Revisiting Broadcast Algorithms for Wireless Edge Networks

André Rosa, Pedro Ákos Costa, João Leitão
NOVA LINCS & DI/FCT/NOVA University of Lisbon, Lisboa, Portugal
af.rosa@campus.fct.unl pah.costa@campus.fct.unl jc.leitao@fct.unl.pt

***Abstract*—With the advent of Edge Computing, suitable, practical, and novel abstractions are required for applications to leverage the existing computational power at the edge. In particular, applications in the domains of smart cities and the Internet of Things (IoT) can rely on devices in the vicinity of data consumers and producers for their operation. While these devices are expected to be equipped with wireless radios, network infrastructure might be unavailable in many scenarios. In those cases, devices must rely on wireless ad hoc networks for coordination and cooperation. In this context, one of the most important primitives is the broadcast of messages, that can be leveraged as a building block to devise more complex distributed services and applications.**

The literature on wireless ad hoc broadcast algorithms is quite vast, with many different algorithms being proposed which explore or combine different techniques or features in their operation. While such protocols are becoming increasingly relevant, understanding how they relate among them is complicated. To address this challenge, in this paper, we introduce a novel framework that allows to abstract the operation of wireless ad hoc broadcast protocols. Leveraging on our framework, we explore a particularly interesting class of these protocols: neighbor-aware ad hoc broadcast protocols; of which we propose 4 novel protocols. Finally, we rely on a materialization of our framework to implement prototypes of these protocols and experimentally study their performance in a testbed composed of 21 Raspberry Pi 3 - model B.

***Index Terms*—Broadcast Algorithms, Wireless ad hoc, Reliability, Experimental Evaluation**

I. Introduction

The edge computing paradigm [1] emerged to address the limitations of current cloud-based applications. These applications produce high volumes of data [2] which render cloud infrastructures unable to timely process and provide responses to application clients. As such, the edge computing paradigm promotes moving computations beyond cloud datacenter boundaries towards data producers and consumers. Edge computing however, can be materialized across very distinct scenarios involving different hardware [3], from fog computing [4] to IoT networks [5], [6] where IoT devices may own some computational power. In this paper, we focus on a concrete scenario of edge computing, where wireless-capable commodity devices form a wireless ad hoc network [7]. Such scenario can be found in application domains related with smart cities and IoT, where devices do not have access to network infrastructure.

To enable the emergence of novel applications leveraging the computational resources in wireless-capable devices, adequate abstractions should be provided. A fundamental primitive for supporting distributed applications is the broadcast of messages [8], where a message sent by a single process is collaboratively disseminated and delivered by all processes in the system. Broadcast primitives are an essential building block of complex distributed applications and protocols [9]–[13] as a means to achieve coordination and collaboration among processes.

Unfortunately, ensuring that all processes deliver a message in a system where nodes interact through a shared wireless medium, without an access point, is a non-trivial task. This is because when two processes (in two distinct nodes) within transmission range of each other, transmit a message at the same time, a collision might occur, leading those messages to not be delivered to any other process within range, or only to a subset. As we discuss further ahead, when collaboratively broadcasting a message throughout the network, nodes are more likely to attempt to retransmit a message simultaneously, increasing the probability of collision. Furthermore, if retransmission policies, akin to the ones employed in wired networks with explicit acknowledgments (e.g., TCP acknowledgement and retransmission mechanisms), to recover lost messages are used, the possibility to saturate the wireless medium grows even further, potentially leading to the well-known broadcast storm problem [14], effectively rendering any form of communication among processes impossible.

With the goal to avoid broadcast storms, some broadcast protocols that were previously proposed in the context of wireless ad hoc networks [15]–[17], tend to be probabilistic in nature, avoiding explicit acknowledgment of messages and avoiding all processes to perform retransmissions, while exploring complementary mechanisms to maximize their reliability (i.e., ensure that the vast majority of processes delivers every broadcasted message). This has led to the emergence of many variants of those techniques and protocols that explore them in different combinations and/or using different parameterizations, leading to a vast number of different alternatives (e.g., [15], [16], [18]–[22]). We note, however, that many broadcast protocols present a similar design pattern, being composed by a retransmission policy, which defines the (local) strategy of each process for deciding when to retransmit, or

The work presented here was partially supported by the Lightkone European H2020 Project (under grant number 732505), NG-STORAGE (FC&T grant PTDC/CCI-INF/32038/2017), and NOVA LINCS (FC&T grant UID/CEC/04516/2013).

2575-8462/19/$31.00 ©2019 IEEE
DOI 10.1109/SRDS47363.2019.00033

not, a given received message, and a delay policy, that encodes when that decision should be made.

This common design pattern motivated us to devise a generic framework for specifying wireless ad hoc broadcast algorithms. The framework captures some of the key aspects that govern the operation of several wireless ad hoc broadcast protocols, which are the **retransmission delay**, the **retransmission context**, and the **retransmission policy** employed by the protocol. These three aspects work together for the efficient operation of a broadcast protocol: the **retransmission delay** provides a waiting period to gather environmental information, which is dictated by the **retransmission context** (e.g., number of neighboring nodes within transmission range), and leveraged by a **retransmission policy** to decide if a retransmission should proceed or not. Our framework promotes these aspects to be parameters for a generic broadcast protocol, enabling a wide range of protocols to be easily specified. We note, however, that such aspects can also be parameterized as "empty", when they are not required by a protocol. Furthermore, to promote the focus of the framework over the key aspects of broadcast protocols, the framework abstracts common, but essential, aspects that are prevalent in the design of such protocols, including the tracking of messages already delivered (to avoid duplicates); the gathering of environmental information that is provided (locally) to each protocol, which is used to affect its local decisions; and the scheduling of retransmissions.

Our framework also allows for efficient and modular implementation of these protocols. To this end, we have built a prototype of a small kernel for designing, implementing, and executing broadcast protocols in wireless ad hoc networks, whose design follows directly from the concepts defined by our framework. We leveraged our framework, and the protocol kernel prototype derived from it, to study and extend the existing state of the art for a particular area of the design space of wireless ad hoc broadcast protocols: neighbor-aware protocols. These protocols take into consideration information regarding the local neighborhood of a node (which can be obtained by static configuration or through the use of a simple companion protocol) to govern the execution of the distributed broadcast process, in particular, in the definition of the retransmission policy and associated delay, employed by each node. We explore 4 novel variants of this class of protocols, specifying them, and performing an experimental study of their performance in a realistic scenario with real devices (Raspberry Pi 3 - model B).

