BlockTree: Location-Aware Decentralized Monitoring in Mobile Ad Hoc Networks

Dominik Stingl, Christian Gross, Leonhard Nobach, Ralf Steinmetz
Multimedia Communications Lab
Technische Universität Darmstadt
Darmstadt, Germany
{stingl,chrgross,nobach,steinmetz}@kom.tu-darmstadt.de

David Hausheer
Peer-to-Peer Systems Engineering Lab
Technische Universität Darmstadt
Darmstadt, Germany
hausheer@ps.tu-darmstadt.de

***Abstract*—Mobile ad hoc networks (MANETs) represent a crucial alternative to deploy applications in urban areas. In those networks, it is inevitable that all nodes are aware of the current system state to adapt their behavior according to the varying conditions. However, existing decentralized monitoring solutions for MANETs only locate the required information at a set of nodes, which are in charge of serving the remaining network, while the availability of information depends on the accessibility of those nodes. To avoid these limitations, BlockTree is a novel, fully decentralized monitoring approach for MANETs that leverages each node's resources to capture and distribute the system state to all nodes. Exploiting its hierarchical structure, BlockTree introduces the concept of location-aware monitoring delivering detailed as well as aggregated information. Through robust communication paired with the stateless design, BlockTree provides accurate results in the presence of fast moving nodes or over an error-prone communication medium.**

I. Introduction

Mobile ad hoc networks (MANETs) provide the opportunity to exchange information between handheld communication devices without the need for a communication infrastructure. Equipped with a communication module that offers an ad hoc mode (e.g., Wi-Fi), the devices directly interact with each other and contribute their resources to enable the deployment of applications in place. Especially in urban areas, MANETs represent a viable alternative to infrastructure-based networks and are particularly useful for applications, where the locality of interaction can be exploited. Examples comprise mobile social networks [14], where the interaction takes place between a group of users, as well as location-based services (e.g., Foursquare or academic approaches [3], [20]), where users consume and generate information about nearby locations [4].

To cope with the varying conditions resulting from the dynamic nature of the environment, it is essential that the participating nodes in a MANET obtain the current system state to adjust the network and the application accordingly. For example, a monitoring mechanism (i) monitors the access frequency of the exchanged content to adjust the replication factor [5], (ii) monitors social metrics [21] or available node resources [14] to identify highly connected or powerful nodes for content placement, or (iii) monitors the node density to identify sparsely populated places [9], as examined in Section V-C. In contrast to a centralized approach, where a server processes the monitored data and reacts on the obtained results, MANETs must cope with the challenge that the data is distributed over the entire network, while no dedicated component is in place to capture the monitored data. Once collected, the dissemination of the generated results among the interested nodes, which require the information to adjust their behavior, constitutes another challenge.

Decentralized approaches addressing those challenges already exist, e.g., for *static* peer-to-peer (P2P) networks (cf. [8], [11], [27], [28]). However, those approaches are not applicable to MANETs, where (i) mobility leads to a highly dynamic network, (ii) the communication range is limited, and (iii) the communication medium is error-prone. To tackle those additional challenges, dedicated monitoring approaches have been developed for MANETs (cf. [2], [14], [22], [23]). They all rely on a small subset of nodes, which are in charge of collecting and storing the monitored information. The data collection, limited to this subset, leads to the situation that the information is only available at a few nodes and must be disseminated to the remaining network. This rather centralized design reveals several drawbacks: (i) collecting nodes including their generated results might become unreachable, (ii) the constantly changing network increases the probability that request or result messages get lost, and (iii) an election of responsible nodes must be organized in a highly dynamic environment.

To overcome these shortcomings of existing approaches, this paper proposes BlockTree, a fully decentralized location-aware monitoring mechanism for MANETs. Through its hierarchical design, BlockTree scales both with the number of nodes and the spatial network size, enabling to monitor information in large areas with a varying node density. To address the locality of interactions in MANETs, BlockTree introduces the concept of location-aware monitoring, providing detailed information about a node's vicinity, while summarizing views on distant places. For communication, receiver-based contention schemes [7], [29] are applied to provide reliable data transmission but avoid unnecessary flooding. Paired with its stateless design, BlockTree exhibits high robustness and accurately monitors the system state even in dynamic networks.

