# Adaptive Broadcast Cancellation Query Mechanism for Unstructured Networks

Rui Lima *
Univ. Lusíada
CLEGI, V. N. Famalicão, PT
rml[at]fam.ulusiada.pt

Carlos Baquero
Univ. do Minho & INESC Tec
HASLab, Braga, PT
cbm[at]di.uminho.pt

Hugo Miranda
Univ. de Lisboa
LaSIGE, Lisboa, PT
hmiranda[at]di.fc.ul.pt

## ABSTRACT

The availability of cheap embedded sensors in mobile devices boosted the emergence of unstructured networks using wireless technologies without centralised administration. However, a simple task such as collecting temperature needs a discovery service to find a thermometer. Usually, the resource discovery relies on flooding mechanisms, wasting energy and compromising system availability. On the other hand, energy efficient solutions based on broadcast cancellation mechanism have a significant impact on latency.

The paper proposes ABC (Adaptive Broadcast Cancellation) a new algorithm that uses the knowledge acquired in the past to accelerate queries towards the resource. Each node listens to its neighbours and the acquired context is stored in a variation of Bloom filters.

## Categories and Subject Descriptors

Networks [**Network protocols**]: Cross-layer protocols; Network types [**Ad hoc networks**]: Mobile ad hoc networks

## General Terms

Protocols, Searching, Broadcast, Cancellation.

## Keywords

Unstructured Networks, Mobile Systems, Resource Searching, Broadcast.

## 1. INTRODUCTION

Many systems have been developed to monitor environmental resources and people location in adverse places. Mining and extraction companies increasingly use mobile devices with autonomous power to gather operating conditions data. In the past, collecting data concerning the temperature, humidity and air quality was performed manually and with a low sample rate. This is not useful for accident prevention as it is a slow and costly process. Mobile sensors and wireless communications make it possible to collect data at multiple places and with a much higher frequency, accuracy and reduced cost. The use of low cost wireless sensors allows to deploy an unstructured network in a mine, without extending copper wires over several kilometres [11]. Unstructured networks with sensors operating in multi-hop mode, allow each node to collect data and send messages through the network as new tunnels are opened. The network topology is being changed over time but the monitoring system still detect landslides that could put workers in hazard.

The mine scenario is challenging for communications and applications. One must cope with an adverse propagating media while simultaneously trying to increase sensors lifetime in spite of the large number of queries performed to the infrastructure. When trying to locate some resource or answer to a query, the simplest approach is to take advantage of the broadcast nature of the wireless propagation media to flood the query. In contrast with more ordered approaches, for example those relying on point-to-point or multicast protocols, this is more robust as it copes well with a changing topology and with the uncertainty that characterises resource location queries. Unfortunately, successive floodings of the network is expensive as it requires a large number of transmissions and occupies bandwidth, contributing to deplete the mobile nodes energy cells. Pure flooding protocols must be avoided when using battery power devices, because they waist considerable amounts of energy on unnecessary retransmissions. Some energy efficient searching mechanisms aims to attenuate the negative impact of flooding by limiting the query scope. This can be achieved by adding additional delay, exploring a trade-off between energy consumption and time.

This paper proposes a new strategy to improve the performance of flooding protocols that limits the query scope, by incorporating a distributed learning mechanism that adapts the protocol behaviour to the context in which it is used. Historical data is used to calibrate artificial delays on the retransmission, privileging the propagation of the flood on the direction where the resource is more likely to be found. This mechanism is combined with previous results on flood cancellation to produce an efficient search algorithm for Wireless Sensor Networks (WSNs). The paper shows that the proposed **Adaptive Broadcast Cancellation** protocol (**ABC**) can significantly improve the latency and reduce the number of retransmissions in long term scenarios where queries are frequently repeated. ABC uses a piggyback approach to learn about the resource locations, which is used to reduce both latency and retransmissions in subsequent queries. To save resources on the devices, ABC uses a new bloom filter based mechanism, **Linear Bloom Filters** to store and propagate compact information about resources in the local

*The work reported in this paper was co-financed by FCT - Fundação para a Ciência e Tecnologia, Portugal (PEst-OE/EME/UI4005/2011) and carried out within the research centre Centro Lusíada de Investigação e Desenvolvimento em Engenharia e Gestão Industrial (CLEGI).

vicinity.

ABC protocol was evaluated using the well tested `ns2` network simulation under several network scenarios. Evaluation of results shows that it is possible to speed up the searching tasks without compromising the energy efficiency.