The remainder of the paper is structured as follows: Section II discusses the key properties and different techniques employed in the design of broadcast algorithms for wireless ad hoc networks; Section III provides an overview of our framework to model the operation of wireless ad hoc broadcast protocols; Section IV delves into the class of neighbor-aware broadcast algorithms (which we dub NABA), presenting a set of novel variants in this class of algorithms; Section V provides a practical evaluation of broadcast algorithms; Section VI discusses the related work; Section VII concludes the paper.

II. Broadcasting in Wireless Networks

In a distributed system, broadcast protocols aim at delivering messages, that can be sent by any process, to all processes. In a wireless ad hoc network, processes communicate via the exchange of messages through the wireless medium. In this context, it is frequent to leverage on one-hop broadcast [23], which allows a node to send the same message to all other nodes within its radio transmission range (neighbours), with a single transmission. The use of one-hop broadcast is relevant since it saves power and reduces the occupation of the wireless medium when compared with sending the message point-to-point to all neighboring nodes. In this paper, we consider multi-hop networks, where not all nodes are directly reachable by every other node. For a message to reach all nodes in the system in such scenarios, nodes have to retransmit received messages such that messages transverse the network, and are received by all participants. Naturally, this retransmission process might lead some nodes to receive a message more than once. Such redundant messages should not be delivered to application layers [8].

A. Performance Metrics and Retransmission Policies

Collaborative broadcast protocols, as it is the case of wireless ad hoc broadcast protocols, strive to achieve two conflicting goals. On the one hand, these protocols strive to maximize the *reliability* of the broadcast process, which usually is defined as the fraction of (correct) nodes that delivered a broadcast message [24]. On the other hand, these protocols aim at minimizing the *cost* of each broadcast message, which is defined as being the number of individual retransmissions that are required to spread the message through the entire network. The conflicting nature of these goals derives from the lack of independence between them, for instance, it is possible to lower the transmission cost of a broadcast protocol by never retransmitting the message but this will have a negative impact on the reliability of that protocol.

Due to this tension between goals of broadcast protocols, at their core one can usually find a retransmission policy, that allows a node to make a (typically local) decision regarding its need to retransmit a message received from another node. Intuitively, such policies will strive to avoid retransmissions that are not going to lead to message deliveries for the application (i.e., transmissions that will not reach nodes that have not yet received the message) and/or ensure that the message is retransmitted if there are nodes within the transmission range that have not received that message yet. In the following, we discuss different retransmission policies that can be found in the literature.

Usually, nodes will only consider retransmitting a message when they receive it for the first time. However, and as we detail further ahead, some policies might consider the number of received duplicates of a message to be retransmitted to make their decision.

1) Always Yes: The simplest retransmission policy that can be employed is to have every node in the system to retransmit each received message. This is the policy implicitly

associated with flooding [14] protocols. Although this policy is simple and (potentially) robust, it promotes a high cost for the broadcast process.

2) Probabilistic: Another frequently observed alternative is to rely on a probabilistic policy, where each node retransmits each message (observed for the first time) with a given probability p, where p is a protocol parameter. This approach is employed by protocols that follow gossip-based approaches [20], [25]–[27]. It reduces the cost of the broadcast process, at the risk of impairing reliability if nodes, that are essential for the success of the dissemination process, decide to not retransmit the message.

3) Counting: Counting is a policy that relies on counting the number of observed duplicates for a message (for a given period of time) to make a more informed decision regarding the need to retransmit that message [14], [15], [25], [28]. Usually, such policy will rely on a parameter c which is the minimum number of counted duplicate messages that are necessary for a node to decide not to retransmit a message. While the intuition of such policy is easy to grasp, it does not take into account the local topology of the network, and hence, it may lead nodes that are bridging different partitions of the network to decide not to transmit messages, disrupting the reliability of the broadcast process.

4) Distance-based: Some broadcast protocols [14], [29] take into account the distance between the sending and the receiving nodes to bias the decision of the later to retransmit a given message. The intuition associated with such policy is that nodes that are farther from the sender will cover a higher number of new nodes if they retransmit a message, compared with nodes that are closer to it. Hence, this policy will take into account the distance to the sender and retransmit a message only if this distance is above a threshold d or adjust locally the probability of retransmission (p) to be proportional to the distance. While this policy can be highly effective [29] it requires nodes to either know their positions (through GPS for instance) or to infer their relative distances by taking into account the power of the received radio signal (which is employed in some variants of PAMPA [29]). However, the hardware support to obtain such information might not be available in commodity devices.

5) Location-based: This policy [18] is similar to the distance-based policy, however, instead of taking into account only the relative distance between the sender and the receiver to perform retransmission decisions, it requires nodes to have complete knowledge of their positions and surroundings, which allows them to make a more precise decision on the utility of their message's retransmissions.

6) Neighbor-based: Neighbor-based retransmission policies [21], [30], [31] rely on topology information, obtained by each node, regarding the nodes in direct range of transmission, or up to some number of hops in the network, to compute the utility of a message retransmission. This information can be leveraged in a different number of ways: from using the observed number of direct neighbors to control the probability of retransmission (combining this with probabilistic retransmission policies) or to adjust the number of duplicate messages that a node has to receive to cancel its own retransmission (combining this approach with counting policies described above). We will revisit this class of retransmission policies in Section IV where we also propose novel variants of wireless ad hoc broadcast protocols leveraging on these policies.

B. Environmental Sensing and Retransmission Context

Some of the retransmission policies discussed above require nodes to gather information (even if this information is imprecise) on their execution environment. In particular: i) *Distance-based policies* need information concerning the relative distance between the message sender and receiver to make a retransmission decision; ii) *Location-based policies* require information on the position of nodes in some space coordinates system for their operation; and finally, iii) *Neighbor-based* policies require information about neighboring relations with other nodes up to a given horizon (i.e., $1-hop$ which is direct neighbors, $2-hop$ which includes knowing the neighboring relations of their direct neighbors, or generally $n-hop$ where if $n \geq$ to the diameter of the network implies that a node has full knowledge about the system topology).