In the following, Section II addresses the requirements of decentralized monitoring in MANETs. Afterwards, the related work is presented in Section III. Given the required back-

ground, BlockTree is presented in Section IV and evaluated in Section V. Finally, Section VI concludes the paper.

II. Decentralized Monitoring in MANETs

Given a MANET, which is populated with moving nodes, a decentralized monitoring mechanism collects locally measured data and subsequently provides meaningful system statistics of the collected data as a final result. In contrast to a centralized approach, decentralized approaches leverage the nodes' resources for collecting, processing, and disseminating the data, as described in [25]. To monitor the data in the network, attributes are specified, each node has to measure. Afterwards, the nodes organize themselves to collect the measurements of all nodes. Out of the collected data, statistics are generated and distributed among the nodes in the MANET. To counteract the otherwise huge load of monitored information, the data is aggregated using data-dependent aggregation functions, such as minimum, maximum, average, or standard deviation. In terms of hierarchical or tree-based approaches, aggregation functions can rely on the hierarchical computation property [18], [28], where an aggregate at a certain hierarchy or tree level consists of data from lower levels.

Out of the relevant non-functional requirements for decentralized monitoring mechanisms in fixed networks [25], the herein considered non-functional requirements are derived:

- **Accuracy** The key task of a monitoring mechanism is the provisioning of *accurate* results, meaning that the monitored system state should not deviate from the effective state but represent it as accurate as possible.
- **Timeliness** A monitoring mechanism must provide the calculated results in a *timely* manner to reflect the current state and not provide stale information.
- **Scalability** The monitoring mechanism must be *scalable* and provide accurate and timely results while (i) the spatial network size increases or decreases and while (ii) the node density inside the network varies.
- **Robustness** The assumed node mobility paired with the limited communication range of Wi-Fi ad hoc, results in a constantly changing topology with frequently changing neighbors. A corresponding monitoring mechanism must be *robust* to cope with these characteristics, while providing its services even in the presence of fast moving, newly arriving or leaving nodes.

III. Related Work

Related approaches of decentralized monitoring mechanisms exist for static P2P networks and MANETs as discussed in the following.

A. Decentralized Monitoring in Static P2P Networks

In static P2P networks, the topology of a decentralized monitoring mechanism constitutes a good criterion to classify existing solutions [19]. Hence, existing approaches can be divided into tree- and mesh-based approaches.

Tree-based approaches [8], [28] rely on a hierarchical structure to collect the locally measured attributes, while the root proactively disseminates the calculated results or reacts on requests. To create and organize the tree, each peer can autonomously identify its parent or determine if it is the root. Especially in MANETs, where the mentioned relation between two nodes is only valid if both nodes are within range, the tree must be continuously reorganized, resulting in a state, where neither data can be collected nor valid results can be provided.

Mesh-based monitoring mechanisms [11], [12], [27] work on arbitrary topologies and are applicable in dynamic networks, because a peer does not require a fixed set of neighbors to communicate with and the gossip-based communication scheme is highly robust for data exchange. On the contrary, these approaches require long-range connectivity between the peers [11], [12] to synchronize them and to disseminate the collected data in the network. In MANETs, this long-range connectivity is hardly obtainable, which leads to a decreasing performance especially in larger networks.

B. Decentralized Monitoring in MANETs

Among existing decentralized monitoring mechanisms for MANETs, approaches are discussed herein, which operate on clusters, as classified in [2]. The approaches integrate all nodes into the monitoring process but divide them into measuring and collecting entities, which are arranged in a hierarchy. To establish the hierarchy, Kwak et al. [14] create proximity-based clusters, where the elected heads are organized in an overlay. HMAN [2] uses a mathematical function to create the hierarchy, incorporating several aspects of a node (e.g. number of neighbors). Riggio et al. [23] present an approach, where measuring nodes become collecting sinks if the distance to other sinks exceeds a certain threshold. In contrast to these autonomous approaches, DAMON [22] relies on a predefined number of dedicated sinks, which must be provided by a network operator. Within the created hierarchy, the measuring nodes are affiliated to one collecting node, which collects the measured data and responds to result requests.

