GRACO: a geographic GReedy routing with an ACO-based void handling technique

Mouna Rekik, Nathalie Mitton, Zied Chtourou

GRACO: a geographic **GR**eedy routing with an ACO based void handling technique

Mouna Rekik[1,2], Nathalie Mitton[1], Zied Chtourou[3]

[1] Univ Lille Nord de France, IFSTTAR, COSYS, LEOST, [2] Inria,
[3] Digital Research Center of Sfax, Tunisia

March 27, 2017

Abstract

Geographic routing has gained much attention as a basic routing primitive in wireless sensor networks due to its memory-less, scalability, efficiency and low overhead features. Greedy forwarding is the simplest geographic routing scheme, it uses the distance as a forwarding criterion. Nevertheless, it may suffer from communication holes, where no next hop candidate is closer to the destination than the node currently holding the packet. For this purpose, a void handling technique is needed to recover from the void problem and successfully deliver data packets if a path does exist between source and destination nodes. Many approaches have been reported to solve this issue at the expense of extra processing and or overhead. This paper proposes GRACO, an efficient geographic routing protocol with a novel void recovery strategy based on ant colony optimization (ACO). GRACO is able to adaptively adjust the forwarding mechanism to avoid the blocking situation and effectively deliver data packets. Compared to GFG, one of the best performing geographic routing protocols, simulation results demonstrate that GRACO can successfully find shorter routing paths with higher delivery rate, less control packet overhead and shorter end-to-end delay.

1 Introduction

Wireless sensor networks (WSN) consist of a large number of densely deployed sensors that have communication, computing, and sensing capacities [1]. Although sensors have power and memory constraints, they are multi-functional with sensing, wireless communication, computation capabilities and low-cost devices. For these reasons, WSNs are widely used in many fields as military surveillance, disaster prediction, and environment monitor [1]. WSNs require efficient routing protocols that adapt to the unpredictable and highly dynamic

environment. The network topology may change dynamically due to node mobility, node failure and various physical properties related to the propagation channel (e.g., obstructions, noise, and power limitations) [2].

Geographic routing [3] is an attractive routing technique for large scale wireless sensor networks due to its low overhead, high scalability and memory-less features. Unlike topology-based routing, it uses only local information about the geographic location of nodes to determine, at each step, the next node to forward the packet. Greedy geographic routing schemes consist of forwarding data packets closer to the destination at each step of the routing process. This can lead to data packets stuck in situations where the current node fails to find a node closer to the destination than itself. These situations are later referred as *the void problem*. Hence, geographic routing algorithms usually combine a greedy forwarding strategy with a recovery mechanism to solve the void problem. Several recovery strategies have been proposed in the literature, face routing is the most prominent and effective one since it was proven to guarantee data delivery. However, it relies on assumption of a planar graph which is not always achievable in realistic wireless networks and may generate long detours.

In this paper, we present GRACO a new geographic routing protocol that combines a modified greedy forwarding (GR) phase and an Ant-Colony-Optimization (ACO)-based recovery strategy. The main contribution of this paper is a geographic routing that dynamically avoids holes and creates multiples paths around them. The ACO-based recovery phase will use ants to discover alternative paths if greedy is not possible. The *ant* packets are sent around the void searching for route to a closer node than the stuck node to destination, then the protocol can switch back to greedy mode until the destination or a new void. GRACO greedy forwarding is assisted by pheromone trails from previous recovery phases, which will prevent to return to the same stuck node.

The remainder of this paper is as follows. Section 2 discusses the most relevant related work in geographic routing, recovery techniques and ACO based routing. Section 4 presents GRACO. Section 7 analyzes simulation results where CRACO and GFG routing are compared. Finally, Section 9 concludes this work.

2 Related work

This section browses the different concepts of the literature relative to GRACO.

2.1 Geographic routing

Geographic routing [3] is a routing concept that exploits geographic information instead of topological connectivity. It is localized, since only local information such as the geographic position of the current node holding a packet and of its one-hop neighbors in addition to the one of the destination are needed to make routing decision. Thus, there is no need to transmit routing messages to establish complete routes or update their states. Nodes do not have to store routing table or write additional information in the packets either. Hence, the

low-overhead, localized and memory-less features make the geographic routing a simple and scalable routing concept.

A geographic forwarding strategy defines the next hop to which forward the packet using only geographic information. Several variants of geographic forwarding strategies have been proposed based on the way to exploit geographic information to forward a packet toward destination. Greedy [4], Most Forward Within Radius (MFR) [5], Nearest with Forward Progress (NFP) [6] and Compass [7] are the most famous geographic forwarding strategies, they differ in the way to exploit geographic information to forward a packet toward destination: the geographic distance, the largest projection, the shortest hop or the angle respectively. Greedy [4] forwards the message to the neighbor that minimizes the Euclidean distance to the destination in each step [3]. Geographic Greedy forwarding is loop free. Indeed, at each step, the message has to move toward the destination and thus can not loop by going through a node it has already visited [3]. However, Greedy forwarding may not always be possible if all neighboring nodes are further away from the destination than the sender itself. This problem is called communication void, local maximum phenomenon or local minimum phenomenon. It is caused by deployment holes where the forwarding process is blocked at a node called stuck node. The occurrence of hole can be caused by many factors, such as sparse deployment, physical obstacles, node failures, communication jamming, power exhaustion, and animus interference [8]. An alternative mode called recovery mode is then needed to

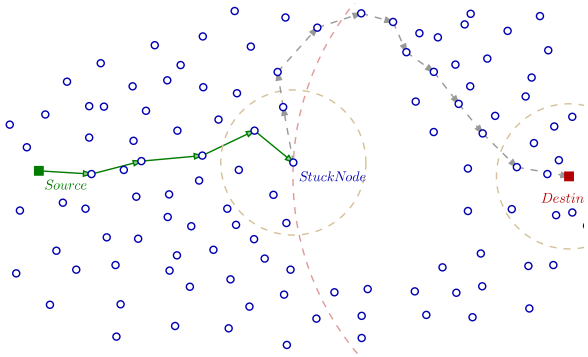


Figure 1: The stuck node has no positive progress neighbors to D. Although a path to D exists, the stuck node can not progress the packet using GF.

guarantee delivery otherwise the packet has to be discarded and the delivery fails.

2.2 Recovery techniques

Solving the problem of communication voids in geographic routing in WSN is the subject of considerable research, many studies have been focusing on this topic and different solutions have been proposed. The existing void handling

strategies can be classified into six categories [9, 10, 11] : flooding-based, planar-graph-based, geometric, cost-based, heuristic, and hybrid.

Intuitively, the simplest recovery technique is flooding, initiated at the stuck node and executed afterwards at all node receiving the stuck packet for the first time. Original flooding and other advanced flooding-based techniques have been proposed to recover from the communication void in geographic routing . These techniques will certainly enable stuck packet to reach the destination if at least a path exists, however, they are inefficient in terms of resource utilization.

Planar graph based recovery techniques rely on a planarization process and a planar graph traversal algorithm. The planarization process consists of creating a plane sub-graph of the network graph where no edges intersect each other. Face routing [12] was used as the basic planar graph transversal algorithm in planar-graph based void handling techniques. In fact, it is the first geographic routing algorithm to guarantee message delivery without flooding [3]. Face routing is applied on a planar graph of the network graph. The plane graph divides the plane into faces. The line segment between the source node and the destination node intersects some faces then, the packets will be forwarded along the boundaries of these faces. Although face routing guarantees delivery, it is energy-consuming since it may generate long detours and makes the packets follow a succession of short edges [13]. Given the advantages of greedy forwarding, it has been combined with face routing in order to reduce the global energy consumption. Indeed, when the greedy fails to forward a packet, face routing is used as a recovery mechanism. Afterwards, several routing algorithms using the combination greedy face routing were proposed [3][14] such as Greedy-Face-Greedy (GFG) [15], Greedy Perimeter Stateless (GPSR) [16], Greedy Other Adaptive Face Routing (GOAFR+) [17]. Planar graph based void handling techniques depends on building and maintaining the planar graph which causes significant overhead especially in the case of large scale networks. Besides, techniques from this class usually create long detours to overcome the void inducing extra costs in terms of delay, overhead and energy consumption to deliver the data.

In geometric void handling techniques, the basic idea is the use of the geometric characteristics to identify void regions. This category includes void prevention solutions, such as the techniques proposed in [18, 19, 20, 21, 22, 23, 24], where holes are discovered in advance for the future use of routing to avoid holes. These solutions usually start by a network discovering phase to detect the boundary nodes and identify the voids and a forwarding mechanism that exclude nodes located on the voids borders from routing process. This is accomplished by spreading information about voids to record the state of the network, and sending feedback, which clearly increases the overhead and causes energy exhaustion especially for the nodes located on holes boundaries. Besides, nodes need much greater resources such as memory storage compared to other void handling techniques in order to record the discovered voids information, especially when holes have a very large boundary.

In cost-based void handling strategies a cost value will be assigned to all nodes in the network depending of the destination and then forwarding packets from a node with a higher cost towards a node with a lower cost. The definition

of cost parameter depends of the protocol. The performance of cost-based void-handling techniques depends on the number of destinations, the number of void nodes, and network dynamics [9]. Techniques from this category works well in static wireless networks with moderate number of void nodes with a limited number of destination nodes under a relatively small network. Otherwise, the recovery phase will generate hight overhead due to cost adjustment process and the maintenance around void regions.

