PAMPA in The Wild: A Real-Life Evaluation of a Lightweight Ad-Hoc Broadcasting Family

Christopher Winstanley[1], Rajiv Ramdhany[1], François Taïani[1],
Barry Porter[2], Hugo Miranda[3]
[1]Lancaster University (UK); [2]University of St Andrews (UK); [3]University of Lisbon (Portugal)
{c.winstanley, r.ramdhany, f.taiani}@lancs.ac.uk
bfp@st-andrews.ac.uk, hmiranda@di.fc.ul.pt

ABSTRACT

PAMPA is a family of broadcast algorithms for adhoc and wireless networks that are both lightweight and robust. PAMPA variants have shown promising performance results in simulations. Simulations can however be misleading, as they often poorly reflect reality. This paper seeks to further our understanding of PAMPA beyond simulations, and reports on the first ever characterisation of the PAMPA family on a real deployment. We use this opportunity to reflect on our findings and lessons learnt when moving from simulations to actual experiments.

Categories and Subject Descriptors

C.2.1 [**Network architecture and design**]: Wireless communication

Keywords

Wireless Sensor Network (WSN), MANETs, Broadcast Algorithms

1. INTRODUCTION

MANETs (Mobile Ad-Hoc Networks) are self-configuring networks of mobile devices connected by wireless links. They typically rely heavily on a best-effort message dissemination service, or *broadcast*, as a fundamental building block to implement higher-level services, including routing and service discovery.

The simplest form of MANET broadcast uses flooding, in which every node retransmits the messages it receives. Flooding is simple and generally robust, but wasteful. A number of algorithms have therefore been proposed to improve on flooding, by limiting retransmissions (communication cost) while still trying to reach the highest number of nodes (delivery). These algorithms differ in the policy they use to select retransmitting nodes. Most of them avoid explicit control messages, and rely instead on the limited amount of information present on each node to drive retransmissions. They also often use some form of *randomisation* (in timeouts, or in decisions) to overcome the network's unpredictability. While this approach works, randomisation may unfortunately cause sub-optimal behaviours, such as nodes retransmitting when they should not, or not retransmitting when they should, leading to nodes being missed out, or unnecessary transmissions being triggered.

PAMPA [11] is a lightweight broadcast algorithm that purposely avoid randomisation by combining two known strategies to the problem of ad-hoc broadcast: counting messages, and measuring received signal strength. PAMPA uses the received signal strength of a message to estimate the distance to the messages' source. PAMPA then uses this distance to count retransmissions in a node's vicinity and decide whether or not to retransmit. PAMPA is a minimalist protocols that does not use control messages; excludes warm-up or calibration phases; and does not assume any particular capabilities on nodes such as location-awareness or directional antennas [5, 7, 10].

Building on PAMPA's insight, a number of variants have been proposed [6] that seek to harden the protocol further. Some of these variants have been shown in simulations to provide delivery rates close to that of flooding while using far less messages. PAMPA and its variants, however, have so far only been tested in simulations. Although extremely useful to rapidly assess a protocol's behaviour, simulations have repeatedly be shown to be unreliable in assessing a protocol's real performance [1, 2, 9].

In this paper, we present an evaluation of PAMPA and its variants based on a real deployment. We use this opportunity to reflect on our findings and lessons learnt when moving from simulations to actual experiments. The rest of the paper is organised as follows: We first present the challenges involved in implementing a broadcast services for wireless ad-hoc networks, and existing works in this area (Sec. 2). We then introduce PAMPA and its variants (Sec. 3), before moving on to our experimental setup (Sec. 4). Section 5 presents our experimental results, and the lessons we learnt, and Sec. 6 concludes.

