Mobi-G: Gossip-based Monitoring in MANETs

Dominik Stingl, Reimond Retz, Björn Richerzhagen, Christian Gross, Ralf Steinmetz
Multimedia Communications Lab
Technische Universität Darmstadt, Germany
{stingl,retz,richerzhagen,chrgross,steinmetz}@kom.tu-darmstadt.de

***Abstract*—Monitoring in mobile ad hoc networks (MANETs) constitutes a crucial service, as it enables the adaptation to the changing conditions in those networks based on the monitored state. Many monitoring mechanisms rely on a hierarchical structure, which must be maintained and suffers from sparsely populated networks, where the hierarchy might not be correctly established. Contrary to this structured design, other mechanisms rely on a simple, flat topology using gossiping to aggregate information. However, current gossip-based approaches provide inaccurate results, since they cannot handle the characteristics of MANETs. To overcome these problems and to exploit the flat topology and the robust communication pattern of gossiping for the monitoring of MANETs, this paper introduces Mobi-G. Mobi-G is a flat approach that gossips to exchange information instead of gossiping to aggregate them. It consists of a flexible protocol that exploits the characteristics of wireless communication, handles the mobility of nodes, and operates even in sparsely populated networks to provide accurate results at minimum cost.**

I. Introduction

Mobile ad hoc networks (MANETs) represent an useful substrate to deploy different types of applications in urban areas on mobile communication devices. Due to the inherent dynamic nature of those networks, arising, e.g., from the autonomy or mobility of nodes, MANETs must be flexible and adapted to the changing conditions of the environment and the network. To implement this adaptation, it is inevitable and mandatory to monitor a MANET to determine its current state. Based on the monitored information, the participating nodes can adapt themselves to the detected and provided state.

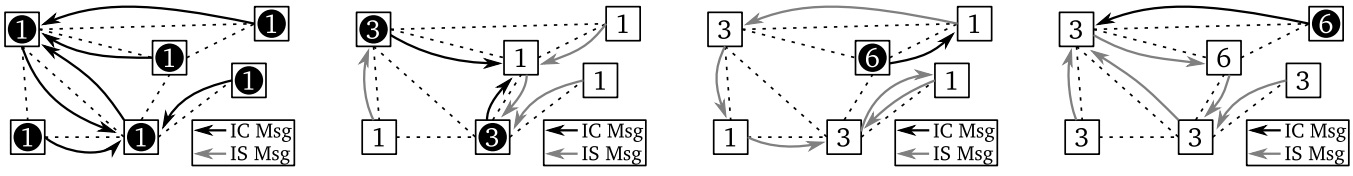
The envisaged monitoring mechanisms, which (i) monitor a MANET, (ii) generate a recent view on the system state, and (iii) disseminate the information among nodes, must handle the sketched dynamics as well. As summarized in [18], a monitoring mechanism for MANETs, fulfilling the aforementioned functional requirements, must provide *accurate* and *timely* information, while (i) scaling with the *spatial size* and *node density* of the network, (ii) withstanding the *mobility* and *fluctuating presence* of nodes, and (iii) handling the limited communication range and unreliable communication medium.

Monitoring mechanisms for MANETs, tackling these functional and non-functional requirements, can be divided into hierarchical and flat approaches. Hierarchical approaches either rely on a set of responsible nodes to create the hierarchy as well as collect and disseminate the data [1], [10], [15], or integrate each node in the hierarchy and monitoring process to increase robustness [6], [18]. Using the hierarchy to structure the nodes and to identify relevant communication partners, hierarchical approaches also operate in larger spatial or populated networks. On the downside, a monitoring mechanism must manage its hierarchy and is bound to the resulting structure and associated communication rules. Especially in sparsely populated networks, hierarchical monitoring mechanisms suffer from these specifications: (i) if a set of nodes collects and disseminates data, the availability depends on the accessibility of those nodes, which decreases in sparsely populated networks; (ii) if all nodes are integrated into the monitoring process, the monitoring mechanism logically partitions the network into areas and requires that these areas are populated with nodes to avoid data loss.

In contrast to hierarchical approaches, other solutions rely on a flat topology and apply gossiping [2] to communicate. These *gossip-based* monitoring mechanisms neither maintain a hierarchy nor rely on predefined relations between nodes or areas. Instead, they can cope with a constantly changing network topology and exchange information with any other node in the network. For fixed communication networks, several approaches exist [7], [21], [23] that provide accurate and timely information about the current system state. Due to the sketched flexible design, the approaches are highly robust and outperform hierarchical approaches in wired communication networks [17], [19]. However, new problems arise when porting these solutions to mobile networks. Especially gossip-based approaches, which rely on the concept of *mass conservation* [8] suffer from the characteristics of MANETs. The upcoming problems comprise (i) the missing long-range connectivity accompanied with a high network diameter and (ii) the high probability of message loss due to the error-prone communication medium or the limited communication range.

To overcome the shortcomings of gossip-based monitoring mechanisms in MANETs, while exploiting their flat topology and robust communication pattern, this paper introduces Mobi-G, which relies on gossip-based communication, but abstains from the concept of mass conservation. Instead, it adapts concepts from Gossipico [22], which was initially developed for wired networks. Mobi-G exploits the positive characteristics of ad hoc communication to cope with the limited communication range as well as the missing long-range connectivity. Based on its flexible protocol, which relies on a time-based synchronization, Mobi-G handles the dynamic nature of MANETs and provides accurate results. The evaluation consists (i) of a detailed parameter analysis to show Mobi-G's applicability and (ii) of a comparison with the hierarchical approach BlockTree [18]. The results outline that Mobi-G is robust, works in large spatial networks, and outperforms BlockTree in sparsely populated networks at considerably smaller cost.

In the following, the design of Mobi-G is presented in Section II. Section III consists of the system parameter and comparative evaluation with BlockTree. An overview about



(a) First cycle of COUNT (b) Second cycle of COUNT (c) Third cycle of COUNT (d) Fourth cycle of COUNT

Figure 1. The first four cycles of the COUNT procedure with the exchange of Information Collecting and Information Spreading messages

related work in the area of monitoring in MANETs is given in Section IV. Finally, Section V concludes the paper.

