Evaluation of Partition-Aware MANET Protocols and Applications with ns-2

Abdelmajid Khelil, Pedro José Marrón, Rüdiger Dietrich, Kurt Rothermel
Universität Stuttgart, IPVS/VS
Universitätsstrasse 38, 70569 Stuttgart, Germany
Tel: (+49) 711-7816-{251, 223, 434}
Fax: (+49) 711-7816-424
{khelil, marron, rothermel}@informatik.uni-stuttgart.de , ruediger.dietrich@gmx.net

Keywords: MANET, Network Partitioning, Protocol Evaluation, ns-2, Hypergossiping

Abstract

Mobile Ad Hoc Networks (MANET) are composed of mobile devices equipped with short range radio capabilities. Communication is possible between devices located in each other's transmission range. Especially in sparse MANETs, node mobility leads to frequent network partitioning, which makes typical networking tasks much more difficult. Although we observe an increasing need for partitioning information, the widely used network simulator ns-2 does not support protocol developers to easily evaluate their protocols concerning network partitioning.

To simplify the evaluation of MANET partition-aware protocols and applications in ns-2, we extend the simulator to provide partitioning information at the simulation time. Developers might be interested in using this information to evaluate their partition-aware protocols and applications and to compare their performance to the optimal case.

Hypergossiping is a partition-aware broadcast protocol for MANETs. Nodes rebroadcast messages upon joining the partitions that have not yet received these messages. Hypergossiping uses a heuristic to detect partition joins. We show the feasibility of our approach and the applicability of the partitioning information we provide for ns-2 users to compare the performance of our approach to the optimal case.

INTRODUCTION

The number of mobile devices equipped with wireless network interfaces is continuously increasing. Many existing wireless technologies such as WLAN and Bluetooth provide besides an infrastructure-based communication mode an ad hoc communication mode. The ad hoc mode allows mobile devices to directly communicate if they enter each others communication range. If nodes can act as routers, multihop communication between nodes is possible. The so formed networks are refered as Mobile Ad Hoc Networks (MANET). MANETs are suitable for scenarios where an infrastructure is very costly or even unavailable.

MANETs show frequent network partitioning [1], i.e. the network goes into groups of nodes that can not communicate directly or indirectly with each other. Partitioning happens due to node movement, communication link failures (e.g. spatial constraints), or node failures (e.g. energy depletion).

Partitioning leads to the reduction and degradation of the quality of network services or even to their unavailability. Therefore, many existing MANET protocols and applications need to consider and handle network partitioning and network merging.

TORA [2] is a MANET unicasting protocol that provides a mechanism to detect network partitioning in order to erase depreciated routes. The "Epidemic Routing for Partially-Connected Ad-Hoc Networks" [3], the "Probabilistic Routing in Intermittently Connected Networks" [4], and the "Delivering Messages in Disconnected Mobile Ad-Hoc Networks" [5] present MANET routing protocols that provide mechanisms to deliver messages even in partitioned networks.

MAODV [6] is a MANET multicasting protocol that handles network partitioning. The protocol initiates the multicast group election procedure once the multicast tree becomes partitioned. If two parts of the tree become connected once again the tree has to be repaired.

Hypergossiping [7] is a MANET broadcasting protocol that considers network partitioning in order to increase the delivery reliability of gossiping in sparse networks. Hypergossiping deploys a heuristic to detect partition joins and rebroadcasts the appropriate messages from buffer on partition join detection.

MANET autoconfiguration aims at assigning dynamically IP-addresses for MANET nodes. Many algorithms have been developed to allow autoconfiguration such as MANETconf [8], Prophet [9], [10], and [11]. These algorithms handle partition merging in order to avoid address conflicts (duplicates).

Because of the importance of handling MANET partitioning some research is done to detect, predict and prevent this partitioning. In [12] the authors present an algorithm to detect MANET partitioning using border nodes. For partition prediction in MANETs, the authors in [13, 14] use velocity-clustering. In [15] the authors predict network partitioning by means of the so-called multiple disjoint paths set. [16] presents methods to predict network partitioning in location-aware MANETs. Partitioning prevention or retardation in MANETs can be realized by changing the trajectories of cer-

tain nodes such as suggested in [17] and [18] or by adapting the transmission range.

We observe an increasing need for partitioning information to develop and evaluate partition-aware protocols and applications for MANETs. Simulation with the tool ns-2 [19] is widely used in the MANET community. Although there are many projects that handle MANET partitioning, ns-2 does not provide extra partitioning information for the protocol evaluation concerning network partitioning. The main contribution of this paper is an extension for ns-2, which simplifies the evaluation and the performance comparison of partition-aware protocols and permits an easy definition of optimal protocols. Furthermore, we show using the example of hypergossiping the practical importance of this extension. We provide patches for common versions of ns-2 and for common operating systems, which allows an easy installation of the framework.

The remainder of this paper is organized as follows. In Section we define key terms, present our system model, and briefly introduce ns-2. Section discusses related work. In Section we present our approach to provide partitioning information for ns-2 users. Using hypergossiping as an example, we show in Section the applicability of the provided information. Finally, Section concludes this paper.