2. BACKGROUND AND RELATED WORK

Wireless ad hoc networks are often in long-term deployment (as in WSNs) where nodes have limited energy resources. In these networks, flooding to disseminate information is especially costly and threaten their very survivability. One basic flooding strategy, *Delayed Flooding*, involves each node re-broadcasting every new packet it receives af-

MidSens'12, December 4, 2012, Montreal, Quebec, Canada.

ter a random delay. Although it achieves a good coverage of nodes, its all-out design consumes non-negligible power, engenders network contention/interferences, and is prone to undetectable packet collisions [12]. Most broadcast protocols for MANETs/WSNs therefore seek to reduce message retransmissions whilst maintaining a coverage comparable to classic flooding. In doing so, they reduce contention, the chance of collisions, save energy, and lower broadcast latencies. To optimise on classic flooding, they typically make use of contextual information such as nodes location or node neighbourhood, or rely only on observations of ongoing packet exchanges.

Location-based protocols use geometric modelling [12] to determine the expected additional coverage that is gained by a node by virtue of its position and only allow the node with the maximum expected additional coverage to retransmit.

Neighbour-knowledge flooding protocols [13, 14] exploit each node's knowledge of its local topology to decide on the node's multicast tree membership. This, of course, necessitates the periodic exchange of explicit control messages for neighbour discovery. Multicast-tree building can occur at run-time (*self-pruning*) where each node removes itself from the multicast tree (by not retransmitting) if all its neighbours are already covered by the retransmission of one of its neighbours. The Scalable Broadcasting Algorithm [13] performs self-pruning at each node receiving a new broadcast message by comparing the sender's neighbours with those of the receiver. The node's retransmission is cancelled if a neighbour's retransmission achieves the same node coverage. As an alternative to run-time multicast tree-pruning, Multipoint-Relaying [14] uses the two-hop neighbourhood information at each node to *pre-select* a subset of neighbours as relay nodes. These nodes form a connected-dominating set in the network so that all two-hop neighbours can be reached from any node via a relay node.

Contextual information is however not always available or its use, desirable. GPS may be a too expensive requirement for low-cost devices and does not work indoors. Neighbour discovery via Hello messages is a source of significant overhead, additional contention and collisions. To address these limitations, a range of broadcast protocols have been proposed that do not require any contextual information.

For instance, *epidemic protocols* such as *GOSSIP1(p)* [8] use a form of probabilistic flooding in which nodes retransmit a message with some fixed probability p ($p < 1$). To prevent retransmissions from dying out (p is too low), *GOSSIP3(p,k,m)* [8] extends this technique by forcing retransmissions in two cases: *i)* if the message has been travelling for less than k hops, and *ii)* if the number of retransmissions listened by any node after a short delay is lower than a threshold m. Unfortunately static gossip probabilities may render these protocols inefficient in heterogeneous network topologies. A low gossip probability is suitable for a dense network but cause the broadcast to die out in the sparser regions. Conversely, a higher probability improves reachability in the sparser regions but is wasteful for denser topologies. Network densities are not usually known a priori. To address this issue, adaptive epidemic protocols such as RAPID [5] and Smart Gossip [10] adapt their retransmission probability to the perceived network density. They assign different gossip probabilities to nodes based on their topological importance; critical nodes broadcast with higher probability. Node densities are evaluated during a warm-up phase through the periodic exchange of messages and neighbour retransmissions count, although this may be costly.

Further, as found in [12] if several neighbours around a node were to retransmit the same flooding message, the expected additional coverage, EAC(k), after a host hears the same message k times dramatically decreases. *Counter-based broadcast* protocols therefore restrict the retransmissions in a node's neighbourhood to a predefined threshold. They wait for a random but bounded delay counting the number of duplicate messages received at a node and only allow retransmission if the counter does not reach a pre-determined threshold. Duplicate-message counting improves efficiency and is therefore a trait found in many WSN broadcast protocols including PAMPA. On its own, counting, like the use of static probability, does not ensure 100% reachability; the problem of selecting suitable thresholds for uneven distributions in network topology still remains.