## 2. RELATED WORK

Unstructured networks have been boosted by wireless technologies and node mobility. Mobile nodes can establish connections to each other, without imposing a particular structure, and turn life easier for end users as there is no need to configure a specific network structure. Nodes can form unstructured networks when they are self-organized to forward incoming messages in order to ensure a multi-hop communication network between themselves. Although without a rigid structure, when a node needs to find some network resource or data, a query must be submitted to search for it. Broadcasting is a communication paradigm where each node can send one message to its neighbour nodes, that are within the radio range. Protocols using broadcast communication paradigm can be more energy-efficient, when compared with unicast, avoiding to send multiple copies of the same message. One of the most simple implementation of broadcast is known as flooding. For example, many routing protocols available for ad-hoc networks such as Dynamic Source Routing (DSR) [7], Ad-Hoc On Demand Distance Vector (AODV) [14], Zone Routing Protocol (ZRP) [2] and Location Aided Routing (LAR) [10] are using the broadcasting communication building block to establish routes between nodes, but with the cost of flooding the entire network [8]. Flooding is a very energy inefficient [15] mechanism because there is no need to continue searching after receiving the requested information and a significant amount of transmissions will never take place if the search stops.

Some solutions [7, 4] have been proposed to control the query searching scope and avoid unwanted floods. The Expanding Ring Search (ERS) [7] algorithm limits the query scope by adding a Time To Live (TTL) handler field to the message. Each intermediate node decrements the TTL packet field when it forwards the query message. If the TTL is initialized to 1, only neighbour nodes (one hop) will receive the query. If the resource was not found, then another query is initiated and the TLL value is incremented so that the query can go for the next hop. This process is carried out until the initiator receives a successful answer. ERS can be energy efficient when the resources are in the firsts hops. However, the overlapping of successive searching areas results on many redundant messages that compromise the energy efficiency for the discovery of distant resources. Other flooding based mechanisms limit the expanding query using geographical data or distance gradients. Unfortunately a comparison of these methods shows they all exhibit a comparable performance [4].

Solutions using chasing packets to stop the query propagation proved to be more energy-time efficient. In the Blocking Expanding Ring Search (BERS$^\star$) [13] and in some variations [16, 1, 21] a new control message was added to stop the retransmission process. When some resource is found an acknowledgement is sent back to the initiator node. The initiator sends the stop message that is forward to every node involved in the searching process and stops the expanding query. BERS family protocols are energy efficient when nodes can wait a minimum time ($delay \leftarrow f(hop)$) for the corresponding hop before retransmission. During the delay time window, if a pending retransmission node receives a stop message, the node aborts the forward operation and the expanding query is blocked. However adding the minimum delay suitable for cancellation reduces the time efficiency. There is a trade-off between time efficiency and energy efficiency for protocols using chasing packets. More recent protocols such as Broadcast Cancellation Initiated from Resource (BCIR) [12] and timeBERS (tBERS) [17], instead off waiting for the initiator node to start the cancellation process, start sending cancellation packets as soon as the resource is found. Both protocols improve time efficiency without additional energy costs. The Tree-based Power-aware Source Routing (TPSR) [19] is a protocol that uses a Pub/Sub mechanism to efficiently collect information over Mobile Ad hoc Networks (MANETs). TPSR provides a working route between subscribers and publishers for a reasonable amount of time while ensuring that the number of hops is small. The optimizations are based on the received signal strength and use bloom filters to merge similar subscriptions to improve protocol efficiency. The Two-Sided Expanding Ring Search (TSERS) [20] considers a searching mechanism where two nodes are simultaneously executing the ERS for a path between each other. All intermediate nodes that receive a query can send back a successfully response to the source of the query. TSERS explores the fact that searching cost is lower for shorter distances than for longer ones.

Linear Bloom Filters, introduced in this paper, provide both a generalization of Bloom Filters [3] and a simple approach to attenuate bloom filters and capture gradients of resources in a network. The idea of using attenuated bloom filters and directing routing towards resources is originally in [18]. Their design is based on a vector of standard binary bloom filters that capture the resource presence at different hop distances; by assigning different weights to each level a metric similar to our *confidence* metric can be derived. Gradient-based routing using Bloom Filters is also used in the Wader design [5]; here the strategy is to decay the information in the filter, by setting random bits to 0 as filters are transmitted, so that resources further away have a lower representation in the local filter.

## 3. ABC QUERY MECHANISM

The previous section showed that broadcast cancellation protocols increase the delay linearly with hop distance between nodes in a blind way, disregarding any additional factor. Increasing delay creates a time window to disseminate broadcasting cancellation messages, thus minimizing the number of retransmissions, but with negative impact on time efficiency. This is the classical approach which is orthogonal to the resource location and network topology. To the best of our knowledge none of the protocols BERS$^\star$ and BCIR$^\star$ take advantage of the knowledge obtained with previous queries to improve the efficiency of the next. The paper proposes the Adaptive Broadcast Cancellation (ABC) protocol considering the fact that in the majority of the scenarios for Wireless Sensor Networks (WSNs) the movement of nodes is slow or nonexistent.

ABC takes advantage of topology and resource location stability to estimate resource location contributing to improve broadcast performance. The ABC protocol abandons the blind delay increase by tuning it with the knowledge

gained in previous queries to estimate the resource location. The key idea is to have nodes propagating queries faster in the direction where the resource is expected to be found. In contrast, when node relay function is assumed to be irrelevant for a specific query, delay is kept at an adequate level to optimise the cancellation process.