Heuristic void handling techniques consists of exploiting some additional resources or using some inherent properties of network topology and some geographic properties of void areas [9] such as using alternative existing communication media in the network, or gradually increasing stuck node's transmission power to search for a positive progress neighbor, or passive participation by not participating in the routing process while the node is enable to find positive progress neighbors, *etc.* However, these techniques are not always effective, or even possible.

Hybrid void-handling techniques, such as the one proposed in [25], combine at least two void-handling techniques together to handle voids more effectively and more efficiently. Usually, this combination increases the complexity of the void handling technique compared to others from different category.

All the above cited recovery techniques work well in static networks. Any change in void localization, such in dynamic networks, will require maintenance process with extra processing, overhead and delay. To overcome these drawbacks, we propose an efficient recovery technique for greedy forwarding based on ant colony optimization. The proposed routing algorithm is a geographic routing that is able to adaptively adjust the forwarding mechanism to avoid the holes, create one or multiple paths around them and effectively deliver data packets.

2.3 Ant colony Optimization

Ant colony optimization (ACO) is a bio-inspired approach from ants foraging behavior. Real ants are able to find the shortest path between their nest and a food source without any visible, central or active coordination mechanisms. Ants drop pheromones, a natural chemical substance, on the path. The path optimization is achieved by exploiting the pheromone quantity deposited. Then, ants select a path based on the pheromone concentration deposited on the set of paths found. The higher the concentration of pheromone on a path, the greater the probability to select it. This indirect communication mechanism is called *stigmergy*. In addition to that, real ants show an impressive behavior when encountering obstacles on their way. Actually, they are able not only to avoid obstacles, but also to find a shortest path around them.

ACO based approaches are very effectively applied to NP-hard problems and result in good optimization. Networking field is one of the many domains that have investigated ACO-approaches to design multi-objective and multi-constraint routing protocols and solve issues like mobility, path optimization, resource utilization and energy awareness. ACO was mainly proposed by

Dorigo [26] [27], and it was widely used to solve network data routing problems. Many routing protocols based on ACO meta-heuristic were proposed in the literature [28][29] [30] [31]. ARA (Ant Colony Routing Algorithm) [32] the first ACO-based routing algorithm aims to reduce routing overhead and most of the existing ACO based routing techniques in WSN and mobile ad-hoc networks are derived from this algorithm [33]. ACO based routing uses two types of agents, forward ant packets and backward ant packets. The first type of ants is used to discover paths toward destination and the second one aims to drop pheromone trails on the established path between the source node and the destination node.

[34], [35] and [33] provide comparative studies of ACO based routing algorithm. Although ACO-based routing algorithms solve many problems such as multiconstraints and multiobjective routing, in addition to path optimization, these algorithms usually produce high overhead, since, in order to converge to an optimized solution, they need a colony-like behavior, *i.e.* a huge number of ant-like packets.

In this paper, we introduce GRACO, a geographic routing algorithm that combines a modified greedy forwarding and a recovery technique based on ACO. It is the first algorithm that uses ACO to assist greedy forwarding. It allows to circonvince voids and provides multi-paths around them.

3 Notations and system models

We assume that all nodes are aware of their location through an hardware device such as GPS or any other location mean.

We model the wireless sensor network as a directed graph G=(V,E), composed of a finite set V of sensors, called also nodes, and a finite set E of links. A wireless link exists between nodes U and v (the link $Uv \in E$) if U and v are within transmission range of each other, *i.e.* $|Uv| <= R$, where $|Uv|$ represents the Euclidean distance between U and v. The directed link from a node U to a node v is denoted $\overrightarrow{Uv}$. The physical set of nodes which are in transmission range of node U is denoted $N(U)$ and called the neighborhood of node U, $N(U) = \{v \in V$ such as $|Uv| \leq R\}$. We denote $|N(U)|$ the cardinality of $N(U)$, $|N(U)|$ is the number of neighbors of U. We also define $N_D(U)$ the subset of $N(U)$ in which each node is nearer from node D than U itself, *i.e.* $N_D(U) = \{v \in N(U)$ such as $|vD| \leq |UD|\}$.

In the following, we call "current node" the node trying to route a packet and we use $NextNode$ to refer to the next hop in the path of a packet.

We denote $C(A, r)$ the circle C of radius r and centered at A. For two nodes A and B, the circle $C(A, |AB|)$ is the circle of radius $|AB|$ and centered at A.

Algorithm 1 GRACO(U) - Run at each node U upon reception of a message $M(U, D)$

```
if U=D then
    EXIT # U is the final destination of M. The routing has succeeded.
else
    Ph_assisted_greedy(U,D)
    if Ph_assisted_greedy fails then
        ACO_Recovery(U,D)
    end if
end if
```

4 GRACO

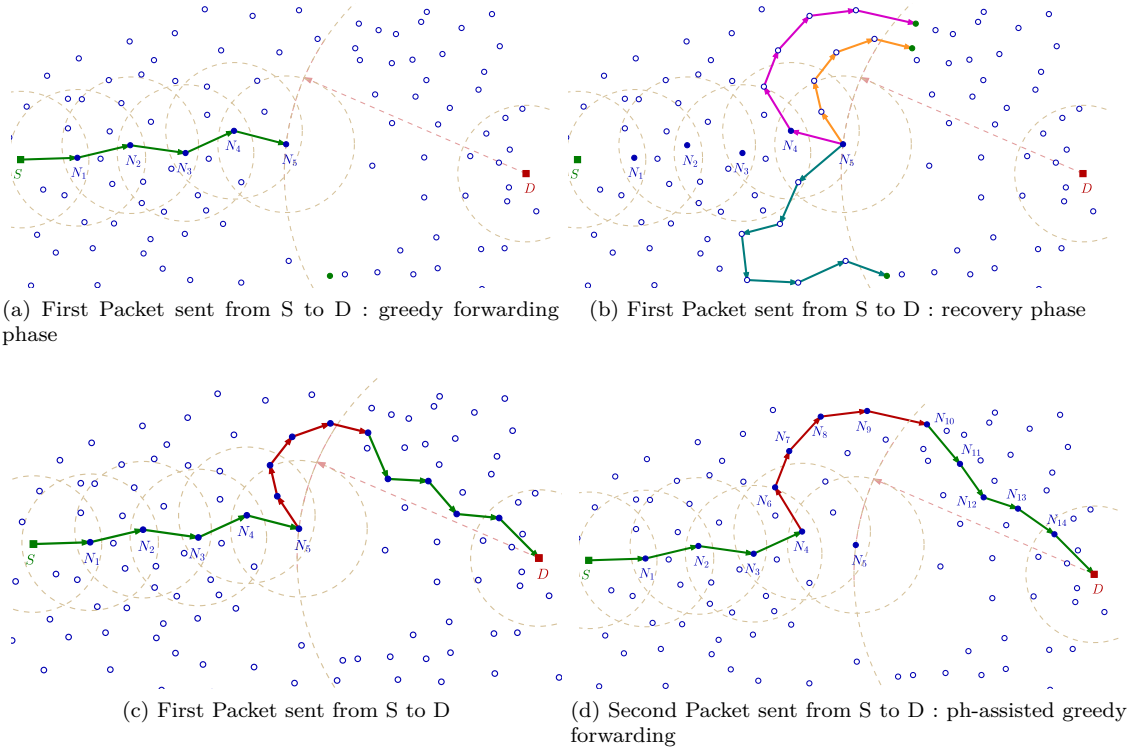
4.1 GRACO overview

GReedy with ACO-based recovery routing protocol (GRACO) is a geographic routing algorithm that combines two modes : a greedy mode and an ACO-based recovery mode triggered when greedy mode fails. Algorithm 1 depicts the GRACO behavior. In GRACO, a data packet is first routed using a modified greedy routing that accounts for the pheromone trails. If a pheromone trail exists for a given destination, it is used to select the next node. Otherwise, the next hop is selected using plain greedy forwarding strategy. However, if the packet reaches a stuck node (a node that has no neighbor closer to the destination that itself neither pheromone trails), the node launches an ACO-based recovery and sends some exploratory ants (Fant) to find a path. The stuck node waits for a backward ant (Bant) to come back, if so, a path is established and data packets can be sent. The data packet will be routed using the same strategy as the Fant until arriving to the unstuck node, and then switch to greedy forwarding again until its destination or another stuck node, in which case, the same mechanism applies again. Both steps are now detailed in the following sections.

4.2 Ph-assisted greedy forwarding

The greedy mode of GRACO consists of a variant of the plain greedy forwarding (GF). Plain GF is enhanced with the use of pheromone trails from older recoveries detailed in section 5. Consider a node S that wants to send a data packet to D. Before applying GF, S checks whether it has recorded pheromone trails to D in order to avoid returning to the same stuck node. If so, that means the greedy failed to progress a previous data packet during a previous attempt for the same destination D and a recovery mode was been launched. For that reason, S will use these pheromone trails to send the packet instead of the GF. Otherwise, If there is no pheromone trail for D, S proceeds by using the greedy method.

The example presented in Fig. 2 explains the ph-assisted greedy forwarding.



(a) First Packet sent from S to D : greedy forwarding phase

(b) First Packet sent from S to D : recovery phase

(c) First Packet sent from S to D

(d) Second Packet sent from S to D : ph-assisted greedy forwarding

Figure 2: Ph-assisted greedy forwarding

In Fig. 2a, S sends a first data packet to D, the packet is forwarded using plain greedy until it arrives to the stuck node N_5, thus, as shown in Fig. 2b, a recovery phrase is triggred in N_5, several paths are found, and pheromone trails are dropped on the nodes belonging to these paths. The first data packet is, then, relaunched from N_5 using one of the paths found, as presented in Fig. 2c. In Fig. 2d, S sends a second data packet to D, the packet is routed using plain greedy forwarding until arriving to N_4 where it finds pheromones to D, then it uses a pheromone based forwarding which helps the packet to avoid the stuck node N_5. Thus, GRACO's greedy mode is a Ph-assisted greedy forwarding. The Ph-assisted greedy forwarding phase is summarized in Algorithm 2.