II. System Design

Mobi-G is designed for MANETs in urban outdoor areas, such as large places or cities, with a focus on pedestrians that move around by foot. It is assumed that the mobile handheld communication devices, such as smartphones or tablets, can communicate over Wi-Fi ad hoc and determine their current position, e.g., using GPS.

As specified in [17], *attributes* are used to define, which aspects of a MANET are monitored. Each node periodically measures the local values of the defined attributes, which are collected and processed. Subsequently, the processed monitoring results are disseminated to inform other nodes about the current system state. Due to the fact that Mobi-G just gossips to *exchange* data, it can apply different techniques to process the monitored information. In contrast, gossip-based monitoring mechanisms [7], [21], [23], which rely on the mass conservation paradigm [8], are mainly limited to aggregation, because they gossip to *aggregate* data.

As Mobi-G is targeted at larger networks with many nodes, it must compress the size of the collected monitoring information. Consequently, it relies on aggregation to process and compress the data. However, it abstains from gossiping to aggregate data, but applies aggregation functions, which comply with the hierarchical computation property [12], [25]. Hence, the collected data are aggregated and, as specified in [17], the *global view* of an attribute is created, ideally incorporating the input from all nodes in the network.

For the previously sketched collection and dissemination of monitoring data, Mobi-G adapts two concepts from Gossipico by van de Bovenkamp et al. [22], which is targeted at fixed networks. In the next section, the initial concepts are presented and serve as background to understand the design of the proposed monitoring mechanism Mobi-G, as described in Section II-B.

A. Basic System Design of Gossipico

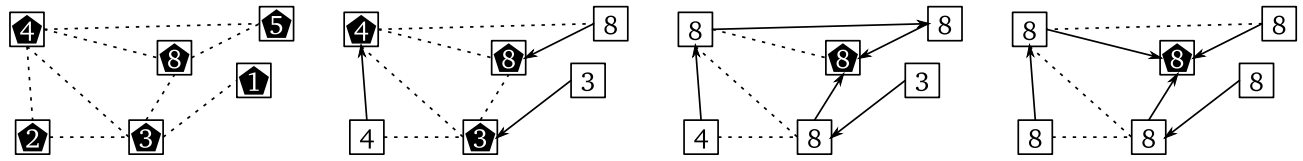
Gossipico consists of two procedures denoted as COUNT and BEACON, which serve as basis for Mobi-G. Similar to other gossip-based approaches, Gossipico relies on cycles to coordinate the periodic execution of COUNT and BEACON. Gossipico abstains from the definition of an epoch to synchronize the protocol and to restart COUNT and BEACON but applies other mechanisms instead. The omission of epochs enables continuous monitoring of attributes, as detailed in the following.

1) The COUNT Procedure: COUNT is used to collect the monitored attributes and disseminate the calculated results in the network. Gossipico introduces the concept of *tokens*, which represent the monitored attributes and are exchanged between nodes, using *Information Collecting (IC)* messages. If a node does not hold tokens, it sends *Information Spreading (IS)* messages to inform other nodes about the observed state of the network. Figure 1 displays the first four cycles of the COUNT procedure after its start. The dashed lines display the known neighbors of a node, while the grey and black arrows represent transmitted IC and IS messages. The black circles illustrate the tokens and the number inside represents an example of a monitored attribute, for instance, the node count. During the first cycle (cf. Figure 1(a)), each node has its own *token*, representing the locally measured values, which is sent to a random neighbor using IC messages. Received tokens are combined with the currently available tokens and forwarded to a neighbor during the next cycle: in Figure 1(b), the tokens are collected and processed by two nodes. Based on this approach, tokens accumulate at different nodes in the network until all tokens are combined (cf. Figure 1(c) and 1(d)). Meanwhile, nodes without tokens send IS messages to inform other nodes about the observed network state. Since gossiping is used to communicate instead of aggregating, the possibilities to combine received tokens are much broader. If aggregation is used to combine the tokens, Gossipico even satisfies the hierarchical computation property [12], [25].

As Gossipico just relies on cycles to synchronize and coordinate the monitoring process, continuous monitoring can be implemented to keep the global view of an attribute always up-to-date. Therefore, nodes always create and send new tokens if the local value of a monitored attribute changes.

2) The BEACON Procedure: To speed up the token collection, the BEACON procedure is introduced, which guides the data to an autonomously selected leader of the network, denoted as *beacon*. The beacon announces its presence in the network by means of gossiping and creates shortest paths for fast data collection. Figure 2 displays the initial, two intermediate, and the final state of the procedure. Again, the dashed lines represent the known neighbors of a node, while the solid arrows represent a potential shortest path to the leader.

For the determination of the single beacon, where all data will be collected, nodes compete with each other. Van de Bovenkamp et al. [22] rely on the metaphor of battling armies to describe the BEACON procedure. Each node starts as a beacon of its own army, also constituting the leader of that army. Beacons are indicated by the black pentagons, as displayed in Figure 2(a). Besides, each node knows the randomly determined army strength, which is represented by the numbers in Figure 2. Furthermore, it is aware of the estimated hop count and the next hop to the beacon of the army, it currently belongs to.



(a) Initial state of BEACON (b) Intermediate state of BEACON (c) Intermediate state of BEACON (d) Final state of BEACON

Figure 2. Different states of the BEACON procedure

During a cycle, a node randomly selects another node for the exchange of *army messages* to determine the strongest army with the corresponding ruling beacon in the network. If they belong to different armies, the stronger army wins and incorporates the defeated node and its army (cf. Figure 2(b)). If both nodes belong to the same army, they just update the information about the next hop and hop count to the beacon. As displayed in Figure 2(c) and 2(d), the upper left node updates its next hop of the path to the beacon, whereas the nodes with a former army strength of three and four join the strongest army.

To handle link or node failures, the basic assumption is that a node detects such a failure in its neighborhood, rebuilds its army, and starts a recount. The recount works if a node only accepts IC messages from its own army and ignores IC messages from other armies. To rebuild an army, a node *revives* its army with a new randomly selected strength. The new army is *immune* against the army, which the reviving node left so that the new army always defeats the previous one.

B. Mobi-G: Bridging the Gap to MANETs

Mobi-G is a flat monitoring mechanism for MANETs that relies on gossiping to communicate and adapts the concepts from Gossipico [22] to manage nodes and data. Mobi-G is designed to exploit the characteristics of MANETs. Therefore, the approach adapts its communication to the characteristics of the wireless communication medium and uses a protocol that handles mobile and fluctuating nodes, while managing the data collection and dissemination.