We note that, contrary for instance, to counting message duplicates, which can be easily handled at the wireless ad hoc broadcast protocol level, such information requires external support, either external (static) configuration provided by users, access to additional information extracted from the device hardware (e.g., from the radio to extract the strength of the radio signal or from a GPS device), or even by a companion protocol being executed alongside the broadcast protocol. To these external sources of information, that support the execution of broadcast protocols leveraging particular retransmission policies, we call *retransmission context*.

While there are many possible retransmission contexts that can be useful for devising wireless ad hoc broadcast protocols in general, and retransmission policies in particular, in this paper we focus on the three previously identified contexts: *distance context*, *location context*, and *neighbor-aware context*.

C. Avoiding Broadcast Storms and Retransmission Delay

Avoiding broadcast storms is essential to achieve high reliability for broadcast protocols. Network collisions in the wireless medium will lead to message losses, which might affect negatively the assumptions of protocols and, consequently, their reliability. Network collisions happen when nodes within transmission range decide to transmit a message at the same time. As such, a simple solution to avoid these phenomena is to delay retransmissions of messages (using some jitter) to minimize the probability of having nodes synchronizing their retransmissions.

Additionally, some of the previously discussed retransmission policies require some period of time to gather information to make a decision. For instance, considering the *counting policy* described above, some time has to pass since the reception of the first copy of a message to offer the opportunity for a node to count duplicate message retransmissions. Note

that, if all nodes that are neighbors of a node that transmitted a message wait for the same time, all of them would retransmit simultaneously because none would observe a duplicate message. Such transmissions would potentially lead to collisions, compromising their usefulness for achieving high reliability.

This implies that defining such retransmission delays might require more sophisticated mechanisms, for instance, the retransmission delay of a node for a given message received for the first time might depend on the *retransmission context* of that node as described above. As an example, consider again the counting retransmission policy. It could be useful to apply a retransmission delay that would be inversely proportional to the distance of the sender, such that nodes that are farther away will decide to retransmit earlier, enabling nodes closer to the sender to avoid their retransmissions, and potentially covering larger number of nodes that have not observed that message yet (in fact, this is very similar to the approach taken by PAMPA [15]).

III. Wireless Broadcast Algorithms Framework

In this section, we present our conceptual framework that captures the operation of a wide range of wireless ad hoc broadcast protocols. Our framework design derives directly from the observations presented previously (Section II). Our key insight for devising this framework is that most protocols to support wireless ad hoc broadcast present similar architectural patterns, differing primarily on the employed *retransmission policy*, their strategies to compute a *retransmission delay*, and what is the required (external) *retransmission context* that provides environmental information for the retransmission policy and the computing of the retransmission delay.

Additionally, we note that some approaches (e.g., [16]) rely on multiple rounds of (potential) retransmissions of a single message, where the protocol reevaluates the need of retransmitting the message. To accommodate these class of protocols, we enrich our framework with an additional parameter denominated *phases* [16] that encodes the number of times, for each newly received message, that a protocol will evaluate its decision to retransmit that message. For simplicity, we assume that the time between these distinct phases is provided by the same function that computes the retransmission delay for protocols that only consider, at most, a single message retransmission.

In the following we present the overview of the workflow of a generic wireless ad hoc broadcast protocol, that lies at the core of our framework, explaining our notation to represent different protocols. We then provide examples of possible materializations for its components, and finally, present the specification of a set of representative wireless ad hoc broadcast protocols using our framework.

A. Overview

Figure 1 presents a simplified flux diagram that captures the workflow of a generic wireless ad hoc broadcast protocol. The diagram describes the workflow for a particular message m.

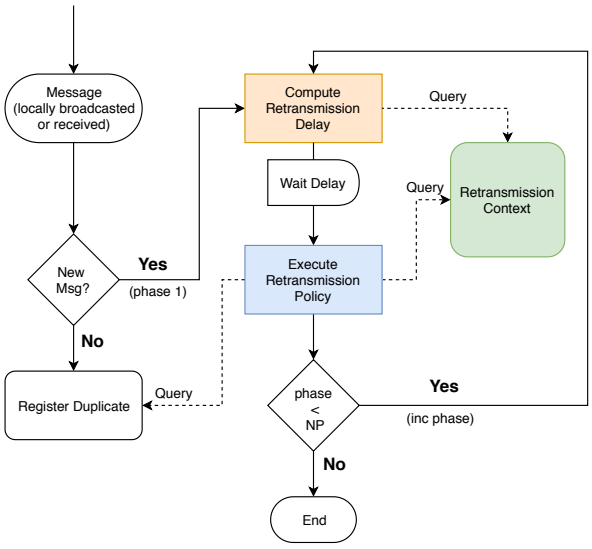

Fig. 1. Workflow for a Generic Wireless Ad hoc Broadcast Protocol

Evidently, multiple workflows for different messages might be concurrently active in the context of a single node.

The workflow starts with the reception of a message. We assume messages have a unique identifier that allows the protocol to easily detect duplicates. Upon the reception of a message, the first step of our generic protocol is to check if this is the first time that the message is observed. If this is not the case (if the message is a duplicate) and if the protocol is still processing this message (i.e., if there is an active workflow for that message) we simply register the reception of the duplicate (and the source of the transmission). This is relevant for protocols that take into account the reception of duplicates in their operation.

However, if the message is received for the first time, this will trigger the start of the retransmission process (starting at retransmission phase 1). As the first step of this process, our generic protocol computes the *retransmission delay* to be applied for this message. The retransmission delay computation is performed by using a *function* that is provided to our framework as a parameter. The complexity of such function can be highly variable (we discuss the design of some of these functions in Section III-B2), it receives as input the received message, the identifier of the node from whom the message was received, and the number of the retransmission phase in which the protocol is (all messages start in phase 1). Note that the function that computes the transmission delay may query the *retransmission context*, which is also a parameter in our framework (more details in Section III-B1).

After computing the retransmission delay, the workflow proceeds by waiting for that amount of time, after which it executes the *retransmission policy* function, which is another parameter provided to our framework. Similar to the retransmission delay, the function that executes our retransmission policy can be arbitrarily complex. It receives as input the same information as the retransmission delay function, and similarly,

it can query the retransmission context to obtain additional information that can be useful for computing the decision to retransmit or not the message. Additionally, the retransmission policy can also query the local information of our broadcast protocol, that keeps track of all duplicate messages received (for messages whose workflows are still under execution).