## 3.1 ABC Algorithm Description

To describe the ABC algorithm we assume that a node $N_i$ can have three primitive actions related with each query: $N_i$ can start a new query, by broadcasting a searching message $m_s$; can start the cancellation process, by sending a cancellation message $m_c$ or; $N_i$ can forward either or both of the messages $m_s$ and $m_c$. Network communication is asynchronous. Messages $m_s$ and $m_c$ are tagged with a query ID, so that $N_i$ can track several concurrent queries. Therefore, the remainder of the text focuses on the dissemination of a single query, knowing that other similar disseminations can be occurring concurrently on the WSN. Before broadcasting the first query there is no network context and each $N_i$ only knows its local resources. However it will be necessary to spread the local context through the network.

There is no information on the number of nodes or topological properties, such as network diameter.

## 3.2 ABC Knowledge Representation

Knowledge on the direction of the resources is abstracted on a space-efficient probabilistic data structure coined as Linear Bloom Filter (LBF), stored by each node. A traditional Bloom filter is represented by a binary array where each position can store a bit $\{0, 1\}$, that is used to test whether an element is a member of a set or not. With the LBF variation of Bloom Filters our approach stores floating point values $c \in [0, 1]$ instead of binary values. This variation extends traditional test operation, permitting to assign a confidence $c$ for an element to be in a set, with 0 representing absence of knowledge and 1 the highest confidence. By default $c$ is set to 0 for all LBF positions. When a node owns some resources, this information can be stored by setting $c$ to 1 on the positions associated to the resources by the hash functions. In ABC, resources owned by neighbours are added to the same structure by increasing the values of $c$. Figure 1 depicts an LBF sample representing the confidence values for elements $\{a, b, c, d, e\}$. By having all their positions with confidence 1 elements $\{a, e\}$ must be stored locally. The high confidence value associated to resource $b$ suggest that it is stored in some node in the neighbourhood, while $c$ and $d$ are lively hosted on more distant nodes.

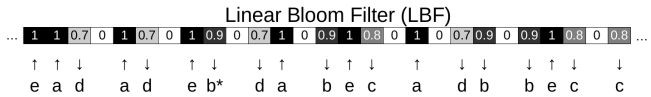



Figure 1: Node context with LBF

**LBF Operations:** Algorithm 1 defines three functions to manipulate the LBF data corresponding to the actions: i) inserting elements into a LBF, ii) merging data from two LFB and iii) getting the probability for some element. Network resources are designated as $elem_i$ in LBF context. Function $LBFins(elem_i)$ is called to insert some $elem_i$ in node $i$'s LBF. It generates the hash keys to address $k$ independent bloom array positions, setting them to 1. Function $LBFmerge(rLBF[])$ combines information of the nodes LBF with that from another node. The remote data is attenuated by a constant factor lower than 1 to privilege local information. Function $LBFget(elem_i, lLBF[])$ returns an estimate of the confidence value $c$ assigned for element $elem_i$. The cumulative attenuation for some resource at Hop distance is $c = att^{Hop}$.

**Algorithm 1:** Linear Bloom Filter (LBF)

```
1  lLBF[] ← [0, ··· , 0];    // local Linear Bloom Filter
2  att ← 0.9;                          // Attenuation factor
3  k ← 4;                        // Number of hash functions
4  Function LBFins(elem)
5  |  for i ← 1 to k do
6  |  |  idx ← hash_i(elem);
7  |  |  lLBF[idx] ← 1;
8  |  end
9  Function LBFmerge(rLBF[])
10 |  for i ← 1 to rLBF.size do
11 |  |  lLBF[i] ← max(lLBF[i], rLBF[i] × att);
12 |  end
13 Function LBFget(elem,lLBF[])
14 |  c ← 1;
15 |  for i ← 1 to k do
16 |  |  idx ← hash_i(elem);
17 |  |  c ← min(c, lLBF[idx]);
18 |  end
19 |  return c;
```

When a node sends a message, his $lLBF[]$ is piggybacked in the message and incorporated in the LBFs of the nodes receiving it, using function $LBFmerge(rLBF[])$. The attenuation factor ensures that the impact of resources stored by neighbours have a greater impact on the nodes LBF than those hosted by distant ones. A sample of $lLBF[]$ is depicted in Figure 1 showing that some node owns resources $\{a, e\}$. The near resource is $\{b\}$ while $\{d\}$ is the most distant known resource.

An interesting result of ABC is that the expected distance between a node and any resource $elem$ can be estimated using the expression in Eq. 1.

$$H_e = \frac{\log(LBFget(elem, lLBF[]))}{\log(att)} \quad (1)$$

## 3.3 ABC Query Dissemination

As illustrated in Alg. 2, a query is initiated with the broadcast of a message $m_s$. When $N_i$ starts a new query it creates a new packet to be disseminated to the network. $N_i$ stores in local memory the query unique ID ($queryID$), to avoid message duplication and spurious retransmissions. $N_i$'s LBF is copied into the message header, thus initiating its dissemination by the network. ABC uses the piggybacking technique to acquire resource context. To prevent further transmissions associated with the learning mechanism, we chose to use the searching messages to carry the information from the resource location. Learning with a piggybacking approach prevents the dissemination of additional messages. The increase in the size of the payload is not significant and is limited to the LBF size.