1) Adapting the Communication: Mobi-G does not require a routing protocol to exchange data but uses its own routing methods to communicate. On the one hand, this decision leads to additional traffic besides an already running routing protocol, because monitoring data cannot be piggybacked. On the other hand, Mobi-G operates independently of the deployed routing protocol. As proposed by Nanda and Kotz [13], the decentralized monitoring mechanism can still operate and monitor the MANET if the underlying routing protocol fails.

Mobi-G mainly relies on broadcasting (i) to counteract the missing long-range connectivity and (ii) to reduce the utilized upload bandwidth. Since broadcasting does not guarantee a successful delivery, it is used for the information exchange, which can cope with data loss. Consequently, IS and army messages are solely broadcasted to (i) spread the global view of attributes, (ii) determine the strongest army, and (iii) refresh the shortest paths. In contrast, IC messages are sent via unicast and are acknowledged by the receiver to increase the probability of a successful transmission and to avoid duplicate tokens.

a) Efficient Information Dissemination in Mobi-G: For the efficient and fast dissemination of information, IS and army messages are spread to the whole network. In contrast to Gossipico, each cycle of Mobi-G triggers a dissemination of IS and army messages, which are directly forwarded, independent of the clocking cycle. To reduce the number of duplicate transmissions, while still providing a reliable dissemination, Mobi-G relies on contention-based forwarding schemes [3], [26]. Each receiving node calculates a *hesitation time* based on the distance between the sender $\overrightarrow{s}$ and the receiver $\overrightarrow{r}$. Given the assumption of the maximum communication range r_{max} of a device, the hesitation time is calculated based on (1), where h_{max} represents the maximum hesitation time. Both parameters, r_{max} and h_{max}, are system parameters, which must be determined and set by the user to configure Mobi-G. Equation (1) assigns smaller values to distant nodes to speed up data dissemination. If a hesitating node receives the forwarded message, it drops its current message.

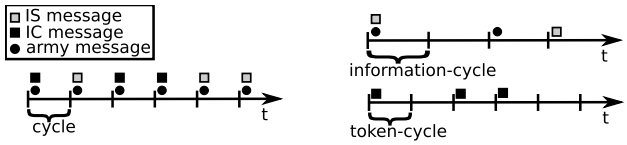
$$f_{diss}(\overrightarrow{r}) = h_{max} * \left(1 - \frac{\|\overrightarrow{s} - \overrightarrow{r}\|_2}{r_{max}}\right) \tag{1}$$

To avoid that redundant information are disseminated, the content of a received message is evaluated: (i) only IS messages with a higher freshness are forwarded; (ii) army messages are passed, if the sender belongs to a stronger army or if a shorter path to the beacon exists.

b) Responding to Missing or Old Information: To avoid that a node operates on stale information due to the unreliable communication or to an intermittent connection, a *response procedure* is introduced. The response procedure identifies nodes with stale information and initiates an update to refresh the information. Therefore, the receiver of a broadcasted message with old information broadcasts a response message containing new information. Similar to the dissemination of IS and army messages, a receiver-based contention scheme is used. In contrast to the calculation of the hesitation time for the dissemination of IS and army messages, the response operation favors nearby nodes to ensure that the former sender $\overrightarrow{s}$ receives the recent updates and does not leave the communication range of the answering node $\overrightarrow{r}$. Consequently, (2) assigns smaller values to nearby nodes.

$$f_{resp}(\overrightarrow{r}) = h_{max} * \frac{\|\overrightarrow{s} - \overrightarrow{r}\|_2}{r_{max}} \tag{2}$$

2) The Monitoring Protocol: The presented modifications are necessary to handle the inherent characteristics of MANETs. To deal with the missing long-range connectivity, the cycle, in which periodic tasks are triggered, is modified. The restart procedure for the collection and dissemination of data as well as for the identification of the ruling beacon is changed. Moreover, paths to the beacon are periodically refreshed to guarantee for a good token convergence despite constantly changing neighbors.



(a) Single cycle synchronization of Gossipico

(b) Dual cycle synchronization of Mobi-G

Figure 3. Adaptation of the cycle synchronization in Mobi-G

a) Cycle Adaptation: Gossipico relies on a single cycle to trigger the periodic transmission of all message types, as shown in Figure 3(a). To counteract the missing long-range connectivity and speed up the token collection, the cycle length should be decreased. On the other hand, shorter cycles lead to an unnecessary high frequency of IS and army messages. Therefore, Mobi-G relies on two parallel cycles (cf. Figure 3(b)): (i) the *token-cycle* triggers the transmission of IC messages and is configured by the *Token-Send-Delay*-parameter; (ii) the *information-cycle* spreads IS and army messages and is configured by the *Refresh-Timeout*-parameter.

An interval of the token-cycle is used to collect and combine incoming tokens or to forward present tokens. If a node does not have a token, it does not trigger the transmission of an IC message. When setting the Token-Send-Delay, the following trade-off has to be considered. A small interval leads to a fast transmission of tokens, helping to tackle the missing long-range connectivity. If the Token-Send-Delay is too small, a receiving node directly forwards the new token. As a result, tokens are not gathered and combined at a node, which leads to separate transmissions and higher traffic.

In contrast, the information-cycle is used to disseminate IS and army messages at a lower frequency. Since IS and army messages are broadcasted, it is not necessary that each node sends a message during each interval. Thus, if a hesitating node already receives an IS or army message, it can drop its hesitating message if the received information match the local information. If the received information differ, a node has two possibilities to react: (i) if the message contains stale information (i.e. lower freshness, weaker army, longer path to the beacon), the receiving node starts the response procedure to correct the stale entries; (ii) if the message contains new information (i.e. higher freshness, stronger army, shorter path to the beacon), the receiving node disseminates the information. When Mobi-G reaches a steady state, where the information at the nodes do not differ, the transmission of information is reduced to a subset of nodes, while the majority drops the hesitating messages.

b) Counteracting Node Mobility: The prevailing mobility in MANETs heavily influences the paths towards the beacon and falsifies the information about the next hop and hop count to the beacon. This problem becomes apparent if the distance between the node and the beacon increases. A node maintains a low hop count to the beacon and ignores subsequent paths with a higher hop count, although, they correctly reflect the current situation. To mitigate this problem, the local hop count of a node increases as a function of time so that longer but valid paths are accepted. In addition, the next hop towards the beacon can leave the communication range of a node, which becomes apparent, if an IC message is not acknowledged. To detect a new next hop, a node directly broadcasts an army message with a high hop count, which triggers the response procedure at the receiving node. During the next interval of the token-cycle, the sending node uses the initiator of the response procedure as next hop to the beacon.