Although claiming to implement decentralized monitoring, only dedicated nodes act as sinks, which receive and manage the data, whereas the majority of nodes must query the sinks to obtain the results. The selected subset must handle all data and requests of the whole network, which results in an increased load especially in dense networks. If a collecting sink becomes unreachable or leaves the network, the collected knowledge is lost. Moreover, the affected nodes must detect this failure and agree on a new sink in a highly dynamic environment. BlockTree differs from those approaches in that it does not elect sinks. Instead, it treats each node equally and provides each node with the information about the system state to avoid that the information is reserved to a subset of nodes.

IV. System Design of BlockTree

BlockTree is a novel decentralized monitoring mechanism for MANETs, which relies on a hierarchical topology to (i) adequately arrange the nodes, (ii) collect and manage data, and (iii) provide location-aware results. BlockTree abstains from creating a hierarchy based on child-parent relations

between single nodes, but relies on relations between dedicated areas to obtain a robust topology. Any node in an area at a lower level of the hierarchy can exchange information with any node in the corresponding area at the next higher level, leading to multiple connections between the two areas. For the autonomous definition of the areas and their relations, BlockTree partitions the MANET into *blocks* that are arranged in a hierarchy. Each node just has to know its current location to deduce the ID and level of its current block. Sent messages contain further information so that each receiving node knows how to process the information. Exploiting the hierarchical structure, BlockTree relies on the hierarchical computation property to aggregate data [18], [28] and implements the concept of location-aware monitoring: results from lower levels incorporate fine-grained information from a small area of the network, whereas results from higher levels incorporate coarse-grained information but from a larger area.

The following sections describe the utilized routing procedures, the logical network partitioning, and present two approaches of BlockTree, which implement the just introduced concepts in different ways.

A. The Routing Procedures

Similar to geographic routing [16], the herein considered routing procedures *receiver-based position unicast* and *receiver-based area dissemination* rely on geographic coordinates and locations to define the destination of a message. To reliably route a message from a source to the targeted location, both routing procedures are inspired by receiver-based contention schemes [7], [29]. A sending node does not make any decision about the receiver, but broadcasts the message. Each receiving node estimates if it is the most suitable candidate among others to forward the message. In the following, the term *receiver-based* refers to this strategy. To avoid the creation of consent among the candidates, a *hesitation time* is introduced as proposed in [7], [29], which is computed as follows:

$$hesTime := hesFactor * maxForwardTime \quad (1)$$

In this equation, $maxForwardTime$ is a constant factor defined by the routing procedure, while $hesFactor$ represents the hesitation factor, which varies between 0 and 1 depending on an underlying formula. Through the hesitation time, better suited candidates obtain a smaller value for an earlier transmission. Hesitating nodes, which receive messages from other nodes during the next hop, drop their delayed messages.

In addition to the payload, a message contains further information so that a receiving node knows how to handle a message. Besides the sender's position $\vec{src}$ and the destination $\vec{dest}$, it always contains the position of the last hop. To avoid duplicate message transmissions, each message contains a unique identifier uid with a corresponding timer $timeout$ that specifies how long a receiving node drops incoming messages with the same uid. Therefore, each node maintains a *message forwarding cache*, which contains all uids of previously initiated and received messages. If the specified time has elapsed, the uid is removed from the cache and a node can again receive messages with that uid. For the remaining calculations, a two-dimensional Cartesian coordinate system is assumed, where the distance is calculated in meters.

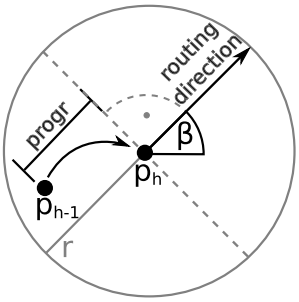


Figure 1. Computation of the distance $progr$ to calculate the hesitation factor

1) Receiver-Based Position Unicast: This routing procedure routes messages to a certain position specified by $\vec{dest}$. The message is intended for nodes, which are currently located next to the defined position. For the correct mode of operation, the routing procedure requires a definition of the assumed maximum communication range of the devices. For BlockTree, this range is specified as system parameter and denoted as r.

