Hypergossiping: A Generalized Broadcast Strategy for Mobile Ad Hoc Networks

Abdelmajid Khelil, Pedro José Marrón, Christian Becker, Kurt Rothermel

Universität Stuttgart, Institute for Parallel and Distributed Systems (IPVS), Universitätstr. 38 70569 Stuttgart, Germany

Abstract

Broadcasting is a commonly used communication primitive needed by many applications and protocols in mobile ad hoc networks (MANET). Unfortunately, most broadcast solutions are tailored to one class of MANETs with respect to node density and node mobility and are unlikely to operate well in other classes. In this paper, we introduce hypergossiping, a novel adaptive broadcast algorithm that combines two strategies. Hypergossiping uses adaptive gossiping to efficiently distribute messages within single network partitions and implements an efficient heuristic to distribute them across partitions. Simulation results in ns-2 show that hypergossiping operates well for a broad range of MANETs with respect to node densities, mobility levels and network loads.

Key words: MANET, adaptive broadcast, network partitioning

1 Introduction

Mobile ad hoc networks (MANETs) are networks formed on-the-fly by mobile nodes equipped with short range communication capabilities. Such networks are suitable for scenarios where an infrastructure is unavailable and communication must be deployed quickly, e.g. in disaster-rescue or military scenarios.

Email addresses: khelil@informatik.uni-stuttgart.de (Abdelmajid Khelil), marron@informatik.uni-stuttgart.de (Pedro José Marrón), becker@informatik.uni-stuttgart.de (Christian Becker), rothermel@informatik.uni-stuttgart.de (Kurt Rothermel).

MANETs also suit well for scenarios where infrastructure is costly. For example an infrastructure to monitor road traffic jams and to warn drivers may be expensive, so authors in [1] suggest to exploit ad hoc communication to form vehicle ad hoc networks and to aggregate speed information to recognize traffic jams.

Broadcasting is a common communication mechanism in MANETs. It is frequently deployed for news spreading (such as alarms and announcements), for resource discovery and advertisement (such as topology discovery and maintenance [2]), and for sensor data dissemination (such as data aggregation [1] and consistency update propagation [3]). Some broadcast applications require timeliness such as route discovery for on-demand routing protocols. Other broadcast applications tolerate higher delays in range of seconds, minutes or even hours depending on the semantic of data. Examples of these delay-tolerant applications are spreading of announcements (e.g. usenet-on-the-fly [4]) or propagation of updates that do not change frequently (e.g. room temperature).

Flooding is a common approach to realize broadcasting in MANETs because of its topology independence. In flooding-based approaches nodes forward a received message to all their neighbors. Subsequently, all nodes within the network should receive the message.

But flooding exhibits some serious problems. At the two extremes we can consider dense MANETs and sparse MANETs with respect to the *node density*, i.e. the number of nodes operating in a given area. While dense MANETs encounter so-called *broadcast storms* [5], where collisions on the Media Access Control (MAC) layer extinguish broadcast messages, sparse MANETs are challenged by frequent *network partitioning* [6], where messages do not reach every node in one flooding round. Two common strategies can be applied to conquer these extreme cases. First, selective strategies, e.g. gossiping [7], cause only a subset of nodes to forward a message reducing the probability of broadcast storms. Second, selective repetition of broadcasts, e.g. hyperflooding [8–10], can be used to overcome network partitions.

Most broadcasting techniques are unfortunately tailored to one class of MANETs and are likely not to operate well in other classes. Our main objective is to provide an adaptive broadcast algorithm for a wide range of MANET operation conditions. The main contribution of this paper is hypergossiping, a novel generalized broadcast mechanism that combines two strategies and provides a configuration depending on the local density of a node, reflected by the number of its neighbors. Using simulation results we show that hypergossiping can be deployed in a wide spectrum of MANETs with respect to node densities, mobility levels and network loads.

The remainder of this paper is organized as follows. The next section describes the system model and the requirements on a generalized broadcast strategy for MANETs. In Section 3 we discuss the related work. Section 4 introduces our generalized broadcast strategy, i.e. hypergossiping. In Section 5 we first define the simulation model and the evaluation metrics, then we calibrate the parameters of hypergossiping, evaluate it and compare it to related work. Section 6 summarizes the paper and gives an overview of ongoing and future work.

2 Preliminaries

This section briefly presents the underlying system model of our approach. Based on the characteristics of the system model we derive some important requirements on our generalized broadcast algorithm.

2.1 System Model

In this work, we consider MANETs that are formed by mobile nodes of similar communication capabilities (communication range and bandwidth). We assume nodes have no knowledge about their position or speed. The MANET may show very heterogeneous spatial distribution of nodes, from locally very sparse to very dense, and very heterogeneous node mobility pattern, from low mobile to highly mobile. We assume that devices do not change their trajectories for communication purposes like in [11].

Broadcast data has typically a temporal and spatial relevance [12]. Broadcast algorithms have to consider this spatio-temporal relevance while broadcasting. In this paper, we consider only the temporal relevance of data and assume that information becomes irrelevant after a certain period of time, i.e. its *lifetime*. Lifetime is application dependent and may be in the range of seconds, minutes, or even hours.

2.2 Requirements

Based on the system model we present some important requirements on our generalized hypergossiping algorithm.

Because node density heavily influences the performance of broadcasting, and MANETs may show a wide range of node densities, the first requirement on a generalized broadcast strategy for MANETs is to adapt to the density of the network, in order to reduce broadcast storms and overcome network partitioning. Since global state in MANETs is hard to obtain and spatial distribution of nodes may change continuously, the second requirement on such a strategy is that nodes independently adapt to local MANET characteristics. Finally, different instances of the adaptive broadcast strategy have to interoperate in order to deliver messages through the network where different instances are present due to heterogeneity of density.

3 Related Work

The design of broadcast algorithms is a fundamental problem in MANETs and several broadcast protocols have been proposed in the literature. In [13], [14] and [15] the authors provide three comparative studies for the existing broadcasting techniques. [14] classifies broadcasting schemes into heuristic-based and topology-based. [13] subclassifies heuristic-based class into probability-based and area-based. We categorize all these protocols into adaptive and non-adaptive protocols.

3.1 Non-Adaptive Broadcast Algorithms

Non-adaptive heuristic-based protocols use heuristics with predefined fixed parameters to reduce broadcast storms. They do not adapt to the time-varying MANET situations that show quite different levels of broadcast storms. Examples of non-adaptive probability-based schemes are gossiping [5,7] and counter-based [5]. Examples of non-adaptive area-based schemes are location-based [5] and distance-based schemes [5].

Non-adaptive topology-based protocols (e.g. Multipoint Relaying Broadcasting [16], Connected Dominating Set Based [17, 18], Minimum Forwarding Set Based [19], deterministic broadcast [20], generic self-pruning [21], LENWB [22], and SBA [23]) require an accurate topology information which is hard to acquire in highly mobile environments and due to collisions. That is why these protocols perform poorly in terms of delivery ratio for highly mobile scenarios [13] [15] or highly congested ones.

The common drawback of all these broadcasting techniques is that they are developed for unpartitionable MANETs, and subsequently break in partitioned ones. Summarizing we conclude that all these schemes are optimized for specific scenarios and do not support a broader range of MANET situations.

In order to suit non-adaptive broadcast schemes to a broader range of operation conditions, some of them have been adapted to local MANET characteristics. The basic idea of adaptive topology-based approaches is to better manage node mobility for the purpose to avoid stale topology information [24, 25]. Adaptive heuristic-based protocols however adapt the heruristic to the number of neighbors [25, 26].

In [24] authors proposed to use two different communication ranges for topology management and for data transmission. They recommend to select a shorter range for topology management and to adapt the difference between the two ranges to node mobility, which requires speed information. In [25] the authors proposed one adaptive topology-based scheme, called neighbor-coverage scheme (NC). The authors adapted the NC scheme by adjusting dynamically the HELLO interval to node mobility reflected by neighborhood variation, so that the needed 2-hop topology information gets more accurate. Despite these optimizations, the adaptive topology-based schemes still have the main drawback that neighborhood information may be inaccurate in congested networks.

