Mobile Information Systems 9 (2013) 261–280
DOI 10.3233/MIS-130162
IOS Press

A multi-hop advertising discovery and delivering protocol for multi administrative domain MANET

Federico Mari[a], Igor Melatti[a], Enrico Tronci[a] and Alberto Finzi[b,*]
[a]*DI, University of Roma "La Sapienza", Roma, Italy*
[b]*DSF, University of Napoli "Federico II", Complesso Universitario di Monte Sant'Angelo, Napoli, Italy*

Abstract. A *Mobile Ad-hoc NETwork* (MANET) is *Multi Administrative Domain* (MAD) if each network node belongs to an independent authority, that is each node owns its resources and there is no central authority owning all network nodes. One of the main obstructions in designing *Service Advertising, Discovery and Delivery* (SADD) protocol for MAD MANETs is the fact that, in an attempt to increase their own visibility, network nodes tend to flood the network with their advertisements. In this paper, we present a SADD protocol for MAD MANET, based on Bloom filters, that effectively prevents advertising floods due to such misbehaving nodes.

Our results with the ns-2 simulator show that our SADD protocol is effective in counteracting advertising floods, it keeps low the collision rate as well as the energy consumption while ensuring that each peer receives all messages broadcasted by other peers.

Keywords: Wireless network, Mobile Ad-hoc NETwork (MANET), Service Advertising, Discovery and Delivery (SADD)

1. Introduction

A *Mobile Ad-hoc NETwork* (MANET) is *Single Administrative Domain* (SAD) if all its nodes belong to a single authority (administrative domain). For example, a *Wireless Network* (WN) consisting of mobile sensors moving in a given area and gathering data (e.g. temperature) is a SAD-MANET since all sensors fall under the same administrative domain. Following [2], a MANET is *Multi Administrative Domain* (MAD) if each network node belongs to an *independent* authority. In other words, in a MAD-MANET each node owns its resources and there is no central authority owning all network nodes. For example, a network consisting of PDAs, laptops, and other WiFi capable devices each belonging to a different user is a MAD-MANET since each node has a different owner (its user). Note that, both network mentioned above are *Peer-to-Peer* (P2P) networks. However, the first one (mobile sensors) is a SAD-MANET whereas the second one (PDAs, laptops, etc.) is a MAD-MANET. Although the devices forming a SAD-MANET and a MAD-MANET may physically be the same, the dynamics of the two networks may be quite different. In fact in a MAD-MANET each node owns its resources (software as well as hardware). For example, a node in a MAD-MANET may modify the software running on its hardware or even modify the hardware if this is at its advantage (*selfish* behavior).

*Corresponding author: Alberto Finzi, DSF, University of Napoli "Federico II", Complesso Universitario di Monte Sant'Angelo, Via Cinthia – 80126 Napoli, Italy. E-mail: finzi@na.infn.it.

1.1. Motivations

WiFi (IEEE 802.11, e.g. see [31]) MANETs consisting of mobile devices such as laptops, cell-phones, PDAs are more and more widespread [56]. This opens up opportunities for many interesting applications. Typical examples are: mobile commerce (*m-commerce*), entertainment, content sharing, emergency management. Here are a few typical scenarios.

1. *Sell:* While you are having a nice walk your handheld device periodically *advertises* a service you may offer, e.g. piano lessons. Upon *discovering* that you offer piano lessons someone (possibly many network *hops* away from you) may ask you for more details that your handheld device will promptly *deliver* to your potential pupil.
2. *Buy:* During the very same walk your handheld device discovers that someone is advertising for math lessons. Your device knows you are interested in math lessons (since you told it) so it will ask for more details that will be delivered by the peer device of your potential teacher.
3. *P2P:* Of course, in much the same way content can be advertised and exchanged between peers thus supporting entertainment as well as emergency management applications.

Protocols supporting the above activities are often called *Service Advertising, Discovery and Delivery* (SADD) protocols. SADD protocols have been extensively studied. For example see [15,22,25,26,28,33, 35–37,51]. However, to the best of our knowledge, all SADD protocols proposed in the literature target SAD-MANETs.

Unfortunately, SADD protocols designed for SAD-MANETs may not work for MAD-MANETs. For example, a WiFi MANET consisting of handheld devices each belonging to a different user is indeed a MAD-MANET. If network nodes behave selfishly they may deviate from the specified protocol if this is at their advantage. Thus, a selfish PDA (user) may refuse to forward packets (to save energy) or may decrease its backoff time (to increase its bandwidth) or may increase its advertisement frequency (to increase its own visibility). Hence, a SADD protocol for MAD-MANETs must deploy suitable countermeasures to protect the network from node misbehaviors that may eventually kill any networking activity.

1.2. Node behavior

In order to design protocols for MAD-MANETs, reasonable hypotheses on node behaviours are needed. Here are some well known classes of node behaviours.

Malicious nodes are willing to spend their resources just to damage the network. For example, malicious nodes may perpetrate a *Denial of Service* (DoS) attack by flooding the network (or part of it) with their messages. Malicious nodes may do so even if this will use up all of their energy without actually doing them any real service.

Selfish nodes act in their best interest. For example, rather than spending its energy flooding the network with messages without getting any reward, a selfish node may refuse to forward packets (to save energy) or may decrease its backoff time after a collision (to increase its bandwidth).

Altruistic or *obedient* nodes (e.g. see [2]) just follow the given protocol. One may think that altruistic nodes do not exist. However it is just a matter of fact that most nodes (agents) in a MAD-MANET are indeed altruistic. That is one can often safely assume that most (although not all) network nodes are altruistic.

Assuming that all nodes are malicious is a too pessimistic hypothesis. No protocol for MAD-MANETs can be designed under such an hypothesis. As for MAD-MANETs the typical approach is to assume

that most nodes are selfish or altruistic (e.g. see [11]). In some cases malicious node can be tolerated following the approach in [2,38]. As for SADD protocols, the main obstruction to overcome is *flooding*. In fact, all nodes in the network will be eager to broadcast their advertisements (otherwise they would not participate in the protocol to begin with). This may result in flooding which, in turn, leads to a *Denial of Service* (DoS) attack. In fact, if *too many* nodes flood the network with their advertisements eventually no one will be able to send anything (DoS attack). Note that flooding is an *attractive* misbehavior for malicious as well as selfish nodes. In fact a selfish node may be interested in increasing its advertising frequency to increase its visibility. On the other hand, a malicious node may increase its advertising frequency since it is an easy way to flood the network and carry out a DoS attack. Of course, flooding is not the only possible misbehavior for nodes in a MAD-MANET. We note, however, that flooding is something that any node participating in the protocol will desire to do and, last but not least, can easily do by simply changing a protocol parameter (namely, the advertisement frequency). This is not the case with other attacks. For example, packet dropping may be desirable for a node, but usually requires some nontrivial work on the protocol implementation.

Resting on the above considerations, in this paper we assume that all network nodes follow the given protocol and may deviate from it only by increasing their advertisement frequency. This models the fact that nodes participating in the protocol are eager to broadcast their advertisements. Accordingly, our goal is to devise suitable countermeasures to guarantee that node attempts to increase their advertisement frequency do not result in an advertisement flood destroying any networking activity.

1.3. Our contribution

We present *MAD-SADD*, a SADD protocol for MAD-MANETs. From a functional point of view our protocol is similar to the SADD protocols for SAD-MANETs proposed in the literature (see Section 1.1). Our main contribution here is in the mechanism that allows our protocol to counteract advertisement *flooding*. In our setting, each node has an *advertisement*, a *profile* and a *full service description*. The advertisement and the profile define the services (e.g. content, resources, consultancy, etc.). The full service description (just *full description* in the following) gives full information about the advertised service. Here are examples of full descriptions. If the advertising node is offering a movie then the full description will be the advertised movie file. If the advertising node is offering, say, piano lessons, then the full description will be a file with the maestro address.

Each node periodically *broadcasts* its advertisement to the network nodes. All nodes cooperate in spreading the advertisement by *forwarding* it to their neighbors (*advertising* phase). Upon receiving an advertisement, a node uses its own profile to evaluate its interest in the received advertisement. If the received advertisement is considered interesting, the interested node starts a *unicast* transmission asking the advertising node for the *full description* (*discovery* phase). Finally, the advertising node delivers such full description using again a unicast transmission (*delivery* phase).

