingly, the response time of the Dell platform is longer than the EeePC platform. Therefore, we performed a second experiment to

[1]Every 6 seconds, whether the device is sending or not, there is a spike of 8% due to the MAC layer in ad hoc mode that we have ignored here.

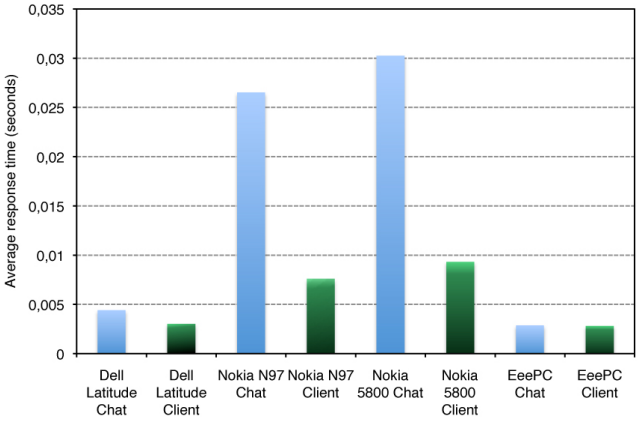

Figure 4: Average response time running RWG.

investigate whether the difference was due to the protocol or the device. The average round-trip time over 100 packets without running the protocol was 0,93 ms for the EeePC and 1,74 ms for the Dell, proving that the difference in the previous experiment was due to device characteristics and not due to the protocol.

The third experiment studies the performance of the protocol in terms of response time at different transmission rates in the EeePC and the Nokia N97 devices. For every transmission rate and device, the MacBook sent 100 messages measuring the average response time of the receiver device placed at 10 metres from the MacBook. Since the response time of the chat application is longer than the *RWGClient*, its behaviour was also tested in this experiment.

Fig. 5(a) shows that the response time of the protocol with the *RWGClient* remains constant in the Nokia N97. Somewhat surprisingly, the response time of the protocol in the EeePC with the *RWGChat* is almost constant whereas in the N97 increases significantly.

The explanation is that the chat application renders the content on the screen, which is CPU consuming. In Fig. 5(b) the CPU load at different transmission rates is shown. One can note that the CPU load of the N97 running RWG with *RWGChat* rises close to 100% during the experiment, with a big impact on the performance in terms of response time. These experiments have shown that transmission rate indirectly affects the response time through the increase in CPU load.

C. Energy consumption

This section describes the experiments that were performed in order to analyse the energy footprint of the RWG protocol in a given device. We know that display, radio transceivers and CPU are the main causes of battery discharge. We performed the experiments on Nokia N97 since it can be used with an accurate measurement tool (NEP) [8]. We used the *RWGClient* on top of the standard

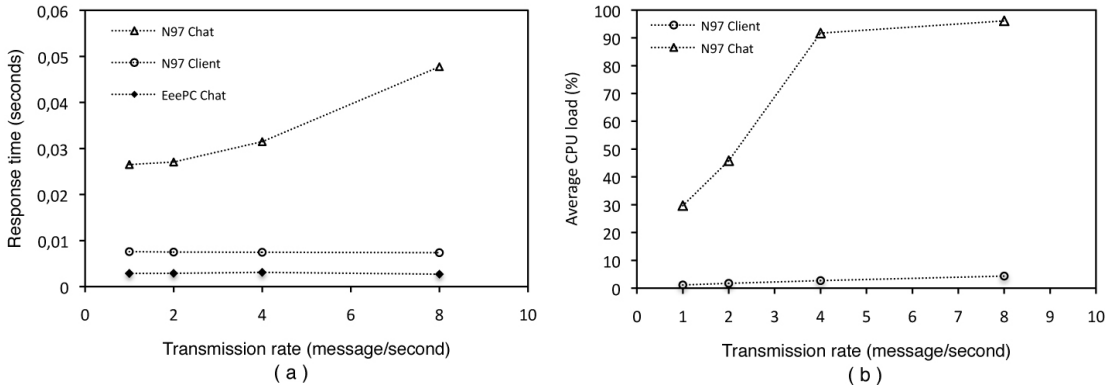

Figure 5: Response time and average CPU load for different devices at different transmission rates.

software on the device and then tried to isolate the impact of the protocol operation as follows.