Nodes are kept in standby until they are waked to start a

**Algorithm 2:** Start Query

```
1 begin
2     pkt ← creatPacket(···);
3     queryList ← queryList ∪ {msg.queryID};
4     pkt.bloom ← node.lLBF;
5     send(pkt);
6 end
```

new query, or until they receive a message. Duplicated messages are ignored, based on the query ID. After processing the message, nodes will decide to forward the search message or start the cancellation process, using the searching mechanism described in BCIR$^\star$ [12] protocol, according to the multi-hop communication paradigm.

## 3.4 ABC Query Handling

ABC extends the BCIR$^\star$ query propagation algorithm which progressively increases message propagation latency to facilitate the dissemination of cancellation messages once the resource is found. This is reflected in the query handling algorithm depicted in Alg. 3. Like in BCIR$^\star$, the first step is to confirm message validity, discarding it if *i*) a cancellation for this message was already received or *ii*) it is a duplicate. Afterwords, the query may produce two outcomes. When the node receives $m_s$ and hosts the corresponding resource, it starts the cancellation process by sending $m_c$, otherwise, the query message is schedule to be retransmitted.

Concerning the query propagation, the differences between BCIR$^\star$and ABC are on the delay applied to the queries by each node. Alg. 3Alg 4 depicts the algorithm followed by ABC which considers 3 components. The resource location independent component is given by a static value (DELAY_MIN) plus some *jitter*, modelled by a random uniform distribution. The location dependent component makes the delay proportional to the number of hops travelled by the message and obeys to the BCIR$^\star$ algorithm. This is the component that facilitates message cancellation. The third component is dictated by the nodes estimation of its distance to the resource, given by lLBF. ABC adds an additional coefficient to attenuate the delay given by $(1 - c_2)$ (see Alg. 4) that is complementary of the confidence index, to speed up the query propagation in the predicted direction of the node hosting the resource. The $calcDelay(msg, proto)$ function in Alg 4 returns the future schedule time for message retransmission according to the message protocol. Table 1 shows the added delay for each protocol.

Figure 2 depicts an example of a query propagation accelerated into the most likely direction. It considers that $N_0$ starts looking for a resource located at $N_1$ and assumes that nodes know the confidence $c$ associated with the expected hop distance ($H_e$). $N_3$ receives $m_s$ and given that $c_1 < c_2$, an attenuation $(1 - c_2)$ is applied to the delay, speeding up the query towards to node $N_1$. On the other hand, given that $N_2$ receives $m_s$ and $c_1 \geq c_2$, no attenuation will be applied.

# 4. EVALUATION

Modeling of ad-hoc network topologies is usually under Random Geometric Graphs (RGG) or approached by more regular topologies, such as those in driven by Manhattan networks. This paper evaluates both of these topology classes,

**Algorithm 3:** Query Handling

```
1  Function handleQuery (msg)
2      LBFmerge(msg.rLBF);
3      pkt.bloom ← node.lLBF;
4      if msg.queryID ∉ queryList then
5          queryList ← queryList ∪ {msg.queryID};
6          if LBFget(msg.elem, lLBF) = 1.0 then
7              cancelList ← cancelList ∪ {msg.queryID};
8              send(CancelPacket(pkt));
9          else
10             time ← calcDelay(msg.header, ABC);
11             insertTxQueue(time, pkt);
12         end
13     end
14 Function handleCancellation(msg)
15     LBFmerge(msg.rLBF);
16     pkt.bloom ← node.lLBF;
17     if IsInTxQueue(pkt) then
18         removeTxQueue(pkt);
19     else
20         if (NodeOwnsResource(msg.elem) ∧
           node.resourceFound[msg.queryID] = False
           then
21             node.resourceFound[msg.queryID] ←
               True;
22             msg.resourceHost[msg.queryID] ←
               node.ID;
23         end
24         if msg.queryID ∉ cancelList ∧
           msg.queryID ∈ queryList then
25             time ← calcDelay(msg.header, FLOOD);
26             insertTxQueue(time, pkt);
27         end
28         cancelList ← cancelList ∪ {msg.queryID};
29     end
```

**Algorithm 4:** Added delay

```
1  const DELAY_MIN ;                    // Minimum delay
2  const JIT_MAX ;                      // Maximum jitter
3  Function calcDelay (msg,proto)
4      jitter ← runif(0, JIT_MAX);      // Random Uniform
5      time_FLOOD ← DELAY_MIN + jitter;
6      time_BCIR* ←
       max(2 · (msg.hop − 1) · time_FLOOD, time_FLOOD);
7      c1 ← LBFget(msg.elem, msg.rLBF);
8      c2 ← LBFget(msg.elem, node.lLBF);
9      if c1 < c2 then
10         time_ABC ← max(2 · (msg.hop − 1) ·
           DELAY_MIN · (1 − c2) + jitter, time_FLOOD);
11     else
12         time_ABC ← time_BCIR*;
13     end
14     return now() + time_proto;
```