PRELIMINARIES

Let us first define some key terms concerning network partitioning, fix our system model, and briefly introduce ns-2.

Terminology

Network split (or partitioning) is the division of the network into two (or more) disjoint groups of nodes that can not communicate with each other. We refer to these groups as *network partitions*. In the simplest case, a partition may consist of only one isolated node. A partition is uniquely identified by its constituting nodes and the time the partition is formed. A MANET where partitioning occurs is said to be *partitioned*.

Partition join (or merging) is the combination (coalescing) of two (or more) partitions into one bigger partition. Partition merging occurs when two partitions come in the communication range of each other.

System Model

We consider MANETs formed by N mobile devices that move according to an arbitrary mobility model within a two-dimensional area of interest. The mobility model can integrate movement constraints, but we do not consider the impact of these constraints on the propagation characteristics of the communication link. We assume that nodes are uniquely identified, e.g. using their MAC addresses. Without lost of generality we enumerate the N nodes using numbers (node-ID) from 0 to $N-1$. We assume that all nodes use the same fixed communication range R. Nodes can communicate once their distance is below R. We model each partition in the MANET as an undirected graph. The nodes constituting the partition present the vertices of this graph. An edge between two nodes is added if their distance is below R. Thus each snapshot of the MANET is a set of undirected graphs. We assume that nodes do not fail, e.g. they do not crash or run out of energy.

The network simulator ns-2

Ns-2 [19] is a discrete event simulator targeted at wired and wireless networking research. For efficiency reasons ns-2 is implemented in OTcl and C++. OTcl is an object-oriented extension of the interpreted language Tcl (the Tool Control Language). Ns-2 uses OTcl for control and C++ for data manipulation, since OTcl permits an easy and dynamic simulation configuration and C++ a fast and efficient manipulation of data and implementation of protocols.

GOD (General Operations Director) is a central omniscient instance of ns-2. GOD stores global state information. The GOD instance implemented by the current ns-2 version manages the shortest path information between nodes. This global information is used by MANET routing protocol developers.

RELATED WORK

In [1] authors provide a quantitative analysis of network partitioning using their own event-based simulator. They define partitioning metrics and present them for different movement patterns. We adopt their metrics and provide them for ns-2 users dynamically at simulation-time.

MANET routing protocol developers need global path length information in order to analyse the path length optimality of their ad hoc routing protocols. The optimal route length is given by the shortest path length. This is the reason GOD continuously provides the length in hops of the shortest path between any two nodes [19]. For this goal the independent utility `calcdest` is provided for ns-2. `Calcdest` annotates movement pattern files generated for ns-2 with lines of GOD information. These lines are needed to load the GOD instance with the appropriate information at the appropriate time. `Calcdest` calculates the number of hops of the shortest path between nodes based on the nominal radio range. The GOD instance does not calculate this on the fly during simulation runs, since it can be quite time consuming. Global path knowledge is loaded into the GOD instance from the movement pattern file using OTcl commands. Currently, the GOD instance is used only to store an array of the shortest number of hops required to reach from one node to one other, and does not provide partitioning information.

PROVIDING PARTITIONING INFORMATION FOR NS-2

In this section, we compute partitioning information for arbitrary movement patterns and provide an interface, which satisfies the most important needs of partition-aware protocols and applications developers.

Approach

We follow a similar approach to that of generating GOD information for MANET routing. We first annotate movement trace files with basic partitioning information. The GOD instance then loads this information at the simulation begin. During simulation GOD aggregates partitioning information and generates dynamic partitioning information. For MANET developers GOD provides a generic interface that simplifies the use of partitioning information during simulation (Fig.1).

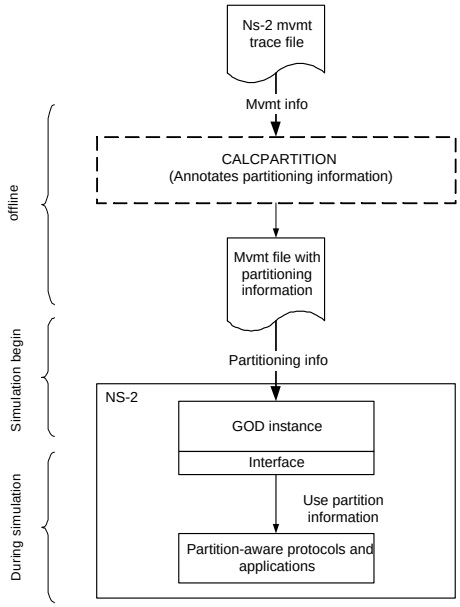


Figure 1: Approach

We do the annotation offline. This increases the reusability of trace files, since the annotation only depends on the node movement provided that the communication range remains constant. This also reduces simulation run-time, since calculating partition information may be quite time-intensive. The generated partitioning information is calculated based on a fixed communication model, where we assume that nodes can communicate if their distance is below a given constant value *R*. This makes the information invalid, if nodes fail or adjust communication range at run-time, which we except in our system model.