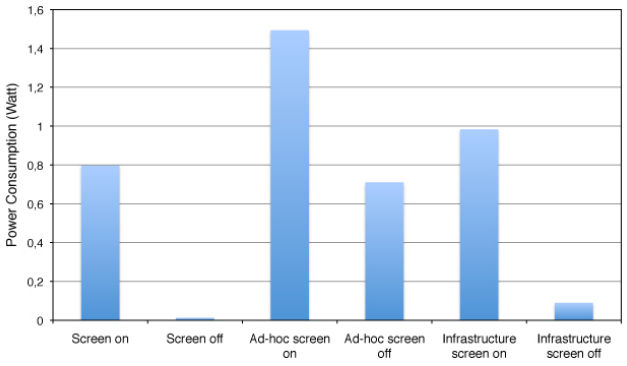

Figure 6: Power consumption of Nokia N97.

First, we measured the power consumption on the Nokia N97 in different states (the protocol was not running). Fig. 6 shows in the two leftmost bars that switching on the screen consumes 0,79 W. The content of the screen was the application menu, which does not have any graphical activity. The implementation of the protocol uses the WLAN interface in ad hoc mode, which means that the power consumption will be around 0,7 W when the protocol is running. This is shown by the 4th bar from the left in Fig. 6. Note that the ad hoc mode consumes more energy than infrastructure since the node is listening to the channel all the time and uses less power saving mechanisms. This is shown in the rightmost bar.

Second, the energy consumption when running the protocol was studied in the following experiments. The *RWGClient* was running on top of the protocol and as stated before it was using the WLAN interface in ad hoc mode. The transmission power was 100 mW by default.

In idle state, without sending any data, the most noticeable increase in power consumption was due to using the WLAN interface. As stated in Subsection IV-A, the CPU usage in idle state is 0% and it increases to around 1% when the mechanism that sends a message every once in a while performs its duty. The following tests verify the impact of that mechanism in idle state. First, RWG was running in idle state without sending any message and the battery (1500 mAh) was discharged after 7:27 hours. In the second experiment, the protocol sent a message every second and the battery lasted 7:18 hours. Therefore, we can conclude that the impact of the mechanism on the lifetime of the idle state is only 2%, not affecting significantly the energy consumption. Consequently, the impact of the protocol on the consumed energy in idle state is due to the use of the WLAN interface.

In operation state, two Nokia N97 were used and a message was sent from one device every second. Our intuition was that the use of more memory can lead to more CPU load, which consumes more energy. Therefore, the test was performed with the message buffer of the phones empty as well as with 500 messages to show the impact of message storage on energy consumption. The energy consumption difference was not significant. Thus, we conclude that the implementation of RWG handles the messages in an energy-efficient manner.

Third, the power consumption increase due to data rate was tested. The consumed power in the idle state (WLAN active in ad hoc mode) was taken as reference value. The *RWGClient* was used to send messages at different transmission rates. The size of the packets was 98 bytes, including MAC, IP, UDP and RWG headers. The average increase in consumed power of the sending period is shown in Fig. 7(a), which shows that, as expected, the consumed power increases when the message transmission rate increases. However, the average increase in consumed power is very small in comparison with the 0,7 W for having the WLAN active in ad hoc mode.

Finally, the average increase in consumed power of some CPU demanding operations was tested. One of the most consuming operations is deleting many messages from the

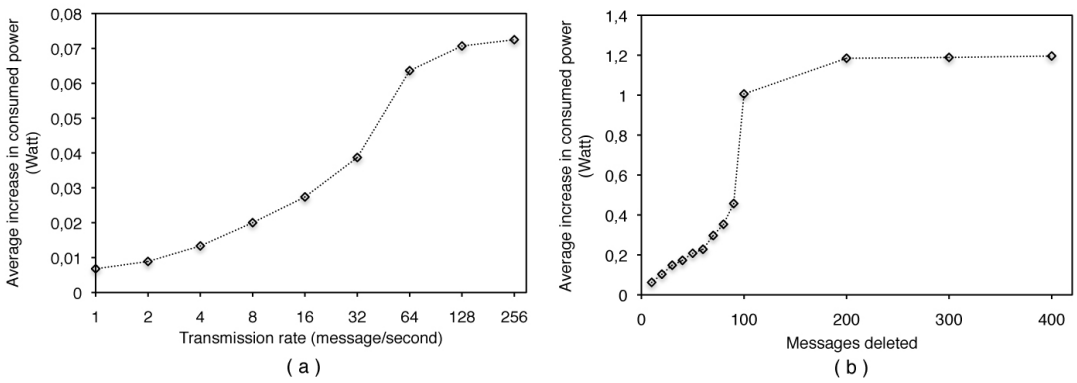

Figure 7: Average increase in consumed power due to (a) data rate in logarithmic scale and (b) a CPU demanding operation. The reference value is the consumed power in protocol's idle state.