To limit collisions, nodes should avoid forwarding *recently* forwarded packets. In a SAD-MANET this is typically done by endowing packets with a *sequence number* field (e.g. see [39,53]). However, in a MAD-MANET sequence numbers do not work since a misbehaving node may broadcast many times the same packet (advertisement) using higher and higher sequence numbers. In this way, packets coming from that node will appear to other nodes as new packets and, accordingly, will be always forwarded. This increases bandwidth usage and visibility of the *misbehaving* node, but decreases bandwidth and visibility of all other nodes.

To counteract such misbehavior, nodes may store in RAM the forwarded messages. In this way, each node will be able to detect if an incoming message (from a certain node) is new or recently seen (and

processed). However, usually handheld devices do not have enough RAM to effectively store all recently seen messages.

We propose a trade-off between not storing messages (which results in an unacceptably high number of collisions) and storing them all (which results in an unacceptably high RAM usage).

More specifically, we propose to store advertisement signatures by using a *Bloom filter* [17]. A Bloom filter is a data structure that can effectively store message signatures. For example, using only 1.2 Kbytes of RAM (easily available on any handheld device) we can store signatures for more than 1000 messages with a false positive probability (i.e. the probability of considering as old an advertise that is actually new) of 9.4×10^{-3}. The Bloom filter is periodically cleared, thus only *recent* advertisements are kept. This allows nodes to propose old advertisements to newcomers.

Our experimental results with the ns-2 simulator confirm the effectiveness of our protocol in counteracting advertisement flooding DoS attacks. Namely, our protocol keeps low the collision rate as well as the energy consumption (*safety*) while ensuring that each peer receives all messages broadcasted from the other peers reachable in one or more hops (*liveness*).

1.4. Comparison with related works

SADD protocols have been widely studied. See for example [14,15,22,28,33–37,45,51]. However all mentioned papers address the problem of SADD protocols for SAD-MANETs, that is, they do not account for selfish or malicious *misbehaviors* of nodes. On the other hand, by exploiting the *obedient* nature of SAD-MANET nodes, the previously mentioned protocols propose routing (e.g. see [4] for a survey) schemas much more sophisticated than ours.

To the best of our knowledge no previously published paper addresses the problem of designing a SADD protocol for MAD-MANETs. There are however protocols for MAD networks (i.e. networks consisting of selfish nodes). An example is the BAR Gossip [38] protocol which allows an altruistic (i.e. following the protocol) broadcaster to stream data to a pool of possibly selfish or even malicious clients. Other approaches rely on specific domain policies and architectures [19].

Flooding of advertisements can be considered as a particular case (an easy one to carry out) of *Denial of Service* (DoS) attack. For this reason we will compare our protocol with those striving to counteract DoS attacks in MANETs. *Denial of Service* (DoS) attacks for ad hoc networks have been studied in [1, 32,48], but in these works advertisement flooding is not addressed.

The SEAD protocol in [29] makes use of elements from a one-way hash chain to provide authentication for both the sequence number and the metric in each entry. The SRP protocol in [47] is based on multiple routes and relies exclusively on the mutual authentication of the end nodes (source and destination). In [45], a probabilistic routing approach is proposed to drive the on-demand discovery process and to reduce the control overhead. Note that the above secure protocols do not solve our problem since they are not able to prevent flooding of legitimate messages (advertising) from legitimate nodes participating in the protocol. The Ariadne protocol [30] enables to secure the routing discovery phase and ensures that all forwarded packets follow the secure route. Here, request-flooding attacks are considered and the proposed solution consists of a rate limit for each node the route requests it is asked to relay. Even if mitigating the effect of flooding at the network layer, this solution does not solve our problem. Indeed, following the approach in [30] the more advertisement messages a node sends, the more it will get broadcasted at the expense of the nodes sending less advertisements. As a result, in our framework the Ariadne approach not only is not sufficient to discourage advertising flood, but encourages it since nodes that are not flooding may never see their advertisement broadcasted.

MANETs have been highly vulnerable to attacks due to the dynamic nature of their network infrastructure. A discussion on security attacks and techniques applicable to MANETs is presented in [5]. A risk aware response mechanism to systematically cope with routing attacks is proposed in [60]. In [46] the authors present an authentication, authorization and security assessment strategy in which, once a device enters a MANET, it is immediately taken out if considered dangerous by the infrastructure. None of the aforementioned approaches can be applied to our context since the DoS attack we are considering here (advertisement flood) does not fall in the class of attacks studied in [5,46,60].

In [58] the authors introduce and analyze *Ad Hoc mobile networks Flooding Attacks* (AHFA). The proposed countermeasures use neighbor suppression to counteract route request flooding attacks and path cutoff to counteract data flooding attacks. Note that this problem is different from the one considered in our paper. In fact, AHFA works at the network layer whereas in our case flooding stems from message advertisement flooding at the application layer and it is perfectly compatible with a legal behavior at the network level.

Since in a MANET nodes are usually constrained by limited computation resources, selfish nodes may refuse to cooperate. Consensus in sparse MANETs is discussed in [3]. In [40], the authors address the noncooperation problem following game-theoretic approach combining reputation and price-based systems. Cooperation in the context of Vehicular Ad-Hoc Networks (VANET) – specializing from MANETs – is studied in [42]. None of these approaches apply to our context where network flooding is the main misbehavior.

The use of bloom filters in networks is discussed in [8]. The deployment of bloom filters in MANET was proposed in [23,52] but only to store requested services or to guide service requests respectively, not as a method to counteract DoS attacks. Furthermore, differently form our approach, in [23] the authors analyze static nodes organized in a grid, while the service discovery protocol in [52] relies on a backbone of directories constituting a virtual network. Instead, in [57], the bloom filter is used as a mechanism for distributed call admission control in MANET.

2. Our scenario

The main assumptions underlying our MAD-SADD design are the following.

Misbehavior. The only possible misbehavior for a node participating in the protocol is network flooding with its own advertisements. As discussed in Section 1.2 this is the most relevant problem to consider in our context.

Strong Identity. Each node has a strong identity. This identity is a unique identifier (ID) for the node. This ID could be a node MAC address or, if available, an IP obtained as in [16,24].

Slow moving. We assume that nodes move *slowly enough* with respect to messages travel time in the network. In particular, nodes move slowly enough so that they can be considered standing still during the *unicast* communications (discovery and delivery). This guarantees that once an advertisement is found to be interesting and more data are requested the path between the source and destination nodes found during the broadcast phase remains valid. This hypothesis is quite reasonable in our setting. In fact we are considering MANETs consisting of handheld devices in a (downtown) metropolitan area such as a square, a mall, a shopping street, a restaurant, etc. In short, relatively crowded scenarios where people move slowly or just stand steel.

Fixed Bandwidth. The bandwidth that each node dedicates to our SADD protocol is fixed. This limits the node resources used by the SADD protocol. For example, even in a densely populated area a node will not spend all of its networking resources forwarding advertisements. This is a typical approach in P2P systems.

3. Some more misbehaviors

In this paper we focus on advertisements *flooding* in MANET, of course there are many other possible misbehaviors in our scenario. Here are some of them.

To begin with, MAD-MANETs protocols have been studied at the infrastructure level in order to counteract misbehavior of selfish nodes at the MAC and network layers. As for the *MAC layer*, selfish nodes may try to increase their bandwidth by decreasing their backoff time (or variations thereof). Paper studying such issues and proposing countermeasures are, for example [6,12,13,41,49,59]. As for the *Network layer*, a selfish node may decide not to forward one or more packets, thus saving energy. There are basically two approaches to encourage nodes to participate in the network operations: micropayments schemes and reputation mechanisms. In a *micropayment* schema honest nodes forwarding packets are remunerated with some suitable form of currency. Micropayments have been studied in [9, 10,55,61,62]. In a *reputation* system schema nodes that refuse to forward packets are punished by denying them service. Reputation systems are studied, for example, in [20,21]. It is worth noting that all works on cooperative (MAD) networks rest on game theory to model node *selfishness*. For a more complete discussion on these topics see [11]. Note that in our setting each node will have many applications running at the same time. For example, beside our SADD protocol, a node may run a VoIP protocol or a file transfer protocol. For this reason the protection mechanisms used at the network layer level cannot directly be used at the application layer level. In particular we cannot avoid advertisement flooding working at the network layer. Finally, *malicious nodes* may modify or even forge *transit traffic* [54]. For example, *malicious* nodes may modify messages when they are forwarding them. More in general, *malicious* nodes are willing to spend their own resources just to damage other nodes.