To handle link or node failures, Gossipico refrains from a time-driven restart mechanism, but restarts COUNT and BEACON if a link or node failure is detected. This detection states a problem in MANETs, since nodes might just have left the communication range. Thus, Mobi-G applies a time-driven instead of the so-called churn-driven restart mechanism. In contrast to Gossipico, but similar to [7], [21], [23], Mobi-G relies on epochs to restart the data collection and dissemination as well as beacon identification. At the beginning of a new epoch, each node becomes the beacon of its own new army, which is immune to the ruling army of the previous epoch. After restart, the nodes determine the strongest army with its corresponding beacon. If Mobi-G applies *discrete monitoring*, each node uses the current local value of an attribute during the restart as input for the new epoch. If Mobi-G applies *continuous monitoring*, each node generates new tokens during an epoch if an attribute changes. Thus, Mobi-G implements continuous monitoring, though relying on a time-driven restart mechanism.

III. Evaluation

The conducted evaluation of Mobi-G consists of a detailed examination of the system parameters to understand how they influence performance and cost. Subsequently, scalability of Mobi-G is evaluated and compared against the hierarchical approach BlockTree [18], which constitutes a state-of-the-art monitoring mechanism for MANETs.

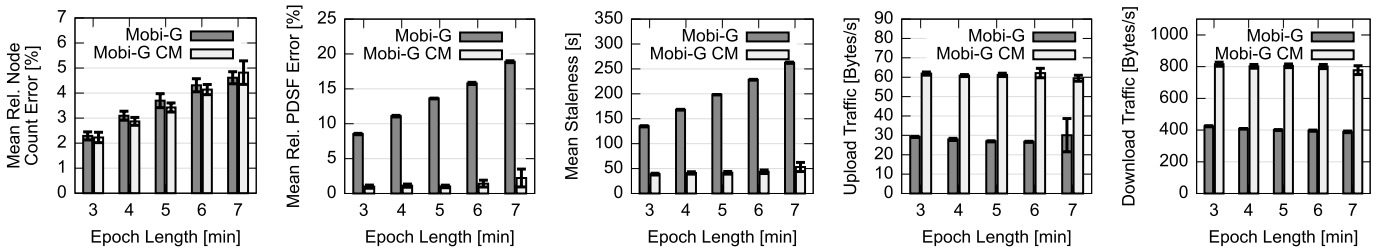
A. Simulation Setup

All experiments are simulated with a modified version of PeerfactSim.KOM [20], which simulates wireless communication and models the environment and node mobility. For the system parameter evaluation, a default scenario is defined, which is used within each experiment. The corresponding scenario parameters are summarized in Table I. A plain quadratic map with an edge length of 1875 m is modeled and populated with 1350 nodes that move according to the Gauss-Markov Mobility Model [11] with a maximum speed of $2\,m/s$. To model the sojourn time of a node, the node session times are exponentially distributed with a mean of 16 min.

Table I. Default values for the scenario parameters

Scenario Parameter	Value
Edge length	1875 m
Number of nodes	1350
Mobility model	Gauss-Markov [11]
Maximum node movement speed	$2\,m/s$
Mean node session length	16 min

According to [17], [19], the performance of a monitoring mechanism is split into accuracy and staleness. For the quantification of *accuracy*, the relative error between the monitored $\hat{x}$ and effective global view x of an attribute is calculated at each node by $|\frac{\hat{x}}{x} - 1|$. The monitored attributes that serve as input for Mobi-G consist of the *node count* as well as of a *Poisson Distributed Sine Function* (PDSF) with a period of one hour. The node count is used, as it constitutes a de-facto standard to evaluate the accuracy [17], [19]. The PDSF attribute is used (i) to examine how Mobi-G monitors a



(a) Mean rel. node count error (b) Mean rel. PDSF error (c) Mean result staleness (d) Upload traffic per node (e) Download traffic per node

Figure 4. Influence of the system parameter epoch length on performance and cost of Mobi-G

fluctuating attribute and (ii) to outline the effect of continuous monitoring. The PDSF assigns each node a separate sine function, which is shifted along the y-axis according to a Poisson distribution. The relative error is calculated based on the monitored and the effective average of PDSF. In terms of *staleness*, $t_{stale} = t_m - t_{avg}$ is measured, which represents the time between the measurement t_m taken by the simulator and the average age t_{avg} of all aggregated values.

Dealing with the arising *cost*, which can be divided into communication, memory, and computation cost for the approximation of the global state (cf. Keshav [9]), the evaluation only focuses on communication cost. Therefore, the up- and download traffic per node is measured.

Each experiment is simulated for two hours. During the first hour, the simulation reaches its steady state and the second hour is used to capture the data for later analysis. If not stated otherwise, each experiment is repeated ten times and the corresponding plots display the average mean of a simulation with a confidence of 95%.