4.3 ACO based Recovery

Similarly to other ACO based algorithms, the ACO recovery uses two types of ants to solve a problem : Fants (Forward-ants) to discover the environment, and Bants (Backward-ants) to "mark" the solutions found. In addition, GRACO relies on a concept of zones, which plays an important role, that we introduce in the following.

Algorithm 2 Ph_assisted_greedy(U,D) - Run at each node U upon reception of a message $M(U, D)$

if $existPh(U, D) =$ **true then**
  $NextNode \leftarrow getNextNodePH(U, D)$ # *There already exists a pheromone trail to D*
else
  if $N_D(U) \neq \emptyset$ **then**
    $NextNode \leftarrow$ N st $||UD| - |ND|| = max_{v \in N_D(U)}||UD| - |vD||$ # *traditional greedy forwarding*
  else
    Ph_assisted_greedy fails
  end if
end if
Send(M($NextNode$,D)) # *to send the message to NextNode*

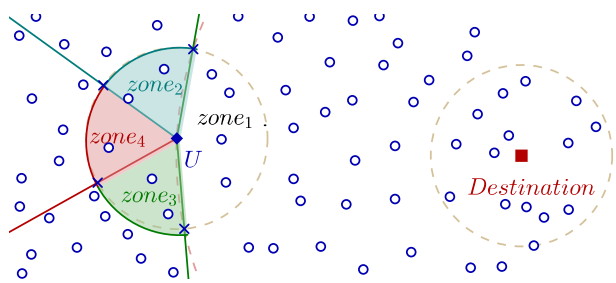


Figure 3: Different zones of U for the destination node D

4.3.1 Zones

Using the concept of zones [36], a node divides its neighborhood into 4 zones based on its position and the position of D. Consider a destination node D, each node U partitions its neighbors into two main zones: the positive progress zone, later called $zone_1$, and a negative progress zone. As shown in Fig. 3, $zone_1$ is represented by the intersection of the two circles $C_1(U, R)$ and $C_2(D, |DU|)$. It gathers nodes in $N_D(U)$. Then, the negative progress zone is then partitioned into 3 sub-zones: $zone_2$, $zone_3$ and $zone_4$. Let α be the positive progress angle and β the negative progress angle, as presented in Fig. 4, α is the angle $(\overrightarrow{UB}, \overrightarrow{UA})$ where A and B are the points of intersection of two circles $C_1(U, R)$ and $C_2(D, |DU|)$, and $\beta = 2\Pi - \alpha$. For a node $v \in N(U)$, θ is the angle $(\overrightarrow{UD}, \overrightarrow{Uv})$. Node v belongs to :
- $zone_2$ if $\frac{\alpha}{2} < \theta \leq \frac{\alpha}{2} + \frac{\beta}{3}$
- $zone_4$ if $\frac{\alpha}{2} + \frac{\beta}{3} < \theta \leq \frac{\alpha}{2} + 2.\frac{\beta}{3}$
- $zone_3$ if $\frac{\alpha}{2} + 2.\frac{\beta}{3} < \theta \leq \frac{\alpha}{2} + \beta$

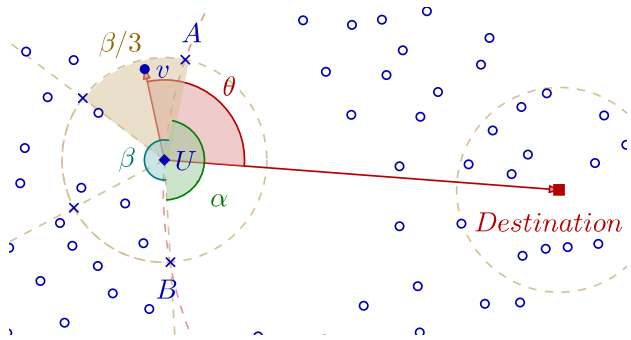


Figure 4: Illustration of α, β and θ

4.3.2 Pheromone initialization

Before starting a route establishment phase to a destination D, the amount of pheromone deposited on the links must be initialized. Each node U that has no pheromone to D and needs to forward a Fant toward D, assigns a pheromone trail to each of its outgoing links for D, *i.e.*, an initial pheromone value ϕ_{0j} is assigned for each neighbor v in $N(U)$ as $j \in \{1, 2, 3, 4\}$ and $zone_j$ being the zone in which node v lies, knowing that $\phi_{01} \geq \phi_{02} (= \phi_{03}) \geq \phi_{04}$. The motivation is that, in most cases, shortest paths pass through the neighbors whose directions are closer to the direction of the destination. Thus, the initial pheromone values being bigger as the zone is closer to the destination, direction will favor ants to choose these zones. As a result, this phase leads to a fast convergence to a shortest path most of the time. The ACO-based recovery strategy assumes that each node maintains a PhTable, a table of pheromone values assigned to its outgoing links for different destinations. Whenever a node receives a packet for a specific destination D, it searches in its PhTable for pheromone for D. If such pheromone trail exists, it will be used to choose the next hop. Otherwise, a pheromone initialization process is launched.
The pheromone initialization process attributes pheromone trails to all the outgoing links of a node. However, the not updated pheromone trails will be evaporated as the time passes. If a link is unused, the pheromone level on it should be completely evaporated by the end, thus, the corresponding PHTable entry is deleted. In this way, the pheromone evaporation process minimises the amount of data stored in the nodes.

4.3.3 Route establishment

The route establishment phase is accomplished using two types of ants: the Fant and the Bant. The main role of the Fant is to explore the neighborhood in order to find an alternative path. To do so, it drops on its way, a pheromone track to the stuck node to be later used by the Bant. The Bant establishes the route to the destination found by the Fant by dropping pheromone trails when it goes back to the stuck node. Fant and Bant are small packets, they only carry

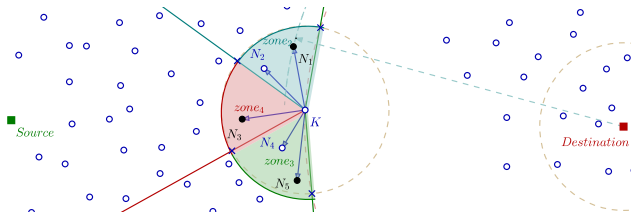


Figure 5: Fant is considered by the closest node in each zone.

Algorithm 3 ACO_Recovery(K,D) - Run at the stuck node K to trigger the recovery phase by diffusing the first Fant

1: $FANT(K, D) \leftarrow CreateFant()$ # Stuck node K creates a Fant
2: $NextNode\_Zone_2 \leftarrow max_{v \in Zone_2(K)} ||KD| - |vD||$ # *the closest node to D in $Zone_2$ of the stuck node K*
3: $NextNode\_Zone_3 \leftarrow max_{v \in Zone_3(K)} ||KD| - |vD||$ # *the closest node to D in $Zone_3$ of the stuck node K*
4: $NextNode\_Zone_4 \leftarrow max_{v \in Zone_4(K)} ||KD| - |vD||$ # *the closest node to D in $Zone_4$ of the stuck node K*
5: Broadcast(FANT($NextNode\_Zone_2, NextNode\_Zone_3, NextNode\_Zone_4, K, D$))

their ID, information about stuck node, destination, previous node and the hop count.

The route establishment starts at the stuck node K. In each zone, K selects the neighbor with the best progress to the destination. The Fant will be considered and forwarded only by the 3 selected neighbors. The IDs of the selected neighbors are stored in the diffused Fant. Whenever a neighbor receives the Fant, it checks whether it is in the list of the selected neighbors, if not it ignores the packet . In the example presented in Fig. 5, the stuck node selects 3 neighbors, N_1,N_3 and N_5, each one is the the closest to destination in its zone.

The recovery phase triggered by the stuck node in summarized in Algorithm 3.

Whenever a node U receives a Fant, it checks whether it is closer to the destination than the Stuck node K or not. If so ($|UD| < |KD|$), U sends back a Bant, the recovery can end and the greedy step resumes. Otherwise, U forwards the Fant. Similar to other ACO routing algorithms [33], a node forwards a Fant to the next node using a stochastic decision based on the values of pheromone trails to select the next hop. Suppose that a Fant is currently residing in node U. U has k neighbors v_1, v_2, ..., v_k. We will note Φ_i the amount of pheromone assigned to v_i (or the link $\overrightarrow{Uv_i}$). The neighbors of U will be partitioned into 4 zones. Consider Φ_{zone_i} is the maximum pheromone amount assigned to the neighbors of $zone_i$, $\Phi_{zone_i} = \max\{\Phi_\mathrm{j} \setminus v_j \in zone_i\}$. In order to forward a Fant,

Algorithm 4 Forward_Fant($FANT(K, D)$) - Run at each node U upon reception of a Fant for D issued by stuck node K

```
update_BRTable(U)
if |UD| < |KD| then
    # Node U allows a progress compared to the stuck node and can stop the recovery, U sends a Bant to K
    BANT(K, D) ← Create_Bant(FANT(K, D))  # Node U creates a Bant to send it back to K in order to drop pheromone to D
    Forward_Bant(BANT(K, D))
else
    if existPh(U, D) = false then
        initializePh(U, D)
    end if
    NextNode ← getNextNodePH(U, D)
    Send(FANT(K, D), NextNode) # to send the Fant to NextNode
end if
```

first of all, U selects a zone with probability P_{zone_i} :

$$P_{zone_i} = \frac{\Phi_{zone_i}}{\sum_{j=1}^{4} \Phi_{zone_j}} \tag{1}$$

U chooses a node v_i in this selected zone with probability P_i:

$$P_i = \frac{\Phi_i}{\sum_{j=1}^{|N(D)|} \Phi_j} \tag{2}$$

Using the concept of zones, the ants are not completely blind, as in the usual ACO based algorithms, they will be attracted, but not forced, to the right direction. Besides the PHTable mentioned before, each node maintains also a back routing table, the BRTable, to store information that will be used later by Bants to go back to the stuck node. Whenever a Fant arrives to a node from one of its neighbors, an entry is added to the BRTable. Fig. 6 shows an example of a Fant F trying to bypass a void. F is launched at the stuck node, and forwarded to N_1, N_2, N_3 until reaching unstuck node N_4. To summarize, Algorithm 4 shows the forwarding strategy of the Fant until it arrives to an unstuck node, a node closer than the stuck node to the destination.