Table 1: Added delay for each protocol

| Protocol | Added Delay |
|---|---|
| $FLOOD$ | DELAY_MIN + $jitter$ |
| $BCIR^\star$ | $2 \cdot (HOP - 1) \cdot$ DELAY_MIN + $jitter$ |
| $ABC$ | $2 \cdot (HOP - 1) \cdot$ DELAY_MIN $\cdot (1 - c) + jitter$ |

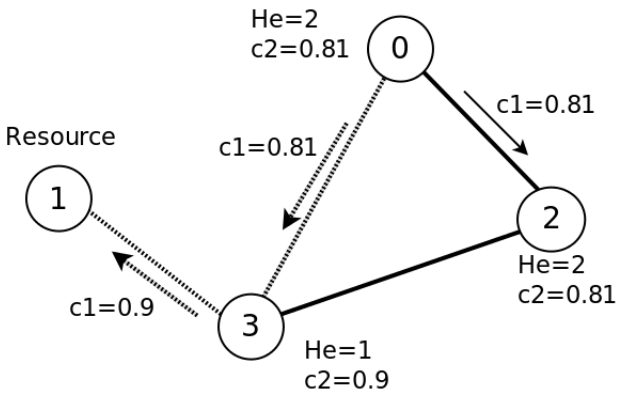



Figure 2: ABC - Delay Attenuation

while examining some more extreme deployments, occurring in mines [11], that exhibit higher path lengths. To evaluate the performance of ABC and understand the impact of the adaptive delay on the number of transmissions and end-to-end delay, an implementation for the `ns2` network simulation was developed. ABC is compared with a naive implementation using plain flooding with BCIR$^\star$ [12], since both use similar cancellation mechanisms. ABC includes the adaptive delay while BCIR$^\star$ operates without knowledge on the location network resources.

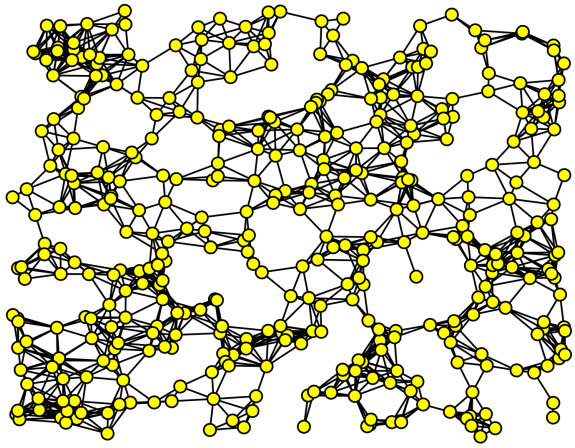
Two metrics are used to study the ABC time and energy efficiency behaviour, namely end-to-end delay and the number of transmissions. The **end-to-end delay (L)** is defined as the average time between the broadcasting search by the initiator node $N_0$ and the moment at which $N_0$ receives the first answer. The energy cost is measured by the average **number of transmissions (R)** performed by every node on each query. It is assumed that any two retransmissions consume the same amount of energy. Two network topology classes were considered *i*) a equidistant grid topology and *ii*) mine topologies created using a modified Random Geometric Graph (RGG).

## 4.1 Network Topologies

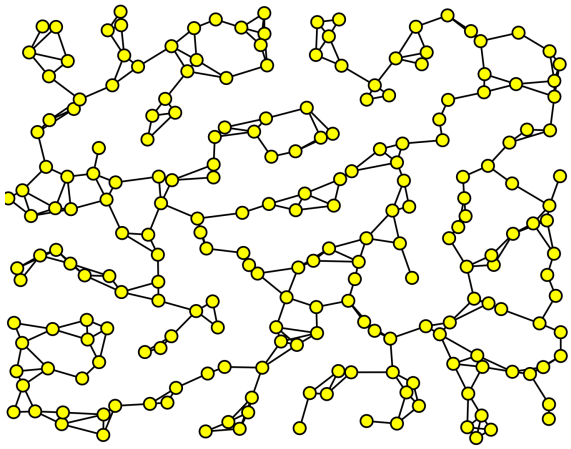
The **equidistant grid** topology was generated with 176 nodes equidistant from each other, ensuring that the interior rows share equitably distance to cover the area. The distance between the rows is greater than the radio range, to avoid direct communication from one row to another. Nodes are distributed along the rows covering an area of $3200 \times 3200\text{m}^2$, resulting in a node density of $\simeq 17$ Nodes/Km$^2$.