4. Bloom filter background

A *Bloom filter* (e.g. see [17]) of *size* m and *signature* k consists of a bit-vector B of size m and k suitably (e.g. see [17]) chosen hash functions mapping strings into B entries.

Two operations are possible on a Bloom filter: `query()` and `insert()`. Operation `insert(u)` stores a k-bit signature of message `u` in B. Operation `query(u)` returns `true` if the k-bit signature of message `u` is in B, `false` otherwise. If `query(u)` returns true we conclude that message `u` has been stored in B. Of course *false positives* are possible. That is, `query(u)` may return true even when message `u` has never been stored in B. However, by choosing suitable values for m and k the probability of getting a false positive can be made very small [17].

More specifically, for a filter of size m and signature k after γ insertions the probability of getting a false positive is approximately $p = (1-e^{-\frac{k\gamma}{m}})^k$. For example, if we plan for γ insertions, by taking $m = 10\ \gamma$ with $k = 5$ (our best choice from [17]) we get $p = 9.4 \times 10^{-3}$. Thus if we want to store say 10^3 messages with the above value of p we may use a bit-vector of size $m = 10\ \gamma = 10^4$, that is about 1250 bytes of RAM. As a result, even a small handheld device can easily recognize 1000 *recently forwarded* messages.

5. Protocol description

An overview for our protocol is in Section 1.3. A detailed presentation follows.

5.1. Communication environment

As for the *link layer*, wireless communication links between network nodes are implemented using the WiFi (IEEE 802.11b) protocol. As for the *transport layer*, nodes (processes) communicate using the UDP protocol. As for our protocol, MAD-SADD, it is an *application layer* protocol. Note that since MAD-SADD takes care of routing we do not use the routing protocols provided at the *network layer* level. As for the *network layer* we assume (see assumptions 2 and 3 of Section 2) that countermeasures are implemented to counteract network layer selfish misbehaviors.

5.2. Message header

MAD-SADD messages are organized into packets. A packet is organized as a record consisting of administrative fields (*header*) and a data field. The administrative fields are the following: 1. *time*: packet time stamp; 2. *source_address*: ID of the node (see assumption 4 of Section 2); 3. *destination_address*: ID of final destination node; 4. *packet_from_address*: ID of node from which the packet has been received; 5. *next_hop_address*: ID of node to which the packet will be forwarded; 6. *seq_number*: sequence number; 7. *type*: packet type (i.e. *short-description*, *query-unicast*, *data-info*, *data-info-ack*, *data*, *ack*, *no-route*).

5.3. Advertising phase: Broadcast

In our scenario, each node is eager to transmit its own advertisement (see assumption 1 of Section 2). In order to meet this requirement, each node periodically sends its own advertisement in broadcast to all the reachable neighbors (see Fig. 1(a)). This happens with a fixed *advertisement frequency* f, i.e. a node broadcasts its own advertisement every $1/f$ seconds.

The advertisement must fit into one packet. For this reason, we also call it *short description* (of the service). Moreover *short-description* is also the type of a packet containing a short description (i.e. an advertisement). A node receiving a short description will request (in unicast) the *full description* of the advertisement only if it is interested in the advertised service.

As an example, in Fig. 1(a) node a (*advertising node*) broadcasts its advertising, reaching the two nodes within its transmission range.

5.4. Advertising phase: Forwarding

When receiving a *short-description* packet, a node is required to forward it in broadcast to all the reachable neighbors (see Fig. 1(b)). Because of assumptions 1 and 3 of Section 2 we can assume that when requested a node will actually forward a packet, without modifying it. However, if all nodes forward in broadcast all the advertising they receive, there would be many collisions, resulting in poor overall performances. On the other hand, as discussed in Section 1.3, in this phase we cannot use sequence numbers as it is usually done in MANETs. We address this problem by forwarding only *new* (i.e. not *recently* received) *short-description* packets. In this way, we reduce the number of forward operations for each node, thus saving energy and decreasing the number of collisions. In order to decide if a newly arrived *short-description* packet m was already received in the past or is new, we proceed as follows. Each node maintains a cache `BF` with the *signatures* of the received short descriptions. The data structure used to implement `BF` is a Bloom filter (see Section 4). Thus, when node i (*interested node*) receives from node j a *short-description* packet m, node i checks whether (the signature of) m is

already in BF (m is old) or not (m is new). If m is in BF, then i checks if m was *recently* forwarded. More specifically, let f be the advertising frequency. If the difference between the time stamp of m and the time stamp of the last message from j is less than $1/f$, then m is discarded, otherwise it is forwarded. This means that j cannot send to i message m more than once within $1/f$ seconds. Of course j can send to i other messages within the same period of time. The above approach allows us to avoid advertisement *flooding*. Note that, since the Bloom filter stores advertisement signatures, we may have false positives, i.e. we may decide that a short description is old when it is indeed new. However, this is very unlikely to happen (see Section 4). As an example, in Fig. 1(b) the advertising is propagated through the network.

Remark: One may be tempted to simplify the above Bloom filter based schema by just saying that after having forwarded a packet a node has an *inhibition* time of $1/f$ seconds where it does not forward any packet. Unfortunately this approach does not work. In fact, if incoming packets are queued then advertisement flooding results in a buffer overflow attack. If incoming packets are not buffered then advertisements from a flooding node will have a greater chance of being forwarded thus making flooding quite interesting (almost needed if a node wants its advertisement to be actually broadcasted). We also note that the Ariadne protocol [30] takes into account flooding based DoS attacks (e.g. route request flooding) and proposes mechanisms to counteract such attacks. However the mechanisms proposed in [30] aim at counteracting DoS attacks stemming from malicious nodes forging route requests. In our case however flooding takes place in the broadcast phase (no routing needed) and stems from a perfectly legal node behavior: sending advertisements. Note also that we may use the mechanisms proposed in Ariadne [30] to secure the discovery (Section 5.5) and delivery (Sects. 5.6, 5.7) phases of our protocol.

5.5. Discovery phase

When node i receives a *new* advertisement message m (i.e. m is not in BF), it evaluates its interest for advertisement m using its profile. For example, this can be done by using a data mining algorithm (e.g. *cosine similarity* [27]) when messages are free format or exploiting the message format (e.g. as in [15, 22,28,33,35,36,51]). Of course any of the above approaches can be used within MAD-SADD. To carry out our simulations (Section 6) here we use a *free format* for profiles and advertisements, and *cosine similarity* to evaluate advertisements.

If node i deems m to be interesting, it discards any other incoming *short-description* packet and behaves as follows.

First of all, node i stores in a k-dimensional vector, say `undo-vector`, the entry values of BF for the k indexes computed by BF hash functions on argument m. Then i inserts the signature of m into BF. Finally, i starts a *unicast* communication with the advertiser of m, call it a. Namely, using a unicast transmission i asks a for the full description M of the advertisement m.

The Bloom filter BF is cleared when the γ-th message is added, where γ is the maximum number of insertions BF has been designed for. In this way, BF will contain only *recently* seen advertisements, thus allowing to forward old advertisements to newly arrived nodes. Routing for the discovery phase will be discussed in Section 5.9. Here, we simply remark that, once the intermediate nodes between i and a are selected, all such nodes will cooperate in forwarding the *query-unicast* packet from i to a (see assumption 3 of Section 2). As an example, in Fig. 1(c) node i sends a *query-unicast* packet to a.

5.6. Delivery phase: Communication setup

When an advertising node a receives a *query-unicast* packet q from an interested node i, q is immediately served, if a is not busy, otherwise q is stored in a queue `Q_queries`.

In order to serve a *query-unicast* packet coming from a node i, the first operation to carry out is the setup of the communication between i and a. This is done in the following way. Node a sends to i a *data-info* packet, containing the total length of M and the number of *data* packets which are needed to completely transmit M. Then, node i sends to a a *data-info-ack* message. All the intermediate nodes in the path between i and a simply forward these packets (by assumption 3 of Section 2, we can assume that the path is still active). Note that during this phase other queries unicast may be received by a. These queries are all enqueued in `Q_queries`.