When a Fant reaches an unstuck node N, a Bant is sent back to the stuck node. An unstuck node is a node closer to destination than the stuck node. After adding an entry to the BRTable, N extracts the information from the Fant, creates a Bant and destroys the Fant. Subsequently, N sends the Bant to the stuck node K. The role of the Bant is to drop pheromone on the path

Algorithm 5 Forward_Bant($BANT(K, D)$) - Run at each node U upon reception of a Bant destined for K

```
update_PHtable(U, D) # the Bant updates pheromone trails in the node U for the destination D
if U = K then
    Ph_assisted_greedy(U,D) # Send the data packet
else
    NextNode ← getNextNodeBRtable(U, D)
    Send(BANT(K, D), NextNode) # to send the Bant to NextNode
end if
```

found in order to establish a track from K to N. Unlike the traditional ACO-based algorithm, the Bant will not necessarily return to the stuck node using the same path of the correspondent Fant, but it will use the pheromone dropped by other Fants that used at least one node of its path. In the example presented in Fig. 7, the Fants find two valid paths to an unstuck node, the first one presented in green is a 4-hop-length path and the second one in pink is a 16-hop-length path. A Bant, in this example, chooses to use the green path to go back to K since it is the shortest path to K. On its way back to the stuck node, the Bant updates pheromone trails in the PHTables. Consider a Bant arrives to node A from node B, it will update pheromone track of the link $\overrightarrow{AB}$ in the PHTable of A :

$$\Phi_{\overrightarrow{AB}} = \Phi_B = \Phi_B + \Delta\Phi$$

where $\Delta\Phi$ is the amount of pheromone added to reinforce the path to D, this value depends on the quality of the path. Algorithm. 5 sums up the Bant forwarding.

In order to improve the performance of GRACO, the stuck node is able to relaunch the recovery phase n times, where $0 \leqslant n \leqslant n_{max}$, if it does not receive any Bant in a certain interval of time.

ACO algorithms are based on the balance between two processes : intensification (deposition of pheromone) and diversification (evaporation). The evaporation rule is used to reduce the effect of old pheromone. Thus, for each node U, the pheromone trail of its neighbor v_i is evaporated periodically using formula (1).

$$\Phi_i = \Phi_i * (1 - \alpha) \tag{3}$$

where α is the pheromone evaporation rate, $0 < \alpha < 1$.

5 Multiple recoveries for the same destination

In its way to the destination, a packet may face different voids in a single path, for that reason, in such cases, the recovery mode will be launched several times. In order to reduce the effect of old recoveries, the pheromone added to a PHTable will be marked by the rank of the recovery launched. A Fant will use only the

pheromone of its recovery to decide where to go next. A Bant returning to mark a good path, will remove old pheromone trails found in its way. Thus, a node that received ants from different recoveries, will have, at the end, only recent recovery pheromones in its PHTable .

Fig. 8 shows an example of multiple recoveries launched in the way from a source S to a destination D. Fig. 8a presents the first recovery mode launched, the ants find during this phase two paths to unstuck nodes. The data packet will then choose one of these paths. If the data packet uses the pink path, it reaches another stuck node, a second recovery mode is then launched. The new Fant will use only its pheromone, otherwise, it cannot go out of the stuck node since the pheromone dropped on the pink path are stronger than the initial trails of the second recovery. The second recovery ants will find the blue as shown in Fig. 8b. Since the second recovery finds a good path to an unstuck node, the first recovery pheromones are no longer useful. The new Bant deletes older pheromone traces and adds the new one. The next time the source node S decides to send a packet to D, this latter will be forwarded using greedy forwarding until node N and then pheromone forwarding, as presented in Fig. 8c.

6 Loop management

In the route establishment phase, a Fant may run into a loop. We prevent this situation using a simple process to detect and manage a loop. The first time a node detects a Fant in a loop, it assigns a loop flag to the packet and returns it back to the previous node. Each node that receives a Fant with a loop flag, tries to get out from this loop by sending it to another node then the one it came from. If it is not possible, the node sends the Fant back to the previous node in its path, using BRtable. And so on until the Fant goes out from the loop, the loop flag is removed. In the case where the Fant returns back to the node that first detected a loop, the Fant is cancelled. Consider the example of a Fant F1 that enters in the loop ABC as shown in the Fig. 9a. F1 loop management process is presented in Fig. 9, and described in the following steps :

1. Node A receives F1, it recognizes duplicate receptions of F1 based on a memorized entry previously added by the same ant. Since A is the first node to detect F1 in loop, it assigns a loop flag to F1. A does not add any information about the duplicated ant in the BRTable. Then, it sends the packet back to the previous node C.

2. C receives F1 and recognizes that F1 is in loop based on the loop flag, C deactivates the link to the previous node (A) and tries to send F1 out of the loop. Since A is no more considered (the link is deactivated) and B exists in the BRTable of C (that means F1 already passed from B), C couldn't get F1 out of the loop, it decides then to return the ant to the previous node (B). C sends F1 to B and deletes B from its BRTable.

3. B receives F1 and recognizes that F1 is in loop based on the loop flag, B deactivates the link to the previous node (C) and it tries to send F1 out

Algorithm 6 Loop_Management(V,D) - Run at each node U upon reception of a Fant for D coming from node V

```
if Loop_Flag = 1 then
    delete_ph_phtable(V,D) # delete_ph_phtable(V,D) deletes the ph trail to neighbor V for destination D in PHTable of the current node U
    if |N(U)| > 1 then
        NextNode ← getNextNodePH(U, D)
        Loop_Flag ← 0
    else
        NextNode ← getNextNodeBRtable(U, D)
    end if
else
    Loop_Flag ← 1
    NextNode ← V
end if
```

of the loop. Since C is no more considered (the link is deactivated) and A exists in the BRTable of B (that means F1 already passed from B), B couldn't get F1 out of the loop, it decides then to return the ant to the previous node (A). B sends F1 to A and deletes the entry of F1 from the BRTable.

4. A receives F1 for the second time, it deactivates the link to the previous node (B) and kills the ant F1.

By deactivating the links that lead to loop, the next ants will not come to the same situation again. Similarly, deleting entries from BRTable will prevent Bants to loop in their way back.

The loop management process is described in Algorithm 6 and Fig. 10 .

7 Simulations and results

In order to evaluate the performance of the GRACO algorithm, we simulated its functioning under the WSNet simulator [37]. WSNet is an efficient event-driven simulator dedicated to WSN, which has been extensively evaluated and compared [38].

For a fair performances evaluation, we choose to compare our results against GFG routing algorithm [12]. GFG is an efficient geographical routing protocol for wireless sensor networks that combines the greedy forwarding with face routing as a recovery mode, it has been proven to guarantee data delivery for arbitrary connected planar graphs [14] with low complexity added in terms of routing overhead and additional latency of the recovery mode compared to other geographic routing protocols [39] [9]. To the best of our knowledge, all more recent geographic protocols that are able to efficiently encounter voids and guarantee data delivery with better performances, are only variants of GFG. These

Parameter	Value
Duration (s)	300
Routing	GFG and GRACO
MAC	idealmac and IEEE 802.15.4 CSMA-CA
Radio	$radio\_half1dk$
Interferences	none
Modulation	bpsk
Field size	300 m×300 m
Communication range	25m
Number of nodes	250, 350, 450, 550, 650, 750, 850, 900
Density	5, 7, 9, 11, 13, 15, 17, 19

Table 1: Simulation parameters

protocols were proposed to improve the performance of GFG by modifying some initial assumptions, such as using adjustable transmission range [40], which is not relevant are our case. To the best of our knowledge, to date, GFG remains the best geographic routing protocol that guarantees the delivery and considers the same initial assumptions that we do.

We evaluated two versions of GRACO, the first one tries to minimize routing overhead while maintaining good performance level, indeed, for a single recovery step, if no Bant is back to the stuck node within an interval of time $T = 1$ms, it is allowed to relaunch another set of 3 ants only once, n_{max} is then set to 1. The second variant of GRACO focuses on guaranteeing the delivery, the stuck node will then send a set of 3 ants periodically as long as no Bant is back. We note $GRACO\_V1$ where $n_{max} = 1$ and $GRACO\_V2$ where $n_{max} \to \infty$.