buffer at the same time. The test consisted of deleting different number of messages from the buffer at the same time when their TTL expired. The buffer was filled with the intended number of messages and deleted by the message deletion operation. The TTL of the messages was fixed to 1 second. Fig. 7(b) shows that the increase in consumed power converges to a maximum when deleting more messages. This maximum is reached when the CPU load is 100%. Even though the consumed power level is the same deleting 300 and 400 messages, the latter consumes more energy due to the deletion operation takes longer. Note that with the radio being on by default due to the ad hoc mode, the quantitative increase is larger due to higher CPU usage (Fig. 7(b)) than due to higher transmission rate (Fig. 7(a)).

To summarise, one could conclude that the energy consumption footprint of the implementation of the RWG protocol is mostly due to the use of the WLAN interface in ad hoc mode, which is more significant than the other aspects.

D. Memory consumption

Since Nokia N97 was the most constrained platform in terms of memory, these tests were performed in that context and NEP was used to gather the data. The Nokia N97 has 128 MB of RAM of which some part is already used by Symbian OS and its components. The amount of free memory in the Nokia N97 was around 60 MB.

The increase in allocated memory of the system when the protocol and the application ran was studied. The average of 10 measurements shows that the memory usage of the minimum configuration to use the RWG protocol (*RWG + RWGClient*) is 1000 KB, whereas the memory usage of the chat application using the protocol (*RWG + RWGChat*) is 7998 KB. The results show that using the *RWGChat* the memory consumption is around eight times the usage of the basic application.

V. Related work

Several works propose different approaches and implementations to communicating in disaster areas [9]–[11], however their survey is beyond the scope of this report.

A similar study to ours is performed analysing the performance of two different Host Identity Protocol (HIP) implementations in Symbian OS [12] in terms of CPU load, memory usage and power consumption but only at initialisation time.

The response time of a payment transaction was analysed for evaluating the performance of an implementation of a payment protocol for vehicular ad hoc networks [13]. Our work studies response time of a different protocol (RWG) in a wider range of devices (e.g., handhelds and laptops) showing the relationship between the resource footprint and the performance in terms of response time.

Many works, as surveyed below, focus on measuring the power and energy consumption of different wireless technologies, whereas the focus of our efforts is on analysing the energy footprint of a wireless message dissemination protocol.

The energy consumption of IEEE 802.11 in ad hoc mode is analysed studying a wireless interface [14] and a particular mobile device (Nokia N95) [15]. Perrucci et al. [8] study the impact of 2G and 3G networks in battery consumption of a Nokia N95 and Balasubramanian et al. [16] compared 2G, 3G and IEEE 802.11 in terms of energy consumption. NEP is often used as measurement tool in Nokia phones, e.g., Xiao et al. [17] studied the energy consumption of mobile Youtube. Approaches requiring more equipment are needed for other platforms, ranging from the use of a power meter and custom software [18] to creating their own measurement framework replacing the battery [19].

In summary, very little prior work can be found in the literature about studies of the resource footprint of MANET protocols in multiple mobile platforms.

VI. Conclusion

This work has presented the implementation of a manycast protocol (RWG) for disaster areas in commercially available devices. It has focused on studying the resource footprint of the protocol on some devices, showing both some qualitative values of interest and some methodologies that can be applied to future devices in order to reveal important aspects for future optimisations.

The study has provided some insights on the impact of the hardware and protocol on the footprint. The CPU usage of the protocol is very low and the memory usage is small. The performed experiments have shown that features of nodes that are usually not considered in simulations such as CPU, response time or energy consumption are worth studying. In particular, we have shown how a platform and its resources impact performance in terms of response time. The transmission rate can indirectly imply significant delays in the response time through the CPU load. Moreover, it has been shown that a faster CPU does not always lead to faster response (e.g., in the netbook and laptop comparison).

Since energy consumption is crucial in disaster area networks, the most power consuming aspects of a mobile phone were considered. The most significant aspect in the energy consumption footprint of the protocol is due to reliance on an IEEE 802.11 ad hoc network, which restricts the lifetime of a mobile phone to less than 8 hours.