A node that wants to send a message to a certain position $\vec{dest}$ broadcasts the message in the network. During each hop, a receiving node p_h first checks if it already received a message with that uid. If the message is further processed, the node verifies if it belongs to the receiver set using (2), where $\vec{p_h}$ denotes p_h's current position.

$$rcv(\vec{p_h}, \vec{dest}) := \begin{cases} \text{true} & \text{if } ||\vec{dest} - \vec{p_h}|| < r \\ \text{false} & \text{else} \end{cases} \quad (2)$$

In any case, the node subsequently calculates the hesitation time using (1). The corresponding hesitation factor $hesFactor$ depends on the *routing direction* between the position $\vec{src}$ of the message initiator and the destination $\vec{dest}$ as well as on the position $\vec{p_{h-1}}$ of the last hop p_{h-1}. Therefore, p_h determines the normalized vector $\vec{\beta}$ indicating the routing direction from $\vec{src}$ to $\vec{dest}$. Orthogonal to $\vec{\beta}$, an imaginary line is placed through p_h's current position $\vec{p_h}$, represented by the dashed line in Figure 1. Afterwards, the distance $progr$ between p_{h-1} and the dashed line is determined to calculate the hesitation factor $hesFactor$:

$$hesFactor(progr) := \begin{cases} \infty & \text{if } progr < 0 \\ 0 & \text{if } progr \geq r \\ 1 - \frac{progr}{r} & \text{else} \end{cases} \quad (3)$$

In case that the previous hop p_{h-1} is on the same side as the destination, the message is not forwarded. Otherwise, $hesFactor$ decreases with an increasing distance between p_{h-1} and the dashed line. If the distance between p_{h-1} and the dashed line exceeds r, $hesFactor$ is set to 0. The routing procedure favors nodes that minimize the distance to the target along the routing direction.

2) Receiver-Based Area Dissemination: This routing procedure disseminates a message to all nodes that currently sojourn in an area defined by $dest$. To use this routing procedure, a node must be situated in the area, it wants to address. A receiving node first checks if it already received the message, given the obtained uid. If the uid is not cached and the node is in the addressed area, it automatically belongs to the receiver set and calculates the hesitation time to forward the message. Equation (4) calculates the corresponding $hesFactor$, which linearly decreases for an increasing distance between the receiver p_h and the last hop p_{h-1} until r is reached. The routing procedure prefers distant nodes to cover a large area, while reducing the propagation time.

$$hesFactor(\vec{p_{h-1}}, \vec{p_h}) := \begin{cases} 0 & \text{if } ||\vec{p_{h-1}} - \vec{p_h}|| \geq r \\ 1 - \frac{||\vec{p_{h-1}} - \vec{p_h}||}{r} & \text{else} \end{cases} \quad (4)$$

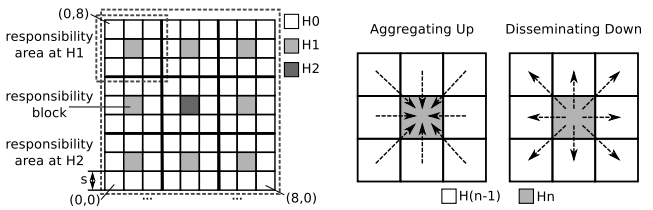
B. Logical Network Partitioning

BlockTree depends on a logical partitioning of the network into *blocks* to create a hierarchical topology with relations between different blocks. Unlike other approaches, which partition the network to organize the replication [15] or to determine location servers for geographic routing [16], BlockTree uses the blocks to collect and disseminate monitored data. The size of a block depends on the provided maximum communication range r, since BlockTree requires that all nodes of the same block are reachable within one hop. Therefore, the length of a block's edge s is calculated based on (5). In contrast, the communication between nodes of different blocks can take place over several hops.

$$s = \frac{r}{\sqrt{2}} \quad (5) \quad , \qquad blockID(\vec{p}) := \left(\lfloor \frac{p_x}{s} \rfloor, \lfloor \frac{p_y}{s} \rfloor \right) \quad (6)$$

Given its current position $\vec{p}$ in the network, a node p can determine the ID of the block it is currently in. Therefore, (6) assigns a unique two-dimensional identifier to each block, resulting in the logically partitioned network, consisting of unique blocks.