Annotation

An annotation is a set of OTcl commands for ns-2. We have to annotate complete partitioning information while keeping the size of trace file as small as possible. But we also have to generate offline as much statistics as possible, to keep simulation time as short as possible.

The tool `calcpartition` is able to annotate every ns-2 movement trace file, independently from the mobility model and from the generation tool. The path of the movement trace file as well as the nominal communication range are provided as arguments to `calcpartition`.

One approach for providing partitioning information is to annotate the partitions constituting the MANET at the beginning of the scenario and at each subsequent time this MANET composition will change, i.e. on each future join or split event. In order to reduce the size of this information we proceed as follows. We first annotate the initial partition composition of the MANET and then we annotate only the event type and the resulting partition(s).

We annotate the initial partitions constituting the MANET by means of OTcl commands under the following form:

$god_ set-part ‹partition-list›

*‹partition-list›= {‹node-list›} ({‹node-list›})**

*‹node-list› = ‹node-ID›(, ‹node-ID›)**

Set-part is a function we added to the GOD class. The subsequent partition join and split events are annotated using the following commands respectively:

$ns_ at ‹time›"$god_ set-join ‹new-partition›"

$ns_ at ‹time›"$god_ set-split ‹partition-list›"

Set-join and set-split are two functions we also added to the GOD class. We annotate on each event only the nodes that are concerned by that event. This decreases the size of the annotated trace file.

The following example of annotation

$god_ set-part { 0, 1 } { 2 } { 3, 4 }

$ns_ at 4.26 "$god_ set-join { 0, 1, 2 } "

$ns_ at 6.54 "$god_ set-split { 3 } { 4 } "

means that at the beginning of the scenario the MANET ist formed by 3 partitions. The first partition contains nodes 0 and 1. The isolated node 2 presents the second partition. Nodes 3 and 4 form the third partition. At time 4.26 s, the partitions { 0, 1 } and { 2 } merge. At time 6.54 s, the partition { 3, 4 } splits into two partitions.

At the end of the scenario file `calcpartition` lists some helpful statistics like the number of joins, the number of splits, the average (min, max) partition size, the average number of partitions, and the average time to next split or join (in ms). These statistics are valid for the movement scenario time and are printed out as OTcl comments.

Interface

After loading the annotated partitioning information from the scenario file, GOD prepares this information for protocol developers. For this, GOD provides an interface for ns-2 users (Fig.1).

This interface is generic and easy to use. It provides sufficient functions to satisfy the needs of the ns-2 users. Protocol developers need to debug and evaluate their protocols by observing important indicators or by defining optimal protocols to compare their performance to the optimal case.

The TORA routing protocol [2] needs to detect partition splits. For global evaluation of TORA concerning network partitioning, nodes have to subscribe for split events. In the optimal case nodes have to react on split events by erasing from their routing table the routes to all destinations that belong to the splitting partition. For this, nodes need to query the node-IDs of the own partition or the node-IDs of the splitting one.

MAODV [6] has to consider both join and split events. In the optimal case, MAODV initiates the multicast election on partition split in both resulting partitions, and repairs the multicast tree on every partition join.

Dynamic IP-address assignment algorithms have to consider only partition joins. In the optimal case, one of the nodes that trigger a partition join has to find out if there are duplicate IP addresses in the new formed partition. For the global evaluation of the assignement algorithms, nodes have to query the IP-addresses of all nodes populating the same partition, in order to check whether duplicate addresses exist.

From the examples above we distinguish two main classes of possible needs of protocol developers. A developer may be interested in some instantaneous or statistical values or in some split or join events. Therefore, our GOD interface for partitioning information provides two user modes. First, the Query-Interface allows users to query instantaneous or statistical partition information. Second, the Subscribe-Interface allows nodes to subscribe for and then to unsubscribe from partition join or split events.

Query-Interface

The query-interface processes the requests of nodes for partitioning information. Developers may need instantaneous or statistical information.

- Instantaneous partitioning information describes the current MANET partitioning topology. Currently the following functions are implemented:
  - *getNumberOfPartitions()*: returns the current number of partitions.
  - *getNodesOfPartition(*$node_i$*)*: returns the IDs of nodes that constitute the partition that contains $node_i$.
  - *getPartitionSize(*$node_i$*)*: return the size of the partition containing $node_i$.
  - *belongToSamePartition(*$node_1$,$node_2$*)*: checks whether $node_1$ and $node_2$ belong to the same partition.
- Statistical partitioning information can be calculated over time, partitions or nodes [1]. Statistics over time are done between two past points of time t_1 and t_2. We currently provide the following functions:
  - *getAverageNumberOfPartitions(*t_1,t_2*)*: returns the average number of partitions over the time interval between t_1 and t_2.
  - *getAveragePartitionSize(*t_1,t_2*)*: returns the average size of partitions over the time interval between t_1 and t_2.
  - *getMinPartitionSize(*t_1,t_2*)*: returns the minimal partition size between t_1 and t_2.
  - *getMaxPartitionSize(*t_1,t_2*)*: returns the maximal partition size between t_1 and t_2.
  - *getPartitionChangeRate(*$node_i$,t_1,t_2*)*: returns the partition change rate of $node_i$, i.e. the number of join or split events that the partition of $node_i$ experiences between t_1 and t_2.
  - *getAveragePartitionChangeRate(*t_1,t_2*)*: returns the average partition change rate over all nodes.
  - *getSeparationTime(*$node_i$,$node_j$,t_1,t_2*)*: returns the cumulative time between t_1 and t_2, during which the nodes $node_i$ and $node_j$ do not belong to the same partition.
  - *getConnectionTime(*$node_i$,$node_j$,t_1,t_2*)*: returns the cumulative time between t_1 and t_2, during which the nodes $node_i$ and $node_j$ belong to the same partition.
  - *getNumberOfJoins(*t_1,t_2*)*: returns the number of partition joins between t_1 and t_2.
  - *getNumberOfSplits(*t_1,t_2*)*: returns the number of partition splits between t_1 and t_2.