In [25] the authors also proposed two adaptive heuristic-based schemes, called adaptive counter-based (ACB) and adaptive location-based (ALB). Using a simulation-based approach the authors derived the best appropriate counter-threshold and coverage-threshold for ACB respectively ALB as a function of the number of neighbors. The authors showed that these adaptive schemes outperform the non-adaptive schemes and recommend ACB if location information is unavailable and simplicity is required. We will compare our strategy to ACB. [26] introduced the density-aware probabilistic flooding. Nodes use the following forward probability: $p = min\{1, 11/n\}$, where n is the number of neighbors. We will also compare our strategy to this scheme.

Although the above adaptive schemes support a broad range of dense MANETs, they still show poor delivery ratio in partitioned ones.

The first step towards a single solution for all MANET situations was an integrated scheme presented in [9, 10]. We refer to this scheme as integrated flooding (IF). Nodes switch at run-time between three flooding schemes, namely, plain flooding, scoped flooding, and hyperflooding. Authors recognize mobility as main cause of broadcast partitioning [27] and switch between these schemes according to the relative node mobility [9]. To this end, nodes include velocity information (speed and direction) in HELLO beacons. If a node has a current value of relative velocity to its neighbors higher than a high_threshold the node switches to hyperflooding mode. If the relative velocity is below low_threshold, scoped flooding is used. Otherwise, the node switches to plain flooding. Alternatively the same authors suggest in [10] that IF switches between the three schemes based on network load. Authors use MAC layer collisions as an indicator of network load. If a node has a current number of collisions higher than a high_threshold the node switches to scoped flooding mode. If the number of collisions is lower than low_threshold, hyperflooding is used. Otherwise, the node switches to plain flooding. The authors however do not mention how to combine both switching criteria. Furthermore these switching criteria may lead to opposite decisions. For example, in low mobile networks with low traffic, the relative velocity based switching would install scoped flooding, but the network load based switching would install hyper-flooding. For these reasons and because we mainly focus on wide ranges of network conditions concerning node density and node mobility and less concerning network load, we consider in this paper only the relative velocity based switching criteria for IF.

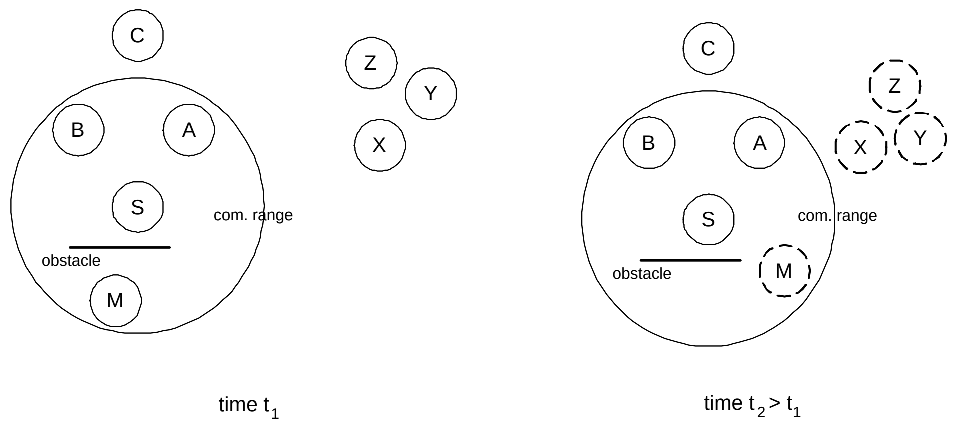
To the best of our knowledge, IF is the single existing adaptive MANET broadcast protocol that considers both connected and partitionable networks. Unfortunately, IF shows the following three drawbacks. First, hyperflooding deploys a very simple broadcast repetition strategy, i.e. on each discovery of a new neighbor, all cached packets are re-transmitted. If the buffering time of packets is high, this strategy leads to a high number of useless but costly broadcast repetitions in highly mobile networks. If the buffering time is low, the strategy shows a poor delivery ratio in low mobile and partitioned networks. Second, scoped flooding uses a predefined forward threshold, which makes IF less efficient than the above adaptive schemes in highly dense scenarios. Third, relying on velocity information presents a strong limitation of the deployment of IF. In Subsection 5.6, we compare our solution to IF.

4 Hypergossiping (HG)

In this section, we first state the problem using a motivation scenario, and then present our generalized solution fulfilling the requirements above. Hypergossiping provides an adaptive broadcast strategy combining a selective flooding strategy, so as to reduce broadcast storms, and a repetition strategy, in order to overcome network partitioning. In contrast to IF, which adapts to node mobility, hypergossiping uses the local node density as the main criteria for adaptation.

In the following we present a motivation scenario and mention the problems that our approach has to deal with. We consider the pedestrian scenario in Fig. 1, where a source node S broadcasts data, which we assume to remain relevant after time t_2.

At time t_1 we observe three partitions {S,A,B,C}, {X,Y,Z} and {M}. The main reason for this network partitioning is the lack of a (multihop) MAC-link between the source node and some destination nodes. This happens if the spatial distance between nodes is larger than the communication range or if obstacles are situated between nodes and prohibit them to communicate. Now we assume that a heuristic-based or a topology-based scheme is implemented for broadcasting. If S initiates a broadcast at time t_1, it sends the packet to all its neighbors using MAC broadcast. Receivers of this packet or a subset of them (e.g. A) may MAC broadcast this packet to their neighbors. We say node A *forwards* the packet. By this way the broadcast reaches nodes A, B and C but does not reach the nodes X, Y, Z, and M. At time t_2 the partitions {X,Y,Z} and {M} join the partition {S,A,B,C}, and the MANET becomes connected. If node A has buffered some still-relevant messages, A should share them with X. If node A repeats the forwarding of buffered messages, we say A *rebroadcasts* these messages. Similarly S has to rebroadcast its buffered messages to M. If node X relays the received messages to its neighbors Z and Y, we say also that X *rebroadcasts* these messages.



(a) MANET is partitioned (b) MANET is connected

Figure 1. Problem statement

Besides network partitioning, broadcast storms may stop the broadcast to continue progressing. For example, if S and C start to transmit simultaneously, collisions may happen at A and B. In this case, the broadcast of S extinguishes at the source and S should rebroadcast the message.

Due to the lack of a global view on the MANET, it is challenging to detect partition joins and to rebroadcast the appropriate messages. In the following we present our novel approach.

4.2 Approach

In general we can consider a MANET as a set of partitions, which may join or split over time. Thus, we decide to superpose the following two strategies to realize hypergossiping. The first strategy allows an efficient broadcasting within a single partition of the MANET. We refer to this strategy as "intra-partition forwarding" (Fig. 2 a)). The second strategy permits an efficient broadcast repetition on partition joins. We call this strategy "broadcast repetition". To this end nodes have to buffer messages during their lifetime and have to rebroadcast these messages or a subset of them on partition joins. After broadcast repetition the first strategy can continue to distribute the message to the joining nodes (Fig. 2 b)). Depending on the mobility of nodes, the node spatial distribution and the lifetime value, messages will succeed to other partitions or not.