When a receives the *data-info-ack* packet, a can start sending M. To this end, M is divided into segments (*data* packets). Each of these *data* packets is then stored in a queue `Q_segments`. Routing for the delivery phase will be discussed in Section 5.9.

5.7. Delivery phase: Sending the full description

Once queue `Q_segments` contains all the needed *data* packets, each one of them is sent to the requiring node i. Namely, for each *data* packet p in `Q_segments`, a sends p to i and waits for the *ack* for p from i. Once the *ack* packet is received, the next packet from `Q_segments` is considered.

When `Q_segments` becomes empty, `Q_queries` is checked again. If `Q_queries` is not empty, i.e. if there are pending unicast queries for a, then node a extracts a new unicast query from `Q_queries` and serves it as explained above. In this way, we force nodes to complete the transmission of full descriptions, before serving new unicast queries.

As for the *data-info* and *data-info-ack* packets, all the intermediate nodes in the path between i and a simply forward the *data* and *ack* packets.

As an example, in Fig. 1(d) node a sends to i a *data-info* packet, which is acknowledged by i with a *data-info-ack* packet. Finally, a sends the *data* packets to i. Each *data* packet is properly acknowledged by i.

At this point, the unicast communication is finished and node i can consider new incoming *short-description* packets.

From Section 2 follows that, as far as we are concerned, no selfish misbehavior takes place during a unicast communication. Thus, as usual, we can use sequence numbers to avoid forwarding many times the same *data* packets.

5.8. Handling link failures

Since nodes are mobile, links may fail. For example, a node may leave the network or run out of energy. Note that link failures may cause problems only during the unicast transmissions, that is during the discovery and delivery phases.

If a node in the transmission chain is not able to deliver a packet (either *data* or *ack*) to the next hop, it notifies this link failure to the previous hop with a *no-route* packet, which will be sent back in the chain to the source. As usual, a receiving node uses a *timeout* to detect transmission failures.

Furthermore, upon a failure, node i (i.e. the one that has received the interesting advertisement m) restores the previous status of the Bloom filter `BF` by using the values stored in the `undo-vector` during the discovery phase (see Section 5.5).

One may wonder why we need to restore the previous values of `BF` in case of communication failure. Suppose that we do not use the `undo-vector` of Section 5.5 and we simply store new advertisements in `BF`. Then a node may never get the *full description* of an advertisement it is interested in. Here is how. Node a broadcasts its advertisement m. Node i receives m, stores it in `BF` and, finding m interesting,

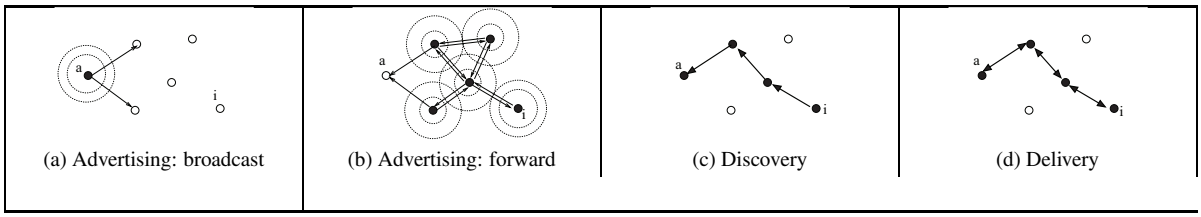

Fig. 1. An example of MAD-SADD execution.

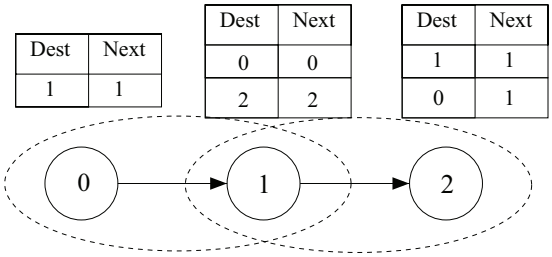

Fig. 2. Routing: advertising phase.

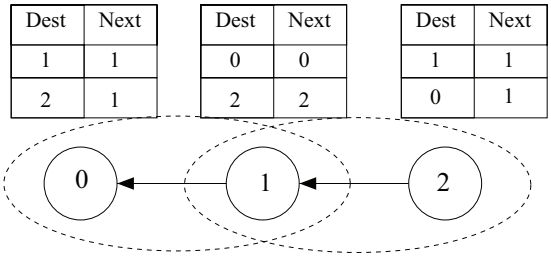

Fig. 3. Routing: discovery phase.

sends a *query-unicast* to a asking for a full description M of m. The unicast communication fails so i never gets M from a. After a while (depending on the advertisement frequency) node a will broadcast advertisement m again. Now i has m in `BF` so it will not check its interest for m and will just forward m. Thus, as long as m stays in `BF`, node i will never consider m any more, so missing the full description of an advertisement i is actually interested in. For this reason, upon a failure, node i must *erase* m from its `BF`. This is achieved by using the `undo-vector` to restore the previous values of `BF` entries modified by storing m in `BF`.

5.9. Routing protocol

While routing is not needed during broadcast communication, when unicast communication takes place we have to address the problem of reaching a specified destination.

Our main contribution concerns the mechanism to thwart node selfish behavior, thus avoiding "advertising flooding". Thus, as for routing, we may use any of the many routing protocols available (e.g. *AODV* [7] and *DSR* [18]). However, by exploiting the advertising and discovery (unicast) phases of MAD-SADD we use here a sort of "lightweight" AODV during the delivery (unicast) phase. In fact, we can piggyback routing information in MAD-SADD packets thus avoiding the overhead of additional routing messages.

Namely, our protocol extracts routing information during both the broadcast and the unicast phase. This is done by extracting the information contained in the headers of *short-description* and *query-unicast* packets in order to set up message routing. An example will clarify the matter.

Consider the situation shown in Fig. 2. Node 0 broadcasts its *short-description* packet, which can be received by node 1 only. Upon receiving the packet, node 1 updates its routing table adding a link to node 0. Then, node 1 broadcasts the same message. Since node 2 is reachable from node 1, node 2 can now receive the *short-description* packet and update its routing table as well. Note that Fig. 2 shows the routing tables after 2 has forwarded the advertisement too.

The routing tables of Fig. 2 are partial and do not contain enough information to establish a bidirectional unicast tunnel between 0 and 2. In fact, whenever node 2 queries node 0 for detailed data, the latter is not able to reroute the requested information towards node 2, since the routing table in 0 does not contain 2 as a possible destination.

This problem can be solved by considering also the information exchanged during the unicast phase. Namely, each node i receiving a *query-unicast* packet p extracts from p the source address s and the previous hop address h. Then, i adds to its routing table an entry with s as destination and h as next hop. For example, in Fig. 3 node 0 adds the last entry (destination $= 2$, next hop $= 1$) by extracting this information from the *query-unicast* packet forwarded by 1.

6. Simulation results

In order to evaluate MAD-SADD performances we implemented it within the ns-2 simulator [43,44]. Of course our goal is not to evaluate performances of the routing protocols. We just use a known one (AODV) which performance has been already widely studied [50].

Our goal here is to evaluate effectiveness of our approach in counteracting DoS attacks consisting of advertisement floods. This entails checking that indeed flooding does not take place (*safety*) and that legitimate messages get indeed transmitted (*liveness*).

We run the following sets of simulations.

1. *Bloom filter performance:* This set of simulations (Section 6.3) aims at evaluating Bloom filter effectiveness in our context.
2. *Advertisement frequency:* This set of simulations (Section 6.5) aims at estimating the best value of MAD-SADD *advertisement frequency*, that is how often an advertisement should be broadcasted. This is the most important MAD-SADD parameter.

6.1. IEEE 802.11b on ns-2 simulator

Our goal is to use MAD-SADD with WiFi (IEEE 802.11b) devices. Accordingly, we analyzed a few WiFi chipset and used their parameters to define ns-2 physical and MAC layer configurations. In order to allow exchange of data between nodes, we extended ns-2 classes with suitable methods allowing data exchange between the application layer and the network layer. In fact, ns-2 primitives *send()* and *receive()* only take as argument the number of bytes to be exchanged but do not directly support data exchange.