The **Manhattan** topology was generated using the BonnMotion [1] tool. The streets with 250 nodes equidistant from each other, ensuring that the interior streets share equitably distance to cover the area. The distance between the streets is greater than the radio range, to avoid direct communication from one street to another. Nodes are distributed along the streets covering an area of $2000 \times 2500\text{m}^2$, resulting in a node density of 50 Nodes/Km$^2$.

The creation of the **mine topologies** mirrors previous works [9] [6] using random geometric graphs (RGG) to model ad-hoc networks topologies. The bi-dimensional random geometric graph model $G(N, r, \ell)$ creates $N$ nodes that are randomly placed inside an unit square area. The topology



(a) Random Geometric (N $\simeq$ 400)



(b) *Modified* Random Geometric (N $\simeq$ 170)

Figure 3: Adaptation for Mine Topologies

defined in the unit square is mapped into the `ns2` using a linear transformation taking into account the radio transmission range. Any two nodes whose euclidean distance is below $r$ are considered connected and linked by an edge. Unfortunately the RGGs created using NetworkX[2] reveal a graph topology that was not suitable to represent mine topologies scenarios because there were too many clusters with high level of redundancy. To create more realistic scenarios a maximum graph degree was defined by removing nodes whose degree exceeds a threshold. The result is a *modified* random geometric graph. This adaptation is exemplified in Figure 3. Figure 3a depicts an original RGG as created by NetworkX. Figure 3b depicts the same graph after imposing a limit to the node degree. The modified RGG is considered to make a better representation of the mine structure, resulting in a topology with longer paths (tunnels) and lower node density.

## 4.2 Experiment Setup

Table 2 summarizes the parameters used for setting `ns2` simulations. The antenna settings, signal power and receiv-

[1] `http://sys.cs.uos.de/bonnmotion/index.shtml`

[2] `http://networkx.github.io`

Table 2: ns2 Simulation Parameters

| Attribute | Value [Default] |
|---|---|
| DELAY_MIN | 10 ms |
| jitter | 7 ms |
| Packet Size | 1000 bytes |
| Max Simulation Time | 17000s |
| Number of topologies | 500 |
| Radio range | 250 m |
| Network links | 11 MBit/s |
| Transmission power | 25 dBm |
| Antenna gain | 1 |
| Antenna Polarization | Omnidirectional |
| ns2 propagation model | TwoRayGround |

Table 3: Packet Drops

| TOPOLOGY (NODES) | FLOOD | BCIR* | ABC |
|---|---|---|---|
| Equidistant Grid (176 nodes) | $\simeq 11\%$ | $\simeq 10\%$ | $\simeq 10\%$ |
| Manhattan (250 nodes) | $\simeq 30\%$ | $\simeq 32\%$ | $\simeq 33\%$ |
| Mine ($\simeq$ 170 nodes) | $\simeq 5\%$ | $\simeq 3\%$ | $\simeq 3\%$ |

ing threshold, are selected to give approximate 250m of radio communication range. To each node is attached one resource. The queries are schedule in random order and search for random resources.

The simulations uses $k = 4$ hash functions and the size of LFB is $m = 256$, as suggested by Eq. (2), by specifying a false positive probability up to $fp = 5\%$ and considering that LBF can keep information for at least 40 resources (i.e. $\#LBF < 40$).

$$m = -\frac{\#LBF \cdot \ln(fp)}{(\ln 2)^2} \quad ; \quad k = \ln(2) \cdot \frac{m}{\#LBF} \qquad (2)$$

The attenuation factor (used in Algorithm 1) is set to $att = 0,9$ corresponding to 10% of attenuation for each hop.

## 4.3 Result Analysis

Results are presented as a function of the distance between the initiator and the first node responding with a successful answer, a metric hereafter designated as HOP count.

For each simulation, all protocols are submitted to a pool of 500 topologies. For each topology, the queries are scheduled using a uniform distribution over the maximum simulation time. The absolute number of queries that received a successful reply is depicted in Figure 4 as a function of the number of hops of distance between the origin of the query and the node hosting the result.

Table 3 shows the percentage of packet drops for each protocol which was observed to be related with the minimum delay before propagation. Smaller delay values quickly lead to increased packet drop rates, typical of broadcast storms.

Evaluation considers only the queries that received a successful response. Figure 4, which shows the absolute number of successfully replied queries in the mine scenario suggest that the excessive amount of traffic produced by flooding can be one of the most significant reasons for query failures, since broadcast storms can prevent the successful propagation of queries.

**Latency:** Figure 5 shows the variation of the average end-to-end delay between the moment a query is broadcast and its sender receives a reply, arranged by the distance from the initiator to the target node (i.e. the node hosting the resource).