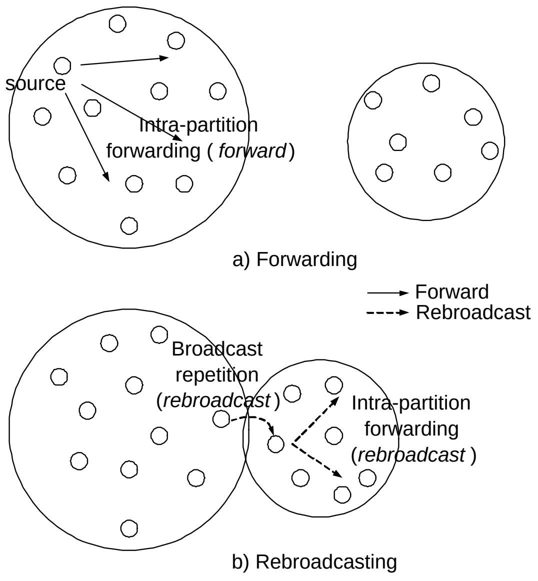

source
Intra-partition forwarding (*forward*)
a) Forwarding
Forward
Rebroadcast
Broadcast repetition (*rebroadcast*)
Intra-partition forwarding (*rebroadcast*)
b) Rebroadcasting

Figure 2. Approach

In our approach, we assume that each node stores the list of IDs of messages received or originated with their remaining lifetime in a so-called *broadcast_table*. Thus nodes are able to decide, whether a received copy of a given message is the first one. Nodes continuously decrement the lifetimes of received packets. Nodes purge entries from the broadcast_table and possibly from the buffer, when the correspondent lifetime expires. When a message is forwarded or rebroadcasted, the remaining lifetime is included in the message.

4.3 Intra-Partition Forwarding

The analysis of the broadcast storm problem [2] suggests gossiping or probabilistic flooding, thus we chose gossiping for intra-partition forwarding.

On receiving the first copy of a given message, gossiping forwards or rebroadcasts the message, with a probability p, to all nodes in the receiver's communication range. In order to reduce broadcast storms while gossiping, we follow a couple of strategies. First, nodes delay each intra-partition forwarding for a random time between 0 and *fDelay*, which reduces collisions. Second, we adapt the gossiping probability p to the node's current number of neighbors, which reduces forward redundancy, contention, and collisions. For this purpose nodes acquire the number of their neighbors by means of periodic HELLO beaconing.

4.4 Broadcast Repetition

The common strategy to overcome partitioning is the repetition of forwarding, i.e. rebroadcasting. For this purpose nodes need two mechanisms; one to detect *when* to rebroadcast and a second to decide *what* to rebroadcast. In the following we introduce our novel partition detection heuristic and rebroadcasting protocol.

4.4.1 Partition Detection

We let nodes share with their neighbors the IDs of recently received or locally originated packets. We call this list "Last Broadcast Received" or *LBR* (Fig. 3). The rational behind this is that two neighboring nodes that belong to the same partition should have received the same broadcasts that had taken place in this partition. By this means nodes are able to conclude whether they are populating the same partition. If a node receives an LBR that "sufficiently" differs from its own LBR, the node can conclude with a certain confidence that it is joining a new partition. We denote the maximum allowed size of an LBR by *maxLBRlength*. Nodes trigger rebroadcasting, only if the overlap between received LBR and own LBR does not exceed a given percentage of the own LBR. We denote this percentage threshold by "intersection threshold" (*IS_threshold*). In order to provide an accurate detection of partition joins, maxLBRlength and IS_threshold have to be dimensioned appropriately. In Subsection 5.4 we show how we calibrate maxLBRlength and IS_threshold.

The above strategy is suitable to detect both causes of broadcast interruption: Network partitioning and broadcast storms. First, if two partitions say P1 and P2 join, some nodes of partition P1 will receive LBRs from other nodes belonging to the formerly partition P2. In this way nodes are able to detect the join event. Second, if a broadcast stops to progress within a partition due to collision or contention, nodes that received the broadcast may detect this on receiving the LBR of one neighbor that has not yet received the packet.

Our strategy is also suitable for MANETs, where nodes may disappear and appear due to put-on-off or reboot. These nodes miss broadcasts taking place while they are unavailable. If a node appears, its LBR-list is empty. Therefore, a neighboring node, whose LBR list is empty, is able to detect this kind of partitioning.

In order to save bandwidth, nodes exploit the existing HELLO beaconing to share their LBRs. Exchanging the LBRs is only necessary if a new neighbor is detected. Thus we do not include the LBR in each HELLO beacon but only in that beacon that just follows the discovery of a new neighbor. This delays the broadcast repetition until the next discovery of a new neighbor, in case of broadcast interruptions caused by broadcast storms.

4.4.2 Rebroadcasting

To allow rebroadcasting nodes should buffer messages that need to be rebroadcasted. If not otherwise stated, we assume that nodes buffer all received and originated messages during their lifetimes, i.e. $m = n$ in Fig. 3.

A node triggers rebroadcasting by MAC-broadcasting a list of the broadcast IDs the node has received yet. We call this list "Broadcast Received" or short *BR* (Fig. 3). Thus neighbors know which packets the sender has already received and can select from buffer the packets that missed this sender. On receiving these new packets the sender gossips them, so they can reach all joining nodes. To increase rebroadcasting efficiency nodes do not rebroadcast immediately upon the reception of BR list(s), but schedule the rebroadcasting for a random time between 0 and *rDelay*. Nodes cancel rebroadcasting, if one neighbor starts to rebroadcast before the scheduled time. To reduce the probability of collisions nodes do not rebroadcast all packets at once but wait a random time between 0 and fDelay before rebroadcasting the next packet.

The pseudo-code description for hypergossiping is given by Algorithm 1. We denote by *card(X)* a function that returns the number of elements of set X, and by *random(x)* a function that returns a random float value $\in [0, x]$.

Algorithm 1 Hypergossiping (HG)

```
 1: Var: p, fDelay, IS_threshold, maxLBRlength, lifetime, rDelay
 2: List: myLBR, myBR, broadcast_table
              ________On receiving a message (msg) M ________
    • if M is DATA do gossiping(p):
 3: if M is received for the first time then
 4:    deliver M
 5:    insert {M.ID, remaining lifetime} to broadcast_table
 6:    insert a copy of M to buffer
 7:    if $myLBR.length < maxLBRlength$ then
 8:       insert {M.ID} to myLBR
 9:    else
10:       use FIFO to insert M.ID to myLBR
11:    end if
12:    if $random(1.0) \leq p$ then
13:       wait (random(fDelay))
14:       send M to all neighbors
15:    end if
16: else
17:    discard M
18: end if
    • if M is HELLO with LBR do partition detection:
19: $is \Leftarrow card(myLBR \cap recvLBR)/card(myLBR)$
20: if $is \leq IS\_threshold$ then
21:    send BR to neighbors
22: end if
    • if M is HELLO with BR do rebroadcasting:
23: set $timeout \Leftarrow random(rDelay)$
24: On receiving a DATA msg with $ID \in (myBR - recvBR)$ before timeout:
25: $exit()$
26: On timeout:
27: for all buffered msg M with $ID \in (myBR - recvBR)$ do
28:    $wait(random(fDelay))$
29:    rebroadcast M
30: end for
              ________On discovering a new neighbor ________
31: insert myLBR to next HELLO beacon
              ________On expiration of lifetime of msg M ________
32: if $M \in buffer$ then
33:    delete M from buffer
34: end if
35: if $M.ID \in myLBR$ then
36:    delete M.ID from myLBR
37: end if
38: delete the entry {M.ID, remaining lifetime} from broadcast_table
```

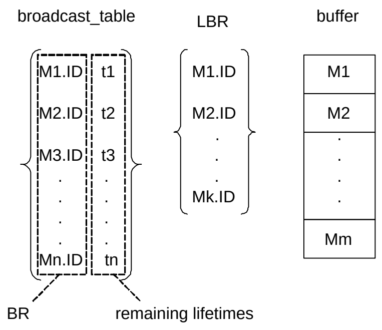


Figure 3. Definition of BR, LBR and buffer

5 Performance Evaluation

In this section, we introduce the simulation model and define our performance evaluation metrics. Afterwards, we study the performance of hypergossiping and compare it to related work.

5.1 Simulation Model

For evaluation we use the network simulator ns-2 [28]. The implementation of the physical and the MAC layer relies on the IEEE 802.11 standard. We use for the physical layer the TwoRayGround propagation model. We generate N mobile nodes in a 1000mx1000m field, where these nodes move according to the random waypoint mobility model. We set min speed equal to max speed in order to prevent the speed decay effect described in [29]. Table 1 summarizes the simulation parameters of our experiments, which show a wide range of node densities and node speeds.