6.2. Simulation environment

Initially, we load each node with an advertisement and a profile that will be kept unchanged during all our simulation campaign. Advertisements and profiles have been chosen so as to have a reasonable overlap between them in order to trigger the delivery phase.

Nodes are positioned in a square region. Each node initial position is chosen at random with a uniform distribution. In the Bloom filter performance simulations (Section 6.3) nodes do not move, while in advertisement frequency simulations (Section 6.5) nodes may move with constant speed in a *waypoint* fashion. We simulate one hour of MAD-SADD execution (*simulation horizon*) in each simulation run. Each node initial energy (i.e. at the beginning of each simulation run) is 3000 J.

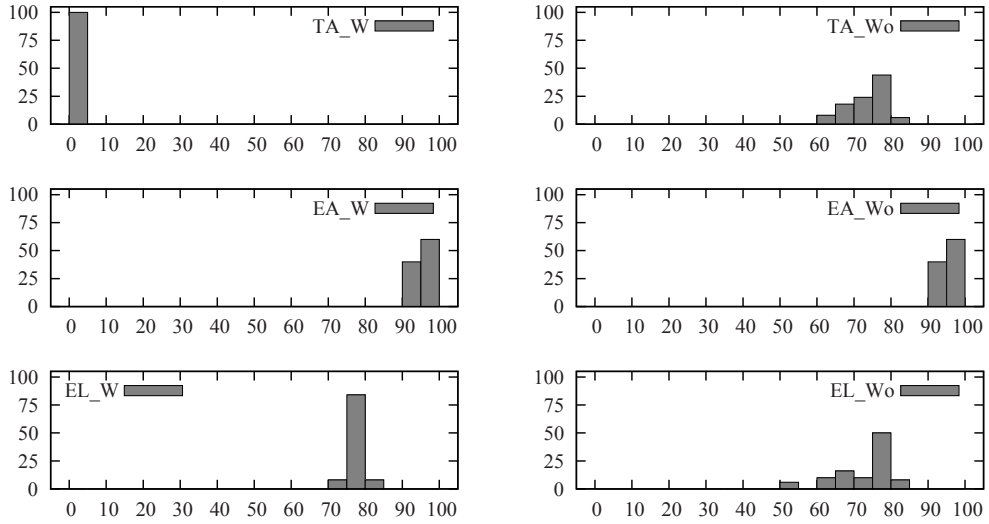

Fig. 4. Bloom filter performances on advertising and energy. The y axis represents percentages of (the total number of) nodes. The x axis represents: *total advestising percentage*, *effective advertising percentage*, *energy left percentage* respectively in the first, second, and third row.

The main settings for ns-2 simulation parameters are the following. The *Device Tx/Rx power* has been chosen so as to have a transmission range of 60 meters. The *Antenna type* has been set to OmniAntenna (omnidirectional). The *Radio propagation model* has been set to `TwoRayGround`. The *Packet length* has been set to 1 Kbyte. The *Full description length* has been set to 3 Kbytes, thus to send an advertisement full description we need 3 packets.

For all graphics in this Section the y axis represents percentages of (the total number of) nodes.

6.3. Bloom filter simulations

We compare MAD-SADD protocol performances with those of the MAD-SADD protocol *without* the Bloom filter (MAD-SADD-NO-BF). In other words, we obtain MAD-SADD-NO-BF from MAD-SADD by skipping MAD-SADD tests checking for *recently* received advertisement.

Here are the simulation parameters we used in our experiments in this section.

1. *Density:* We employ 80 nodes, which are located in a $300x300m^2$-wide area. Each node position is chosen uniformly at random within the given area.
2. *No mobility:* Nodes do not move from their initial position.

This is a reasonable setting, since node mobility only affects MAD-SADD ability to reach a node, while here we are only interested in studying Bloom filter performances within MAD-SADD.

Our results are in Figs 4 and 5. Figure 4 shows 6 graphics, divided in three rows and two columns. Graphics in the first (leftmost) column show simulation results obtained with MAD-SADD (i.e. *with* Bloom filter) while graphics on the second column (rightmost) show simulation results obtained with MAD-SADD-NO-BF (i.e. MAD-SADD *without* Bloom filter).

As for Fig. 4, the y axis represents percentages of nodes, while the x axis meaning is the following.

1. *First row:* (Labels *TA_W* and *TA_Wo*) The x axis represents the *total advertising percentage*, i.e. the percentage of received *short-description* packets with respect to the *total* number of sent *short-description* packets. Note that a node broadcasts its advertisement many times (depending on the

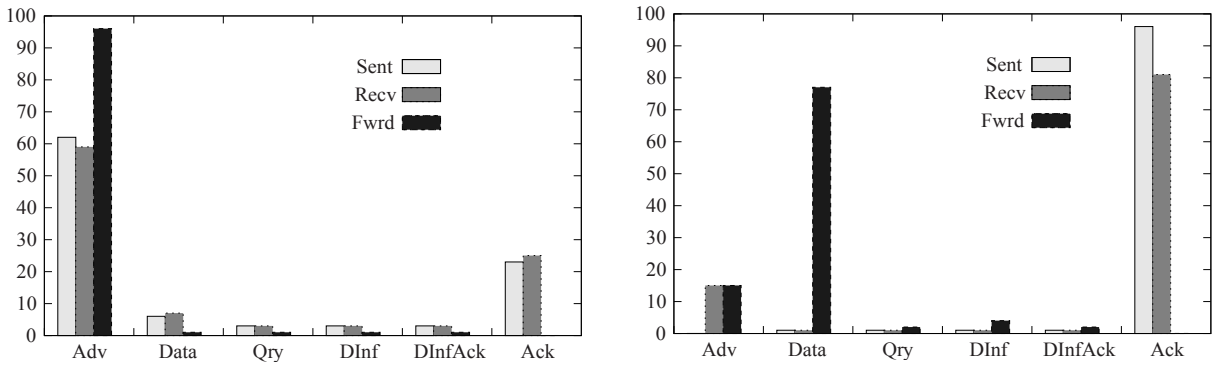

Fig. 5. MAD-SADD performances on packets with (left) and without (right) Bloom filter. The y axis shows percentages of (the total number of) nodes.

advertisement frequency). On the other hand, the same advertisement message can reach the same node many times because of multiple paths. Typically the number of sent *short-description* packets is much greater than the number of advertisement messages.

2. *Second row:* (Labels *EA_W* and *EA_Wo*) The x axis represents the *effective advertising percentage*, i.e. the percentage of received advertisement messages with respect to the sent advertisement messages. Note that here we are counting advertisement messages and not packets, thus each message is counted once (the first time it is sent/received).
3. *Third row:* (Labels *EL_W* and *EL_Wo*) The x axis represents the *energy left percentage*, i.e. the percentage of energy remaining in the devices at the end of the simulation run.

As an example, from the first row of Fig. 4 we can see that without Bloom filter (right column) about 80% of the total advertising is received by about 44% of the nodes.

Figure 5 shows the statistics on the type of exchanged packets *with* Bloom filter (left side) and *without* Bloom filter (right side). Namely, we detail how *sent*, *received* or *forwarded* packets are distributed (in percentage on the total number of packets) among the possible types of packets, i.e. *short-description* (column *Adv*), *data* (column *Data*), *query-unicast* (column *Qry*), *data-info* (column *DInfo*), *data-info-ack* (column *DInfAck*), and *ack* (column *Ack*).

6.4. Reading bloom filter simulations

The simulation results in Section 6.3 show the following.

1. *Total advertising:* From the first row of Fig. 4 we see that when the Bloom filter is used each node receives just a small percentage of the total number of sent *short-description* packets (which includes already seen, i.e. undesired, advertisement messages). In fact, graphics *TA_W* shows that 100% of the nodes receives at most 5% of the generated advertising. On the other hand, if the Bloom filter is turned off, most of the nodes receive most of the (undesired) advertising. For example graphics *TA_Wo* shows that 74% of the nodes receives at least 71% of the (undesired) advertising, and that 50% of the nodes receives at least 76% of the (undesired) advertising.
2. *Effective advertising:* From the second row of Fig. 4 we see that the Bloom filter is accurate enough so that no *desired* advertisement message is lost. In fact, graphics *EA_W* and *EA_Wo* are equal. This means that there is no difference between turning on and off the Bloom filter when considering only *new* advertisement messages. Namely, in both cases all the nodes receive at least 91% of the desired advertising.