Table II. SYSTEM PARAMETER VARIATIONS OF MOBI-G

System Parameter	Value
Epoch length [min]	3, 4, 5, 6, 7
Token-Send-Delay [s]	0.5, 1, 2, 4, 8, 16
Refresh-Timeout [s]	0.5, 1, 2, 4, 8, 16

B. System Parameter Evaluation

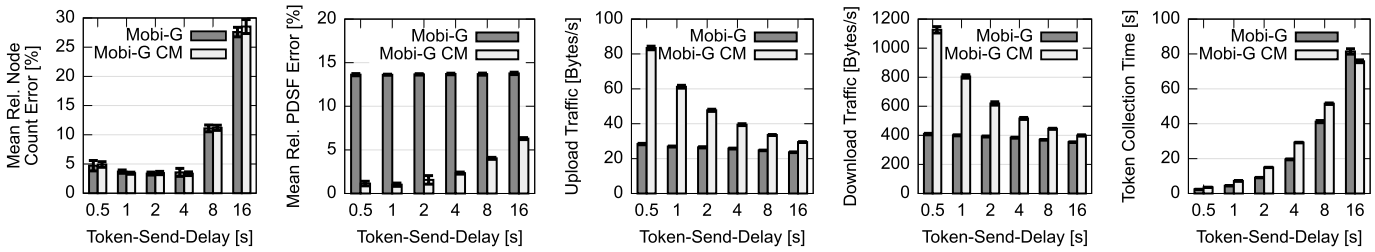
Table II lists the evaluated system parameters, where the underlined values represent the default values, while other parameters are varied. The experiments comprise the results for both discrete monitoring, labeled as *Mobi-G*, and continuous monitoring, labeled as *Mobi-G CM*.

1) Epoch Length: Starting with a brief evaluation of the impact of *continuous monitoring*, Figure 4(a) shows that continuous monitoring does not increase the accuracy when counting active nodes. Since nodes just disappear without sending a new token to inform other nodes, continuous monitoring cannot reflect the changing number of nodes. In contrast, the results in Figure 4(b) outline the positive impact of continuous monitoring on accuracy when measuring data from fluctuating attributes. If continuous monitoring is disabled, Mobi-G obtains its values from the beginning of an epoch, whereas continuous monitoring collects and integrates new values during the whole epoch. The same trend can be observed for staleness (cf. Figure 4(c)), where the continuous integration of new values leads to fresh results. Considering the traffic (cf. Figure 4(d) and 4(e)), it can be observed that the performance gain comes at the expense of an increased communication overhead. In terms of discrete monitoring, the current global view of an attribute remains constant and is disseminated only one time during an epoch, once all tokens have been collected at the beacon. In contrast, continuous monitoring results in a constant collection of tokens and update of the global view, doubling the traffic of Mobi-G.

Focusing on *epoch length*, Figure 4(a) outlines that a shorter epoch length reduces the mean relative monitoring error of node count. The monitoring error, originating from leaving nodes, can only be corrected by the time-controlled restart mechanism, which depends on the epoch length. Thus, an earlier restart due to a shorter epoch length leads to a reduced integration of offline nodes in the global view of an attribute. This effect can also be observed for discrete monitoring in terms of PDSF (cf. Figure 4(b)), because an earlier restart enables the integration of recent values and increases the accuracy of the monitoring results. Since continuous monitoring always integrates recent values, the influence of a short epoch length nearly disappears. For the same reasons, the mean result staleness decreases for a shorter epoch length in terms of discrete monitoring, whereas there is little impact on staleness when considering continuous monitoring (cf. Figure 4(c)).

As displayed in Figure 4(d) and 4(e), there is a slight influence of the epoch length on traffic. Though, a shorter epoch leads to a frequent identification of beacons and collection of tokens, the more frequent execution of these operations does not affect the traffic. Most of the traffic originates from IS and army messages, which are not influenced by the epoch length.

2) Token-Send-Delay: Changing the Token-Send-Delay influences the time to collect all tokens at the reigning beacon. Figure 5(e) displays the mean lifetime of a token from its creation to its arrival at the beacon as a function of the Token-Send-Delay. If tokens are aggregated on their way to the beacon, the older creation time is kept as reference. The displayed results have been filtered to remove outliers and comprise 95% from the overall data. Figure 5(e) displays the immense impact of the Token-Send-Delay on the token collection time, which influences the accuracy of Mobi-G's discrete and continuous monitoring in different ways. Based on Figure 5(a), it can be observed that a longer token collection time results in an increased relative node count error. On the other hand, the results for a shorter interval reveal that a reduction does not improve the performance but increases the traffic below a certain parameter value (cf. Figure 5(c) and 5(d)). Figure 5(b) displays the negligible impact of Token-Send-Delay on the accuracy of discrete monitoring of PDSF due to the already existing high error. For continuous monitoring a higher Token-Send-Delay decreases the accuracy, since the continuously created tokens require more time to reach the reigning beacon. The plot for staleness is omitted, since the results resemble the ones for the mean relative PDSF error.



(a) Mean rel. node count error (b) Mean rel. PDSF error (c) Upload traffic per node (d) Download traffic per node (e) Token collection time

Figure 5. Influence of the system parameter Token-Send-Delay on performance and cost of Mobi-G

Figure 5(c) and 5(d) show that a higher Token-Send-Delay decreases the resulting traffic, because a higher value reduces the cycles to transmit tokens during an epoch. In terms of discrete monitoring, the traffic reduction is small, since the tokens are ideally collected at the beginning of an epoch, while the remaining cycles are not used. In contrast, continuous monitoring results in a constant transmission of tokens. With a longer Token-Send-Delay, there are fewer cycles to transmit tokens and the probability increases that they are aggregated at a node, leading to the displayed traffic reduction.

3) Refresh-Timeout: For the evaluation of Refresh-Timeout only the mean relative node count error is examined in terms of performance, since the results for PDSF and staleness exhibit similar tendencies. The considered results reveal that an increasing Refresh-Timeout has little impact on the accuracy (cf. Figure 6(a)), but reduces the traffic (cf. Figure 6(b) and 6(c)). The traffic decreases for an increasing Refresh-Timeout, because there are fewer cycles per epoch to send IS and army messages. Considering accuracy (cf. Figure 6(a)), the small influence on accuracy can be explained when looking at the results for the army conquest time and information dissemination time, which depend on the periodic transmission of IS and army messages. The army conquest time in Figure 6(d) displays the duration for the strongest army to conquer the whole network since its creation. The information dissemination time in Figure 6(e) shows the time interval from the dissemination of an IS message by the reigning beacon and its reception at the nodes. Similar to the token collection time (cf. Figure 5(e)), the displayed results comprise 95% from the overall data. In Figure 6(d), it can be observed that a longer Refresh-Timeout increases the mean time to conquest the whole network about 20 s. Based on the results in Figure 6(e), the influence on the information dissemination time is even less, because a longer Refresh-Timeout extends the mean dissemination time about one second for discrete and about 0.5 s for continuous monitoring. Compared to the high influence of the Token-Send-Delay on the token collection time (cf. Figure 5(e)) and on accuracy (cf. Figure 5(a)), it becomes apparent that the small increase of the army conquest and information dissemination time just slightly affects discrete and continuous monitoring.