The partitioning of the environment into blocks is performed without any central control or coordination. Instead, it is assumed that the whole environment, which is represented in a two-dimensional Cartesian coordinate system and can have its origin at the intersection of the equator with the prime meridian, is completely divided into blocks. Based on the logically partitioned environment, each node autonomously calculates the ID of the block it is currently in. To calculate the block ID, a node must identify its current position in the underlying two-dimensional Cartesian coordinate system of BlockTree. Therefore, a node determines its geo-location specified by latitude and longitude coordinates using, e.g., the Global Positioning System (GPS). Afterwards, common projection techniques, such as Lambert's Conformal Conical Projection [10], can be used to transform the geo-location into the two-dimensional Cartesian coordinate system.



(a) Hierarchy of C-BlockTree (b) Data flow inside the hierarchy of C-BlockTree

Figure 2. Hierarchy and data flow of C-BlockTree

C. BlockTree: The Concentrating Approach

The *concentrating* approach of BlockTree, denoted as *C-BlockTree*, consists of two phases. During the first phase, the data is collected from each node and distributed among the nodes during the second phase. For the collection, a hierarchical structure is created with dedicated locations at each level of the hierarchy, represented by the grey blocks in Figure 2(a). The concentrating blocks are located in the middle of their corresponding areas to guarantee that they are equally accessible from the whole area. At each level (except at Level 0), the data from a level beneath is *concentrated* at these blocks, which in turn are responsible to forward the data to the next higher level. Based on these design decisions, the resulting hierarchy resembles a tree, where the data of lower levels is concentrated in the middle of outer regions until all data is collected at the highest hierarchy level in the middle of the whole MANET. Finally, the dissemination of the results is simply broadcasted from the highest level to the whole network.

1) Creating the Hierarchy: To create the hierarchy (cf. Figure 2(a)), a level is calculated for each block. Starting at the bottom of the hierarchy, a block at Level 0, abbreviated as H0, only consists of nodes. Every block at H1 is surrounded by eight blocks of H0. Together, the nine blocks form a *responsibility area* at H1, which is maintained by the light grey block, also denoted as *responsibility block*. To form a responsibility area at H2, a responsibility block is surrounded by eight blocks of H0 and in addition by eight responsibility areas of H1. Further levels above are defined accordingly. This iterative creation either ends if the predefined maximum level of the hierarchy is reached or if the populated area is covered by the created hierarchy. Given the assumptions for the logical network partitioning (cf. Section IV-B), each node can autonomously calculate the level for each block.

2) Protocol Behavior: The protocol of C-BlockTree works completely unsynchronized, thus, the periodically executed operations of the protocol can be started at any time. The interval between two consecutive executions of the operations is defined by the system parameter *Update Interval*. The protocol consists of the following three operations.

a) Leaf Broadcasting: Leaf Broadcasting is an operation, where each node in the network periodically broadcasts its locally measured values. This broadcast requires no routing to a certain point or area, but is used to inform all nodes inside the block about the recently monitored values. A node

only processes this information from nodes of the same block. It stores the received sender ID and the monitored values in the *leaf information table*. The table holds all values of nodes inside the same block and provides each node with sufficient information to initiate the *Aggregating Up* operation. If the information of a node is not regularly updated, the corresponding entry is purged.

b) Aggregating Up: Every node periodically triggers the Aggregating Up operation to provide a responsibility block with recently aggregated values. Using receiver-based position unicast, a message is sent to all nodes in the responsibility block at the next higher level, as displayed on the left-hand side in Figure 2(b). Even nodes that are currently located at the highest level must send their partial aggregation result to a destination at the next level. This can either result in an unsuccessful attempt to deliver the message, as the destination is not populated or a new node at the next higher level is discovered, thus, increasing the hierarchy.