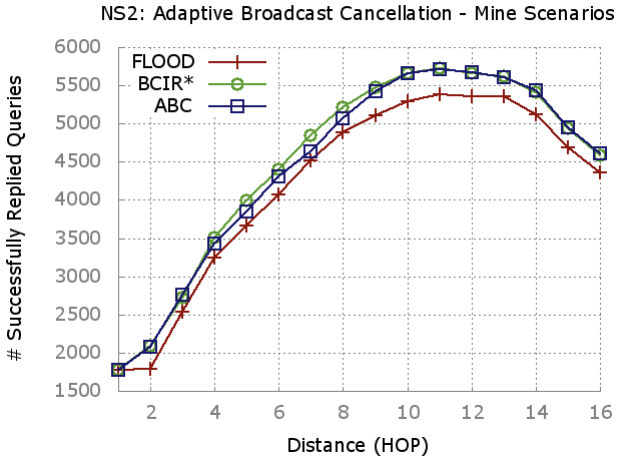



Figure 4: Absolute number of successful queries

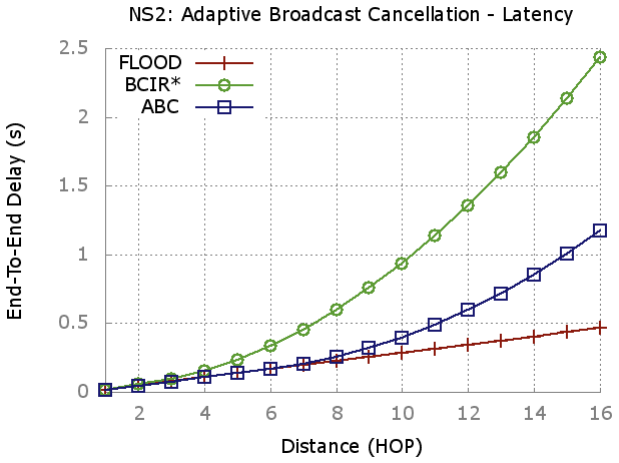



Figure 5: End-To-End Delay

The figure shows that ABC protocol average end-to-end time is considerable smaller when compared with BCIR*. For example, the time difference between BCIR* and the FLOOD baseline for $H = 4$ is similar to the difference between ABC and the FLOOD baseline for $H = 8$. When the resources are near the initiator node (small HOPs) both ABC and FLOOD have similar performances. This result can be attributed to the contribution of Bloom filters in the accelerating of the query propagation in the direction of the resource, a unique property of ABC which makes it competitive with flooding for resources located up to 7 hops.

**Energy impact:** Assuming that the transmissions are responsible for the greatest impact on energy consumption in a system based on wireless communications, we compare the energy efficiency by counting the number of transmissions used until all messages related with a query cease to be propagated. As depicted in Fig. 6 the energy performance of ABC is improved when compared with BCIR*. A comparison between the mine and the equidistant grid topologies

shows that the adaptation mechanism of ABC preserves the energy consumption in a pattern comparable to BCIR$^\star$, specially in the first hops. This is very significant considering that ABC was able to improve latency and that BCIR$^\star$ is capable of significantly reducing transmissions with respect to flooding.

## 5. CONCLUSIONS

Mechanisms to improve the performance of resource discovery in MANETs typically trade-off the number of messages by latency. This paper presented ABC, an algorithm that makes an innovative approach capable to attenuate such trade-off by speeding the query propagation in the predicted direction of the resource.

Evaluation showed that a sense of the resource location can significantly reduce the end-to-end time, while still benefiting from a reduction in energy cost for successive discovery queries.

ABC achieves these goals with a small penalty in the query message size, resulting from the piggyback of a LBF which epidemically disseminates the resource location direction.

## 6. REFERENCES


[1] M. A. Al-Rodhaan, L. Mackenzie, and M. Ould-Khaoua. Improvement to blocking expanding ring search for manets.

[2] N. Beijar. Zone routing protocol (zrp).

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[4] Q. Chen, S. S. Kanhere, and M. Hassan. Performance analysis of geography-limited broadcasting in multihop wireless networks. *Wireless Communications and Mobile Computing*, 13(15):1406–1421, 2013.

[5] D. Guo, Y. He, and Y. Liu. On the feasibility of gradient-based data-centric routing using bloom filters. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):180–190, Jan 2014.

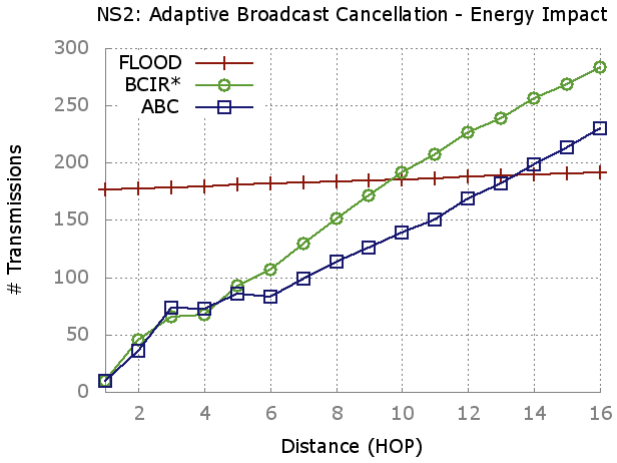
[6] M. Haenggi, J. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti. Stochastic geometry and random graphs for the analysis and design of wireless networks. *Selected Areas in Communications, IEEE Journal on*, 27(7):1029–1046, September 2009.