4) Summary: Based on the results, it can be observed that continuous monitoring provides highly accurate and fresh results for varying attributes at the expense of an increased traffic. In terms of node count, the difference between continuous and discrete monitoring is negligible. To increase accuracy and freshness of discrete monitoring as well, the epoch length represents a cost-effective alternative, since a shorter epoch considerably improves accuracy and staleness, while the influence on traffic is negligible. The results for the Token-Send-Delay reveal that this parameter strongly influences the accuracy for discrete and continuous monitoring depending on the monitored attribute. A large Token-Send-Delay reduces the traffic at the expense of inaccurate results, whereas too small values increase the traffic without improving accuracy. The results for Refresh-Timeout outline that an independent transmission of IS and army messages through the separate information-cycle is a beneficial extension, since the traffic is reduced, while accuracy and staleness remain nearly constant.

Based on these outcomes, the final system parameter configuration of Mobi-G consists of (i) an Epoch Length of 3 min, (ii) a Token-Send-Delay of 2 s for the provisioning of accurate results without an excessive creation of traffic, and (iii) a Refresh-Timeout of 8 s to reduce the resulting traffic while keeping the results accurate and fresh.

C. Comparative Evaluation

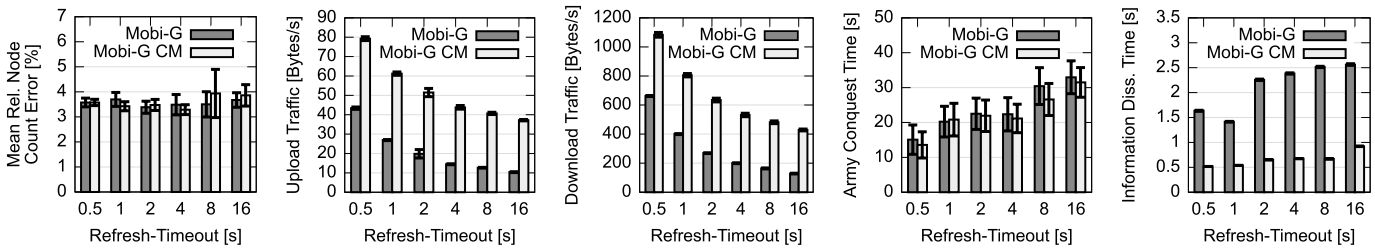
After the system parameter evaluation, this section outlines how Mobi-G scales (i) with the spatial network size and (ii) the node density. The variations for both experiments are listed in Table III. When considering different spatial network sizes, the number of nodes is increased as well to obtain the same node density in each scenario. For node density, the spatial network size is fixed, while the number of nodes is varied.

Table III. SCENARIO PARAMETER VARIATIONS FOR THE COMPARATIVE EVALUATION

Scenario Parameter	Value
Edge length [m]	312, 625, 1250, 1875, 2500
Node density [nodes/km^2]	96, 192, 288, 384, 480

For a better assessment of the results, Mobi-G is compared against BlockTree [18], labeled as *BT* in the plots. The relevant parameters of Mobi-G are configured as summarized in Section III-B4, while BlockTree's most important system parameter *Update Interval* is set to eight seconds, which triggers the periodic operations. Although BlockTree already performs well for a larger Update Interval [18], the configuration is oriented at the Refresh-Timeout of Mobi-G so that both monitoring mechanisms trigger most of the operations with the same interval. In contrast to the previous setups, each experiment is repeated five times.

1) Spatial Network Size: The results for the mean relative node count error in Figure 7(a) outline that BlockTree outperforms Mobi-G for networks with an increasing spatial size. While a hierarchy helps to organize the collection and dissemination of data, a flat and gossip-based design suffers from the larger network diameter, leading to the increasing error. Considering the accuracy of PDSF (cf. Figure 7(b)), the error for Mobi-G with discrete monitoring is nearly constant,



(a) Mean rel. node count error (b) Upload traffic per node (c) Download traffic per node (d) Army conquest time (e) Information diss. time

Figure 6. Influence of the system parameter Refresh-Timeout on performance and cost of Mobi-G

whereas it increases for continuous monitoring with a growing spatial size. However, the results outline that BlockTree cannot keep up with Mobi-G's continuous monitoring. Its *on-demand* creation and collection of tokens leads to a higher accuracy, especially when monitoring fluctuating attributes. On the other hand, continuous monitoring does not suffice to provide as fresh results as BlockTree (cf. Figure 7(c)), which benefits from its short Update Interval.

Regarding the cost (cf. Figure 7(d) and 7(e)), BlockTree's good performance comes at the expense of considerably higher traffic. A growing spatial network size increases the traffic of all three approaches but also the gap between BlockTree and both alternatives of Mobi-G. For Mobi-G, a larger area does not necessarily lead to increased traffic on all nodes, but only on those nodes, that currently reside on the route towards the beacon. Since many tokens can already be combined on their way to the beacon, the collection traffic can be reduced. BlockTree handles the increasing size by creating a new hierarchy level. This level leads to an additional information exchange, which affects every node and results in the drastically increasing traffic.

2) Node Density: In terms of accuracy, it can be observed that the node density influences Mobi-G and BlockTree (cf. (Figure 8(a) and 8(b)). While Figure 8(b) still hides the bad influence of sparsely populated areas on BlockTree, the results in Figure 8(a) reveal this problem. BlockTree suffers from sparsely populated areas, because parts of the hierarchy are not populated so that information are not collected and disseminated over the hierarchy as envisaged. In contrast, Mobi-G compensates the low density by its flat design and the gossip-based communication, which does not rely on predefined paths for data collection and dissemination. As long as army messages identify paths to the beacon, Mobi-G is able to monitor sparsely populated networks.

Figure 8(c) and 8(d) display the typical traffic characteristics of receiver-based contention schemes for monitoring mechanisms, as observed in [18]. Denser populated areas increase the download traffic per node (cf. Figure 8(d)), because the probability increases that new or redundant data will be received. On the other hand, Figure 8(c) outlines that the upload traffic per node decreases with an increasing node density. Especially when disseminating IS or army messages, there are more potential senders involved, which reduces the probability that a single node disseminates the information several times. Similar to the results for the increasing spatial network size, Mobi-G produces considerably less traffic than BlockTree. Moreover, there is only a small increase of Mobi-G's download traffic when monitoring denser network.