3. *Energy left:* From the third row of Fig. 4 we see that Bloom filter usage allows MAD-SADD to save energy. In fact, graphics *EL_W* shows that by enabling the Bloom filter 100% of the nodes have at least 71% of their initial energy at the end of the simulation. On the other hand, if the Bloom filter is turned off, only 68% of the nodes achieve the same result.

One may wonder why using Bloom filter leads to energy saving, since Bloom filter usage has a significant computational cost on each node. The answer is that such higher computational cost incurred by each node is more than compensated by a reduction in the number of packets exchanged between nodes. This translates into a remarkable energy saving. More specifically from Fig. 5, we can see the following.

1. *Acks:* When the Bloom filter is not used (right side), nearly all of the sent packets and about 81% of the received packets are acks. On the other hand, only 23% of sent packets and 25% of received packets are acks when the Bloom filter is used (left side).
2. *Advertisements:* When the Bloom filter is used (left side), most of sent and received packets have type *short-description* (i.e. are advertisement messages), while only 1% of the sent packets (and 15% of the received ones) have type *short-description* when the Bloom filter is turned off.
3. *Forward:* When the Bloom filter is used, nearly all the forwarded packets have type *short-description*, while 77% of the forwarded packets have type *data* when the Bloom filter is turned off.

Shortly, Fig. 5 shows that using a Bloom filter most of the packets are advertisements (*short-description*) while the percentage of "administrative" packets (e.g. *ack*) and data packets is small. On the other hand, turning off the Bloom filter most of the packets are administrative (namely *ack*) and data packets.

Note in fact that each time a node receives an "interesting" advertisement it starts the discovery and delivery phases, that in turn trigger date exchange. Thus when Bloom filter is turned off the discovery and delivery phases are triggered many times for the very same "interesting" advertisement. This increases the (undesired) data traffic.

6.5. Advertisement frequency simulations

We present simulation results aiming at finding an effective value for MAD-SADD *advertisement frequency* in our scenario. Of course if we change the node density or the node deployment area the results will be different but nevertheless they can be obtained following the procedure outlined here.

First of all, note that, in order to save as much energy as possible, one may want to set the advertisement frequency as low as possible. However, since nodes are moving, in order to reach as many nodes as possible with its advertisement message, a node should set the advertisement frequency to a large enough value. Thus an advertisement frequency that is a reasonable trade-off between energy saving and node coverage has to be found.

To this end we proceed as follows. Our *candidate* frequencies are (in Hz): $f_1 = 1/30$, $f_2 = 1/60$, $f_3 = 1/120$, $f_4 = 1/180$.

For each f_i ($i = 1, 2, 3, 4$) we run three sets of simulations, each set representing a given scenario in a squared area: *low density scenario*, i.e. 35 nodes in a $300m^2$-wide area; *medium density scenario*, i.e. 60 nodes in a $300m^2$-wide area; and *high density scenario*, i.e. 60 nodes in a $100m^2$-wide area.

In all these scenarios, the initial position of each node is picked uniformly at random. However, since here we are interested in MAD-SADD nodes coverage, we have to consider moving nodes. To this end, at the beginning of each simulation run, each node i picks at random a time $t_i \in [0h, 1h]$, a (*waypoint*) position p_i (within the given area) and a speed $v_i \in [0.5m/s, 1.5m/s]$ (that is the typical speed for a

Table 1
Summary for the low density scenario

	Data sent	Data recvd	Collisions	Del perc
f_1	42	40	37,965	92.89
f_2	44	42	14,524	93.99
f_3	42	38	6,098	92.44
f_4	34	30	4,000	88.52

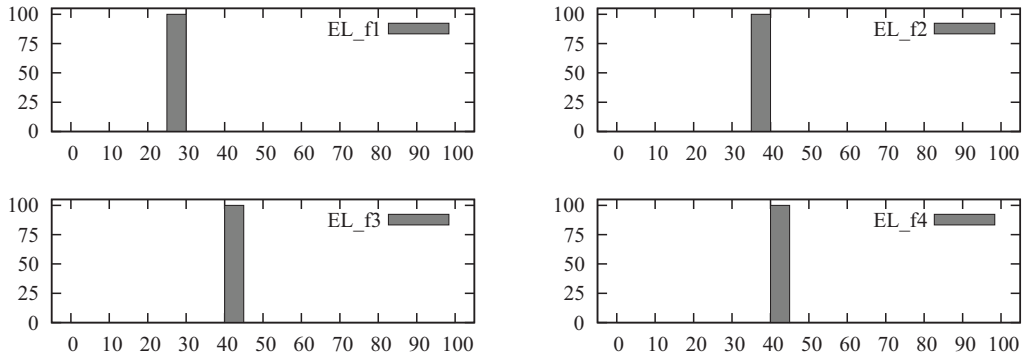

Fig. 6. MAD-SADD performances in the low density scenario. The y axis represents percentages of (the total number of) nodes; in the x axis we have *energy left percentage* (EL).

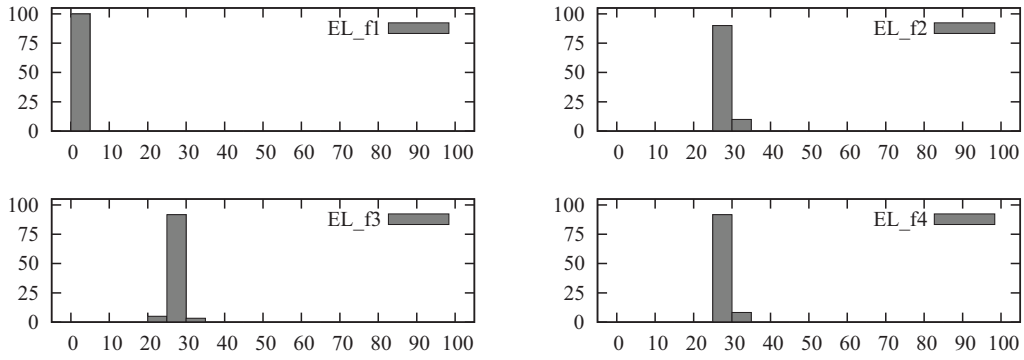

Fig. 7. MAD-SADD performances in the medium density scenario. The y axis represents percentages of (the total number of) nodes; in the x axis we have *energy left percentage* (EL).

walking person). Then, at time t_i from the beginning of the simulation, node i will begin moving towards p_i with speed v_i.

Our results are shown in Fig. 6 and Table 1 for the low density scenario; in Fig. 7 and Table 2 for the medium density scenario; in Fig. 8 and Table 3 for the high density scenario.

Figure 6 contains 4 graphics, which represent the *energy left percentage* for the four possible advertising frequencies.

Columns in Table 1 have the following meaning. Column *Data Sent* (resp. *Data Recvd*) shows the number of *data* packets sent (resp. received) during the simulations. Column *Collisions* shows the number of collisions detected. Finally, column *Del Perc* shows the fraction between the number of all received packets received and the number of all sent packets.

Table 2
Summary for the medium density scenario

	Data sent	Data recvd	Collisions	Del perc
f_1	180	168	1,599,865	80.47
f_2	806	767	624,019	70.83
f_3	757	706	591,936	58.96
f_4	747	688	599,373	59.09

Table 3
Summary for the high density scenario

	Data sent	Data recvd	Collisions	Del perc
f_1	2,041	1,790	15,982,439	80
f_2	2,233	2,120	6,515,836	86.12
f_3	2,294	2,171	2,718,257	87.03
f_4	2,396	2,249	1,596,387	87.23

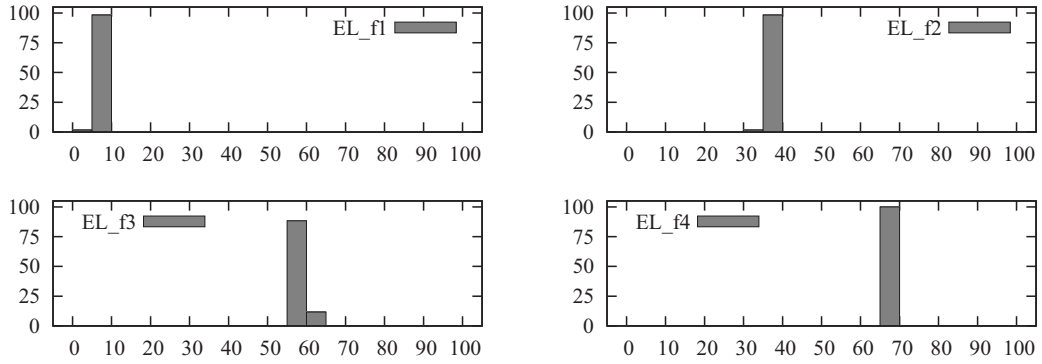

Fig. 8. MAD-SADD performances in the high density scenario. The y axis represents percentages of (the total number of) nodes; in the x axis we have *energy left percentage* (EL).