If the retransmission policy decides to retransmit the message, it will request the transmission to the local device. Independently of the decision, the workflow proceeds by verifying if the protocol has reached the last retransmission phase, by comparing the current phase of the protocol with the number of retransmission phases associated with that protocol (parameter NP). If the generic protocol was configured to execute additional retransmission phases, the current phase is incremented, and the protocol goes back to the computation of the retransmission delay. Otherwise, the workflow for the current message terminates (and information about duplicates received for that message can be garbage collected).

B. Framework Parameters

Notice that there are four parameters associated with the execution of our generic wireless ad hoc protocol described above. This implies that a wireless ad hoc broadcast protocol is defined in our framework by specifying the values of these four parameters. More precisely, in our framework a concrete wireless ad hoc broadcast protocol can be specified as a tuple: (RC,RDF,RPF,NP), where RC stands for the *Retransmission Context*, RDF represents the *Retransmission Delay Function*, RPF denotes the *Retransmission Policy Function*, and finally, NP denotes the number of retransmission phases configured for that protocol.

In the following, we will discuss examples of possible values for these parameters.

1) Retransmission Context: We now briefly discuss some of the retransmission contexts that can be employed in the design of concrete wireless ad hoc broadcast protocols.

a) $\perp$*:* This is the non-existing context and is employed to denote protocols that do not take into account any environmental external information.

b) POWERAWARE*:* This is a context that records, for each received message, the radio signal strength associated with that transmission. It will further normalize this intensity to a common reference (that depends on the recent history of received transmissions). As it will be shown further ahead, this retransmission context is employed in the design of the families of protocols PAMPA [15] and Flow-Aware [16].

c) GPS*:* This is a context that provides to each node its own set of GPS coordinates. Furthermore, this retransmission context will also tag any outbound message with the location of the transmitting node (this effectively allows nodes that receive a message to compute the distance to the sender).

d) NEIGHBORS(H)*:* This is a particularly interesting retransmission context since it does not require specialized hardware. This context will provide each node information about their neighbors and their neighbors' neighbors and so forth, up to an horizon of H network hops. While this can be materialized by static configuration files in static settings (i.e., where nodes are stationary), it can also be materialized through a simple neighboring protocol where nodes exchange periodic announcement messages, where they include information about their neighbors (up to $H-1$ hops).

2) Retransmission Delay Function: We now provide some examples of possible retransmission delay functions that can be employed when building concrete wireless ad hoc broadcast protocols. We remind the reader that these functions have as (base) input the message m being processed, the sender s from whom the message was received, and the current retransmission phase p in which the protocol is for message m. As it will become obvious, not all retransmission delay functions use these inputs. We, however, represent them for completeness of presentation.

a) RANDOM(T,m,s,p)*:* This is a simple function that computes a small random delay (up to T) that aims at minimizing the probability of two (neighboring) nodes retransmitting a message received from the same peer simultaneously.

b) DISTANCEBYPOWER(T,m,s,p)*:* This function assumes a context where the transmission power associated with a given received message is known. Based on this, this function computes a retransmission delay of at most T, that is proportional to the power of the radio signal of the received message (i.e, weaker radio signals will lead to smaller delays).

c) NEIGHBASED(T,m,s,p)*:* This function assumes a context where nodes have information about their neighbors, in particular the number of neighbors that they have. Based on this information, this function computes a delay, with a maximum value of T, that is inversely proportional to the number of neighbors. This implies that nodes with a higher number of neighbors will use smaller delays, while nodes with fewer neighbors will use delays close to T.

3) Retransmission Policy Function: We now present some examples of possible retransmission policies that can be employed when building concrete wireless ad hoc broadcast protocols. We remind the reader that retransmission policies have the same (base) input values as the functions that compute the retransmission delays.

a) TRUE(m,s,p)*:* This is a simple policy that always decides to retransmit a message.

b) PROBABILITY(λ,m,s,p)*:* This is another simple policy that returns a positive decision with a probability λ, independently of the other input parameters of the function.

c) COUNT(c,m,s,p)*:* This encodes a counting retransmission policy, where the decision to retransmit a message is only positive if the number of duplicates received (including the first reception) for that message is below c.

d) ADDITIONALCOVERAGE(ϵ,m,s,p)*:* This denotes a retransmission policy that leverages a retransmission context that allows nodes to infer their own location and the location of nodes from whom they originally received m and any duplicate of m. Based on this information, the node will compute the amount of area that a retransmission performed by itself will cover, considering the area already covered by the nodes that already retransmitted the message. If this area is

above a given threshold ϵ it will decide to retransmit message m, otherwise, the node will decide not to retransmit m.

e) COVEREDNEIGHBORS(k,m,s,p)*:* This denotes a retransmission policy that, similar to the previous one, takes into account the nodes from whom a copy of m was received and assumes a context that allows nodes to know the neighborhood of nodes (to an horizon of 2 hops). Based on this information, it computes the number of new nodes that will receive the message by its retransmission. If this value is above k, a retransmission is performed, otherwise, the node refrains form retransmitting the message.

C. Examples of Protocol Specification

To illustrate the ability of our framework to specify and capture the particularities of existing wireless ad hoc broadcast protocols, we now provide their specification resorting to the examples of retransmission delay functions, retransmission policy functions, and retransmission contexts discussed previously. We have selected a representative protocol for each of the classes discussed in Section II. We note that to improve readability we only present the parameters that are specific for each retransmission delay and retransmission policy functions. Parameters have values that illustrate a concrete (and possible) materialization of these protocols.

1) Flood-based Broadcast: Flood based broadcast protocols can be easily described in our framework as: ($\perp$, RANDOM(100), TRUE, 1). This implies that the protocol operates with no need of a retransmission context, applying a random delay of at most $T = 100ms$ before transmissions, where the retransmission policy is to always retransmit every message, with a single retransmission phase (i.e., each message is retransmitted by each node exactly one time). This is the most simple of the broadcast protocols that we consider.