IV. Related Work

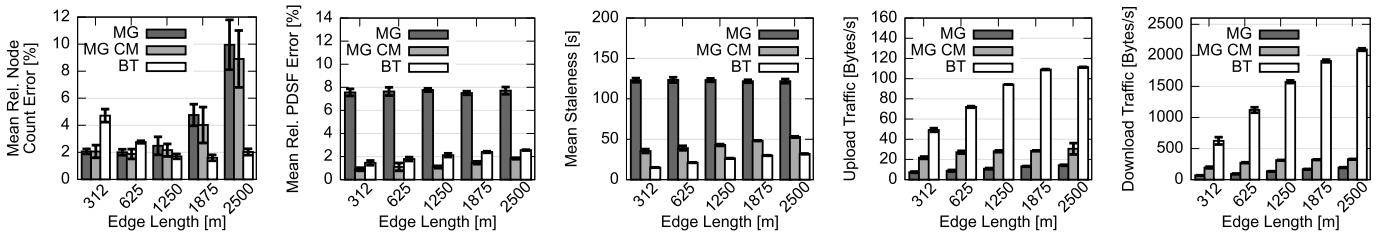
In the area of monitoring mechanisms for MANETs, several approaches have been developed, trying to provide an accurate view on the network, while handling the characteristics of MANETs. Out of these approaches, an overview about gossip-based and hierarchical approaches is given and discussed.

Starting with gossip-based monitoring mechanisms, some approaches exist that monitor a mobile network but do not directly address MANETs. The local push-sum protocol (LPS) [4] constitutes an extension of push-sum [8], which is targeted at *static* wireless networks, where nodes just communicate with their direct neighbors and do not generate a global view over the whole network, but only over a part of it. Mobi-G and LPS rely on leaders to coordinate and accelerate the data collection. In contrast, Mobi-G relies on gossiping to communicate, whereas LPS gossips to aggregate.

Guerrieri et al. [5] monitor delay tolerant networks (DTNs) relying on the Pairwise-AVG algorithm [7] and applying concepts of population algorithms. At the beginning, each node generates a token for a monitored attribute, which is collected and aggregated by a node if two nodes meet. This procedure is repeated until the aggregate of an attribute contains all tokens. Mobi-G relies on tokens as well, but uses beacons to speed up the collection and dissemination. The evaluation shows that the beacon concept provides accurate results in MANETs.

Wuhib and Stadler [24] introduce a modified version of G-GAP [23] that targets MANETs and adapts Push-Sum [8]. Besides G-GAP, the authors introduce an optimal version of G-GAP, which is denoted as G-GAP-NL. For G-GAP-NL it is assumed that no message loss occurs over the wireless communication medium. The evaluation outlines that monitoring mechanisms, which gossip to aggregate data, just perform well under simplified assumptions (i.e., no message loss). In contrast, Mobi-G is capable of handling the typical characteristics of MANETs and provides accurate results.

In addition to the surveyed gossip-based monitoring mechanisms, hierarchical approaches for MANETs are introduced and discussed. The underlying hierarchy serves for the collection of data and is created in a decentralized fashion using different techniques to identify nodes for the higher levels: the approaches (i) rely on proximity-based clustering with cluster heads [10]; (ii) apply mathematical functions [1] or simple comparisons with the neighborhood [16] to determine appropriate and powerful nodes; (iii) become a node at a higher level depending on the density of higher level nodes [15]; or (iv) are provided and defined by a network operator [14]. To obtain a partial or global view of an aggregate, the remaining



(a) Mean rel. node count error (b) Mean rel. PDSF error (c) Mean result staleness (d) Upload traffic per node (e) Download traffic per node

Figure 7. Influence of the spatial network size on Mobi-G and BlockTree

nodes can query the nodes from higher levels. In contrast, Mobi-G does not require an underlying hierarchy to collect and disseminate data. To avoid that the availability of information depends on the accessibility of a single node for a longer time, a node just becomes a beacon for a short time. In addition, the beacon must not serve the whole network, but incorporates the remaining nodes to disseminate the information.

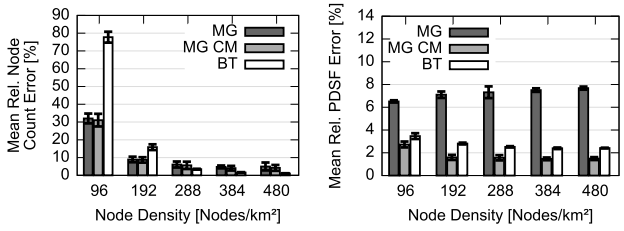
Besides the presented hierarchical approaches, other solutions like BlockTree [18] or the Grid Box Hierarchy [6] are proposed, which try to integrate each node into the monitoring process. BlockTree and the Grid Box Hierarchy logically partition the network into blocks, which are then arranged in a hierarchy. Dependent on the block and level, the nodes coordinate their actions and determine where to send the data, e.g., exchange information within a block or with other blocks. Both approaches rely on hierarchies to organize the nodes and to handle the data, whereas Mobi-G operates on a flat network, where the data can be exchanged between two nodes without considering an underlying hierarchy. The evaluation outlines that these loosely coupled relations between nodes are particularly beneficial in sparsely populated areas, while hierarchical approaches suffer from static relations between nodes or areas, as shown for BlockTree.

V. Conclusion

Mobi-G is a flat monitoring mechanism for MANETs, which does not rely on mass conservation and gossips to aggregate but instead uses gossiping for a robust exchange of monitoring information. It adapts the concepts of Gossipico [22] to collect and disseminate the data as well as to organize the nodes.

The system parameter evaluation outlines that Mobi-G provides accurate results and even captures fluctuating attributes when using continuous monitoring. In contrast to this high accuracy, discrete monitoring generates a less accurate view but halves the traffic. With its system parameters, i.e, epoch length and Refresh-Timeout, Mobi-G increases or preserves the accuracy of discrete or continuous monitoring respectively, while reducing the communication cost.