As mentioned above, nodes get neighborhood information using HELLO beaconing. For all simulations in this work we use a random beaconing period between 0.75s and 1.25s. A neighbor is removed from neighbor list, if during 2s no beacon is received from this neighbor.

We use the following communication load model: At the beginning of the simulation s from N nodes initiate broadcasting at a random time between 1 and 3 s, and continue to send packets with a constant packet rate. This load model is suitable for the different application scenarios mentioned in the motivation, where data sources may send updates with a frequency that ranges from one update every hour to one update every second. A room-temperature sensor may trigger an update of the room's temperature every hour, since the temperature normally does not vary so frequently in rooms. But a vehicle has to trigger an update of its speed more frequently in order to allow jam recognition on the highway [1]. That is why we assume that packet rates

Parameters	Value(s)
Simulation area	1000m x 1000m
Number of nodes	N in {50, 100, 200, 300, 500}
Com. range	R = 100m
Bandwidth	r = 1 Mbit/s
Data packet size	280 bytes
Movement pattern - Max(=min) speed - Pause	Random Waypoint - v in {0, 1, 3, 5, 12.5, 20, 30} m/s - Uniform betw. 0 and 2s
fDelay	10ms
rDelay	10ms
Lifetime	[5 .. 1800] s
Packet rate	[0.001 .. 1] packet/s

Table 1
Simulation parameters

ranging from 0.001 packet/s to 1 packet/s cover a wide range of our operation scenarios. We use a fixed lifetime value during a simulation, i.e. all senders use the same lifetime for all packets they generate.

For the performance evaluation we consider the first s packets generated, the following packets generate a background traffic. Simulations stop some seconds after the lifetimes of the first s packets expire. For the same simulation scenario we ran 10 passes with 10 different movement traces and considered the average.

5.2 Evaluation Metrics

For the evaluation of broadcast protocols the following metrics are typically used:

- *REachability (RE)*: the ratio of mobile nodes receiving the packet to the total number of mobile nodes. This metric measures the delivery reliability of the broadcast algorithm.
- *MNF(R)*: Mean Number of Forwards (and Rebroadcasts) per node and packet. MNF(R) in combination with RE is an efficiency metric of the broadcast algorithm.
- *Delay*: Average end-to-end delay over all receivers.

Metric	Symbol	Value
Gossiping		
REachability	G_RE	$= \frac{card(R(G))}{N}$
Mean Number of Forwards	MNF	$= \frac{card(Forwd)}{N}$
Average end-to-end delay over all receivers	delay	$= \frac{1}{card(R(G))} \sum_{i \in R(G)} (t_i - t_s)$
Hypergossiping		
REachability	HG_RE	$= \frac{card(R(HG))}{N}$
Mean Number of Forwards and Rebroadcasts	MNFR	$= \frac{card(Reb)+card(Forwd)}{N}$
Average end-to-end delay over all receivers	delay	$= \frac{1}{card(R(HG))} * \sum_{i \in R(HG)} (t_i - t_s)$
Rebroadcasting gain	gain	$= \frac{card(R(H))}{card(Reb)}$

Table 2
Evaluation metrics

In Table 2 we illustrate the above metrics for both gossiping and hypergossiping. We denote by t_s the origination time of the packet and by t_i the arrival time of the packet at node i. With respect to a given broadcast packet we define the following five sets of nodes:

(1) *Forwd*: Nodes that forward the packet.
(2) *Reb*: Nodes that rebroadcast the packet.
(3) *R(G)*: Nodes receiving the packet during the first round of the broadcast, i.e. only using forwarding and without any rebroadcasting.
(4) *R(H)*: Nodes reached by means of rebroadcasting.
(5) *R(HG)*: Nodes reached by HG, i.e. by means of either forwarding or rebroadcasting. This results in $R(HG) = R(H) + R(G)$.

Gain is the mean number of additionally reached nodes per rebroadcast. It presents a suitable efficiency metric for broadcast repetition strategies.

5.3 Adaptation of Gossiping

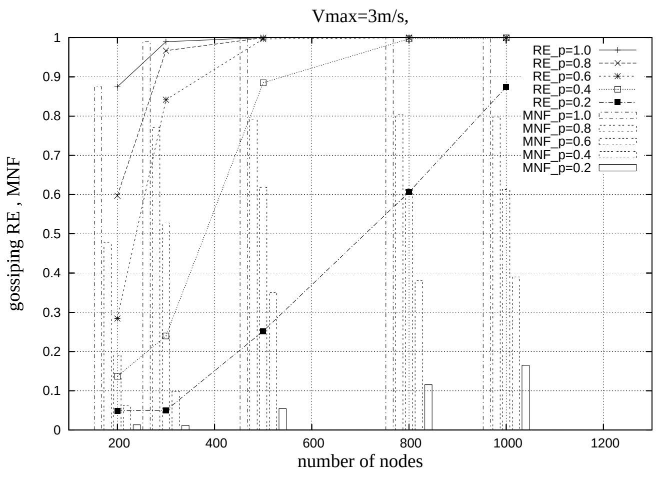
In this section, we adapt gossiping to local node density by determining the appropriate gossiping probability as a function of the number of neighbors. For this study, we use the same parameters as given in Table 1 with $v = 3m/s$, and one sender ($s = 1$) that sends one single packet during simulation time. We set the simulation time to $20s$.

Fig. 4 a) shows the reachability and MNF of gossiping for different probabilities and different numbers of nodes. We can easily conclude that the reachability of gossiping strongly depends on these parameters. For 500 nodes a probability higher than 0.6 provides a reachability of 100%. Whereas for lower probabilities the reachability drops. We also conclude from Fig. 4 a) that MNF increases linearly with the used probability value. This is evident, if we consider that all nodes use the same probability p, and subsequently the ratio of nodes that do forward the packet is also p. The main goal of gossiping adaptation is to increase the efficiency of gossiping while maintaining the reachability very high. Therefore, we have to select for gossiping the minimal probability value that maintains reachability at about 100%. From the observations above we recommend a probability of 0.6 for 500 nodes. Now if we repeat this calibration process for different numbers of nodes, we get the appropriate probability for the correspondent node densities.

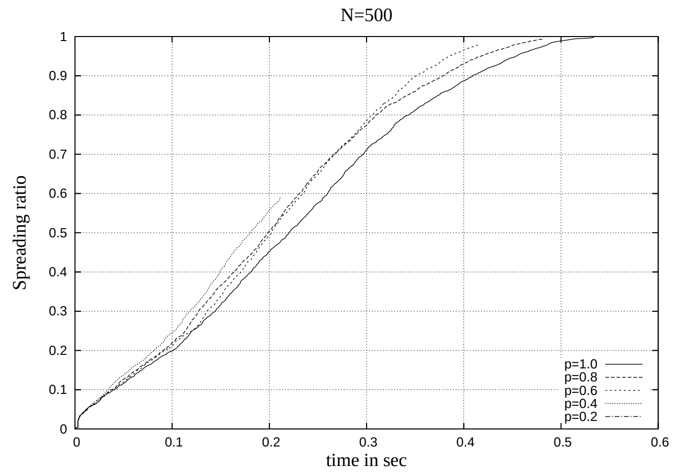
We define *spreading ratio* at time t as the ratio between the number of nodes that received the broadcast until time t to the total number of nodes. Fig. 4 b) shows the spreading ratio of gossiping over time t for 500 nodes and different probabilities. We conclude that only higher probabilities than 0.6 provide a reachability near to 100%. We also conclude that probability 0.6 provides faster propagation than higher probabilities. This is due to more collisions and higher contention if more than 60% of nodes forward the packet. So investigating the spreading ratio provides an alternative approach to fix the appropriate gossiping probability.

In previous work [30], we proposed an epidemic analytical model for information dissemination in MANETs. According to that model, we can describe information spreading using the so-called *infection rate* (denoted by a and measured in infections/s). Infection rate depends on the MANET characteristics and on the broadcast algorithm. Given the infection rate for the considered MANET and the considered broadcast algorithm, we can use an analytical expression to compute the spreading ratio over time t. Infection rate is therefore a measure for reachability and delay. It is shown in [30] that, the higher is the infection rate the lower is the mean delay.