2) Probabilistic Broadcast: Probabilistic broadcast protocols are very similar to flood-based protocols, with the key difference that nodes only retransmit each message with a given probability. In this example, we consider a protocol where the probability of retransmitting a message is $\lambda = 0.8$ and a random delay of $T = 100ms$. Such a probabilistic protocol can be described in our framework as: ($\perp$, RANDOM(100), PROBABILITY(0.8), 1).

3) Counting Broadcast: Here we consider a simple counting broadcast protocol, that retransmits a message after a random amount of time, of at most $T = 1000ms$, if the number of duplicates is below $c = 2$. Such a protocol is specified in our framework as: ($\perp$, RANDOM(1000), COUNT(2), 1).

4) Distance-based Broadcast (PAMPA and Flow-Aware): PAMPA [15] is a solution that delays the retransmission of a message m considering the strength of the radio signal of the first reception of m. Hence, the protocol operates with access to an external source of information that allows obtaining these (normalized) values for received messages, which we refer as using the POWERAWARE retransmission context.

In particular, the weaker the signal the smaller the delay used by a node (in this example we consider a delay of at most $500ms$). When this delay expires, a node will only retransmit a message m if it has not observed 2 additional copies of that message transmitted by other nodes. In this protocol, each node attempts to retransmit a message a single time. Hence, PAMPA can be described as: (POWERAWARE, DISTANCEBYPOWER(500), COUNT(2), 1)

Flow-Aware [16] is a variant of PAMPA that introduced the use of two phases of retransmission. Naturally the specification of this protocol is very similar to PAMPA, and a simplification can be captured by: (POWERAWARE, DISTANCEBYPOWER(500), COUNT(2), 2).

5) Location-based Broadcast: We consider a simple variant of the algorithm described in [18], whereas nodes are assumed to have access to a local GPS device that allows them to obtain their location. We denote this by noting that this protocol operates using the GPS retransmission context. This context transparently captures messages sent by nodes, and adds control information indicating the location of the node. The protocol operates as follows: when a node receives a message m, it will attempt to retransmit the m after a small random delay (to minimize collisions). To decide if a retransmission is useful, the node takes into account the location of all nodes from whom it received copies of m and computes the amount of area that a retransmission performed by itself would cover, that was not yet covered by previously received transmissions of that message. If this area is above a given threshold (in this case we assume $\epsilon = 25\%$) the node proceeds with the retransmission, otherwise, it will consider that the additional coverage offered by its transmission is not sufficiently large and avoids the retransmission. Each node attempts to retransmit each message a single time. In our framework we specify such a protocol as: (GPS, RANDOM(100), ADDITIONALCOVERAGE(25%), 1).

6) Neighbor-aware Broadcast (LENWB): Finally, we provide an example of a neighbor-aware broadcast protocol, in particular, the LENWB protocol described originally in [32]. This protocol continuously transmits hello packets containing its local perception of its neighbors, which enables every node to build a knowledge of the network topology with an horizon of 2 hops. We capture this by stating that this protocol operates with a retransmission context of NEIGHBORS(2). The protocol operates as follows: when it receives a message m it will apply a delay, before attempting to retransmit the message, that depends on the number of neighbors of that node (nodes with fewer neighbors will compute higher delays, which we assume will be at most $500ms$). After this delay, the node compares its neighbors list with the list of all nodes from whom it received a copy of m. If its transmission will only reach nodes that have been covered by previous transmissions of m it decides not to retransmit the message, otherwise it retransmits m. Each node only attempts to retransmit a message a single time. In our framework this protocol can be described as: (NEIGHBORS(2), NEIGHBASED(500), COVEREDNEIGHBORS, 1).

IV. NABA: NEIGHBOR-AWARE BROADCAST ALGORITHMS

In this section, we leverage the framework previously described to study the family of protocols that relies on neigh-

boring information, that we name **N**eighbor-**A**ware **B**roadcast **A**lgorithms, or simply, NABA. We consider this class of protocols to be of particular relevance, due to the fact that they can be implemented in commodity hardware without resorting to specialized hardware (such as GPS receivers or radios that are capable of measuring the strength of the radio signal associated with received messages).

A. Neighbor-Aware Retransmission Context

Broadcast protocols that leverage information about the local network topology for their operation can obtain this information through a companion protocol that periodically transmits an announcement containing the node's identifiers. If the n-hop neighborhood is also required, such announcements can also carry the information obtained so far for neighbors up to $n-1$ hops.

We notice, however, that not all nodes are important for the successful retransmission of a broadcast message, an observation also made by other neighbor-based broadcast algorithms [19], [30], [33]. Hence, a simple discovery protocol, such as the one described above, could be enriched to compute the relationships between nodes regarding message retransmission coverage. These relationships can be computed by performing comparisons of neighbor sets between neighboring nodes, attributing to each neighboring a label: i) *Reduntant*; ii) *Covered*; or iii) *Critical*. We refer to this enriched context as LABELNEIGH(H), being $H = 2$ in our solution.

These labels are computed between two neighboring nodes a and b in the following way. The neighbor b of a node a is said to have a relation of *Reduntant* if, and only if, the neighbors of b are the same as the neighbors of a (excluding a and b). The neighbor b of a node a is said to have a relation of *Covered* if, and only if, the neighbors of b are a strict subset (i.e., all neighbor of b are contained in) of a's neighbors. The neighbor b of a node a is said to have a relation of *Critical*, if it is neither *Redundant* nor *Covered*.

B. Neighbor-Aware Retransmission Policies

Neighbor-based broadcast algorithms whose retransmission policy is only based on the coverage of neighbors can be enriched with additional information. The retransmission policies presented here, all present the dependency of a neighbor retransmission context. We now present 3 different neighbor-aware retransmission policies:

a) NEIGHBORCOUNTING(c,m,s,p)*:* This retransmission policy enriches a counting policy (with parameter c) with information about the number of neighbors of the local node. The decision to retransmit the message is only positive when the number of received duplicates is lower than the minimum between c and the number of neighbors of the current node.

b) PBNEIGHCOUNTING($c1$,$c2$,m,s,p)*:* This retransmission policy enriches the first one, with two threshold parameters $c1$ and $c2$, where $c1 < c2$. If the number of received duplicates of m is greater than or equal to the minimum between the number of neighbors of that node and $c2$, it does not retransmit the message. Otherwise, if the number of received duplicate messages c is lower than $c1$, the message is retransmitted. If the number of received duplicates c is between $c1$ and $c2$, a probability based on c is calculated to retransmit the message, having higher probabilities for lower values of c.

c) CRITICALNEIGH(m,s,p)*:* This retransmission policy leverages the labels computed by the neighbor-aware retransmission contexts to perform retransmission decisions. A message is only retransmitted if the sender s has been labeled as *Critical*, or $s = \perp$. This policy can also leverage multiple phases to ensure that *Critical* nodes are able to propagate messages, as such, when the phase p is greater than 1, the policy counts the number of all retransmissions performed by *Critical* neighbors, retransmitting the message if at least one such neighbor has failed to transmit the message.