Subscribe-Interface

This interface propagates partitioning events to the interested nodes. Nodes can subscribe for or unsubscribe from partition join or split events. The current subscribe interface provides the following major functions:

- *subscribeJoin()*: Allows nodes to subscribe for all join events. The subscribers receive a notification each time a partition join occurs. The join notification mainly contains the nodes constituting the merging partitions, and the IDs of both nodes that realized the join.

- *subscribeSplit()*: Allows nodes to subscribe for all split events. The split notification mainly contains the nodes constituting the resulting partition.
- *unsubscribeJoin()*.
- *unsubscribeSplit()*.

CASE STUDY: HYPERGOSSIPING

So far we have discussed the partitioning framework. We now show the usability of some provided global partitioning information for the evaluation of partition-aware protocols and applications in ns-2. For this, we experiment with the behavior of hypergossiping (HG), a partition-aware broadcast protocol for MANETs, if the GOD information is used to provide nodes with perfect partitioning information at zero cost.

Hypergossiping (HG)

Hypergossiping [7] combines two strategies to distribute messages to all nodes. The first strategy is called *gossiping* (probabilistic flooding) and aims at efficient distribution of messages within the same partition. The second strategy is called *broadcast repetition* and aims at overcoming network partitioning. Hypergossiping buffers messages and rebroadcasts them on partition joins. For this, hypergossiping utilizes a *partition join detection heuristic* to detect partition joins and a *rebroadcasting protocol* to send the appropriate messages.

The partition join detection heuristic works as follows. Nodes share with their new neighbors an ID-list of Last Broadcast Received (LBR list). If a node encounters a node that has a sufficiently different LBR it assumes that a partition join has just occured and triggers the rebroadcasting protocol.

The rebroadcasting protocol first sends the complete ID-list of Broadcasts Received (BR list). Nodes receiving this BR list have then to rebroadcast messages from their buffer that have not yet received by the sender of the BR list. To reduce redundant rebroadcasts hypergossiping deploys a suppression mechanism: Nodes schedule the begin of rebroadcasting for a random time between 0 and *rDelay* and cancel the schedule, if one neighboring node starts to rebroadcast the messages before the scheduled time.

Evaluation of HG Without Global Partition Information

For the evaluation of broadcast protocols the following metrics are typically used [7]:

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

For the evaluation and calibration of the broadcast repetition strategy we used in previous work the metric *gain*. Gain is the mean number of additionally covered nodes per rebroadcast [7].

The above metrics describe qualitatively the performance of the broadcast repetition strategy. The determination of the distance to the optimal case is not possible, since the values of these metrics in the optimal case are impossible to compute without global partitioning information.

Evaluation of HG Using Global Partition Information

In this section, we use global partitioning information in order to determine the optimality of hypergossiping, i.e. gossiping and the broadcast repetition strategy, concerning network partitioning.

Global Information Needed

We need two kinds of global information for the global evaluation of hypergossiping. First, we need a global view concerning network partitioning. Gossiping is mainly interested in the partition size information, since its reachability is limited by the partition size, where it takes place. The broadcast repetition strategy of hypergossiping is mainly interested in partition joins and in the nodes that caused the joins. These nodes have to eventually initiate the rebroadcasting protocol. This first global view is provided by the framework presented in Section . Secondly, we need broadcast global view, i.e. the knowledge about the spreading of broadcast messages at every point of time. This global view is needed by hypergossiping developers, first, to validate if gossiping reaches all nodes within a single partition, and secondly, to determine the messages that should be rebroadcasted on partition join. This knowledge is also required for the evaluation of the partition join detection heuristic. A partition join should only then be detected by the heuristic, if the partition has to rebroadcast messages to the joining partition.

Evaluation of Gossiping

MANET partitioning strongly impacts the reachability of gossiping. Gossiping aims at reaching efficiently all nodes of the partition where the broadcasting node is located. In this section, we aim at investigating the reliability of gossiping. We define the optimal gossiping reachability (OG_RE) as the ratio of the size of the partition containing the gossiping source node to the total number of nodes. Because of collisions gossiping may not reach all nodes within a single partition.