Future work includes optimisations to reduce the expensive IEEE 802.11 ad hoc mode in terms of energy consumption to increase the lifetime of nodes in disaster area networks. This includes adopting energy efficient ad hoc MAC layers from related research areas like wireless sensor networks or combinations with cellular technologies. Porting the protocol to more devices that are increasing their market share (e.g., iPhone and Android devices) would be interesting. Although adaptations of the source code may be necessary, fortunately the same architecture can be reused.

Acknowledgements

This work was supported by a grant from the Swedish Civil Contingencies Agency (MSB), the national Graduate school in computer science (CUGS) and the Department of Education, Universities and Research of the Basque Government.

References

[1] P. M. Aoki, R. Luk, and M. Ho, "When mobile experience comes apart at the seams – emerging markets infrastructure brings us back to nomadic computing in more ways than one," in *Workshop on Mobile and Ubiquitous User Experience, (Ubicomp), LNCS 4717, Springer*, 2007.

[2] R. Mahapatra, T. A. Abbasi, and M. S. Abbasi, "A propose architecture of manet for disaster area architecture," *J. Comput. Theory and Eng.*, vol. 2, no. 1, pp. 31–34, 2010.

[3] Y. Bai, W. Du, Z. Ma, C. Shen, Y. Zhou, and B. Chen, "Emergency communication system by heterogeneous wireless networking," in *Proc. of IEEE Wireless Communications, Networking and Information Security (WCNIS)*, June 2010.

[4] S. Underwood, "Improving disaster management," *ACM Communications*, vol. 53, pp. 18–20, Feb. 2010.

[5] M. Asplund and S. Nadjm-Tehrani, "A partition-tolerant manycast algorithm for disaster area networks," in *Proc. of IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2009.

[6] F. Gao and M. Hope, "Collaborative middleware on symbian os via bluetooth manet," *WSEAS Trans. Commun.*, vol. 7, no. 4, pp. 300–310, 2008.

[7] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," in *Proc. of ACM Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*, 2004.

[8] G. P. Perrucci, F. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, "On the impact of 2G and 3G network usage for mobile phones' battery life," in *European Wireless*, 2009.

[9] J. Kim, D. Kim, S. Jung, M. Lee, K. Kim, C. Lee, J. Nah, S. Lee, J. Kim, W. Choi, and S. Yoo, "Implementation and performance evaluation of mobile ad hoc network for emergency telemedicine system in disaster areas," in *IEEE Engineering in Medicine and Biology Society (EMBC), 2009.*

[10] M. Luglio, C. Monti, C. Roseti, A. Saitto, and M. Segal, "Interworking between manet and satellite systems for emergency applications," *Int. J. Satell. Commun. Netw.*, vol. 25, no. 5, 2007.

[11] H. Tazaki, R. Van Meter, R. Wakikawa, T. Wongsaardsakul, K. Kanchanasut, M. Dias de Amorim, and J. Murai, "Selecting an appropriate routing protocol for in-field manemo experiments," in *Proc. of ACM Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks (PE-WASUN)*, 2009.

[12] A. Khurri, D. Kuptsov, and A. Gurtov, "Performance of host identity protocol on symbian os," in *Proc. of IEEE Communications (ICC)*, June 2009.

[13] J. Téllez Isaac, S. Zeadally, and J. C. Sierra, "Implementation and performance evaluation of a payment protocol for vehicular ad hoc networks," *Electronic Commerce Research*, vol. 10, pp. 209–233, June 2010.

[14] L. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. of IEEE INFOCOM*, 2001.

[15] M. Pedersen, F. Fitzek, G. P. Perrucci, and T. Larsen, "Energy and link measurements for mobile phones using ieee 802.11b/g," in *IEEE Workshop on Wireless Network Measurements (WiNMEE)*, 2008.

[16] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *Proc. of ACM Internet Measurement Conference (IMC)*, Nov. 2009.

[17] Y. Xiao, R. S. Kalyanaraman, and A. Yla-Jaaski, "Energy consumption of mobile youtube: Quantitative measurement and analysis," in *Proc. of IEEE Next Generation Mobile Applications, Services, and Technologies (NGMAST)*, 2008.

[18] A. Gupta and P. Mohapatra, "Energy consumption and conservation in wifi based phones: A measurement-based study," in *Proc. of IEEE Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2007.

[19] A. Rice and S. Hay, "Measuring mobile phone energy consumption for 802.11 wireless networking," *IEEE Pervasive and Mob. Comput.*, vol. 6, pp. 593–606, Dec. 2010.