The message consists of the sender's level n_{snd} and the partial aggregation result, which comprises current values of the *hierarchy table* (described below) and the leaf information table. The uid of the message consists of the sending node's block ID. Relying on the block ID, only one node per block sends the message per interval, while any further attempt from a node inside the block is suppressed for a certain time.

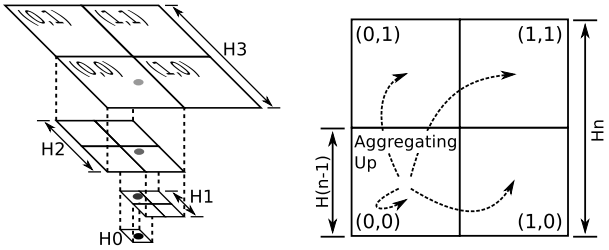
On every received Aggregating Up message, the receiver first checks if it is a parent of the sender based on n_{snd}. If so, the block ID and the partial aggregation result are stored in the hierarchy table. The hierarchy table of a node holds partial aggregation results with the corresponding block ID (i) of the eight surrounding child blocks if the node belongs to a level ≥ 1 and (ii) of every responsibility area beneath if the node belongs to a level > 1. Similar to the leaf information table, stale values are purged.

c) Disseminating Down: This operation is only executed by nodes that currently sojourn in the highest responsibility block. As displayed on the right-hand side in Figure 2(b), the final aggregation results are flooded to the whole network and comprise all values from the hierarchy and leaf information table. To disseminate the aggregated results, receiver-based area dissemination is used. To avoid that the network is flooded with the results several times, the uid of the message consists of the sending node's block ID and is used to block further flooding attempts for a certain time.

On a received message, the receiver stores the results with the corresponding block ID in its result table. C-BlockTree can also be configured that every node at a level greater than zero, additionally, executes the Disseminating Down operation. In this case, a receiving node stores the obtained results in the specified slot for the level in the result table. While this modification results in higher bandwidth consumption, the nodes obtain an overview on the different hierarchy levels.

D. BlockTree: The Planar Approach

The *planar* approach of BlockTree, denoted as *P-BlockTree*, modifies some key aspects of C-BlockTree. It simplifies the communication and creates a more flexible hierarchy to handle sparsely populated or empty blocks and networks of arbitrary spatial shapes. The main differences between P- and C-BlockTree are the following: (i) nodes are part of all hierarchy levels (cf. Figure 3(a)), (ii) only area dissemination is used to communicate, (iii) data is broadcasted to federations of blocks instead of single blocks, as the concentration of data in single blocks might lead to data loss in sparsely populated or empty regions, and (iv) P-BlockTree just consists of one phase to collect and already disseminate the data (cf. Figure 3(b)).



(a) Hierarchy of P-BlockTree (b) Data flow inside the hierarchy of P-BlockTree

Figure 3. Hierarchy and data flow of P-BlockTree

1) Creating a Hierarchy: In contrast to C-BlockTree, where a responsibility block of a higher level is surrounded by blocks and responsibility areas of lower levels, an area at level n consists of several areas of level $n-1$. The data is not concentrated any longer in the middle of a certain area, but exchanged among several areas of the same level. This property facilitates the creation of (i) a flexible hierarchy, which adapts to arbitrary shapes of the network and (ii) handles sparse or empty blocks due to the collection of data at federations of blocks

Starting from the bottom, the area at H0 corresponds to a block. Together with a set of neighboring blocks at H0 they form H1. These neighboring blocks are also denoted as *sectors*. Several sectors of H1 in turn form H2 and so on until either a predefined level is reached or the whole network is covered.