Evaluation of the Broadcast Repetition

Using the partitioning and the spreading gobal knowledge we now perform two studies to evaluate the broadcast repetition strategy of hypergossiping. The goal of the first study is to determine the optimal values for the performance metrics (RE, Delay and MNFR) and thus to show the quality of the broadcast repetition strategy. The purpose of the second study is to count the correct, wrong and redundant decisions of the broadcast repetition strategy.

- Study 1: Optimal Broadcast Repetition

For this study we define the following new protocol: HG with optimal broadcast repetition, short HG-OBR. The gossiping implementation is not modified. Optimal broadcast repetition means optimal partition join detection and optimal rebroadcasting protocol. Optimal partition join detection is easily given by the partitioning GOD interface. Nodes simply have to subscribe to join events. Optimal rebroadcasting is given if both nodes that triggered the partition join rebroadcast exactly the messages that the opposite node has not yet received. Optimal rebroadcasting does not send messages through MAC, but calls the receive procedure of the opposite node and waits for *fDelay* until the next call for next message. Note that optimal rebroadcasting prohibits redundant rebroadcasts.

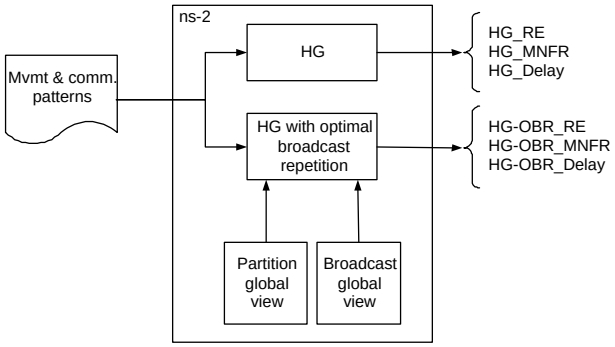


Figure 2: Optimal broadcast repetition (study 1)

For HG-OBR we denote by HG-OBR_RE, HG-OBR_MNFR and HG-OBR_Delay the values for RE, MNFR and Delay respectively. Thus the broadcast repetition strategy of HG can be compared qualitatively to the optimal case. To enable a fair comparison we use for both protocols the same movement and communication patterns (Fig.2). For random number generation we use the same seed in order to increase the similarity of both scenarios.

This approach has two main advantages. First, we do not need new evaluation metrics. Secondly, the approach provides an aggregated (coarse-grained) measurement of the distance to the optimal broadcast repetition.

Nevertheless this approach shows two main drawbacks. First, it does not allow an easy back trace to the weak points of the broadcast repetition strategy; it is not clear if the difference to the optimal case is due to the partition join detection heuristic or to the rebroadcasting protocol. Secondly, two simulation runs are needed; One run for HG and another run for HG-OBR.

- Study 2: Hypergossiping Observation

This approach is an online monitoring of HG with respect to network partitioning (Fig.3). Using the partitioning and the broadcast global view, we validate each decision made by the broadcast repetition strategy, i.e. each decision made by the partition join detection and each decision made by the rebroadcasting protocol.

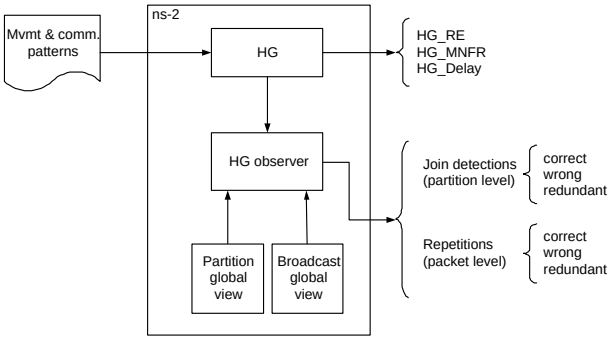


Figure 3: Hypergossiping observation (study 2)

This approach allows a fine-grained evaluation of the broadcast repetition strategy. It allows an easier back trace of the weak points of the broadcast repetition strategy, since a separated evaluation of the partition join detection heuristic and the rebroadcasting protocol is given. Furthermore just one single simulation run is needed.

For this study we define new metrics. We consider two levels for the definition of these metrics: the partition level and the packet level. On the partition level we define metrics that validate the decisions made by the partition join detection heuristic. On the packet level the metrics validate the decisions made by the rebroadcasting protocol.

1. Partition level: At this level we count the correct and wrong partition join detection decisions as well as the redundant decisions, i.e. decisions made by more than one node from each partition. We assume two partitions, say P_1 and P_2, join and that node n_1 from P_1 and node n_2 from P_2 triggered the join. We define the following three metrics:

   (a) *Correct Detections (CD)*: We increment CD, if node n_1 (resp. n_2), has to rebroadcast a list of packets HL_1 (resp. HL_2) and at least one node of P_1

(resp. P_2) detects the partition join. We also increment CD, if node n_1 (resp. n_2) has no packets to rebroadcast and no node of P_1 (resp. P_2) detects the join.