In the following we show how we used these results to adapt gossiping. In case of gossiping, the infection rate basically depends on the node density of the MANET and on the gossiping probability p. In order to adapt the gossiping probability to the node density, we have to select that gossiping probability that maximizes the infection rate. We vary node density and the gossiping probability p and compute for each parameter combination the corresponding infection rate. Fig. 5 a) shows the measured infection rates and their interpolation. Fig. 5 b) shows the optimal probability, which should be used by nodes depending on node density.



(a) Approach 1



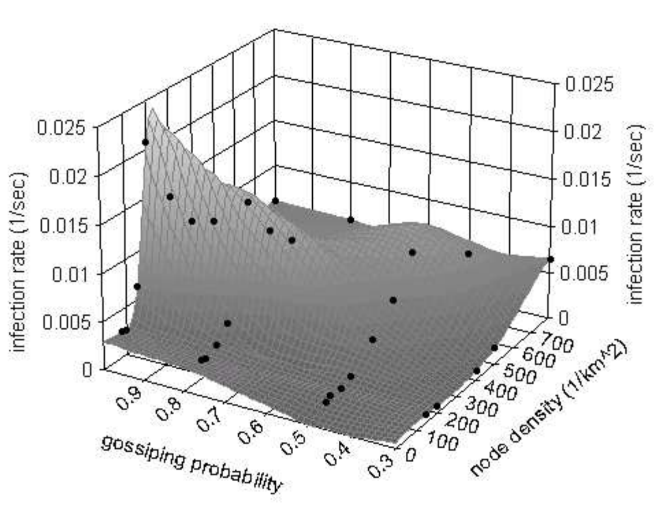
(b) Approach 2

Figure 4. Adaptation of gossiping

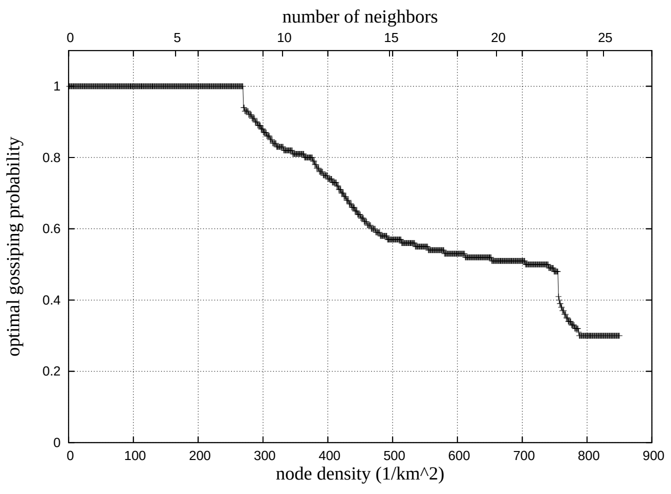
Consistent with our second requirement of generalized broadcasting strategy, we let every node set locally and independently the gossiping probability. Given n the number of neighbors and R the communication range, a node computes easily its local density by:

$$d = \frac{n+1}{\pi * R^2} \tag{1}$$

According to this value the node has to set on-the-fly the gossiping probability. To avoid the computation of local node density, which also assumes that nodes know their communication range, we recommend that nodes select the gossiping probability depending on the current number of neighbors n. By



(a) Infection rate



(b) Optimal probability

Figure 5. Adaptation of gossiping (Approach 3)

scaling the x-axis of Fig. 5 b) using formula (1), we get the probability p as a function of n. We provide the discreate values of this function as a lookup table that maps number of neighbors to the probability values. At run-time we let nodes have access to this lookup table in order to set the gossiping probability dynamically depending on their current number of neighbors.

5.4 Calibration of Partition Detection

To evaluate and calibrate the partition detection heuristic we may use the global view given by the simulator. A metric for the accuracy of the heuristic is the ratio of correct detections to all detections. This ratio has to be maximized while dimensioning the heuristic. In this paper, we evaluate our partition detection by measuring the efficiency of rebroadcasting. A suitable efficiency metric for rebroadcasting is the mean number of additionally reached nodes per rebroadcast, i.e. its gain. The higher its gain, the more efficient is rebroadcasting. To dimension the partition detection parameters, i.e. IS_threshold and maxLBRlength, we select the values with maximal gain.

For calibration we arbitrarily fix the lifetime to 60s and the node speed to 30m/s, and we vary IS_threshold in {0%, 25%, 50%, 75%, 100%} and maxLBRlength in {1, 5, 10, 25, 50, 100}. For every combination we compute the gain and select the combination that maximizes the gain. LBR serves anyway for the identification of a certain partition until the next join. An identification should consider the size of the partition (reflected by number of nodes) and the number of broadcasts originated per unit of time in that partition (reflected by the packet rate). That is why we repeated the calibration process for a wide range of number of nodes and packet rates. The combinations, for which the gain is maximal, are listed in Table 3. We repeated these steps for lifetime values of 600s and 1800s and concluded that these combinations are almost independent from the lifetime value. We explain this as follows. Higher lifetimes means higher number of partition joins within lifetime period. Since gain is a relative metric that measures the mean efficiency of the broadcast repetition strategy over all partition joins and if we assume that this efficiency remains almost constant for every join, we can conclude that the mean efficiency is independent from the number of joins and therefore independent from the lifetime. A variation of node speed can be also interpreted as a variation of the lifetime with respect to the calibration process. Since both variations lead to a variation of the number of partition joins within the lifetime period. Therefore, we conclude that also node speed has no significant impact on the calibration process.

In Table 3 we observe that the denser the network or the more congested it is, the smaller the IS_threshold but the higher maxLBRlength should be selected. In this work, we use a simple calibration of partition detection. We use for MANETs with higher densities than 200 $nodes/km^2$ the tuple (0%, 100), otherwise we use the tuple (25%, 100). This calibration is suitable for most of simulated scenarios in Table 3. Consistent with our second requirement, we let every node select locally and independently the IS_threshold value: At run-time a node sets IS_threshold to 25%, if its current number of neighbors is lower than 6, and 0% otherwise.

N	50	100	200	300	500
n	0.57	2.14	5.28	8.42	14.7
1 packet/s	25%, 100	25%, 100	0%, $\geq$ 50	0%, 100	0%, 100
0.1 packet/s	50%, 100	25-50%, 100	25-50%, $\geq$ 25	$\leq$ 50%, $\geq$ 25	0%, $\geq$ 50
0.001 packet/s	25-75%, $\geq$ 25	$\leq$ 75%, $\geq$ 25	$\leq$ 75%, $\geq$ 10	$\leq$ 50%, $\geq$ 25	0%, $\geq$ 25

Table 3
Calibration of partition detection

5.5 Performance of Hypergossiping

After adaptation and calibration we now evaluate hypergossiping for a wide range of node densities and speeds, and packet rates. Finally, we compare the performance of hypergossiping to that of IF, ACB and density-aware probabilistic flooding. We select for all scenarios 25 senders. If node speed is 0 m/s, the nodes are static and do not discover new neighbors; therefore they do not trigger broadcast repetition; that is why hypergossiping goes into simple gossiping.

5.5.1 Impact of Node Density and Mobility

In this section, we investigate the performance of hypergossiping for a wide range of node densities and node speeds.

Node mobility may contribute to overcome network partitioning. It then follows that the higher is the mobility, the higher is the reachability and the lower is the delay. Fig. 6 shows that the impact of node mobility on reachability is more significant for lower lifetimes. For very short lifetimes the reachability of hypergossiping tends to the reachability of simple gossiping. Fig. 6 illustrates that reachability saturates at around 80% for 50 nodes. We explain this as follows. At a packet rate of 0.0005 packet/s every sender originates only 1 packet within lifetimes up to 2000s. So within lifetimes considered in Fig. 6 only 25 messages are relevant in the MANET at a given time point. LBRs can store all IDs of these messages. The partition condition (*overlap* $\leq 25\%$) becomes over time stronger and some partitions could not be detected. In [31] we investigated the accuracy of our partition detection heuristic with more details using the global view given by the simulator.