Like counter-based protocols, *distance-based broadcast protocols* [12] use the notion of expected additional coverage to determine the value of a node's retransmission. Instead of using neighbour retransmission count, they look at the signal strength of these retransmissions to estimate additional coverage. They assume that signal strength (or Received Signal Strength Indicator - RSSI) is an indication of a transmitter's distance to the receiving node. The greater the distance (i.e the lower the first message's RSSI), the more additional nodes the receiving node can cover with its retransmission. To exploit this notion, distance-based protocols wait for a random but bounded time after the first message reception listening to further retransmissions. If the maximum signal strength of all received retransmissions fall below a certain threshold, the node is allowed to retransmit. Both counter-based and distance-based broadcast protocols use random listening periods which can be counter-productive: nodes of higher topological importance may be preempted from retransmitting if the listening timer of less 'important' nodes in their vicinity expire first.

3. PAMPA AND ITS VARIANTS

PAMPA [11], which provides the basis for our work, eliminates the need for a bounded random time by combining both the counting and distance-based strategies. PAMPA has since then been expanded to include hardening mechanisms, in particular in heterogeneous topologies [6], with promising results obtained in simulations.

3.1 Vanilla PAMPA

PAMPA is at its core a counter-based broadcast algorithm, which uses distance (as derived from a transmission's RSSI) to calibrate how long a node waits while counting (Algorithm 1). More precisely, PAMPA apportions the waiting time of a node to the signal strength of the original message reception (line 5). If within this waiting time a node hears *less* than a given number of retransmissions (line 11), the node retransmits (line 12), otherwise it does not.

3.2 Variants

Although PAMPA is thus more "informed" than other alternatives, its perception of the local node topology remains limited. In particular, PAMPA is not designed to perform well in heterogeneous topologies where some nodes may perform a key role in the message propagation. This is a situation likely to arise in actual wireless networks, for example

```
1  upon receiving(msg) begin
2  |  if msg ∉ messages then
3  |  |  messages ← messages ∪ {msg};
4  |  |  count_msg ← 0;
5  |  |  SetTimer(msg,delay(msg.RSSI));
6  |  else
7  |  |  count_msg ← count_msg + 1;
8  |  end
9  end
10 upon timeout(msg) begin
11 |  if count_msg < n then
12 |  |  Broadcast(msg)
13 |  end
14 end
```

Algorithm 1: The PAMPA Algorithm [11]

when two parts of a network on opposite sides of a river are connected by a small number of sensors deployed over a bridge. To address this problem, a number of variants to PAMPA, were proposed in [6]. All consist in counting only *some* of the retransmissions (see l:7 of Alg. 1), based on two types of information: (i) the retransmission path just followed by a retransmission (*Common Parenting*), and (ii) the relative position of a retransmission with respect to the original copy of a message (*Dynamic Thresholding*).

Common Parenting looks at the next-before-last hop of a retransmission, which we call its *parent node* (Fig. 1). Common parenting records the parent node of the original message reception (lines 3-5 in Algorithm 1), and only counts those retransmissions that have a *different* parent than the original reception at line 7. The motivation for this mechanism is that different parent nodes will tend to denote a higher diversity of propagation paths, which is a sign that the message is propagating well.

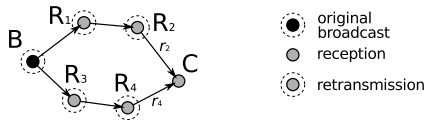

Figure 1: R_1 and R_3 are the parent nodes of r_2 and r_4 respectively.

Dynamic Thresholding uses the RSSI more thoroughly to decide whether a message should be retransmitted or not. It uses a similar approach to distance-based protocols in that it only counts retransmissions that have a higher (resp. lower) RSSI than a particular threshold. However, instead of relying on a fixed threshold, which is difficult to define statically, it uses the RSSI of the original message reception.

Dynamic Thresholding comes in two sub-variants, depending whether only retransmissions with a *lower* (*Thresholding*) or *higher* (*Antithresholding*) RSSI than the original one are counted. Both *Thresholding* and *Antithresholding* have a geometric interpretation: *Thresholding* will cause a node C to only count retransmissions from an doughnut area we have termed its *outer-ring* (Fig. 2). By contrast, *Antithresholding* will primarily count retransmissions from a node's *inner-strip* and *forward bubble* on the same figure (since the nodes located between C and R will have a longer delay than C, and are unlikely to retransmit before C).