Based on the policies described above, we derived the following protocols:

NABA1: (NEIGHBORS(1),NEIGHBASED(1000),NEIGHBORCOUNTING(2),1)
NABA2: (NEIGHBORS(1),NEIGHBASED(1000),PBNEIGHCOUNTING(1,4),1)
NABA3: (LABELNEIGH(2),NEIGHBASED(1000),CRITICALNEIGH(),1)
NABA4: (LABELNEIGH(2),NEIGHBASED(1000),CRITICALNEIGH(),2)

In the following section, we experimentally access the performance of these variants and compare it with the existing state of the art.

Label	Specification
Flood	($\perp$, RANDOM(1000), TRUE, 1)
Counting	($\perp$, RANDOM(1000),COUNT(2),1)
Gossip	($\perp$, RANDOM(1000), PROBABILITY(0.8), 1)
LENWB	(NEIGHBORS(2), NEIGHBASED(1000), COVEREDNEIGHBORS, 1)
NABA1	(NEIGHBORS(1),NEIGHBASED(1000),NEIGHBORCOUNTING(2),1)
NABA2	(NEIGHBORS(1),NEIGHBASED(1000),PBNEIGHCOUNTING(1,4),1)
NABA3	(LABELNEIGH(2),NEIGHBASED(1000),CRITICALNEIGH(),1)
NABA4	(LABELNEIGH(2),NEIGHBASED(1000),CRITICALNEIGH(),2)

TABLE I
LABEL AND SPECIFICAITON OF TESTED BROADCAST PROTOCOLS

V. EVALUATION

In this section, we describe our practical assessment of a variety of broadcast algorithms resorting to our framework. The framework and all the modules used in our evaluation were implemented in the C language resorting to Yggdrasil [7], a framework to develop wireless ad hoc protocols that encourages protocols to be developed with clean event-driven interfaces.

We have implemented a small and parameterizable kernel for wireless ad hoc broadcast protocols, that fundamentally is an implementation of the generic protocol previously presented in Section III. We further implemented the retransmission delay and policy functions described previously that do not require specialized hardware. We implemented the neighbor-aware contexts as companion protocols, that execute alongside the broadcast protocol and expose an interface to allow other protocols to obtain context information from it.

In the following, we describe our experimental methodology and discuss our experimental results.

A. Experimental Methodology

We have conducted experiments with the protocols described in Table I. Notice that we describe the protocols that

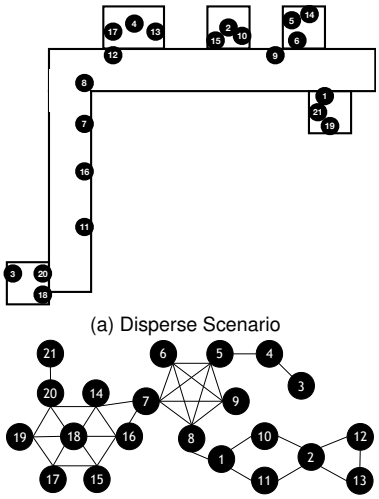

Fig. 2. Node distribution in each scenario

we used in our experiences using the notation of the framework described previously. All time references are in milliseconds.

The companion protocols that gather information about the local topology was configured to issue messages every six seconds, to minimize the contention in the wireless medium due to this protocol.

We have conducted our experiments using a practical testbed, composed of 21 Raspberry Pi 3 - model B, in two different scenarios. One where the nodes are dispersed through our department building, along two hallways, with approximately 30 meters, and several rooms (illustrated schematically in Figure 2a). Due to the unpredictability of the computed neighborhood in each experiment in the first scenario, we also conducted experiments in a second scenario, where all nodes were collocated in a single room, and we have artificially restricted neighborhood relationships, effectively producing a logical overlay network (the logical network is illustrated in Figure 2b) [34]. We note that this last scenario is highly challenging for broadcast protocols because, although devices are filtering messages from nodes with whom they lack a logical relation, their messages still collide, which results in significant contention in the wireless medium.

All reported experiments had a duration of 10 minutes (600 seconds) with grace periods in the beginning (60 seconds in the disperse scenario and 5 seconds in the local scenario) and the end (60 seconds in both scenarios). This allowed our experimental deployment to stabilize. Each experiment is executed 3 times and the results show the average of all (independent) runs. In the following we present the experimental results in both scenarios.

B. Experimental Results

Our experimental evaluation was focused on the relevant performance metrics of wireless ad hoc broadcast protocols previously discussed: reliability (average fraction of correct processes that delivered a broadcast message), and cost (the average number of transmissions performed to disseminate a

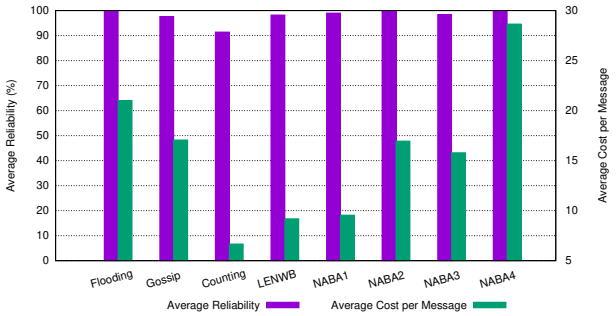

Fig. 3. Broadcast Algorithms in Disperse Scenario

message). A reliability of 100% implies that all messages were delivered to all correct processes in the system. Notice that the cost is relative to the total number of processes in the system. Solutions where each node performs a single transmission per message results in a cost of 21 (as our experimental setup is composed of 21 nodes).