We now arbitrarily set the lifetime value to 600s and the packet rate to 0.001 packet/s. This means that the packet's lifetime expires before the source orig-

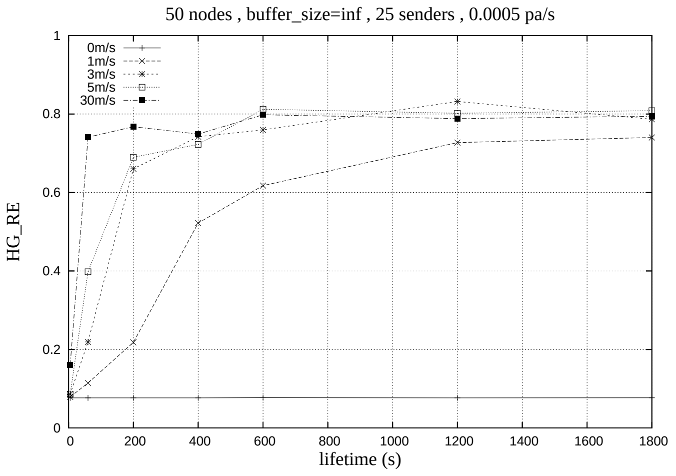


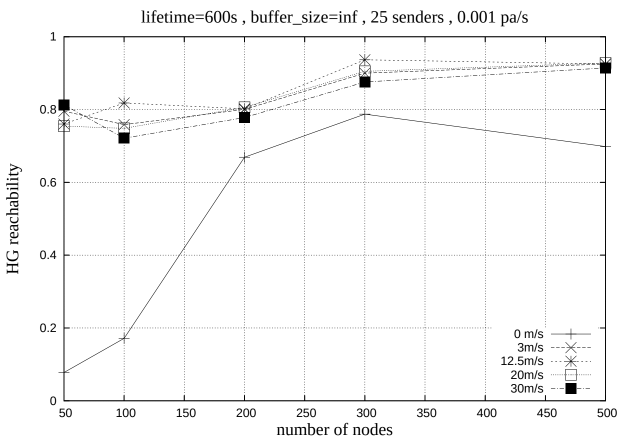
Figure 6. Imapct of lifetime on reachability

inates the following packets. Fig. 7 a) shows that rebroadcasting can strongly increase reachability in sparse MANETs; For 50 nodes and 3m/s reachability increases from 8% to 76%. Hypergossiping also increases reachability if gossiping reachability drops because of collisions; Reachability increases from 70% to 92% for 500 nodes and 3m/s. Hypergossiping keeps the MNFR very low while increasing reachability; Nodes namely forward and rebroadcast a given message in average maximal 1.1 times (Fig. 7 b)). The bend in reachability and MNFR at 200 nodes, in Fig. 7 a) and b), is due to our simple calibration of partition detection, where IS_threshold jumps from 25% to 0% by node densities around 200 $nodes/km^2$ (see Subsection 5.4). In ongoing work, we are looking for smoothing the selection of IS_threshold.

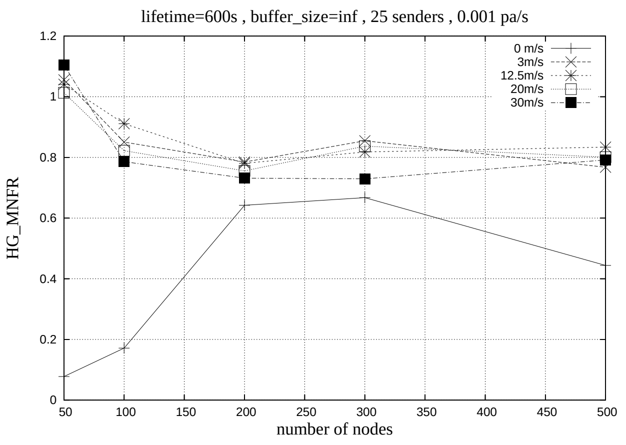
5.5.2 Impact of Network Load

In this section, we discuss the performance of hypergossiping for a wide range of packet rates. For this study, we consider 50, 300 and 500 nodes and node speed of 30 m/s. The lifetime is arbitrarily set to 200 s. We vary the packet rate from 0.001 packet/s to 1 packet/s.

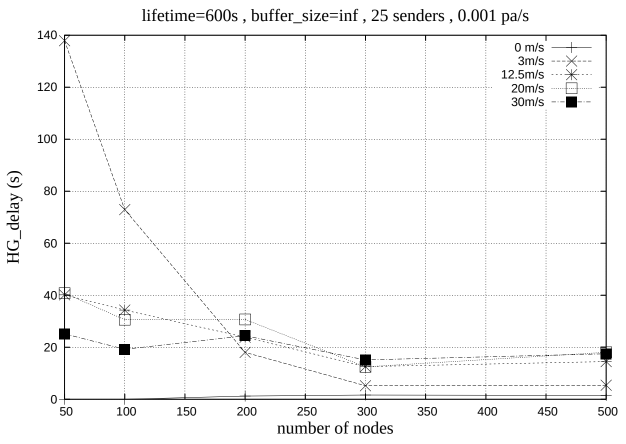
Fig. 8 shows that even for high network traffic hypergossiping provides a very high reachability. Actually the reachabilty surprisingly increases if packet rate increases. Similar to the saturation effect in Fig. 6, We explain this effect as follows. If the packet rate increases, nodes update faster their LBRs, so that partition detection condition gets weaker and saturation effect less important. This also leads to higher MNFR. This effect shows again the need for adapting IS_threshold to packet rate.



(a) Reachability



(b) MNFR



(c) Delay

Figure 7. Impact of node density and mobility

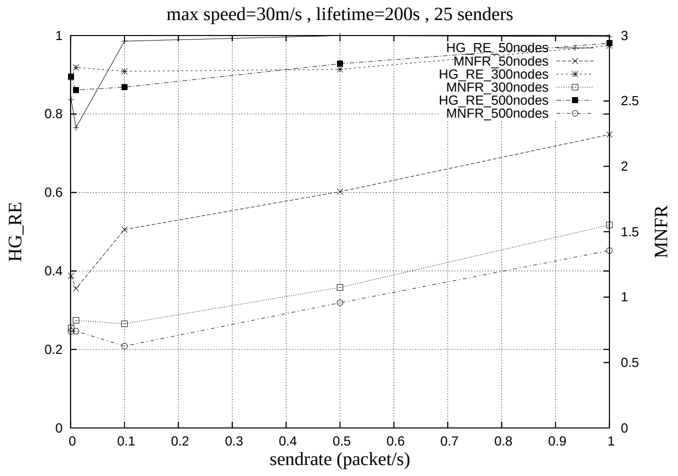


Figure 8. HG_RE and MNFR versus packet rate

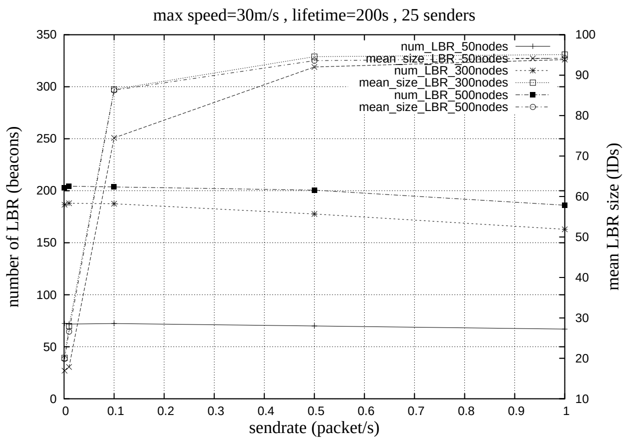
5.5.3 Message and Storage Overhead

Now we investigate buffer size, BR size, LBR size, number of BR beacons and number of LBR beacons. For this study, we arbitrarily set lifetime to 200s and node speed to 30m/s, and vary packet rate and number of nodes. The number of BRs and the number of LBRs are counted within the simulation time, which is set to 250s. The buffer size and BR size are metrics for the storage overhead. BR- and LBR-size and -number quantify the additional bandwidth use, caused by the exchange of BR and LBR beacons.

