NABA: Neighbor-Aware 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 computational power at the edge. In particular, applications in the domain of smart cities and IoT can rely on devices in the vicinity of data consumers and producers for their operation. However, we can expect these devices to lack access to network infrastructure and to be wirelessly capable, requiring and allowing them to form a wireless ad hoc network to coordinate and cooperate. In this context, one of the most important primitives is the broadcast of messages, that can be leveraged as a building block for more complex distributed applications.**

Although the literature of broadcast algorithms in wireless ad hoc networks is vast, most solutions fail to be practical in common devices as they rely on tailored hardware primitives. Therefore, in this paper we present a novel class of broadcast algorithms that do not present such hardware constraints. To achieve this, we generalize previous techniques to instead leverage on the local neighbourhoods of nodes to guide retransmission policies. In addition, we provide a generic framework to define various broadcasting algorithms by simply specifying retransmission and delay policies. Finally, we evaluate our solutions, and relevant state of the art, in a testbed composed of 23 Raspberry Pi 3 - model B.

***Index Terms*—Broadcast Algorithms, Wireless ad hoc, Reliability, Latency**

I. Introduction

The edge computing paradigm [**?**] emerged to address the limitations of current cloud-based applications. These applications produce high volumes of data [**?**] 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. However, the edge can be materialized in various forms [**?**], from fog computing [**?**] to IoT networks [**?**], [**?**] with computational power. Given this, in this paper we focus on a concrete scenario of edge computing, where wireless capable commodity devices form a wireless ad hoc network [**?**]. Such scenario can be the case in application domains of smart cities and IoT, where devices lack access to network infrastructure.

To promote novel application to emerge that leverage the computational resource in these wireless capable devices, adequate abstractions should be provided. One of the most fundamental primitive for distributed applications is the broadcast of messages [**?**], where all processes of a system should deliver a given message disseminated by one process. Broadcast primitives have been the exploited by complex distributed applications [**?**] and protocols [**?**] as a means to provide coordination.

Unfortunately, providing reliable broadcast protocols in wireless ad hoc networks is not a trivial task. This is due to the shared nature of the wireless medium, which render reliable broadcast protocols [**?**] that rely on reliable channels (e.g., TCP channels) and on the acknowledgment of messages unsuitable as they occupy the wireless medium leading to high collisions [**?**] and contention. This observation leads to the conclusion that a successful broadcast protocol for wireless ad hoc networks should be best-effort while maximizing its reliability.

Best-effort broadcast protocols have been previously proposed by the scientific community in the context of wireless ad hoc networks [**?**], [**?**], [**?**]. However, previous solutions leverage properties provided by hardware components (e.g., GPS, message reception power) that might not be available in common devices. As such, current solutions will be unable to be leveraged seamlessly by applications which should be abstracted by hardware properties.

This motivates the necessity to provide best-effort broadcast abstractions, which provide high reliability in the case of communication faults, and that strip hardware constraints. As such, in this paper we present a novel class of broadcast algorithms that leverage local topological information (i.e., direct neighborhoods) to schedule retransmission policies.

The remainder of the paper is structured as follows: Section II discusses key properties that broadcast protocols in wireless ad hoc networks should guarantee; Section III surveys related work; Section IV presents neighbor-aware broadcast algorithms (NABA), as a novel class of broadcast of algorithms; Section V present a practical evaluation of NABA and classical algorithms using an experimental testbed composed of 21 Raspberry Pis 3; Section VI concludes the paper.

II. Broadcasting in Wireless edge networks

Explain basic properties of broadcasts

Properties that should be guaranteed (maximized and minimize):

The work presented here was partially supported by the Lightkone European H2020 Project (under grant number 732505) and NOVA LINCS (through the FC&T grant UID/CEC/04516/2013).