1) Stable Scenario: In this set of experiments, we have configured each node to perform a broadcast every 2 seconds with a probability of 50%. Our goal was to study the performance of different alternatives in executions where nodes do not fail (still collisions may happen in the wireless medium). In each experiment, approximately a total of 3150 messages were broadcast. Plots encode simultaneously the average reliability (left y-axis, purple bars) and cost (right y-axis, green bars) for each broadcast protocol.

a) Disperse Scenario: Figure 3 shows the results we have obtained in this setting where nodes are scattered across the hallways of our department. Flooding shows expected results, it achieves perfect reliability at the cost of having each node retransmitting each message once. We note that the success of this protocol is in part due to the high random delay of $1000ms$ which significantly minimizes the risk of collisions in the wireless medium. However, such a solution could not be sustained if the transmission rate of nodes was higher. The gossip solution achieves slightly lower reliability, however, the cost is lower due to its probabilistic nature (the cost is reduced by approximately 20% as expected). The counting solution has reliability slightly above 90%. The reason for this is the c parameter set to 2. This confirms our previous observation, that the lack of information about the local network topology leads some nodes, that are bridging different partitions of the network, to not retransmit messages. However, it exhibits the lowest cost among all tested protocols.

Concerning the neighbor-aware broadcast solutions, we note that the one with best overall performance is NABA2, having reliability of 100% at the lowest cost. This is an interesting result, as this protocol relies on a very simple retransmission context (it only requires each node to keep track of its direct neighbors) and it combines this solution with a retransmission policy that always retransmits messages when no duplicate messages are received, except when nodes have fewer or equal neighboring nodes to parameter $c1$ (in this case $c1 = 1$),

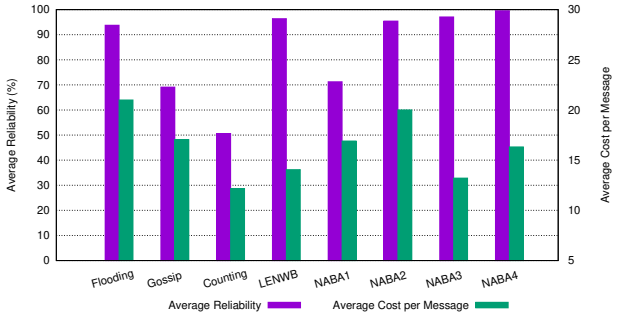

Fig. 4. Broadcast Algorithms in Local Scenario

avoiding redundant retransmissions. When more duplicates are received this protocol decides to retransmit with a probability that lowers with the number of duplicates (becoming zero after the reception of 4 duplicates). The only other protocol to achieve a reliability of 100% is NABA4, that attempts to identify critical nodes to retransmit messages and, for those, it can attempt to retransmit messages twice. However, this second retransmission leads to the cost of this solution becoming too high, even above that of flooding. LENWB and NABA1 also performed well, having reliability close to 100% with low cost. NABA3 showed similar reliability but at a higher cost. An interesting aspect here is that the policies COVEREDNEIGHBORS and NEIGHBORCOUTING(2) appear to be effectively equivalent. This happens because our experiments were conducted in a static topology, where nodes have no mobility. At a high level, both policies strive to ensure coverage of all neighbors. However, it is expected that NEIGHBORCOUTING (2) should present superior results in dynamic topologies.

b) Local Scenario: Figure 4 reports our results for the local scenario, which employs the logical network to restrict the origins of messages that nodes can receive and process, effectively lowering the natural redundancy promoted by the use of one-hop broadcast. Notice that contention in the wireless medium is higher in this scenario. The results show that NABA3 and NABA4, which resort to the LABELNEIGH(2) context, achieve the best performance overall, with high reliability and lower cost. In this context, the second phase of retransmission in NABA4 becomes useful, making this the only protocol to achieve 100% reliability. The other protocols are significantly affected by the lack of redundancy in the wireless network. The key take away from these results is twofold: i) identifying nodes that are in strategic positions in the network is essential for achieving adequate performance in sparse networks; and ii) multiple retransmission phases are adequate only for this type of network.

2) Faulty Scenarios: To effectively evaluate the effects of faults in the evaluated protocols, we have performed experiments in our local scenario, where we introduced three simultaneous node crashes (on nodes 9, 10, and 18, see Figure 2b). These node crashes make the network more sparse further reducing redundant communication paths. In these

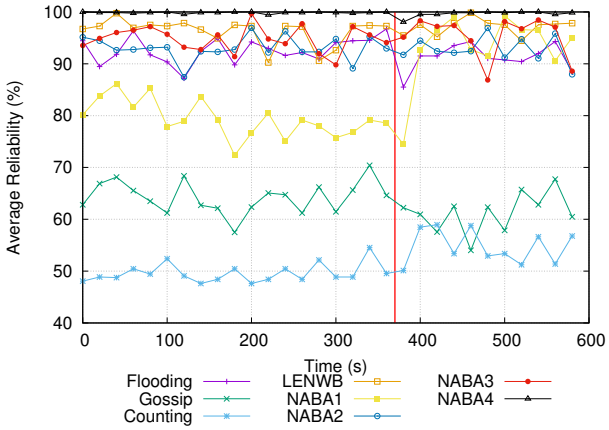

Fig. 5. Average reliability in the faulty scenario

experiments, all nodes transmit every 20 seconds and failures occur around the $370s$ mark. Figure 5 report the reliability over time with faults being marked by the vertical line).

All protocols are able to sustain their reliability to values that are consistent with those reported in Figure 4, with the exception of NABA1 whose reliability increases to become on par with most of the solutions (notice that only NABA4 is able to achieve a reliability of 100% with a small drop after failures). The results obtained for NABA1 are explainable by the fact that failures reduce redundancy, leading nodes to be unable to receive two redundant messages before executing the retransmission policy (which incidentally is executed at a later time for some nodes), hence the protocol operation becomes similar to flood.

VI. RELATED WORK

Having already discussed several wireless ad hoc broadcast protocols, here we focus on frameworks for defining them.

The authors of [35] present a generic conceptual framework for designing peer sampling services based on gossip protocols in wired environments. Gossip protocols have been the basis for many dissemination abstractions [36] in wired networks and many wireless ad hoc broadcast protocols follow their design. However, this framework does not discuss a generic broadcast protocol as our does.