The comparative evaluation reveals that Mobi-G does not suffer from the missing long range connectivity and monitors large spatial networks. Nevertheless, the accuracy of Mobi-G decreases for an increasing spatial network size, whereas BlockTree benefits form its hierarchical structure. However, Mobi-G's continuous monitoring outperforms BlockTree even for larger spatial networks, when monitoring fluctuating attributes. Moreover, BlockTree produces considerably more traffic, which drastically increases for a growing spatial network size, due to BlockTree's hierarchical structure. In terms



(a) Mean rel. node count error (b) Mean rel. PDSF error

(c) Upload traffic per node (d) Download traffic per node

Figure 8. Influence of node density on Mobi-G and BlockTree

of node density, the results outline that Mobi-G provides all nodes with accurate results in densely populated areas. The advantages of the flat topology and gossip-based communication become apparent in sparsely populated scenarios. While BlockTree does not provide accurate results, since the hierarchy cannot handle sparsely populated or empty regions, the loose and periodically refreshed relations between the nodes suffice to collect and disseminate the data.

For future work, it is planned to extend the conducted evaluation to address robustness with respect to (i) a varying node mobility, (ii) different mean session lengths, and (iii) an increased message loss probability. Besides robustness, it is intended to examine the impact of a flat topology on the provisioning of location-aware monitoring results, as specified in [18]. In parallel, it is planned to investigate transitions between flat and hierarchical monitoring mechanisms to exploit their complementary advantages in dynamic environments with varying conditions.

Acknowledgment

This work has been funded by the German Research Foundation (DFG) in the Collaborative Research Centre (SFB) 1053 "MAKI – Multi-Mechanisms-Adaptation for the Future Internet".

References

[1] N. Battat and H. Kheddouci, "HMAN: Hierarchical Monitoring for Ad Hoc Network," in *IEEE/IFIP EUC*, 2011.

[2] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié, "Epidemic Information Dissemination in Distributed Systems," *IEEE Computer*, vol. 37, no. 5, pp. 60 – 67, 2004.

[3] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, "Contention-Based Forwarding for Mobile Ad Hoc Networks," *Ad Hoc Networks*, vol. 1, no. 4, pp. 351–369, 2003.

[4] J. Geibig and D. Bradler, "Self-Organized Aggregation in Irregular Wireless Networks," in *IFIP Wireless Days*, 2010.

[5] A. Guerrieri, I. Carreras, F. D. Pellegrini, D. Miorandi, and A. Montresor, "Distributed Estimation of Global Parameters in Delay-Tolerant Networks," *Computer Communications*, vol. 33, pp. 1472–1482, 2010.

[6] I. Gupta, R. van Renesse, and K. P. Birman, "Scalable Fault-Tolerant Aggregation in Large Process Groups," in *IEEE DSN*, 2001.

[7] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-Based Aggregation in Large Dynamic Networks," *ACM Transactions on Computer Systems*, vol. 23, no. 3, pp. 219–252, 2005.

[8] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," in *IEEE FOCS*, 2003.

[9] S. Keshav, "Efficient and Decentralized Computation of Approximate Global State," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 69–74, 2006.

[10] K. Kwak, G. Huerta-Canepa, Y. Ko, D. Lee, and S. J. Hyun, "An Overlay-Based Resource Monitoring Scheme for Social Applications in MANET," in *IEEE COMPSAC*, 2009.

[11] B. Liang and Z. J. Haas, "Predictive Distance-Based Mobility Management for PCS Networks," in *IEEE INFOCOM*, 1999, pp. 1377–1384.

[12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 131–146, 2002.

[13] S. Nanda and D. Kotz, "Mesh-Mon: A Multi-Radio Mesh Monitoring and Management System," *Special Issue: Modeling, Testbeds, and Applications in Wireless Mesh Networks in Computer Communications*, vol. 31, no. 8, pp. 1588–1601, 2008.

[14] K. Ramachandran, E. Belding-Royer, and K. Almeroth, "DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks," in *IEEE SECON*, 2004.

[15] R. Riggio, M. Gerola, D. Miorandi, A. Zanardi, and F. Jan, "A Distributed Network Monitoring Framework for Wireless Networks," in *IFIP/IEEE IM*, 2011.

[16] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "An Adaptive Management Architecture for Ad Hoc Networks," *IEEE Communications Magazine*, vol. 41, no. 2, pp. 108–115, 2003.

[17] D. Stingl, C. Groß, , and K. Saller, "Decentralized Monitoring in Peer-to-Peer Systems," in *Benchmarking Peer-to-Peer Systems*, ser. LNCS, T. S. Wolfgang Effelsberg, Ralf Steinmetz, Ed. Springer, 2013, vol. 7847, pp. 81–113.

[18] D. Stingl, C. Gross, L. Nobach, R. Steinmetz, and D. Hausheer, "BlockTree: Location-Aware Decentralized Monitoring in Mobile Ad Hoc Networks," in *IEEE LCN*, 2013.

[19] D. Stingl, C. Gross, K. Saller, S. Kaune, and R. Steinmetz, "Benchmarking Decentralized Monitoring Mechanisms in Peer-to-Peer Systems," in *ACM ICPE*, 2012.

[20] D. Stingl, B. Richerzhagen, F. Zöllner, C. Gross, and R. Steinmetz, "PeerfactSim.KOM: Take it Back to the Streets," in *IEEE HPCS*, 2013.

[21] W. W. Terpstra, C. Leng, and A. P. Buchmann, "Brief Announcement: Practical Summation via Gossip," in *ACM PODC*, 2007.

[22] R. van de Bovenkamp, F. Kuipers, and P. Van Mieghem, "Gossip-Based Counting in Dynamic Networks," in *IFIP NETWORKING*, 2012.

[23] F. Wuhib, M. Dam, R. Stadler, and A. Clem, "Robust monitoring of network-wide aggregates through gossiping," *IEEE Transactions on Network and Service Management*, vol. 6, no. 2, pp. 95–109, 2009.

[24] F. Wuhib and R. Stadler, "Adaptive real-time monitoring in mobile wireless networks," Royal Institute of Technology, Tech. Rep., 2008.

[25] P. Yalagandula and M. Dahlin, "A Scalable Distributed Information Management System," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 379–390, 2004.

[26] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 349–365, 2003.