A. Reliability

B. Cost

C. Latency

III. Related Work

A. Classical Algorithms

1) Flooding: Flooding [**?**] is the most simple broadcast algorithm and works by having all the processes on the network retransmit each message they receive, for the first time, propagating them to their neighbours. Although this algorithm being a simple and robust solution, it's very expensive and wasteful (high cost) because, on the one hand, by forcing every node to retransmit every message, it incurs in a lot of redundant retransmissions, since all the reachable nodes may have already received the messages, and, on the other hand, since the number of simultaneous retransmissions is (potentially) high, there's an elevated risk of incurring in collisions. One way to try to mitigate collisions is to make every node delay each retransmission by a random amount of time. Therefore, the probability of simultaneous retransmissions greatly diminishes leading, in theory, to fewer collisions. However, this mechanism also slows down the message propagation. Nevertheless, there are still a lot of redundant retransmissions due to the fact that every node still retransmits all the messages.

2) Probabilistic Algorithms (GOSSIP): To reduce the number of redudant messages, each process could retransmit each message with a given probability P - *Probabilistic Algorithm* [**?**], [**?**], [**?**], [**?**], [**?**]. Therefore, the cost of this algorithms would be aproximatly P times the cost of *Flooding*. Clearly, if $P = 1$, this algorithm is *Flooding*. However, due to the non deterministic behaviour of these algorithms, some processes that abstain from retransmiting may be critical to the propagation of the message to isolated processes or to other network partions, terminating the propagation of the message before reaching all the processes.

3) Counter-based Algorithms: On these approaches [**?**], [**?**], [**?**], while the message is waiting to be transmitted, each process may receive several redundant transmitions (copies) of it, due to the transmissions of other process(es). Thus, a process can abstain from retransmitting a given message if it has received a certain amount C or more copies, being C a parametrizable thresold of the algorithm. However, like the previous algorithms, some processes that abstain from retransmiting may also be critical to the propagation of the message to isolated processes or to other network partions, terminating the propagation of the message before reaching all the processes.

4) Distance-based Algorithms: In this variants [**?**], the relative distance between nodes is used to influence the decision to retransmit (or not) a given message. If the distance d, between a process that received a message and the sender of that message, is small, there's little additional coverage the transmission can provide, and thus these processes can abstain from retransmitting. However, if d is large, the additional coverage will also tend to be large, and thus that processes should retransmit the message. In short, let d_{min} be the distance to the node from which the message was received, if $d_{min} < D$, being D a parameter threshold, that node abstains from retransmiting the message, retransmiting it otherwise.

In general, this distances are usually estimated through the Received Signal Strength Indicator (RSSI) of each message. However, in a grand part of cases, it is not possible to obtain this information, and thus these algorithms could not be used.

5) Location-based Algorithms: By leveraging on location information [**?**], such as GPS coordinates, of the nodes of the network, it's possible to estimate the additional coverage a give retransmittion by a given node may present, and thus utilize this information to abstain form retransmiting or not the messages. However, parallel to the previous situation, in a grand part of cases, it is also not possible to obtain this information.

6) Cluster-based Algorithms: In this algorithms the processes periodically announce their presence, enabling any other process to determine its neighbours. Then, the nodes run a cluster formation protocol, that maintains the clusters and elects their leaders - the cluster head. Within a cluster, a member that can communicate with a node in another cluster is a gateway. In a cluster, the cluster heads retransmission covers all the other nodes in that cluster, if its transmission suffers no collisions. The gateway nodes are responsible to propagate the broadcast message to nodes in other clusters. Thus, there's no need for a non-gateway member to retransmit the message [**?**].

B. Practical Algorithms

1) PAMPA: Power-Aware Message Propagation Algorithm: *PAMPA* is the most well-known algorithm for broadcast in *ad hoc* networks. This algorithm results from the combination of a distance based algorithm and a counting algorithm. Thus, it requires that every process is able to retrieve the strength of the received message (Received Signal Strength Indicator, or RSSI), in order to estimate the distance to the emissor process. It uses the RSSI to calculate the amount of delay to employ to each retransmission, being that messages with higher rssi, i.e., the receiving process is closer to the emissor, have larger delays than messages with lower rssi, i.e., the receiving process is farther from the emissor. Thus, *PAMPA* sorts the retrasmissions by the distance between the receiving and emitting processes. Additionally, *PAMPA* resorts to a counting strategy, to cancel some transmitions, that counts the repetions received of any pending message. If the number of repetions reaches an given (parametrizable) thresold, the retransmission of the message is canceled, and consequently only a limited number of nodes retransmits each message, [**?**].