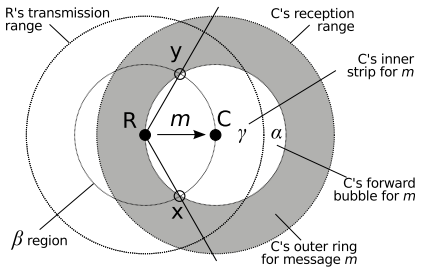

Figure 2: Dynamic Thresholding: The right-hand node will only accept messages from nodes situated in the shaded area.

A family of protocol variants. The criteria used by Common Parenting and Dynamic Thresholding to filter retransmissions are orthogonal, and can thus be combined to create further variants. This is shown on Fig. 3, where the types of retransmissions counted by variants are shown in a 2×2 matrix. PAMPA counts all retransmissions, while Common Parenting (CP), Thresholding (TH) and Antithresholding (ATH) only count retransmissions meeting a single criteria (shown as ones in the figure).

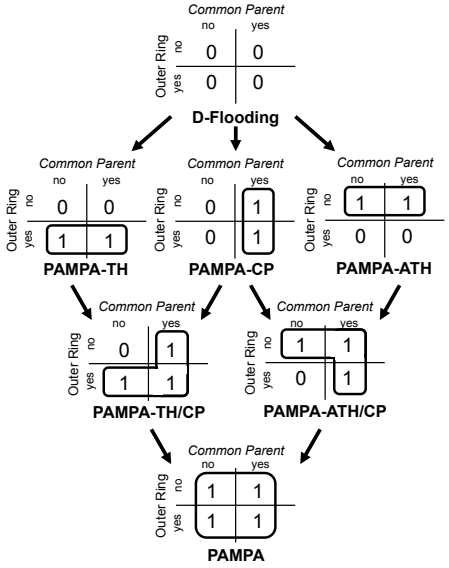

Figure 3: Family of protocols: 1s indicate which retransmissions are counted

By combining the CP criteria with TH and ATH, one obtains two more protocols, PAMPA-TH/CP and PAMPA-ATH/CP. These are more selective than PAMPA in their counting but less than the original three variants. Delayed Flooding can be captured in this scheme by simply not counting any retransmission, and retransmitting all messages.

4. EXPERIMENTAL SETUP

Most of the performance study of broadcast protocols has been on simulation (see, for example [6]). However, even the most detailed simulation models may not capture the various limitations of ad hoc wireless environments. They may

Figure 4: An example TelosB mote.

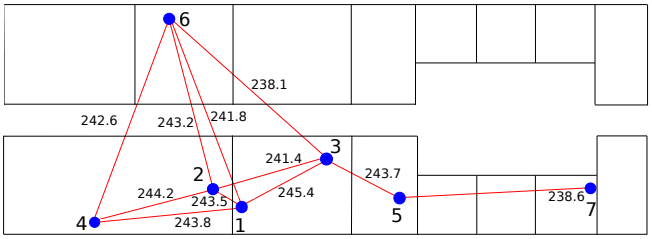

Figure 5: Node layout in the office environment. Red lines indicate a link between the two nodes with the average RSSI of the link.

not include limitations of the radio interface hardware such as limited buffer space, limitations due to the protocol stack implementations or various random sources of errors in the wireless physical layer such as multipath propagation, radio noise, asymmetric radio links, etc. For a real-life evaluation of PAMPA and its variants, the algorithms are implemented and deployed in a live testbed (a WISEBED instance [3]) of sensor mote devices. In particular, the algorithms are built using the Lorien OS's component-libraries and toolchain, deployed and executed on TelosB motes. TelosB motes (depicted in Fig. 4) are low-power wireless sensor devices, each equipped with a 802.15.4 radio module achieving data rates of up to 250 kbps and an indoor transmission range between 20 m to 30 m.