We generate random topologies in a $300m \times 300m$ region with 250 to 900 nodes and $R = 25$m. Each combination of topology and algorithm is run 50 times, which ensures a 95% confidence interval. To evaluate the impact of voids on the performance of each routing protocol, each topology contains necessarily void zones with different diameters and shapes. In each topology we choose a set of 10 pairs of (source, destination) in such a way that there is always at least one void to be handled during the routing process.

The performance of GRACO is mainly measured based on data packet transmission delay, delivery rate and the routing path length (in terms of hop count) in order to measure the robustness and effectiveness of the routing protocol against voids. Besides, we measure the global data delivery cost including recovery cost. The simulation results of GRACO and GFG are given with 95% confidence intervals.

We compared the performances of GRACO and GFG in two steps. We started by simulating the performance of GRACO_V1, GRACO_V2 and GFG with an ideal MAC layer, thus no collision is simulated. The choice of using an

IdealMac was made to isolate the impact of routing algorithms only. In a second step, we compared the functioning of GRACO_V2 and GFG with more realistic MAC layer, the IEEE 802.15.4 mac. We choose to simulate only GRACO_V2 since it performs a better delivery rate than GRACO_V1 with reasonable cost.

7.1 Data delivery cost

We measure data delivery cost as the total number of packets sent in the network to deliver data from a source to a destination, including routing and recovery control packets.

Fig. 11a presents the data delivery cost varying with the number of data packets sent for an Idealmac layer. The cost of GRACO_V2 is clearly higher than GRACO_V1 since the number of ants used in GRACO_V2 is bigger than GRACO_V1. At the beginning, GFG seems to use less packets to deliver the data than GRACO, but when the number of data packets sent increases, the cost on GFG increases fast. GRACO_V1 uses less cost starting the second data packet sent. However, GRACO_V2 becomes cheaper than GFG in the 10th data packet sent between same source and same destination.

Simulations run on IEEE 802.15.4 show similar results. As shown in Fig. 12a, the total cost of GFG is less than the one of GRACO_V2 in the beginning, but starting from the 4th data packet sent, the cost of GFG becomes much higher than GRACO_V2's one.

The simulations proved that the more data packets are sent between the same source and destination, the cheaper become GRACO comparing to GFG. These results are explained by the fact that the number of packets sent during the recovery phase in GRACO is bigger than GFG in one recovery mode. However, when the number of data packets sent between the same source and the same destination increases, GRACO becomes cheaper than GFG since GRACO launches the recovery mode only when the first data packet is sent, the next ones will use the routes already found. As for GFG, it launches the recovery mode every time a data packet is sent. Thus, the cost on GRACO for multiple packets between the same source and destination is the same as for only one packet, in contrary with GFG, the cost used to deliver multiple data packet will be the cost of sending one packet multiplied by the number of data packets sent. In Fig. 11b and Fig. 12b, it can be seen that the cost per packet decreases when the number of data packets increases. On the other hand, the cost of GFG to deliver a single packet remains constant.

7.2 Hop count

Fig. 13 displays the average length of the paths generated by GRACO and GFG for different network densities. The path length is defined as the number of hops followed by the data packet to travel from its source to its destination.

The simulations show that GRACO creates shorter paths than the ones established by GFG. GFG may generate extremely long paths in some cases [41] especially in low density networks as shown in Fig. 13. In the other hand, the

ACO and the zone concept used in GRACO helps the algorithm create shorter paths than the one created by GFG.

In order to bring out the multipath feature of the recovery strategy used in GRACO, we count the number of paths found by each routing protocol. As presented in Fig. 14, GFG uses the same path in all iterations, however, GRACO is able to find multiple paths if possible. The few cases, that GRACO finds only one path in all iterations is where it is the only path possible.

7.3 End-to-end delay

The end-to-end delay accounts for the duration from the time a packet is ready for the transmission at the original source until it is received and decoded correctly at its final destination. In case of a set of data packets, it represents the delay between the source sends the first data packet and the destination receives the last one. Sources and destinations are randomly chosen in a way they are in different sides of a void to trigger recovery process.

Fig. 15 and Fig. 16 depict the end-to-end delay depending on the number of data packets sent with Ideal MAC layer and 802.15.4 MAC layer.

We can see that GRACO delivers data packets in shorter end-to-end delays than GFG. Actually, data packets will be delivered faster using GRACO, as it generates shorter paths than GFG. Furthermore, when the number of data packets sent increases, the delay gap between GFG and GRACO becomes larger. Indeed, when sending several packets from a source to the same destination, GRACO launches the recovery only once, when the first data packet attempts to bypass the void, the next ones will use the pheromone trails dropped on the paths found during the previous recovery phase and will directly follow a short path to the destination. At the contrary, GFG launches its recovery mode every time a data packet has to bypass the void. Thus, GRACO delivers the data packets faster than GFG.

7.4 Packet delivery rate

Packet delivery rate (PDR) is the ratio of data packets successfully received by their destinations to all data packets sent by the sources.

$$PDR = \frac{number\ of\ data\ packets\ received}{number\ of\ data\ packets\ sent}$$

Fig. 17 plots the data delivery rate of GRACO_V1, GRACO_V2 and GFG simulated with an ideal MAC layer. It shows that both of GFG and GRACO_V2 deliver all packets sent successfully. In the other hand, GRACO_V1 performs a delivery rate in the worst case around 92% . Thus, GRACO_V2 guarantees delivery with an extra cost compared to GRACO_V1, however, it remains cheaper than GFG in terms of end-to-end delay and the routing cost specially when the number the data packets sent increases .

GRACO_V1 suffers from packet loss in some rare cases where Fants get lost in the network specially in multiple voids cases, due to the random factor in the stochastic decision to choose next hop. Even when the first data packet

reaches its destination, the following data packets will suffer from the impact of these pheromone trails, which explains the fact that the delivery rate of the first data packet is always higher than multiple data packets. However, the effect of these pheromone trails will be reduced with the time as a result of pheromone evaporation. Indeed, the pheromone evaporation process is an important factor in ACO meta-heuristic, called the diversification factor, it helps to eliminate pheromone trails that may misguide ants and then to adapt to the dynamic nature of the environment. At the end the bad pheromone trails will be totally evaporated and new paths will be found. This depends of the evaporation factor and the intensity of the pheromone deposited. For that reason, we can see, in Fig. 17 that the delivery rate starts to increase again when the number of data packets grows.

Fig. 18 shows the data delivery rate varying with the number of data packets sent with an IEEE 802.15.4 MAC layer. Due to packet collisions measured using the IEEE 802.15.4 MAC layer, the data delivery rate of GFG and GRACO degrades. However, the delivery rate of GFG is affected more than GRACO_V2; the GFG delivers less than 80% of the data, while GRACO_V2 delivers up to 90% of the data.

This can be explained by the fact that the number of control packets used by GFG is bigger than GRACO which has a significant impact on the performance of the routing process with IEEE 802.15.4 MAC layer.

8 Discussion on memory overhead

GRACO is not a memory-less routing algorithm. Indeed, as explained in Section 2.3, the ACO-based recovery is accomplished using pheromone trails, these trails will be memorized in PHTables within the nodes on recovery paths. A PHTable of a node memorizes the pheromone trails on its outgoing links. If a node belongs to several recovery paths for different destinations, it keeps pheromone trails for each of these destinations.

All pheromone trails evaporate periodically, the non-updated pheromone will be deleted faster than the updated ones, thus, a node will have only updated pheromone.

In addition to PHTables, each node in a recovery path stores a BRtable to help Bants to go back to the stuck node. Every entry in a BRTAble will be deleted after an interval of time that we fixed.

To ensure the loop management process, each node stores the IDs of the received Fants in order to recognize the duplicated ones. The node will delete these IDs after a time interval equal to the time-to-live of the Fant.

Hence, unlike GFG, GRACO is not memory-less routing protocol. In fact, nodes of recovery paths store local information about pheromone trails to their outgoing links, back routing information and Fant IDs for loop management process. However, these information are volatile, and will be deleted after a while, since they are only useful for a short time.

The time interval to delete stored data is a recurring problem of cache memory, and is left out of the scope in this paper. In the other hand, simulation results show that GRCAO outperforms GFG in terms of data routing cost, end-to-end delay, data delivery rate and the length of generated paths. Besides, the proposed recovery strategy creates multiple path around the void.

9 Conclusion

In this paper, we presented GRACO, a new geographic routing protocol able to handle the void problem with very low added complexity. It combines a pheromone assisted greedy forwarding phase and an ACO-based recovery phase to efficiently avoid holes and recover from local minima. The proposed routing protocol is fully localized, distributed, scalable, and does not require any graph structure. Nodes, that are part of recovery paths, need to store local information about pheromone trails, back routing and loop management. However, all this information is stored for a short time, the evaporation factor reduces the pheromone information, and other stored information will be deleted when not useful anymore.

Besides the efficient hole avoidance, simulation results show that both GRACO_V1 and GRACO_V2 deliver data packets faster and cheaper in terms of data delivery cost and hop count compared to GFG routing protocol. GRACO_V1 provides a high delivery rate with a minmum cost, in the other hand, GRACO_V2 guarantees delivery with an extra-cost in term of end-to-end-delay and data delivery cost that remains lower that GFG's guaranteed delivery cost. Simulation results with 802.15.4 MAC layer show that GRACO outperforms GFG in terms of end-to-end delay, data delivery cost and data delivery rate.