Although *PAMPA* is more robust than the previous simple algorithms, it still does not perform well in heterogeneous topologies, where some nodes may be critical to the propagation of messages. In order to address this problem, several variants of *PAMPA* were proposed in [**?**] and evaluated in [**?**], based on two types of information: the retransmission path

TABLE I
PAMPA VARIANTS ACCORDING TO COMMON PARENT AND DYNAMIC THRESHOLD.

Common Parent	Dynamic Threshold	Variant Name
No	No	D-Flooding
No	Anti-threshold	PAMPA-ATH
No	Threshold	PAMPA-TH
Yes	No	PAMPA-CP
Yes	Anti-threshold	PAMPA-ATH/CP
Yes	Threshold	PAMPA-TH/CP
No	Threshold & Anti-threshold	PAMPA Vanilla

(*Common Parent*), and a threshold on the distance to the sender of the message (*Dynamic Threshold*).

a) Common Parent: This strategy consists on comparing the nodes that are along the retransmissions path that lead to a message delivery. In particular, the node at one hop (the parent) and two hops (the parent's parent) and thus, only counting messages originated by different two hops parent nodes. This approach reflects a higher diversity of propagation paths, and consequently, messages in which the emitting node's parent are probably less useful to be retransmitted.

b) Dynamic Threshold: Dynamic Threshold uses the power of the received message (RSSI) to additional influence the decision of retransmission of the message in the sense that only retransmissions that have an higher (or lower) RSSI than a predetermined threshold are counted. This threshold is set to be equal to the RSSI of the first message received. This criteria has two sub-variants: Threshold and Anti-threshold. In Threshold, only retransmissions with a lower RSSI than the first reception are counted. On the other hand, in Anti-threshold, only retransmissions with an higher RSSI are counted.

The combination of the two previous strategies originates five variants of PAMPA, that are described in the Table I, alongside of *Delayed Flooding (D-Flooding)* and the original *PAMPA* (*PAMPA Vanilla*).

2) Flow-Aware Broadcast Algorithm: In spite of existing several variants of *PAMPA*, there are still some cases when critical nodes cancel their retransmissions, i.e., decide to not retransmit when they should. Therefore, there were proposed in [**?**] several improvements to it. These proposed algorithms are divided in two classes. The first tries to extend the awareness of the nodes in relation to what is happening with the propagation, allowing them to retransmit the message again if they detected a failure in the process (Two-phased approach). The other approach, opts to enlarge, at run-time, the delay before a retransmission to avoid having nodes to incorrectly retransmit messages thus, increasing the opportunity of other (more appropriate) nodes to retransmits (Dynamic Timeout).

a) Two-phased approach: This strategy tries to identify failures in the retransmission process and recover from this. To this end, it extends the observation period for some nodes, that become responsible for retransmitting if they suspect that a failure has occurred. In the first phase, nodes behave like the *PAMPA* algorithm, by having nodes that are expected to provide the most coverage, retransmit the messages (i.e, nodes that are more distant to their parents). Additionally, nodes that did not retransmit in the first phase, enter a second phase where they continue listening for retransmissions of the message. This phase has a timer with a delay equal to the one defined for the first phase. When this second timer expires, nodes re-evaluate their initial decision of not retransmitting. However, unlike *PAMPA*, a retransmission in the second phase depends on the number of distinct nodes two hops away (different parents) that were observed transmitting that message, instead of simply the total number of retransmissions observed.

b) Dynamic Timeout: This strategy opts to extend, at run-time, the delay of a pending retransmission, with the objective of postponing the decision of nodes as to avoid unnecessary resource consumption. In particular, this approach tries to avoid nodes that are in close proximity of a previous transmission to retransmit the message, allowing other (farther away) nodes to have a chance at retransmitting. This approach is similar to *PAMPA*, where the first delay is proportionally to the signal strength of the first received retransmission. However, for every retransmission listened, a new delay is calculated that replaces the current delay (if it's greater than the previous).

IV. NABA: NEIGHBOR-AWARE BROADCAST ALGORITHMS

A. System Model

B. Design of NABA

C. Variants

V. EVALUATION

A. Broadcast Algorithms Framework

B. Experimental Methodology

C. Experimental Results

VI. CONCLUSION