4.1 Topology Selection

For our experiment, we use a WSN testbed containing seven TelosB motes, all running the same variant of the PAMPA algorithm. The TelosB motes belong to an indoor live WSN testbed (part of the WISEBED experimentation facility) and are selected as to realise a topology where the notion of 'keyness' of certain nodes is exercised. The layout of the nodes can be seen in Figure 5. Although the 'bridge' topologies tested in simulations [6] are hard to reproduce on the testbed due to the fixed position of nodes, the selection of key nodes (such as Node 3 in Figure 5) allows similar protocol behaviour (for example, overcancellation) to be reproduced. The testbed is deployed in an 'office' environment, which includes a mixture of brick support, plasterboard separations and glass walls that radio signals must overcome, as well as thick wooden fire doors. The room height is around 2.7m. Most experiments are performed at night to avoid the added interference of office workers. This set-up allows us to evaluate all protocols under comparable conditions.

4.2 Experiment Planning

A particular node (Node 1, Figure 5) in the WSN is selected as the broadcast-originator whose sole responsibility is to broadcast a message every 10 seconds and not attempt to receive/retransmit any messages. The same node is used throughout all experiments. The periodic interval ensures that broadcasts between different experiment runs do not overlap and interfere with each other. Each broadcast message contains a unique identifier (*uid*) in its header that is used by a receiving node to differentiate between multiple messages. This identifier is part of the soft-state saved in each node's message history to detect duplicate messages. Also included in the message history is the RSSI of each message, obtained through a query to the radio module of the TelosB motes for the received message's network parameters. Only the RSSI value of the first arriving message is recorded and used as a threshold for the retransmission-based protocols.

4.3 Computed Metrics

In each experiment several variables are measured to determine the performance of each protocol on the network. These variables can be used to calculate the Retransmission Ratio and the Delivery Ratio.

Retransmission Ratio is a key metric for broadcast protocols in WSNs and represents the communication cost paid for each successful delivery of a broadcast. The highest possible retransmission ratio is 1, corresponding to a run where all nodes that received the message retransmitted it. More efficient protocols will have lower retransmission ratios. The retransmission ratio of a run is defined as the average ratio between broadcast deliveries and broadcast retransmissions:

$$\text{Retransmission Ratio} = \frac{1}{n_{\text{bcasts}}} \sum_{b=1}^{n_{\text{bcasts}}} \frac{\#\text{retransmission}_b}{\#\text{delivery}_b}$$

where n_{bcasts} is the number of broadcasts (in our experiments, 50), and $\frac{\#\text{retransmission}_b}{\#\text{delivery}_b}$ is the retransmission ratio of broadcast b, i.e. the proportion of nodes that, having received b, retransmit it.

Delivery Ratio is the average proportion of nodes reached by a broadcast on the network. The higher the delivery ratio, the more effective the protocol and the larger the coverage of each broadcast. A delivery ratio of 1 means that every message broadcast across the network was received by every node. The delivery ratio can be obtained from the results through the following computation:

$$\text{Delivery Ratio} = \frac{1}{n_{\text{bcasts}}} \sum_{b=1}^{n_{\text{bcasts}}} \frac{\#\text{delivery}_b}{n_{\text{nodes}} - 1}$$

where n_{bcasts} is the number of broadcasts (in our experiments, 50), and $\frac{\#\text{delivery}_b}{n_{\text{nodes}}-1}$ is the delivery ratio of broadcast b, i.e. the proportion of nodes that receive b. Although a high delivery ratio indicates a high degree of of coverage of the network nodes by a broadcast protocol, its overall performance can only be determined in combination with the protocol's retransmission ratio.

5. EVALUATION RESULTS

This section presents the results of our evaluation of the PAMPA suite of protocols based on a real deployment. In doing so, it enables the identification of the most suitable

PAMPA variant for this deployment scenario. It also serves to compare our findings with the simulation-based evaluation of the same protocol suite in [6]. In the comparison, we also factor in the difference in network topologies, the use of a MAC protocol in the simulation and its absence in our real deployment. Maximum node coverage and fewer retransmissions (i.e. less energy consumption) are the defining traits of the ideal WSN broadcast protocol with the former as the overriding factor. Where broadcast protocols have dissimilar node coverage, the one with the highest average delivery ratio is deemed more successful irrespective of their average retransmission ratios. Node reachability bears higher precedence in typical WSN deployments, say, for software reconfiguration or data dissemination.