[7] D. B. Johnson, D. A. Maltz, and J. Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

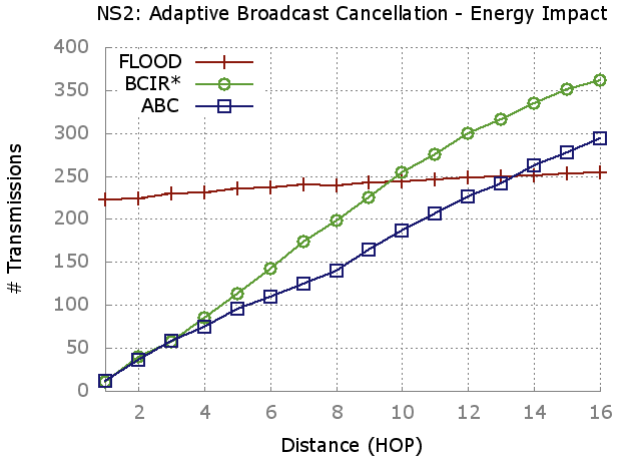
[8] D. Kasamatsu, Y. Kawamura, M. Oki, and N. Shinomiya. Broadcasting method based on topology control for fault-tolerant manet. *Distributed Computing Systems Workshops, International Conference on*, 0:105–110, 2011.

[9] H. Kenniche and V. Ravelomananana. Random geometric graphs as model of wireless sensor networks. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 4, pages 103 –107, feb. 2010.

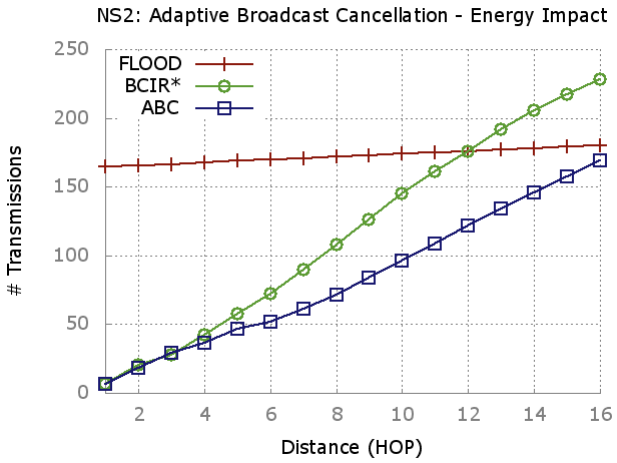
[10] Y.-B. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wirel. Netw.*, 6(4):307–321, July 2000.


(a) Equidistant Grid Topology




(b) Manhattan Topology




(c) Mine Topology

Figure 6: Number of Transmissions (R)

[11] M. Li and Y. Liu. Underground coal mine monitoring with wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(2):10:1–10:29, Apr. 2009.

[12] R. Lima, C. Baquero, and H. Miranda. Broadcast cancellation in search mechanisms. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 548–553, New York, NY, USA, 2013. ACM.

[13] I. Park, J. Kim, and I. Pu. Blocking expanding ring search algorithm for efficient energy consumption in mobile ad hoc networks. In *WONS 2006 : Third Annual Conference on Wireless On-demand Network Systems and Services*, pages 191–195, Les Ménuires (France), Jan. 2006. INRIA, INSA Lyon, IFIP, Alcatel. http://citi.insa-lyon.fr/wons2006/index.html.

[14] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pages 90–100, New Orleans, LA, USA, Feb. 1999.

[15] S. Preethi and B. Ramachandran. Energy efficient routing protocols for mobile adhoc networks. In *Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on*, pages 136 –141, april 2011.

[16] I. Pu and Y. Shen. Enhanced blocking expanding ring search in mobile ad hoc networks. In *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, pages 1 –5, dec. 2009.

[17] I. M. Pu, D. Stamate, and Y. Shen. Improving time-efficiency in blocking expanding ring search for mobile ad hoc networks. *J. of Discrete Algorithms*, 24:59–67, Jan. 2014.

[18] S. Rhea and J. Kubiatowicz. Probabilistic location and routing. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1248–1257 vol.3, 2002.

[19] S. Schnitzer, H. Miranda, and B. Koldehofe. Content routing algorithms to support publish/subscribe in mobile ad hoc networks. In *IEEE 37th Conference on Local Computer Networks Workshops (LCN 2012 Workshops)*, pages 1053–1060, Oct. 22–25 2012.

[20] S. Shamoun and D. Sarne. Two-sided expanding ring search. In *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*, pages 1–8, Jan 2014.

[21] A. Shintre and S. Sondur. Improved blocking expanding ring search (i-bers) protocol for energy efficient routing in manet. In *Recent Advances and Innovations in Engineering (ICRAIE), 2014*, pages 1–6, May 2014.