(b) *Wrong Detections (WD)*: We increment WD, if node n_1 (resp. n_2), has to rebroadcast a list of packets HL_1 (resp. HL_2) and no node of P_1 (resp. P_2) detects the partition join. We also increment WD, if node n_1 (resp. n_2) has no packets to rebroadcast and at least one node of P_1 detects the join.

(c) *Redundant Detections (RD)*: We increment RD if more than one node makes a correct detection.

2. Packet level: At this level we validate each decision to rebroadcast or not to rebroadcast a buffered message, whether it is correct, wrong or redundant. Similar to the metrics on partition level we define the following three metrics.

   (a) *Correct Repetitions (CR)*

   (b) *Wrong Repetition (WR)*

   (c) *Redundant Repetition (RR)*

Algorithm 1 shows the pseudo-code for the HG observer, which monitors HG and increments the metrics CD, WD, RD, CR, WR and RR.

Algorithm 1 HG observer

```
1: On partition join P1 with P2:
2: for all i ∈ {1,2} do
3:   if Pi have to rebroadcast a set of packets HLi then
4:     if Pi_detected_join then
5:       CORRECT_DETECTION++
6:       for all packet ∈ HLi do
7:         if rebroadcasted then
8:           CORRECT_REPETITION++
9:         end if
10:        if redundant_rebroadcasted then
11:          REDUNDANT_REPETITION++
12:        end if
13:        if not_rebroadcasted then
14:          WRONG_REPETITION++
15:        end if
16:      end for
17:    else
18:      WRONG_DETECTION++
19:    end if
20:  else
21:    if Pi_detected_join then
22:      WRONG_DETECTION++
23:      if a packet is rebroadcasted then
24:        WRONG_REPETITION++
25:      end if
26:    else
27:      CORRECT_DETECTION++
28:    end if
29:  end if
30: end for
```

Simulations

In this section, we introduce the simulation model and present the simulation results for the global evaluation of hypergossiping concerning network partitioning.

Simulation Model

We generate N mobile nodes in a 1000mx1000m field, where these nodes move according to the random waypoint mobility model. Table 1 summarizes the simulation parameters of our experiments.

We use a random HELLO-beaconing period between 0.75 s and 1.25 s. A neighbor is removed from the neighbor list if during 2 s no beacon is received from this neighbor. We use the following communication load model: At the beginning of the simulation 30 nodes initiate broadcasting at a random time between 1 and 3 seconds, and continue to send packets with a constant send rate. Broadcast messages remain relevant during their lifetime. We assume that nodes buffer all received messages as long as they are relevant. For the same simulation scenario we ran 10 passes with 10 different movement traces and considered the average.

Table 1: Simulation parameters

Parameters	Value(s)
Simulation area	1000m x 1000m
Number of nodes	$N \in [30,300]$
Com. range	R = 100m
Bandwidth	r = 1 Mbit/s
Data packet size	280 bytes
Movement pattern - Max speed - Pause	Random Waypoint - $v \in \{3,10,20,30\} m/s$ - Uniform betw. 0 and 2s
fDelay	10 ms
rDelay	100 ms
Lifetime	200 s
Simulation time	250 s
Send rate	0.01 packets/s

Simulation Results

In this section, we present the simulation results for the global evaluation of hypergossiping. We also show the usefulness of information that we provide in GOD instance to understand the behaviour of hypergossiping and to cristallize out some improvement opportunities for hypergossiping.

- Global Evaluation of Gossiping

The reachability of gossiping should correlate with the partition size. Fig.4(a) shows that the gossiping reachability is as expected below the optimal gossiping reachability (partition_size / total_number_of_nodes). This is due to collisions, which prohibit gossiping to progress, and which become more frequent with increasing number of nodes. Fig.4(b) shows the frequency histogram of the ratio of the number of nodes reached by gossiping to the sender's partition size. This figure shows that in most cases gossiping reaches either more than 90% of the partition nodes or less than 10% of nodes.

This study shows the importance of partition size information we provide.

- Global Evaluation of the Broadcast Repetition

Next we show the simulation results for the global evaluation of the broadcast repetition strategy.

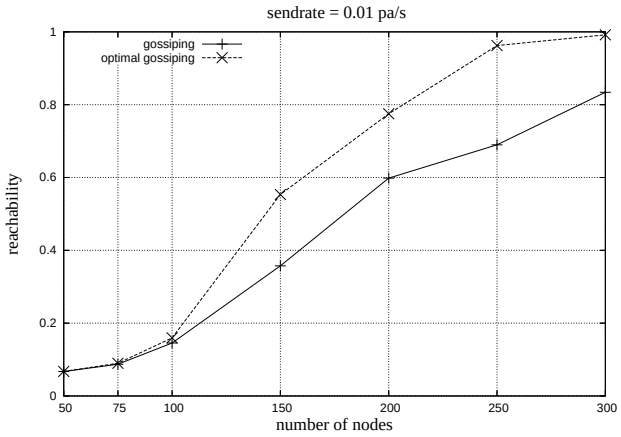
- Study 1: Optimal Broadcast Repetition

Fig.5 shows HG_RE and the values for HG_RE, if the broadcast repetition is optimal, i.e. HG-OBR_RE. For 30 m/s and starting from 100 nodes, the HG-OBR_RE slightly decreases for increasing number of nodes; this is due to collisions (we use the real implementation of gossiping). As expected the HG-OBR_RE is higher than the HG_RE. But for 300 nodes, where the MANET consists of a very large partition and some isolated nodes, hypergossiping reaches a little more nodes than hypergossiping with optimal broadcast repetition. This is due that the broadcast repetition of hypergossiping is able to remedy the gossiping breaks caused by collisions besides overcoming network partitioning [7].