The meaning for the graphics of Figs 7 and 8 is the same as that for the graphics of Fig. 6. Analogously, the meaning for the columns of Tables 2 and 3 is the same as that for the columns of Table 1.

6.6. Reading advertisement frequency simulations

Using the simulation results of Section 6.5 we may suggest an effective value for the advertisement frequency.

First of all, note that we are interested in maximizing the packet delivery percentage and the energy left in each node, while minimizing the number of collisions. These are conflicting requirements, thus a trade-off has to be found.

With these targets, from Fig. 6 and Table 1 we can see that f_2 and f_3 are to be preferred for the *low density* scenario. In fact, f_1 leads to a high number of collisions, while all the nodes have no more than 30% of energy left at the end of the simulation runs. On the other hand, f_4 leads to a low packet delivery percentage. A similar reasoning may be done for the *medium density* scenario (Fig. 7 and Table 2), where f_2 turns out to be our best choice. Finally, in the *high density* scenario (Fig. 8 and Table 3), f_3 and f_4 turn out to be preferable to the other frequencies.

As for Table 2, one may be puzzled by the values in columns *Data Sent* and *Data Received* corresponding to row f_1. In fact, such values are much smaller than the other values in the same columns. The reason for this is that in the medium density scenario the nodes quickly run out of energy when using advertising frequency f_1. This can be seen from Fig. 7 which shows that for frequency f_1 (upper left graph) at the end of the simulation run each node has no more than 5% of its initial energy left whereas for frequencies f_2, f_3, f_4 nodes have much more energy left. For this reason the number of data packets exchanged in the f_1 scenario is much smaller than those of scenarios f_2, f_3, f_4.

Since f_3 is one of the best choices in 2 out of 3 scenarios (and leads to acceptable value for number of collisions and energy left in the remaining scenario), we finally choose f_3 as the best choice for MAD-SADD.

7. Conclusions

We presented a *Service Advertising, Discovery and Delivery* (SADD) protocol for *Multi Administrative Domain* MANETs. The main obstacle to overcome in designing SADD protocols for MAD MANETs is devising effective mechanisms to prevent selfish or malicious nodes from *flooding* the network with their advertisements. We have shown that by using *Bloom filters* it is possible to effectively counteract advertising floods. Our results with the ns-2 simulator show that our protocol keeps low the collision rate as well as the energy consumption (*safety*) while ensuring that each peer receives all messages broadcasted by other peers (*liveness*). Extending the proposed protocol to MAD-MANETs with malicious nodes able to modify or forge messages appears to be an interesting future work.

References

[1] I. Aad, J.P. Hubaux and E. Knightly, Impact of denial of service attacks on ad hoc networks, *IEEE Transactions on Networking* **16**(4) (2008), 791–802.

[2] A.S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.P. Martin and C. Porth, Bar fault tolerance for cooperative services, *SOSP '05* (2005), 45–58.

[3] K. Alekeish and P. Ezhilchelvan, Consensus in sparse, mobile ad hoc networks, *IEEE Transactions on Parallel and Distributed Systems* **23**(3) (2012), 467–474.

[4] J. Al-Karaki and A. Kamal, Routing techniques in wireless sensor networks: A survey, *IEEE Wireless Communications* **11**(6) (2004), 6–28.

[5] M.S. Al-Mazrouei and S. Narayanaswami, Mobile adhoc networks: A simulation based security evaluation and intrusion prevention, *ICITST '11* (2011), 308–313.

[6] E. Altman, R. El-Azouzi and T. Jimenez, Slotted aloha as a stochastic game with partial information, *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[7] AODV. http://tools.ietf.org/html/rfc3561 (2007).

[8] A. Broder and M. Mitzenmacher, Network applications of bloom filters: A survey, *Internet Mathematics* (2002), 636–646.

[9] L. Buttyán and J.P. Hubaux, Enforcing service availability in mobile ad-hoc wans, *MobiHoc '00*, IEEE Press (2000), 87–96.

[10] L. Buttyán and J.P. Hubaux, Stimulating cooperation in self-organizing mobile ad hoc networks, *Mob Netw Appl* **8**(5) (2003), 579–592.

[11] L. Buttyán and J.P. Hubaux, Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing, *Cambridge University Press*, 2007.

[12] M. Cagalj, Thwarting selfish and malicious behavior in wireless networks, *Ph.D. thesis*, Lausanne (2006).

[13] L. Cao and H. Zheng, Spectrum allocation in ad hoc networks via local bargaining, *IEEE SECON '05* (2005), 475–486.

[14] Z. Chen, H.T. Shen, Q. Xu and X. Zhou, Instant advertising in mobile peer-to-peer networks, *ICDE '09* (2009), 736–747.

[15] L. Cheng, Service advertisement and discovery in mobile ad hoc networks. *CSCW '02* (2002), 16–20.