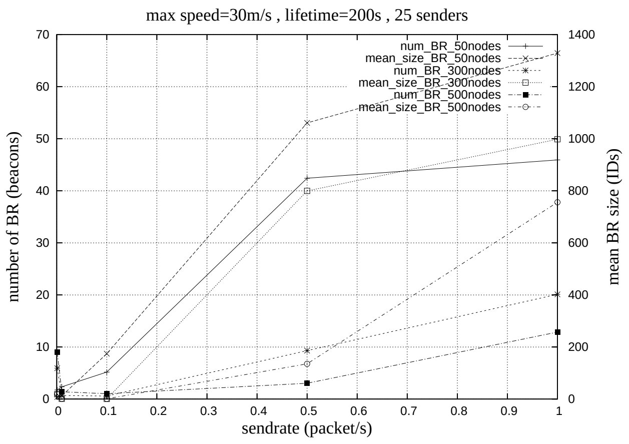
The number of LBR beacons depends on the frequency of discovering new neighbors, which depends on the node density and node speed (Fig. 9 a)). The number of LBR beacons increases with node density. It increases slightly with higher packet rates because of more frequent collisions, which makes neighborhood information less accurate. The mean size of LBR increases with increasing packet rate and node density but it is limited by maxLBRlength value, i.e. 100 IDs.

Similar to LBR, the size and number of BRs increase with increasing packet rate (Fig. 9 b)). In contrast to LBR, size and number of BRs decrease with increasing node density. This is due to calibration of partition detection, which uses stronger detection condition (0%) for dense networks and thus triggers less BR-beacons. Less BR beacons leads to higher delay and thus lower mean BR sizes.

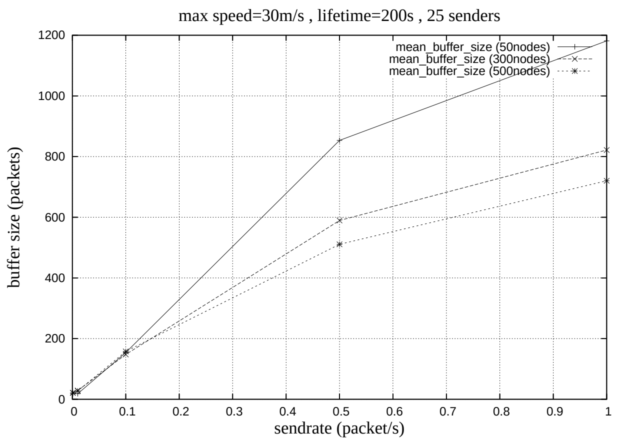
Fig. 9 c) illustrates buffer size versus packet rate. The mean buffer size is computed over time and over all nodes. The buffer size increases with packet rate. For high packet rates the buffer size decreases with node density. This is due to the number of collisions, that increases with increasing packet rates. Ongoing work is aiming to limit buffer overhead, by deploying appropriate replacement strategies (e.g. those that take residual lifetime into consideration)



(a) Number and size of LBR beacons



(b) Number and size of BR beacons



(c) Buffer size

Figure 9. LBR-, BR-, and buffer-overhead

and by adapting the buffer capacity to node characteristics (e.g. to mobility).

5.6 Comparison to Related Work

In this section, we compare hypergossiping to the density-aware probabilistic flooding (PROB-FLOOD) [26], to the adaptive counter-based scheme (ACB) [25] and to the integrated flooding (IF) [9, 10]. For this study, we consider a wide range of node densities and speeds and arbitrarily fix the packet rate to 0.001 packet/s.

The adaptive thresholds are shown in Fig. 10. For ACB we use the dynamic threshold given in [25]. ACB uses a random time span to count redundant packet receptions and possibly forwards the message after this span. This time period is comparable to the random forwarding delay of gossiping (fDelay) and PROB-FLOOD. We choose the same value for these parameters, i.e. 10 ms, which is also used in [13].

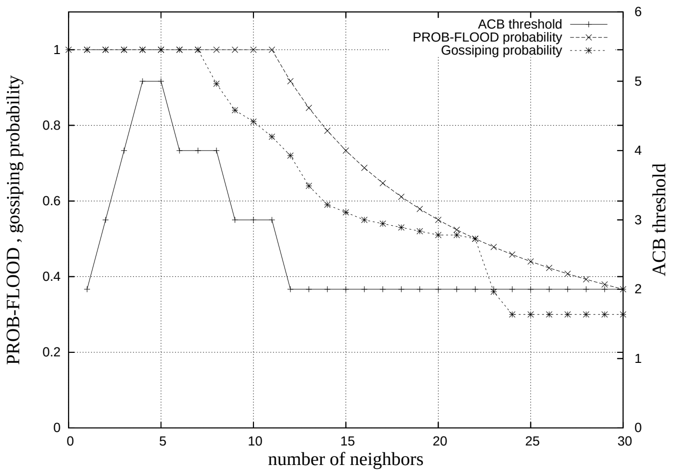
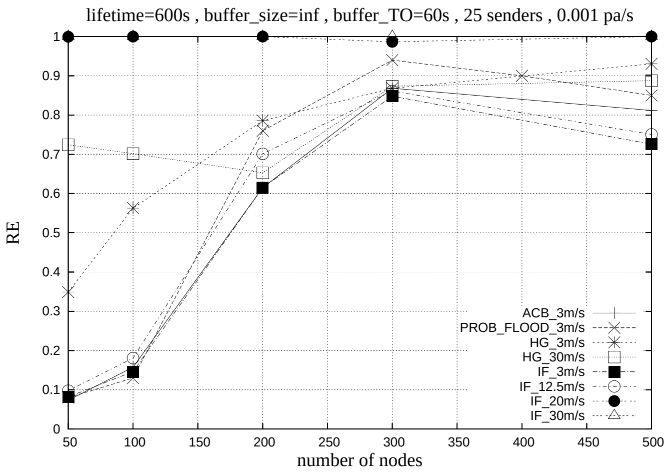


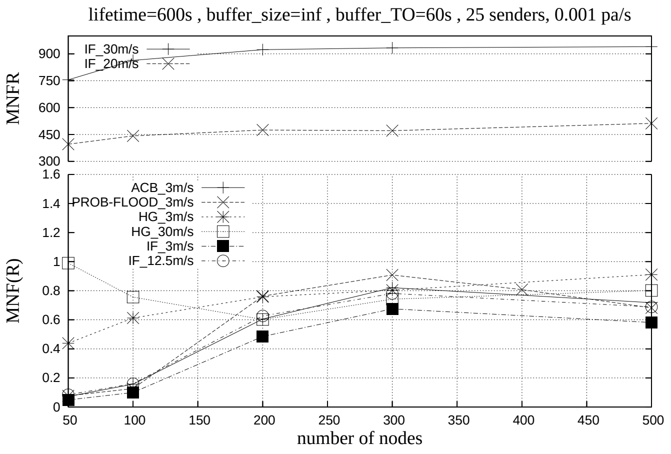
Figure 10. Adaptive thresholds

For IF we use the following parameters (most of them are stated in [9]). In scoped flooding a node forwards a new received message, if at least 15% of its neighbors are not covered by the previous forwards. We note here that scoped flooding is a topology-based scheme that needs 2-hop topology information, which means that nodes have to include their neighbor list in HELLO beacons. Hyperflooding holds packets for a fixed time period and rebroadcasts them on discovering a new neighbor. Nodes install scoped flooding if their relative speed to all their neighbors is lower than $10m/s$. If the relative speed is higher than $25m/s$ hyperflooding is selected. Otherwise plain flooding is deployed. We note that for node speed values until $12.5m/s$, nodes will never reach the higher switching threshold of IF (i.e. $25m/s$) and thus hyperflooding will never be installed in such configuration. For higher speed values some nodes may install hyperflooding mode, which should increase the reachability of IF. For node speed equals $3m/s$ the max relative node speed is $6m/s$, that is why only scoped flooding is installed by IF.