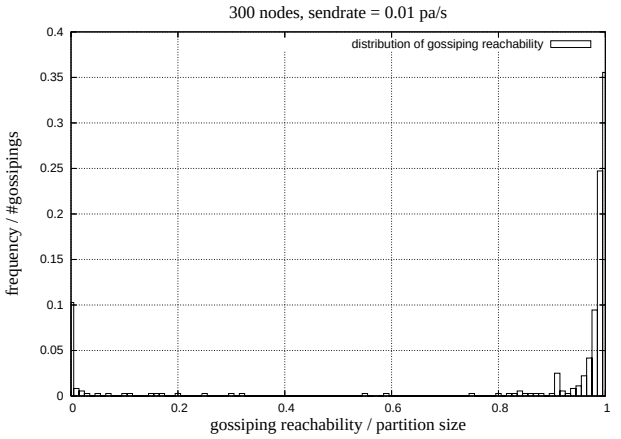
The comparison to the optimal case shows that there is still improvement opportunities to increase the reachability of hypergossiping in sparse MANETs. The improvement potential is higher for higher mobility.

- Study 2: Hypergossiping Observation

Fig.6(a) shows the number of the correct, wrong and redundant detections of the broadcast repetition strategy as well as the doubled number of joins. In Fig.6(b) we present the number of correct, wrong and redundant broadcast repetitions. The Fig.6(a) shows that the number of correct and wrong detections correlates well with the doubled number of joins. We notice that the number of wrong detections is relatively high. Simulation results show that these are mainly the joins that were not detected by the heuristic. We also notice that



(a) G_RE , OG_RE



(b) Frequency histogram of the ratio G_RE to OG_RE

Figure 4: Gossiping evaluation

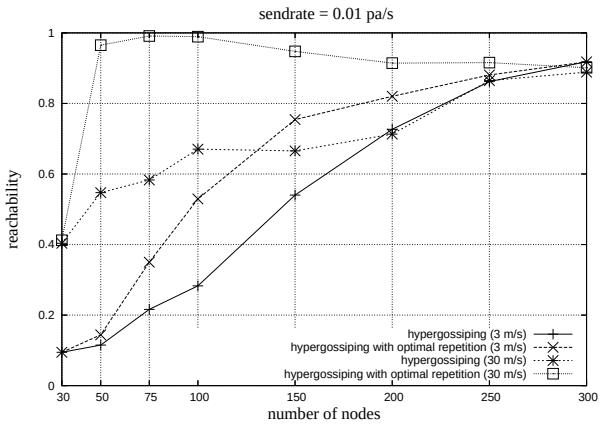
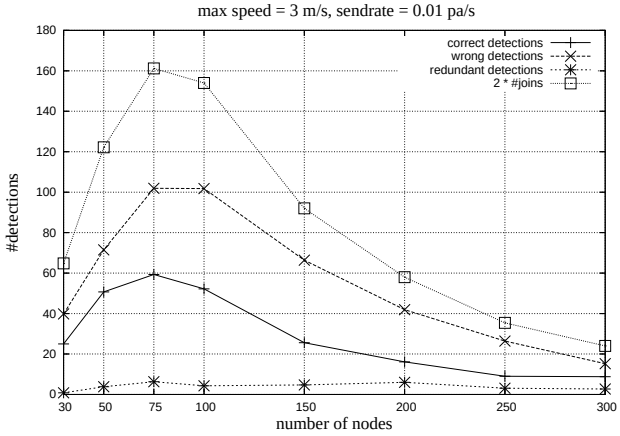
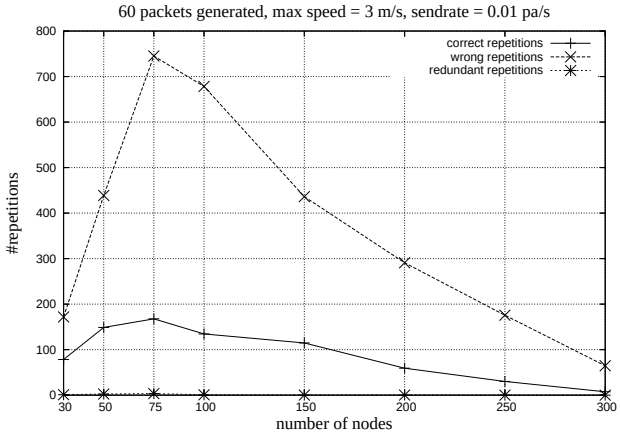


Figure 5: Optimal broadcast repetition

the number of redundant detections respectively the number of redundant broadcast repetitions is very low. This means that our detection heuristic respectively suppression mechanism works well to minimize redundant detections respectively repetitions.