Figure 6 and Figure 7 show the delivery ratios and retransmission ratios respectively for each PAMPA variant over 10 experiment runs. In both figures, the box-and-whisker plots gives the degree of confidence in and the mean (shown as a cross) of the computed metric values for each broadcast protocol variant. The best performing broadcast protocol in our experimental evaluation is vanilla PAMPA, achieving an average delivery ratio of 75% (See Figure 6) at relatively low retransmission cost (only 44%, See Figure 7). It is surprising to find PAMPA achieving an even higher average delivery ratio than D-Flooding (71%) until one realises the effect of not benefitting from the collision-avoidance mechanisms of a MAC protocol. In the absence of an RTS/CTS dialogue, D-Flooding results in high channel contention and packet collisions. The ensuing corruption of networks packets degrades the overall delivery ratio of the protocol. By retransmitting less, PAMPA causes less packet collisions and is therefore able to reach out to more nodes. Vanilla PAMPA performs markedly *better* than its variants which is in stark contrast to the evaluation results from simulation where the opposite is the case (with the exception of the PAMPA-TH variant). Ellis et al. report PAMPA achieving delivery ratios of higher than 90% in only 37% of 'bridge' topologies selected. We attribute the difference in behaviour of PAMPA in simulation and real deployment to three conditions: *1)* the sparser/smaller network topology in our real deployment, *2)* the absence of a MAC protocol in our implementation, and *3)* the not inconsequential effects of the aforementioned real-life deployment-level limitations not captured by simulation.

With regards to node coverage, PAMPA-CP and PAMPA-TH/CP are the second-best performers after PAMPA achieving a healthy average delivery ratios of 0.72 and 0.73 respectively but prove strikingly costlier, posting average retransmission ratios of 0.89 and 0.83 respectively. Clearly, common-parenting curbs overcancellation and produces more retransmissions including those at key nodes. When directional look-ahead is applied to PAMPA-CP, only the retransmissions from nodes in the outer ring (see Figure 2) *and* with no common parents are counted, reducing the wastefulness of common-parenting. This is translated into a reduction of the average retransmission ratio by 0.06. In simulation, by contrast, both PAMPA-CP and PAMPA-TH/CP fared better than PAMPA (the second-worse simulation performer) in terms of node coverage and posted somewhat lower retransmission ratios (0.68 and 0.58 respectively).

The worst performers in our evaluation are the PAMPA-ATH/CP, PAMPA-TH and PAMPA-ATH variants providing modest average delivery ratios (0.65, 0.65 and 0.62 respectively) whilst still incurring high average retransmission

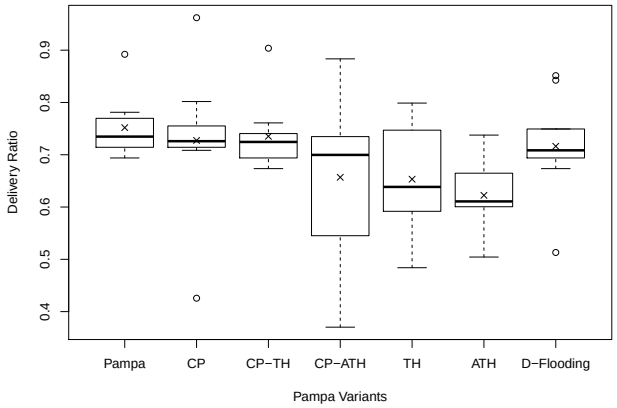

Figure 6: Average Delivery Ratio, results for each algorithm variant and d-flooding with 10 experiments run for each.