Frameworks for broadcast protocols in wireless networks have also been explored in [30] and [31] however, these only consider neighbor-aware broadcast protocols that rely on the computation of connected dominating sets, to define which nodes should retransmit messages. Our framework is more general and able to model other classes of protocols.

VII. CONCLUSION

In this paper, we have presented a conceptual framework to specify and easily define wireless ad hoc broadcast protocols. This framework is based on a generic broadcast protocol, whose behavior is parameterized across three different dimensions: retransmission delay, retransmission policy, and retransmission context. We employed our framework to study

a particularly interesting class of protocols, those that are neighbor-aware (NABAs). We have implemented a prototype of a kernel to support the construction and execution of wireless ad hoc broadcast protocols based on the proposed framework and conducted an experimental work comparing, in practice and using commodity hardware, different broadcast protocols. Our results show that neighbor-aware protocols exhibit interesting results and should be further pursued to enable novel edge applications for wireless ad hoc networks to emerge.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 10 2016.

[2] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. Ieee, 2010, pp. 27–33.

[3] J. Leitão, P. Á. Costa, M. C. Gomes, and N. M. Preguiça, "Towards enabling novel edge-enabled applications," Tech. Rep., 2018. [Online]. Available: http://arxiv.org/abs/1805.06989

[4] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 10 2015.

[6] D. G. Reina, S. L. Toral, F. Barrero, N. Bessis, and E. Asimakopoulou, "The role of ad hoc networks in the internet of things: A case scenario for smart environments," in *Internet of Things and Inter-Cooperative Computational Technologies for Collective Intelligence*. Springer, 2013, pp. 89–113.

[7] P. Á. Costa, "Practical aggregation in the edge," Master's thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2018.

[8] R. Guerraoui and L. Rodrigues, *Introduction to Reliable Distributed Programming*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[9] P. Verissimo and L. Rodrigues, *Distributed Systems for System Architects*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[10] J. Liang, S. Y. Ko, I. Gupta, and K. Nahrstedt, "Mon: On-demand overlays for distributed system management," in *Proceedings of the 2Nd Conference on Real, Large Distributed Systems - Volume 2*, ser. WORLDS'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 13–18.

[11] R. Baldoni, S. Bonomi, A. Cerocchi, and L. Querzoni, "Improving validity of query answering in dynamic systems," in *Proceedings of the Third International Workshop on Reliability, Availability, and Security*, ser. WRAS '10. New York, NY, USA: ACM, 2010, pp. 4:1–4:6.

[12] D. Frey, R. Guerraoui, A. . Kermarrec, and M. Monod, "Boosting gossip for live streaming," in *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, Aug 2010, pp. 1–10.

[13] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec 2007.

[14] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless networks*, vol. 8, no. 2-3, pp. 153–167, 2002.

[15] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen, "A power-aware broadcasting algorithm," in *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, Sep. 2006, pp. 1–5.

[16] D. Lima and H. Miranda, "Flow-aware broadcasting algorithm," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 1601–1608.

[17] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. ACM, 2002, pp. 194–205.

[18] M.-T. Sun, W. Feng, and T.-H. Lai, "Location aided broadcast in wireless ad hoc networks," in *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*, vol. 5. IEEE, 2001, pp. 2842–2846.

[19] W. Peng and X. Lu, "Ahbp: An efficient broadcast protocol for mobile ad hoc networks," *Journal of Computer Science and Technology*, vol. 16, no. 2, pp. 114–125, Mar 2001.

[20] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 2. IEEE, 2003, pp. 1124–1130.

[21] T. J. Kwon and M. Gerla, "Efficient flooding with passive clustering (pc) in ad hoc networks," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 1, pp. 44–56, Jan. 2002.

[22] W. Peng and X.-C. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*. IEEE Press, 2000, pp. 129–130.

[23] M. Torrent-Moreno, S. Corroy, F. Schmidt-Eisenlohr, and H. Hartenstein, "Ieee 802.11-based one-hop broadcast communications: Understanding transmission success and failure under different radio propagation environments," in *Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '06. New York, NY, USA: ACM, 2006, pp. 68–77.

[24] J. Leitao, J. Pereira, and L. Rodrigues, "Hyparview: A membership protocol for reliable gossip-based broadcast," in *DSN'07*, June 2007, pp. 419–429.

[25] M. D. Colagrosso, "Intelligent broadcasting in mobile ad hoc networks: 3 classes of adaptive protocols," *EURASIP J. Wireless Comm. and Net.*, vol. 2007, no. 1, p. 010216, 2006.

[26] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," in *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, June 2002, pp. 1707–1716 vol.3.

[27] R. Hu, "Efficient probabilistic information broadcast algorithm over random geometric topologies," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[28] M. B. Yassein, S. F. Nimer, and A. Y. Al-Dubai, "A new dynamic counter-based broadcasting scheme for mobile ad hoc networks," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 553 – 563, 2011, modeling and Performance Analysis of Networking and Collaborative Systems.

[29] C. Winstanley, R. Ramdhany, F. Taïani, B. Porter, and H. Miranda, "Pampa in the wild: a real-life evaluation of a lightweight ad-hoc broadcasting family," *Journal of Internet Services and Applications*, vol. 5, no. 1, p. 5, Apr 2014.

[30] J. Wu and F. Dai, "Broadcasting in ad hoc networks based on self-pruning," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 3, March 2003, pp. 2240–2250 vol.3.

[31] J. Wu and F. Dai, "A generic distributed broadcast scheme in ad hoc wireless networks," *IEEE Transactions on Computers*, vol. 53, no. 10, pp. 1343–1354, Oct 2004.

[32] J. Sucec and I. Marsic, "An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks," Tech. Rep., 2001.

[33] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks," INRIA, Research Report RR-3898, 2000. [Online]. Available: https://hal.inria.fr/inria-00072756

[34] P. Á. Costa and J. Leitão, "Practical continuous aggregation in wireless edge environments," in *Proc. of 37th IEEE International Symposium on Reliable Distributed Systems (SRDS'18)*. Salvador, Brazil: IEEE, 2018.

[35] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "The peer sampling service: Experimental evaluation of unstructured gossip-based implementations," in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, ser. Middleware '04. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 79–98.

[36] J. Leitao, J. Pereira, and L. Rodrigues, "Epidemic broadcast trees," in *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, Oct 2007, pp. 301–310.