For hypergossiping we arbitrarily fix the lifetime to $600s$ and use the same buffering strategy as IF. IF buffers all packets for a given fixed time, called *buffer_timeout*. We arbitrarily set the value of buffer_timeout to $60s$. ACB and PROB-FLOOD show almost mobility-independent performance and thus we present for these protocols only results for $3m/s$.



(a) Reachability



(b) MNF(R)

Figure 11. Comparison to related work

We can easily conclude from Fig. 11 a) that hypergossiping reachability outperforms PROB-FLOOD and ACB reachability for sparse networks and highly dense networks while keeping MNFR below 1. This is due to that HG remedies both causes of broadcast interruption: Network partitioning and broadcast storms.

Comparing HG and IF we conclude that IF does not provide an efficient partition detection strategy: IF provides namely a very high reachability for highly mobile MANETs, but MNFR ranges from 400 to 900 rebroadcasts per packet and node (Simulation results show that even for very low buffer timeout values, MNFR is very high for highly mobile scenarios: for $buffer\_timeout = 5s$ and $N = 100$ nodes, $MNFR = 11$). For lower mobility IF installs only scoped flooding or plain flooding and subsequently performs similar to ACB and PROB-FLOOD for sparse networks but worse than these schemes for dense networks, lack of adaptation of scoped flooding to node density.

6 Conclusion and Future Work

In this paper, we introduced hypergossiping, an adaptive MANET broadcast algorithm, which presents our first steps towards a single broadcast solution for a wide range of MANET operation conditions. Hypergossiping covers MANETs with larger node densities and speeds, and network loads. We presented a novel method to adapt gossiping probability to node density and reduce the broadcast storms. Moreover, we presented a novel heuristic for repetition of broadcasts in order to overcome different causes of broadcast interruptions, such as network partitioning.

We are working on more sophisticated adaptation techniques of partition detection parameters. In future work, we will investigate different buffer management strategies to reduce buffer overhead. We expect that these strategies have to take into consideration the residual lifetime of buffered packets and that adapting the buffer capacity to node mobility contributes to reduce the buffering overhead of hypergossiping.

References

[1] J. Tian, P. J. Marrón, K. Rothermel, Location-based hierarchical data aggregation for vehicular ad hoc networks, in: Proceedings of Communication in Distributed Systems 2005 (KiVS), Kaiserslautern, Germany, 2005.

[2] Z. Cheng, W. Heinzelman, Flooding strategy for target discovery in wireless networks, in: Proceedings of the 6th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Venice, Italy, 2003.

[3] J. Hähner, K. Rothermel, C. Becker, Update-linearizability: A consistency concept for the chronological ordering of events in manets, in: Proceedings of the first IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), Fort Lauderdale, USA, 2004.

[4] C. Becker, M. Bauer, and J. Hähner, Usenet-on-the-fly: supporting locality of information in spontaneous networking environments, in: Proceedings of CSCW 2002 Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments, New Orleans, USA, 2002.

[5] S. Y. Ni, Y. C. Tseng, Y.-S. Chen, J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, in: Proceedings of the 5th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MOBICOM), Seattle, USA, 1999.

[6] J. Hähner, D. Dudkowski, P. J. Marrón, K. Rothermel, A quantitative analysis of partitioning in mobile ad hoc networks, in: extended abstract in Proceedings of the Joint Int. Conf. on Measurement and Modeling of Computer Systems (Sigmetrics-Performance), New York, USA, 2004.

[7] Z. Haas, J. Halpern, L. Li, Gossip-based ad hoc routing, in: Proceedings of IEEE INFOCOM, New York, USA, 2002.

[8] K. Obraczka, G. Tsudik, K. Viswanath, Pushing the limits of multicast in ad hoc networks, in: Proceedings of the International Conference on Distributed Computing Systems (ICDCS), Phoenix, USA, 2001.

[9] K. Viswanath, K. Obraczka, An adaptive approach to group communications in multi hop ad hoc networks, in: Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Vienna, Austria, 2002.

[10] K. Viswanath, K. Obraczka, An adaptive approach to group communications in multi-hop ad hoc networks, in: Proceedings of the International Conference on Networking (ICN), Atlanta, USA, 2002.

[11] W. Zhao, M. Ammar, E. Zegura, A message ferrying approach for data delivery in sparse mobile ad hoc networks, in: Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Tokyo, Japan, 2004.

[12] A. Ouksel, O. Wolfson, B. Xu, Opportunistic resource exchange in inter-vehicle ad hoc networks, in: Proceedings of the IEEE International Conference on Mobile Data Management (MDM), Berkeley, USA 2004.

[13] B. Williams, T. Camp, Comparison of broadcasting techniques for mobile ad hoc networks, in: Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Lausanne, Switzerland, 2002.

[14] Y. Yi, M. Gerla, T. Kwon, Efficient flooding in ad hoc networks: a comparative performance study, in: Proceedings of the IEEE International Conference on Communications (ICC), Anchorage, USA, 2003.

[15] F. Dai, J. Wu, Performance analysis of broadcast protocols in ad hoc networks based on self-pruning, in: IEEE Transactions on Parallel Distributed Systems, Vol. 15, No. 11, 2004.

[16] A. Laouiti, A. Qayyum, L. Viennot, Multipoint relaying: An efficient technique for flooding in mobile wireless networks, in: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS), Hawaii, USA, 2002.

[17] J. Wu and H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), Seattle, USA, 1999.

[18] I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks, in: IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 1, 2002.

[19] G. Calinescu, I. Mandoiu, P. J. Wan, A. Zelikovsky, Selecting forwarding neighbors in wireless ad hoc networks, in: Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), Rome, Italy, 2001.

[20] S. Basagni, D. Bruschi, I. Chlamtac, A mobility transparent deterministic broadcast mechanism for ad hoc networks, in: ACM/IEEE Transactions on Networking, Vol. 7, No. 6, 1999.

[21] J. Wu, F. Dai, Broadcasting in ad hoc networks based on self-pruning, in: Proceedings of IEEE INFOCOM, San Francisco, USA, 2003.

[22] J. Sucec, I. Marsic, An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks, CAIP Technical Report 248 - Rutgers University, 2000.

[23] W. Peng , X. C. Lu, On the reduction of broadcast redundancy in mobile ad hoc networks, in: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Boston, USA, 2000.

[24] J. Wu, F. Dai, Mobility management and its applications in efficient broadcasting in mobile ad hoc networks, in: Proceedings of IEEE INFOCOM, Hong Kong, China, 2004.

[25] Y. C. Tseng, S. Y. Ni., E.Y. Shih, Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc networks, in: IEEE Transactions on Computers, Vol. 52, No. 5, 2003.

[26] J. Cartigny, D. Simplot, Border node retransmission based probabilistic broadcast protocols in ad-hoc networks, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS), Hawaii, USA, 2003.

[27] C. Ho, K. Obraczka, G. Tsudik, K. Viswanath, Flooding for reliable multicast in multi-hop ad hoc networks, in: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), Seattle, USA, 1999.

[28] S. McCanne, S. Floyd, Ns network simulator.
URL `http://www.isi.edu/nsnam/ns/`

[29] J. Yoon, M. Liu, B. Noble, Random waypoint considered harmful, in: Proceedings of IEEE INFOCOM, San Francisco, USA, 2003.

[30] A. Khelil, C. Becker, J. Tian, K. Rothermel, An epidemic model for information diffusion in MANETs, in: Proceedings of the 5th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Atlanta, USA, 2002.

[31] A. Khelil, P. J. Marrón, R. Dietrich, K. Rothermel, Evaluation of partition-aware MANET protocols and applications with ns-2, in: Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Philadelphia, USA, 2005.