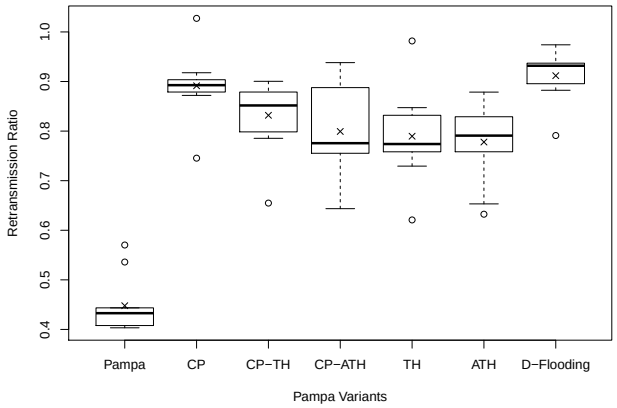

Figure 7: Average Retransmission Ratio, results for each algorithm variant and d-flooding with 10 experiments run for each.

ratios (0.8, 0.78 and 0.77, respectively). Considering only the PAMPA-ATH and PAMPA-ATH/CP variants, one can find that the application of common-parenting to PAMPA-ATH curbs overcancellation leading to an improvement in average delivery ratio of 0.03 but incurring an additional retransmission overhead of same magnitude (0.03). The simulation results for PAMPA-ATH and PAMPA-ATH/CP depict a different picture altogether; they performed consistently better than PAMPA, reducing the average delivery loss of PAMPA by 13-14% approximately.

PAMPA-TH is expected to perform less well than PAMPA in sparse networks where the presence of nodes outside the signal-strength threshold (outer ring) is more likely. As only the retransmissions for these nodes are counted in PAMPA-TH, this leads to higher message drop rates (poorer delivery ratio) in sparser topologies than for PAMPA. This is reflected in our results. On comparison with its simulation results, one finds that it surprisingly performs better in real deployment than in simulation. The simulated PAMPA-TH exhibited the worst average delivery ratio (node coverage)

and the second-most costly retransmission ratio.

6. CONCLUSIONS AND LESSONS LEARNT

In this paper, we have evaluated the performance of the PAMPA family of broadcast protocols on a real deployment. The results have confirmed PAMPA's strengths: The vanilla version of the protocol was able to reach more nodes than a naive flooding strategy (75.2% vs. 71.6%), while incurring much lower retransmissions costs (44.8% vs 90.2%). These results are also in line with earlier simulation results, obtained with different node densities and topologies, in which PAMPA achieved a delivery ratio of 77% for a retransmission ratio of 51% [6].

The results of the PAMPA variants have been more chequered, however, and different from earlier simulation results in their relative performance to PAMPA. For instance, whereas PAMPA-CP performed better than PAMPA in simulations for a moderate additional cost [6], it performed worse in our real deployment, for a retransmission ratio (89.1%) close to that of flooding.

Several good reasons can explain this divergence: First, as mentioned earlier, simulations only imperfectly reflect reality [2, 1], highlighting the need for real deployments. Second, but more importantly, earlier simulations results for PAMPA were obtained on large topologies (several hundreds of nodes), which are difficult to compare against the smaller deployment used in this paper. This is particularly important for the CP mechanism, which would be expected to perform better in denser networks with longer and more diverse propagation paths, leading to less retransmissions. It would be interesting to look at implementing this small topology in a simulator and seeing whether similar results are achieved. Another reason for this may be the reliance on RSSI to affect the protocol variant's decisions. RSSI is often not accurate enough to determine distances, leading to poor decisions made by the nodes when dropping packets. In further research the hardware should be tested for distance and RSSI measurements and then calibrated accordingly.

This later points highlights the difficulty in assessing wireless broadcast protocols, and more generally any distributed wireless mechanisms: Simulation campaigns tend to focus on large networks to demonstrate scalability, and to benefit from statistical effects (such as the diversity of propagation paths) which are hard to observe at smaller scales. By contrast, testbed deployments tend to be much smaller, only involving a few tens of nodes, or less. This experimental discrepancy is probably a reason to focus more on the validity of simulations results, than on their size.