Since a node is a member of each level, it only has to know (i) the dissemination area at level n, given its position and (ii) the sector ID of the corresponding level n, where the node currently sojourns. Given n and $blockID(\vec{p})$, $sectorID$ is defined as

$$sectorID(blockID(\vec{p}), n) := \left(\lfloor \left(\frac{id_x}{d^{n-1}} \right) \rfloor \bmod d, \lfloor \left(\frac{id_y}{d^{n-1}} \right) \rfloor \bmod d \right), \quad (7)$$

where d is the number of sectors along a dimension. Throughout this paper, d is set to 2 resulting in four sectors per level.

2) Protocol Behavior: P-BlockTree works completely unsynchronized and relies as well on the system parameter Update Interval to trigger the periodic operations. It only consists of Leaf Broadcasting and Aggregating Up. Leaf Broadcasting is implemented in the same way as for C-BlockTree, whereas Aggregating Up differs, as explained in the following.

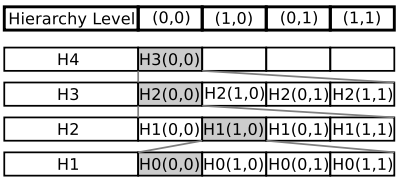


Figure 4. The hierarchy table of P-BlockTree

a) Aggregating Up: A node triggers the operation not only once, but for every *active* level of the hierarchy. A level n is denoted as *active* if it consists of at least two sectors of level $n-1$, so that at least one message from a neighboring sector at the same level can be received. Given the example with four levels (H0 to H3) in Figure 3(a), a node periodically starts this operation for level H1 to H3.

At any level, a sector disseminates its partial aggregation results to all nodes in its neighboring sectors, as displayed in Figure 3(b). In turn, all nodes in the disseminating sector receive the partial aggregation results from the neighboring sectors. For the data exchange, a node does not send separate messages but uses receiver-based area dissemination to distribute the partial aggregation result in the whole sector of the next higher level. For the identification of the message, the uid is a tuple, consisting of the sector ID and the level n_{snd} of the sending node. Using this tuple, the number of disseminations per sector is reduced, because the majority of sending attempts is blocked for a certain time.

If such a message is received, the partial aggregation result is stored in the hierarchy table. As displayed in Figure 4, the table consists of a row for each active level and of a column for every sector the receiving node is currently in. Sector ID and n_{snd} are used to identify the corresponding cell in the table. The results from the neighboring sectors at the same level are stored within the same row and yield to the partial aggregation result of a sector at the next higher level. Exploiting this hierarchical table structure, P-BlockTree implements the concept of location-aware monitoring, because the table contains fine-grained results at a lower level and coarse results from a higher level, which cover a larger area.

V. Evaluation

The evaluation is targeted at answering the question how BlockTree meets the identified non-functional requirements *scalability* and *robustness* (cf. Section II), while providing *accurate* and *timely* results at ideally low costs. Therefore, the evaluation starts with a comparative analysis (cf. Section V-B) comparing both approaches of BlockTree with respect to scalability. Afterwards, Section V-C details the evaluation of the most promising approach in terms of robustness and focuses on the provisioning of location-aware monitoring results.

A. Simulation Setup

The evaluation of this paper is based on simulations using a modified version of PeerfactSim.KOM [24] for mobile networks [26]. The parameters for the default scenario are listed in Table I. During each set of experiments, one of these parameters is varied, while the rest is fixed. The parameters model a quadratic area, which is populated with nodes that move with the specified maximum speed using the Gauss-Markov movement model [17]. The mean session length defines the mean for the exponentially distributed session length, which models the sojourn time of an active node in the area.

The simulated wireless communication module for Wi-Fi is oriented towards the IEEE 802.11b standard. It bases on measurements by Anastasi et al. [1], modeling (i) a *communication range* that varies between 110-130m for a *transmission rate* of 1Mbps and (ii) a *message loss probability* of 10%. For the identification of neighboring nodes and potential communication partners, the model relies on a unit disc graph.

To configure BlockTree, *Update Interval* and the assumed maximum communication range r must be defined. Update Interval, which triggers the periodic operations of BlockTree and defines different timers to purge message- and aggregation tables, is set to 15s. Relying on the measured communication range of Anastasi et al. [1], r is set to 120m, resulting in a block size of approximately 85m.