As future work, we intend to propose GRACO as a routing solution for Smart Grid Neighborhood Area Netowrk (NAN). Smart grid is the next generation of electrical power grid. It integrates information and communication technologies (ICT) to allow for reliable and sustainable electric power delivery and to integrate renewable energies. There is no smart grid for all, every energy grid will develop its own applications and services. Therefore, many services and concepts related to smart grid such as Virtual Power Plant (VPP) [42][43] and distributed energy and storage integration into the grid [44] are expected to require partial or fully distributed control and the ICT deployed in parallel to the electrical grid should be scalable and should not introduce additional constraints to the smart grid development. For this purpose, we plan to study and analyze the applications supported by NAN in order to define NAN communication requirements, and to implement an extension of GRACO that takes account of the QoS required.

References

[1] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 2002.

[2] A. Dvir and N. Carlsson. Power-aware recovery for geographic routing. In *Wireless Communication and Networking Conference (WCNC)*, 2009.

[3] N. Mitton, T. Razafindralambo, and D. Simplot-Ryl. Position-Based Routing in Wireless Ad Hoc and Sensor Networks. In *Theoretical Aspects of Distributed Computing in Sensor Networks*. Springer, 2010.

[4] G. G Finn. Routing and addressing problems in large metropolitan-scale internetworks. isi research report. 1987.

[5] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communication*, 1984.

[6] T.C. Hou and V.OK Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communication*, 1986.

[7] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Canadian Conference on Computational Geometry*, 1999.

[8] Z. Jiang, J. Ma, W. Lou, and J. Wu. An information model for geographic greedy forwarding in wireless ad-hoc sensor networks. In *Conference on Computer Communication (Infocom)*, 2008.

[9] C. Dazhi and P.K. Varshney. A survey of void handling techniques for geographic routing in wireless networks. *IEEE Communication Surveys Tutorials*, 2007.

[10] M.Y. Sanavullah Shri Alarmelu.V, R. Poonkuzhali. An efficient void handling technique for geographic routing in [MANET: A survey. *Journal of Advanced Research in Computer Science and Software Engineering*, 2012.

[11] S. Parvin, M.A. Sarram, G. Mirjalily, and F. Adibnia. A survey on void handling techniques for geographic routing in vanet network. *International Journal of Grid & Distributed Computing*, 8(2), 2015.

[12] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless networks*, 2001.

[13] N. Gouvy, N. Mitton, and J. Zheng. Greedy Routing Recovery Using Controlled Mobility in Wireless Sensor Networks. In *International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW)*, 2013.

[14] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *Proceedings of the 12th annual International Conference on Mobile computing and networking*, pages 390–401. ACM, 2006.

[15] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, DIALM '99, pages 48–55, New York, NY, USA, 1999. ACM.

[16] B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[17] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72. ACM, 2003.

[18] I. Tsvietkov. A novel method of locating voids in MANET structure. In *The First International Scientific-Practical Conference Problems of Infocommunications Science and Technology*, pages 57–58. IEEE, 2014.

[19] N. Senouci, M. K. El Ouahed, and H. Haffaf. Detecting boundary nodes in wsn. In *Proceedings of the International Conference on Wireless Networks (ICWN)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

[20] J. Yang, Z. Fei, and J. Shen. Hole detection and shape-free representation and double landmarks based geographic routing in wireless sensor networks. *Digital Communications and Networks*, 1(1):75–83, 2015.

[21] X. Fan and F. Du. An efficient bypassing void routing algorithm for wireless sensor network. *Journal of Sensors*, 2015, 2015.

[22] D. Zhang and E. Dong. A bypassing void routing combining of geographic and virtual coordinate information for wsn. In *The 22nd International Conference on Telecommunications (ICT)*, pages 118–122. IEEE, 2015.

[23] Fucai Yu, Shengli Pan, and Guangmin Hu. Hole plastic scheme for geographic routing in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, pages 6444–6449, 2015.

[24] Z. Fei, J. Yang, and H. Lu. Improving routing efficiency through intermediate target based geographic routing. *arXiv preprint arXiv:1502.04316*, 2015.

[25] G. W. Denardin, C. H. Barriquello, A. Campos, and R. N. do Prado. A geographic routing hybrid approach for void resolution in wireless sensor networks. *Journal of Systems and Software*, 84(10):1577–1590, 2011.

[26] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In *New Ideas in optimisation*. (D. Corne, M. Dorigo, F. Glover, eds.) McGraw-Hill, 1999.

[27] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of Metaheuristics*. Springer US, 2003.

[28] A. M. Zungeru, L. M. Ang, and K. P. Seng. Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison. *Journal of Network and Computer Applications*, 2012.

[29] V. Uchhula and B. Bhatt. Article:comparison of different ant colony based routing algorithms. *IJCA Special Issue on MANETs*, 2010.

[30] D. Sutariya and P. Kamboj. A survey of ant colony based routing algorithms for MANET. *European Scientific Journal*, 2014.

[31] R. Huang and X. Guanghui. Swarm intelligence-inspired adaptive routing construction in wsn. In *Wireless Communication Networking and Mobile Computing (WiCOM)*, 2010.

[32] M. Gunes, U. Sorges, and I. Bouazizi. Ara-the ant-colony based routing algorithm for MANETs. In *International Conference Parallel Processing Workshops*, 2002.

[33] S. D. Shirkande and R.A. Vatti. ACO based routing algorithms for ad-hoc network (WSN, MANETs): A survey. In *Communication Systems and Network Technologies (CSNT)*, 2013.

[34] Y. Kambayashi. A review of routing protocols based on ant-like mobile agents. *Algorithms*, 2013.

[35] P. Sutariya, D.and Kamboj. A survey of ant colony based routing algorithms for MANET. *European Scientific Journal*, 9(10), 2014.

[36] S. Kamali and J. Opatrny. Posant: A position based ant colony routing algorithm for mobile ad-hoc networks. In *International Conference Wireless and Mobile Communication (ICWMC)*, 2007.

[37] A. Fraboulet, G. Chelius, and E. Fleury. Worldsens: Development and prototyping tools for application specific wireless sensors networks. In *Symp. on Information Processing in Sensor Networks (IPSN)*, April 2007.

[38] E. Ben Hamida, G. Chelius, and J. M. Gorce. On the complexity of an accurate and precise performance evaluation of wireless networks using simulations. In *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 395–402. ACM, 2008.

[39] C. H. Lin, S. A. Yuan, S. W. Chiu, and M. J. Tsai. Progressface: An algorithm to improve routing efficiency of gpsr-like routing protocols in wireless ad hoc networks. *IEEE Transactions on Computers*, 59(6):822–834, 2010.

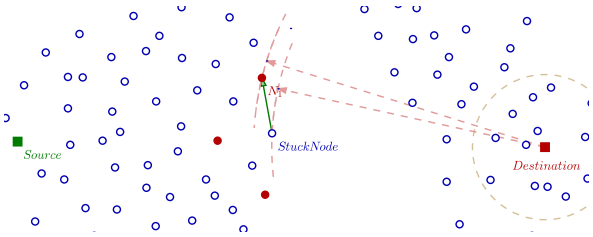
[40] E. Hamouda, N. Mitton, B. Pavkovic, and D. Simplot-Ryl. Energy-aware georouting with guaranteed delivery in wireless sensor networks with obstacles. *International Journal of Wireless Information Networks*, 16(3):142–153, 2009.

[41] K. Seada. Modeling and analyzing the correctness of geographic face routing under realistic conditions". accepted for publication. In *Elsevier Ad Hoc Networks Journal special issue on Recent Advances in Wireless Sensor Networks*, 2007.

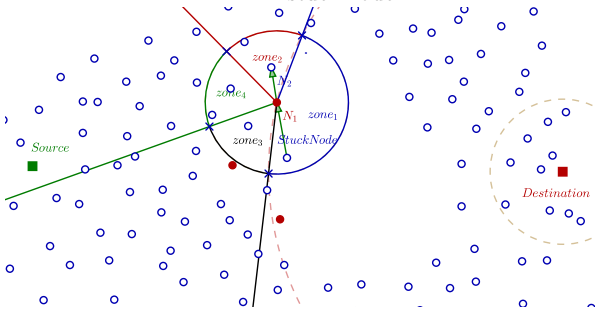
[42] M. Rekik, Z. Chtourou, and N. Mitton. Performance and applicability of a new geographic routing protocol for virtual power plants. In *The 6th International Renewable Energy Congress (IREC)*. IEEE, 2015.

[43] M. Rekik, Z. Chtourou, and N. Mitton. A scalable geographic routing protocol for virtual power plant communications. In *The 15 IEEE International Conference on Environment and Electrical Engineering*, 2015.

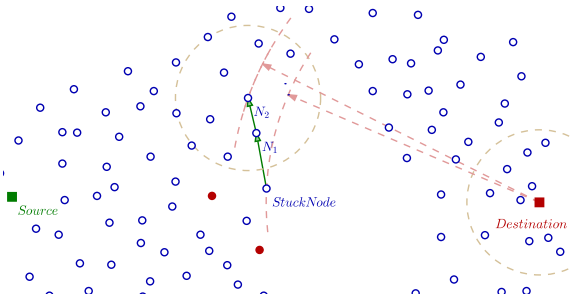
[44] M. Rekik, Z. Chtourou, and N. Mitton. Geographic routing protocol for peer-to-peer smart grid neighborhood area network. In *The 15 IEEE International Conference on Environment and Electrical Engineering*, 2015.