[16] S. Cheshire, B. Aboba and E. Guttman, Dynamic configuration of IPv4 link-local addresses, *RFC 3927*, (2005) http://www.ietf.org/rfc/rfc3927.txt.
[17] P.C. Dillinger and P. Manolios, Bloom filters in probabilistic verification, *FMCAD*, *LNCS* **3312** (2004), 367–381.
[18] DSR. http://tools.ietf.org/html/rfc4728 (2007).
[19] A. Durresi, P. Zhang, M. Durresi and L. Barolli, Architecture for mobile heterogeneous multi domain networks, *Mob Inf Syst* **6**(1) (2010), 49–63.
[20] S. Eidenbenz, G. Resta and P. Santi, Commit: A sender-centric truthful and energy-efficient routing protocol for ad hoc networks with selfish nodes, *IPDPS '05: IEEE Intl. Parallel and Distributed Processing Symposium*, 2005.
[21] D. Figueiredo, M. Garetto and D. Towsley, Exploiting mobility in ad-hoc wireless networks with incentives, *Tech. Rep. 04-66*, University of Massachussetts, Computer Science, 2004.
[22] C. Frank and H. Karl, Consistency challenges of service discovery in mobile ad hoc networks, *MSWiM '04*, ACM Press, New York, NY, USA (2004), 105–114.
[23] P. Goering and G. Heijenk, Service discovery using Bloom filters, *Twelfth Annual Conference of the Advanced School for Computing and Imaging* (2006), 219–227.
[24] E. Guttman, C. Perkins and J. Kempf, Service templates and service: Schemes, RFC Editor, 1999.
[25] A.M. Hanashi, I. Awan and M. Woodward, Performance evaluation with different mobility models for dynamic probabilistic flooding in MANETs, *Mob Inf Syst* **5**(1) (2009), 65–80.
[26] J. Haillot and F. Guidec, A protocol for content-based communication in disconnected mobile ad hoc networks, *Mob Inf Syst* **6**(2) (2010), 123–154.
[27] D. Hand, H. Mannila and P. Smyth, Principles of Data Mining, *The MIT Press*, 2001.
[28] S. Helal, N. Desai, V. Verma and C. Lee, Konark – a service discovery and delivery protocol for ad-hoc networks, *WCNC 2003* **3** (2003), 2107–2113.
[29] Y.C. Hu, D.B. Johnson and A. Perrig, Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks, *In Ad Hoc Networks Journal* **1**(1), (2003), 175–192.
[30] Y.C. Hu, A. Perrig and D.B. Johnson, Ariadne: A secure on-demand routing protocol for ad hoc networks, *Wireless Networks* **11**(1–2), (2005), 21–38.
[31] IEEE: IEEE standard 802.11. IEEE (1999).
[32] X. Jin, Y. Zhang, Y. Pan and Y. Zhou, Zsbt, A novel algorithm for tracing dos attackers in manets, *EURASIP Journal on Wireless Communications and Networking* **2006** (2006), 1–9.
[33] M. Khambatti, K.D. Ryu and P. Dasgupta, Push-pull gossiping for information sharing in peer-to-peer communities, *PDPTA-03*, CSREA Press (2003), 1393–1399.
[34] Y.S. Kim, Y.S. Shim and K.H. Lee1, A cluster-based web service discovery in MANET environments, *Mob Inf Syst* **7**(4) (2011), 299–315.
[35] N. Klimin, W. Enkelmann, H. Karl and A. Wolisz, A hybrid approach for location-based service discovery in vehicular ad hoc networks, *WIT '04: Workshop on Intelligent Transportation*, 2004.
[36] U.C. Kozat and L. Tassiulas, Network layer support for service discovery in mobile ad hoc networks, *INFOCOM-03* **3** (2003), 1965–1975.
[37] E. Kulla, M. Hiyama, M. Ikeda, L. Barolli, V. Kolici and R. Miho, MANET performance for source and destination moving scenarios considering OLSR and AODV protocols, *Mob. Inf. Syst.* **6**(4) (2010), 325–339.
[38] H.C. Li, A. Clement, E.L. Wong, J. Napper, I. Roy, L. Alvisi and M. Dahlin, Bar gossip, *OSDI '06: USENIX Operating Systems Design and Implementation*, USENIX Association (2006), 191–204.
[39] A. Liu, H. Yu and L. Li, An energy-efficiency and collision-free mac protocol for wireless sensor networks, *Proc. of Vehicular Technology Conference 2005* **2** IEEE (2005), 1317–1322.
[40] Z. Li and H. Shen, Game-Theoretic Analysis of Cooperation Incentive Strategies in Mobile Ad Hoc Networks, *IEEE Transactions on Mobile Computing* **11**(8) (2012), 1287–1303.
[41] A. MacKenzie and S.B. Wicker, Stability of multipacket slotted aloha with selfish users and perfect information, *INFOCOM-03* **3** (2003), 1583–1590.
[42] H. Mousannif, I. Khalil and S. Olariu, Cooperation as a service in VANET: Implementation and simulation results, *Mob Inf Syst* **8**(2) (2012), 153–172.
[43] K. Fall and K. Varadhan, ns notes and documentation: http://www.monarch.cs.rice.edu/ftp/monarch/wireless-sim/nsDoc.pdf (2007).
[44] ns-2. http://www.isi.edu/nsnam/ns/ (2007).
[45] F. Palmieri and A. Castiglione, Condensation-Based Routing in Mobile Ad-Hoc Networks, *Mob Inf Syst* **8**(3) (2012), 199–211.
[46] F. Palmieri, U. Fiore and A. Castiglione, Automatic security assessment for next generation wireless mobile networks, *Mob Inf Syst* **7**(3) (2011), 217–239.
[47] P. Papadimitratos and Z.J. Haas, Secure data transmission in mobile ad hoc networks, *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security*, ACM (2003), 41–50.

[48] S. Parvin, F.K. Hussain and S. Ali, A methodology to counter DoS attacks in mobile IP communication, *Mob Inf Syst* **8**(2) (2012), 127–152.
[49] C. Peng, H. Zheng and B. Zhao, Utilization and fairness in spectrum assignment for opportunistic spectrum access, *MONET* **11**(4) (2006), 555–576.
[50] C.E. Perkins and E.M. Royer, Ad hoc on-demand distance vector routing. *2nd IEEE Workshop on Mobile Computing Systems and Applications* (1999), 90–100.
[51] O. Ratsimor, D. Chakraborty, A. Joshi and T. Finin, Allia: alliance-based service discovery for ad-hoc environments, *WMC '02*, ACM Press (2002), 1–9.
[52] F. Sailhan and V. Issarny, Scalable service discovery for manet, *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, IEEE Computer Society (2005), 235–244.
[53] G. Thomas, Capacity of the wireless packet collision channel without feedback. *IEEE Transactions on Information Theory* **46**(3) (2000), 1141–1144.
[54] D. Venugopal and G. Hu, Efficient signature based malware detection on mobile devices, *Mob Inf Syst* **4**(1) (2008), 33–49.
[55] W. Wang, X.Y. Li, S. Eidenbenz and Y. Wang, Ours: optimal unicast routing systems in non-cooperative wireless networks, *MobiCom '06*, ACM Press (2006), 402–413.
[56] D. Woelk, B. Haskell, J.L. Carter, R. Brice, Rusin and A.A. Helal, Any Time, Anywhere Computing: Mobile Computing Concepts and Technology, *Kluwer Academic Publishers*, Norwell, MA, USA (1999).
[57] D. Yi, A Novel Call Admission Control Routing Mechanism Using Bloom Filter in MANET, *NSWCTC 2009*, 2009, 675–678.
[58] P. Yi, Z. Dai, Y. Zhong and S. Zhang, Resisting flooding attacks in ad hoc networks, *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing*, vol. 2, IEEE Computer Society, (2005), 657–662.
[59] J. Zander, Jamming games in slotted aloha packet radio networks, *Proc. of Military Communications Conference '90* (1990), 830–834.
[60] Z. Zhao, H. Hu, G.J. Ahn and R. Wu, Risk-Aware Mitigation for MANET Routing Attacks, *IEEE Transactions on Dependable and Secure Computing* **9**(2) (2012), 250–260.
[61] S. Zhong, J. Chen and Y.R. Yang, Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks, *INFOCOM-03* **3** (2003), 1987–1997.
[62] S. Zhong, L.E. Li, Y.G. Liu and Y.R. Yang, On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: An integrated approach using game theoretical and cryptographic techniques, *MobiCom '05*, ACM Press (2005), 117–131.

Enrico Tronci is currently an Associate Professor with the Computer Science (CS) Department of Sapienza University of Rome (Italy). Previously he was: a researcher with the CS Department of the University of L'aquila (Italy), a Post-Doct at LIP (Laboratoire pour l'Informatique du Parallelisme) at the ENS (Ecole Normal Superior) of Lyon (France). He received his Ph.D degree from Carnegie Mellon University, Pittsburgh, USA and his Master degree in Electrical Engineering from Sapienza University of Rome. His current research interests comprise: Model checking algorithms for automatic verification and synthesis of reactive systems. He has served as conference chair, program committee member, reviewer in many International Journals and Conferences. He has authored more than 50 scientific papers on International Journals and Conferences. He has been recently involved in more than a dozen research projects sponsored by the European Community, European Space Agency, as well as private companies.

Alberto Finzi is Assistant Professor at DSF, Università degli Studi di Napoli "Federico II" (Italy). He received his Ph.D degree in Computer Engineering from Sapienza Università di Roma (Italy). His research interests include: V & V methods for autonomous systems, multi-agent systems, autonomous and adaptive systems, planning and scheduling systems. He has been recently involved with several research projects sponsored by the EC (European Community), NASA (National Aeronautics and Space Administration), ESA (European Space Agency), ASI (Italian Space Agency), FWF (Austrian Science Fund), MIUR (Italian Ministry for University and Research), and private industries.

Igor Melatti is currently Researcher at at the Computer Science Department of the Sapienza University of Rome. He obtained his Ph.D. in Computer Science and Applications from the University of L'Aquila in 2001, after having graduated in the same institution in 2005. He held Post-Doc positions at the School of Computing of the University of Utah (2005) and at the Computer Science Department of the Sapienza University of Rome (2006–2010). His main research interests comprise: Formal methods, Automatic synthesis of reactive programs from formal specifications, Hybrid systems, Automatic verification algorithms, Model checking, Software Verification.

Federico Meli holds a Post-Doc position at the Computer Science Department of Sapienza University of Rome (Italy). In 2009, under the supervision of Prof. Enrico Tronci, he received his Ph.D degree from Sapienza University of Rome. His current research interests comprise: formal methods, automatic verification algorithms, model checking, hybrid systems verification, automatic synthesis of control software from formal specifications, automatic verification of Nash Equilibria in multi administrative distributed systems.