In this context, the fact that the vanilla version of PAMPA obtained very close results both in large simulations and in a small real deployment is probably a strong indication of PAMPA's robustness, an aspect probably worth researching further in the future.

In terms of methodology, our results confirm once again the importance of real deployments to evaluate protocols. This somewhat obvious point comes however with a number of caveats: Real deployments are costly, time consuming, and tedious. Although we were able to rely on the infrastructure of the WISEBED testbed [3], failures, in particular of nodes, were frequent, and considerably increased the efforts required to run experiments. The causes of failures were diverse: batteries would get depleted, passers-by would unplug parts of the infrastructure, the memory of motes would get corrupted. Even without failures, repeating experiments (resetting, re-synchronising, and re-starting each node) made obvious the importance of a powerful management and scripting interface to conduct large numbers of tests in a batch mode, even on a modest network.

The above difficulties in running real experiments, and the importance of these experiments for evaluation, are both key arguments that support the recent efforts made to develop shared wireless testbeds [4, 3], an evolution that should provide critical feedback to researchers and practitioners working with wireless distributed networks.

7. ACKNOWLEDGEMENTS

This work has been partially supported by the European Commission under contract number IST-2008-224460 (WISEBED) and grant agreement number 257992 (SmartSantander).

8. REFERENCES

[1] T. Andel and A. Yasinsac. On the credibility of manet simulations. *Computer*, 39(7):48 –54, july 2006.

[2] E. Ben Hamida, G. Chelius, and J. M. Gorce. Impact of the physical layer modeling on the accuracy and scalability of wireless network simulation. *Simulation*, 85(9):574–588, 2009.

[3] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, and D. Pfisterer. Flexible experimentation in wireless sensor networks. *CACM*, 55(1):82–90, Jan. 2012.

[4] C. B. des Roziers, G. Chelius, T. Ducrocq, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, T. Noël, and J. Vandaele. Using senslab as a first class scientific tool for large scale wireless sensor network experiments. In *10th Int. IFIP TC 6 Conf. on Networking - Vol. Part I*, NETWORKING'11, pages 147–159, 2011.

[5] V. Drabkin, R. Friedman, G. Kliot, and M. Segal. Rapid: Reliable probabilistic dissemination in wireless ad-hoc networks. In *26th IEEE Int. Symp. on Reliable Dist. Sys. (SRDS 2007)*, pages 13–22, 2007.

[6] C. Ellis, H. Miranda, and F. Taïani. Count on me: lightweight ad-hoc broadcasting in heterogeneous topologies. In *Int. Workshop on Middleware for Pervasive Mobile and Embedded Comp.*, M-PAC'09, pages 1–6, 2009.

[7] B. Garbinato, A. Holzer, and F. Vessaz. Six-shot broadcast: A context-aware algorithm for efficient message diffusion in manets. In *OTM 2008*, LNCS, pages 625–638. 2008.

[8] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.*, 14(3):479–491, 2006.

[9] S. Kurkowski, T. Camp, and M. Colagrosso. Manet simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, Oct. 2005.

[10] P. Kyasanur, R. Choudhury, and I. Gupta. Smart gossip: An adaptive gossip-based broadcasting service for sensor networks. In *IEEE Int. Conf. on Mobile Adhoc and Sensor Sys. (MASS'06)*, pages 91–100, 2006.

[11] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. A power-aware broadcasting algorithm. In *Procs. of The 17th Annual IEEE Int'l Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'06)*, 2006.

[12] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *5th annual ACM/IEEE Int. Conf. on Mobile Comp. and Networking*, MobiCom '99, pages 151–162, 1999.

[13] W. Peng and X.-C. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *1st ACM Int. Symp. on Mobile ad hoc networking & computing*, MobiHoc '00, pages 129–130, 2000.

[14] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *35th Annual Hawaii Int. Conf. on Sys. Sciences (HICSS'02) - Vol. 9*, HICSS '02, pages 298–, 2002.