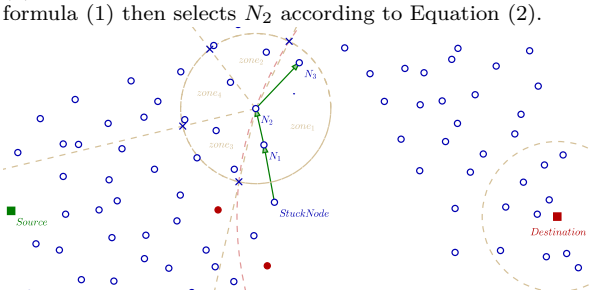
(a) Step1: Fant F_1 reaches N_1, N_1 is not closer than the stuck node



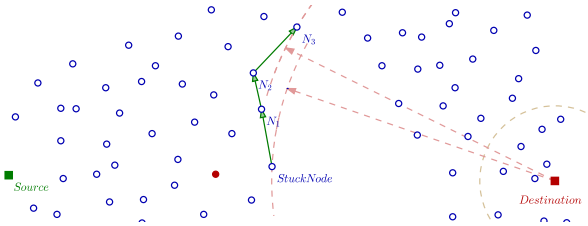
(b) Step2: N_1 forwards F_1 to N_2, N_1 chooses $zone_2$ using formula (1) then selects N_2 according to Equation (2).



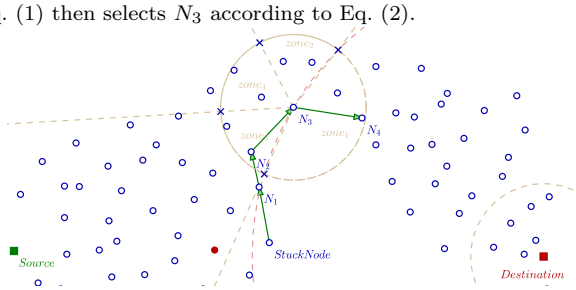
(c) Step3: N_2 receives F_1, N_2 is not closer than the stuck node to the destination.



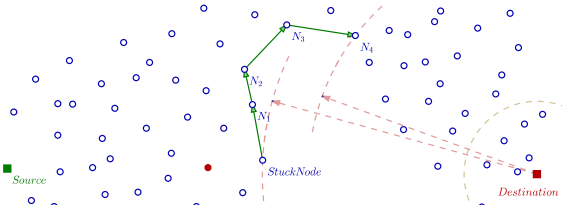
(d) Step4: N_2 forwards F_1 to N_3, N_2 chooses $zone_1$ using Eq. (1) then selects N_3 according to Eq. (2).



(e) Step5: N_3 receives F_1, N_3 is not closer than the stuck node to the destination



(f) Step6: N_3 forwards F_1 to N_4, N_3 chooses $zone_1$ using Eq. (1) then selects N_4 according to Eq. (2).



(g) Step7: N_4 receives F_1, N_4 is closer than the stuck node to the destination, N_4 sends a Bant to the stuck node to mark the path found.

Figure 6: An example of a Fant searching for a path around a void

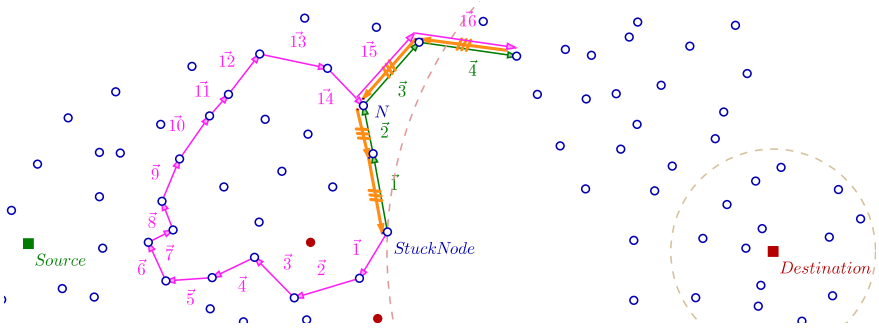
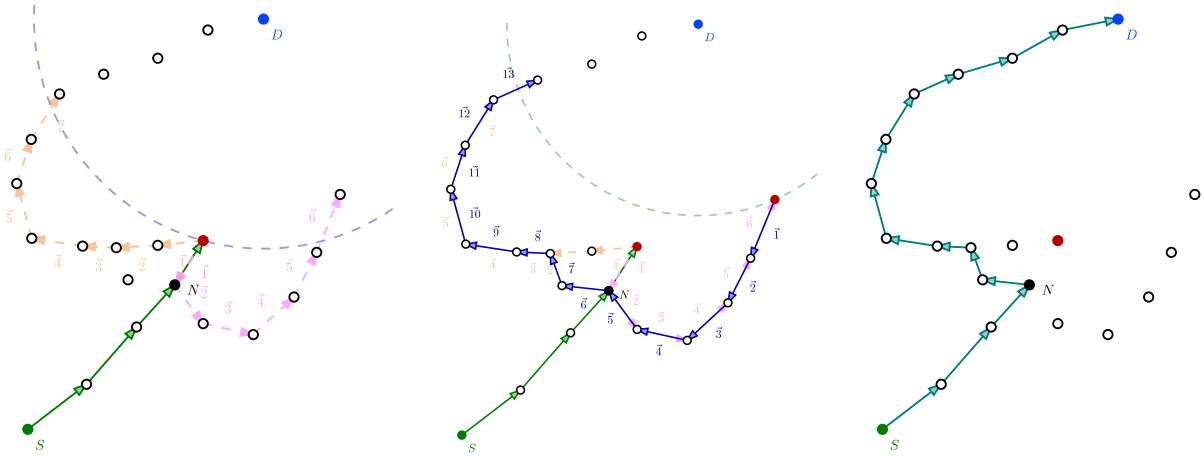


Figure 7: Bant returns to the stuck node

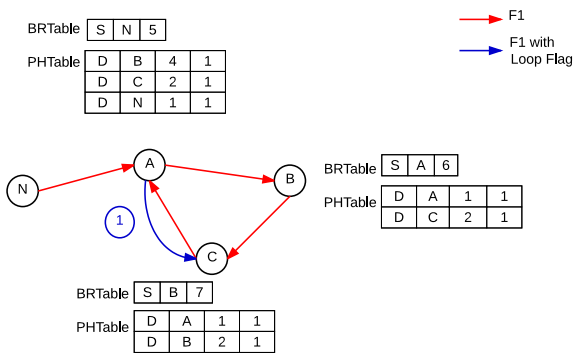


(a) First Recovery

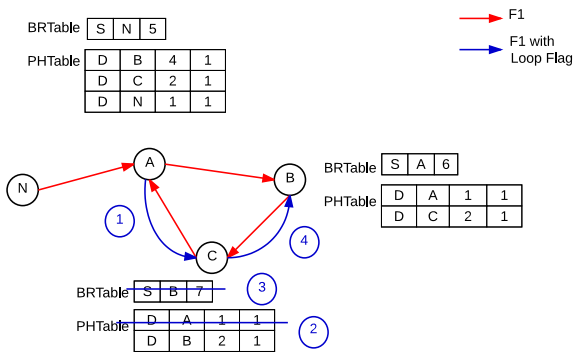
(b) Second Recovery

(c) path used by the second data packet sent by the source

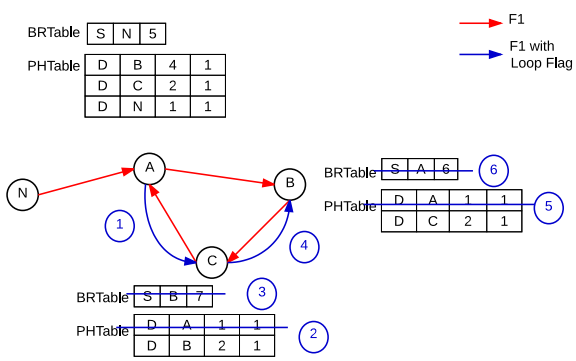
Figure 8: Multiple recoveries



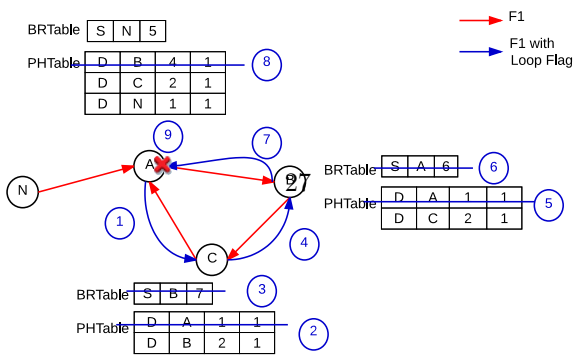
(a) Node A detects F1 in a loop, A assigns a loop flag to F1 and returns it back to C



(b) C receives F1 with A LoopFlag, it deactivates the link to A in its PHTable, sends F1 back to B and deletes B from its BRTable



(c) B receives F1 with A LoopFlag, it deactivates the link to C in its PHTable, sends F1 back to A and deletes A from its BRTable



(d) A receives F1 for the second time, it deactivates the link to B in its PHTable and kills the ant F1