(a) Correct, wrong and redundant detections



(b) Correct, wrong and redundant repetitions

Figure 6: Hypergossiping observation

The observation of the broadcast repetition shows that there is still some potential to improve the partition join detection heuristic. One has to increase the number of correct detections, in order to increase the reachability of hypergossiping efficiently.

CONCLUSION

The observation of the increasing need for partitioning information while developing MANET protocols and applications encouraged us to provide generic partitioning information for the widely used network simulator ns-2. In this paper, we provide a utility, `calcpartition`, to annotate arbitrary movement files with partitioning information and an interface to easily use this information during simulation. We release the source code of `calcpartition` and a patch for the required ns-2 modifications for ns-2 community [1].

We showed by example the applicability of the provided information. For this, we evaluated hypergossiping, a MANET broadcast protocol that considers network partitioning. This global evaluation showed some improvement potentials for hypergossiping, which we will consider in future work.

REFERENCES

[1] J. Hähner, D. Dudkowski, P. J. Marrón, and Kurt Rothermel. A quantitative analysis of partitioning in mobile ad hoc networks. In *Proc. of the Joint Int. Conf. on Measurement and Modeling of Computer Systems (Sigmetrics-Performance) (extended abstract)*, pages 12–16, June 2004.

[2] Vincent Park and M. Scott Corson. Temporally-Ordered Routing Algorithm (TORA). Internet Draft (draft-ietf-tora-spec-04.txt), July 2001.

[3] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, 2000. Duke University.

[4] Anders Lindgren, Avri Doria, and Olov Schelén. Poster: Probabilistic routing in intermittently connected networks. In *Proceedings of The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, June 2003.

[5] Ritesh Shah and Norman C. Hutchinson. Delivering messages in disconnected mobile ad hoc networks. In *Proc. of ADHOC-NOW 2003*, pages 72–83, 2003.

[6] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *ACM/IEEE MOBICOM*, 1999.

[7] Abdelmajid Khelil, Pedro José Marrón, Christian Becker, and Kurt Rothermel. Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. In *Proceedings of The 2005 Conference on Communication in Distributed Systems (KiVS)*, February 2005.

[8] Sanket Nesargi and Ravi Prakash. Manetconf: Confi guration of hosts in a mobile ad hoc network. In *Proc. of IEEE Joint Conference of Computer and Communication Societies (INFOCOM 2002)*, pages 1059–1068, June 2002.

[9] H. Zhou, L.M. Ni, and M.W. Mutka. Prophet address allocation for large scale manets. In *Proc. of IEEE Joint Conference of Computer and Communication Societies (INFOCOM 2003)*, pages 1304–1311, April 2003.

[1] Available for download from http://canu.informatik.uni-stuttgart.de/calcpartition/

[10] J.P.O. Grady, A. McDonald, and D. Pesch. network merger and its influence on address assignment strategies for mobile ad hoc networks. In *Proc. of IEEE Vehicular Technology Conference Fall 2004*, September 2004.

[11] N. H. Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *Proc. of the third ACM international symposium on mobile ad hoc networking and computing (mobihoc 2002)*, pages 206–216, 2002.

[12] Hartmut Ritter, Rolf Winter, and Jochen Schiller. A partition detection system for mobile ad-hoc networks. In *First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, 2004.

[13] Karen Wang and Baochun Li. Group mobility and partition prediction in wireless ad-hoc networks. In *Proceedings of IEEE International Conference on Communications (ICC 2002)*, pages 1017–1021, April 2002.

[14] Karen Wang and Baochun Li. Effi cient and guaranteed service coverage in partitionable mobile ad-hoc networks. In *Proc. of IEEE Joint Conference of Computer and Communication Societies (INFOCOM 2002)*, pages 1089–1098, June 2002.

[15] Michaël Hauspie, David Simplot, and Jean Carle. Partition detection in mobile ad-hoc networks using multiple disjoint paths. In *Proc. 1st International Workshop on Objects models and Multimedia technologies (OMMT)*, 2003.

[16] B. Milic, N. Milanovic, and M. Malek. Prediction of partitioning in location-aware mobile ad hoc networks. In *Proceedings of the Hawaii International Conference on System Sciences, HICSS-38*, 2005.

[17] M. Ahmed, S.V. Krishnamurthy, R.H. Katz, and S. Dao. Trajectory control of mobile gateways for range extension in ad hoc networks. *Comput. Networks*, 39(6):809–825, 2002.

[18] Qun Li and Daniela Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of the sixth ACM/IEEE International Conference on Mobile Computing and Networking (mobicom 2000)*, pages 44–55, August 2000.

[19] S. McCanne and S. Floyd. Ns network simulator.