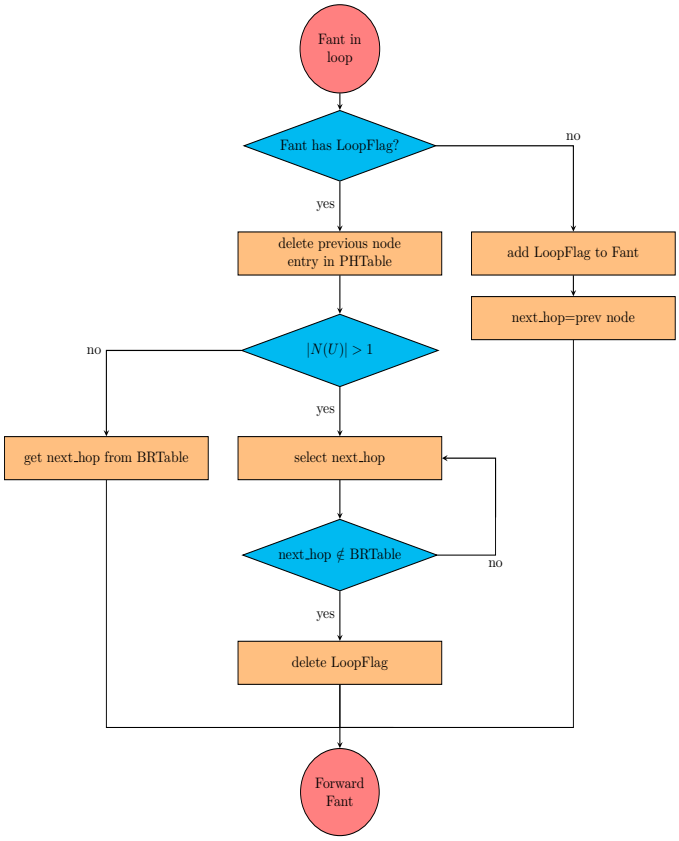
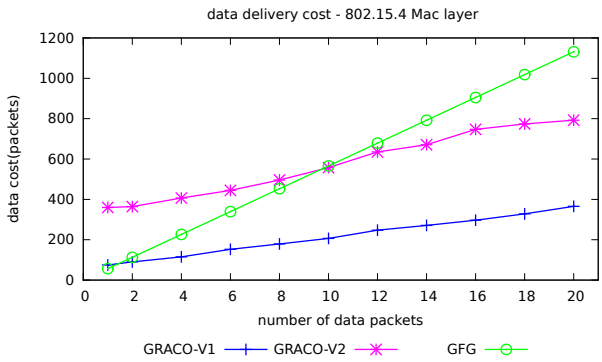
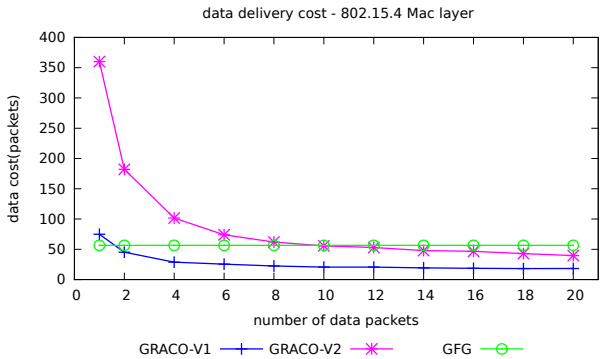


Figure 10: Loop Management

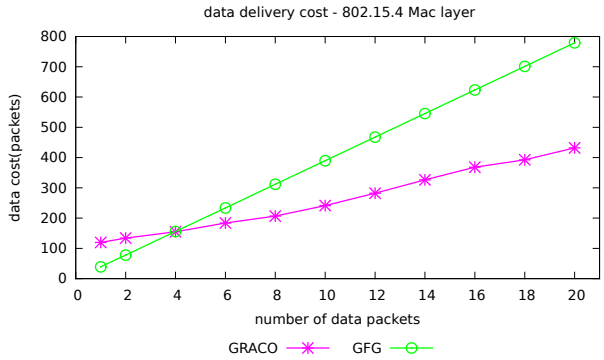


(a) Data cost varying wrt the number of packets sent

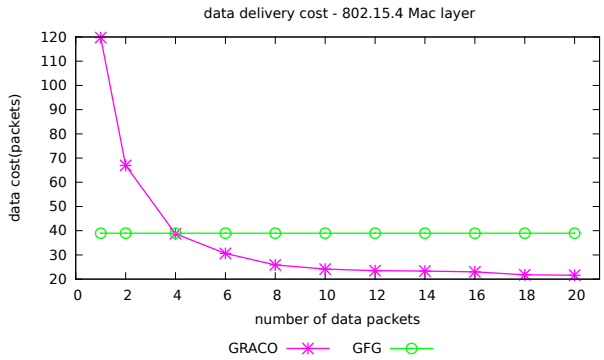


(b) Data cost per packet

Figure 11: Data cost - Ideal MAC Layer



(a) Data cost varying wrt the number of packets sent



(b) Data cost per packet

Figure 12: Data cost - 802.15.4 MAC Layer

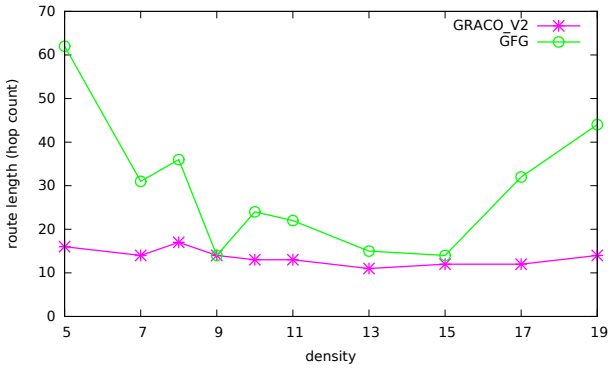


Figure 13: hop count

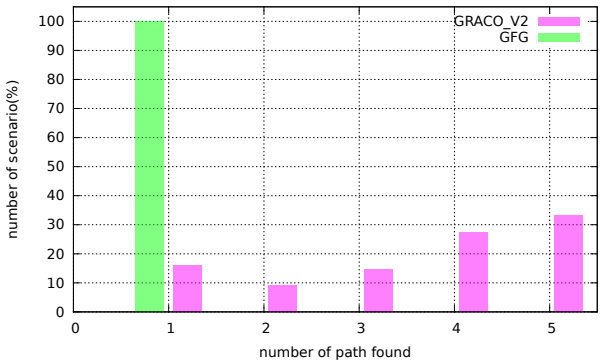


Figure 14: path count

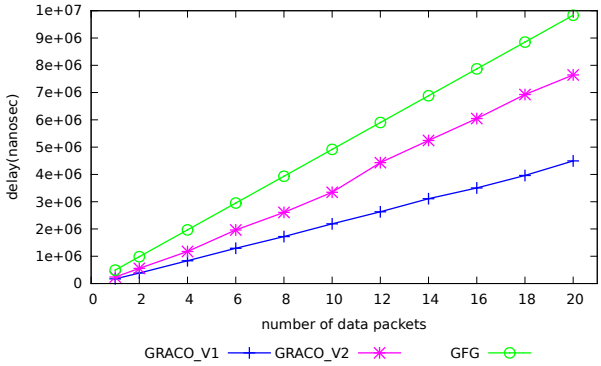


Figure 15: end-to-end delay varying with the number of data packets sent - Ideal MAC Layer

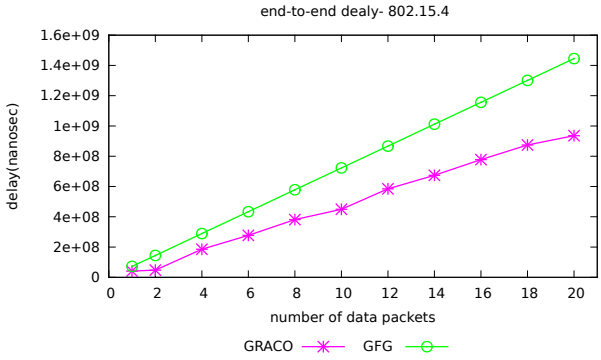


Figure 16: end-to-end delay varying with the number of data packets sent - 802.15.4 MAC Layer

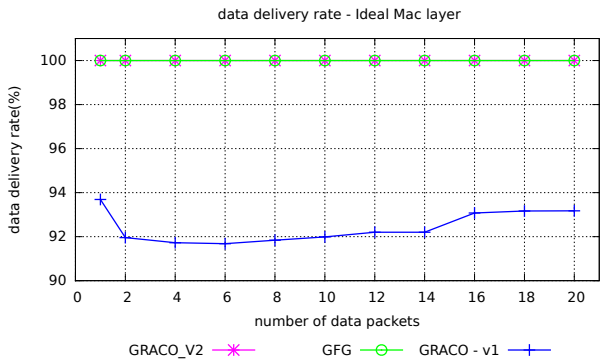


Figure 17: delivery rate varying with the number of data packets sent with an Ideal Mac layer

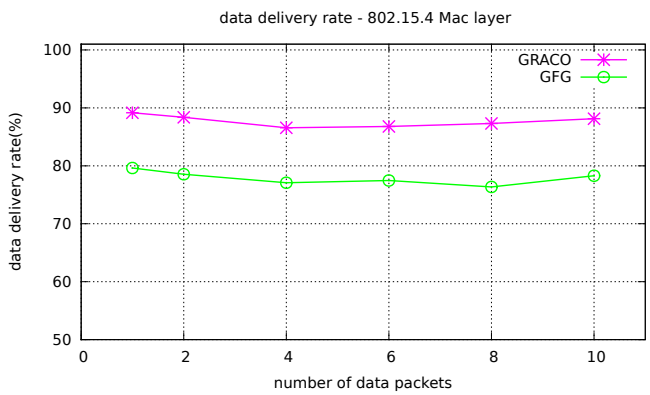


Figure 18: delivery rate varying with the number of data packets sent with 802.15.4 Mac layer