Each experiment is simulated for two hours. The first hour is used to reach a steady state so that the number of nodes levels out at the given value. After reaching the steady state, measurements are taken during the second hour.

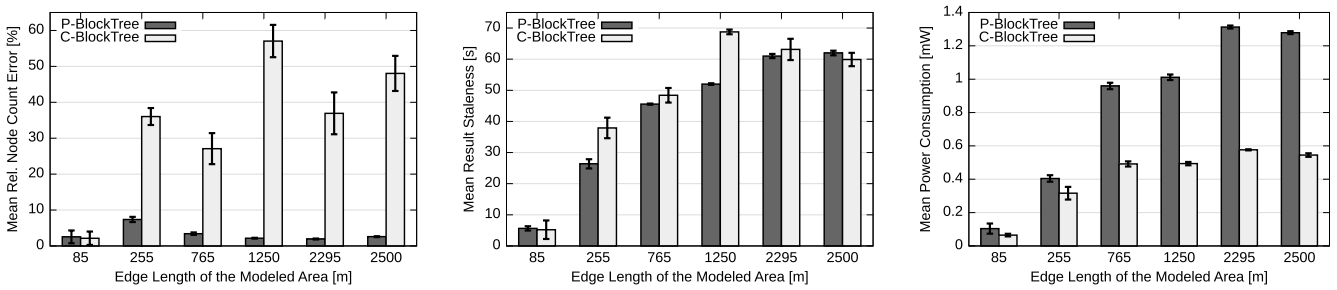
Table I
Default values for the scenario parameters

Parameter	Value	Parameter	Value
edge length	2500m	maximum movement speed	$2\frac{m}{s}$
# of nodes	2400	mean session length	16min

B. Comparative Evaluation

For the evaluation of scalability, the first set of experiments examines how a growing spatial network size influences the approaches given a fixed node density. The second set of experiments varies the number of nodes for a fixed spatial network size and evaluates the effect of node density. To quantify how the considered approaches meet the non-functional requirements, the delivered monitoring results are evaluated in terms of accuracy and staleness, while measuring the arising costs. To examine accuracy, the number of active nodes is monitored, since the determination of active nodes constitutes a representative problem for collecting network statistics, as denoted by Kostoulas et al. [13]. The relative error $|\frac{\hat{x}}{x}-1|$ quantifies the accuracy of the returned results, where $\hat{x}$ represents the monitored and x the effective node count. Dealing with staleness, $t_{now}-t_{avg}$ represents the age of a monitoring result, where t_{avg} is the average age of all included monitored values and t_{now} is the point in time of receiving the result. To quantify the costs, the average power consumption is calculated based on measurements by Feeney [6]. Each experiment is repeated five times. The corresponding plots display the mean and the 95% confidence interval.

1) Varying the Spatial Network Size: Table II lists the values for the edge length of the quadratic area. The number



(a) Mean result accuracy in terms of node count (b) Mean staleness of the provided results (c) Mean power consumption for communication

Figure 5. Comparative evaluation for different spatial network sizes

of nodes is modified as well to obtain a comparable density for all experiments. Four values (85, 255, 765, and 2295m) are chosen according to the hierarchy of C-BlockTree to evaluate how it performs in scenarios, where its hierarchy can be properly established (cf. Figure 2(a)). The experiments are completed with two arbitrary values (1250 and 2500m) to evaluate performance and cost in networks of an arbitrary spatial size. The lower part of Table II shows the resulting maximum hierarchy level for both approaches.

Table II
PARAMETER VARIATION OF THE SPATIAL NETWORK SIZE AND THE RESULTING MAXIMUM HIERARCHY LEVELS FOR BOTH APPROACHES

Parameter	**Value**					
edge length [m]	85	255	765	1250	2295	2500
# of nodes	3	27	243	600	2187	2400
P-BlockTree [H]	0	2	4	4	5	5
C-BlockTree [H]	0	1	2	3	3	3

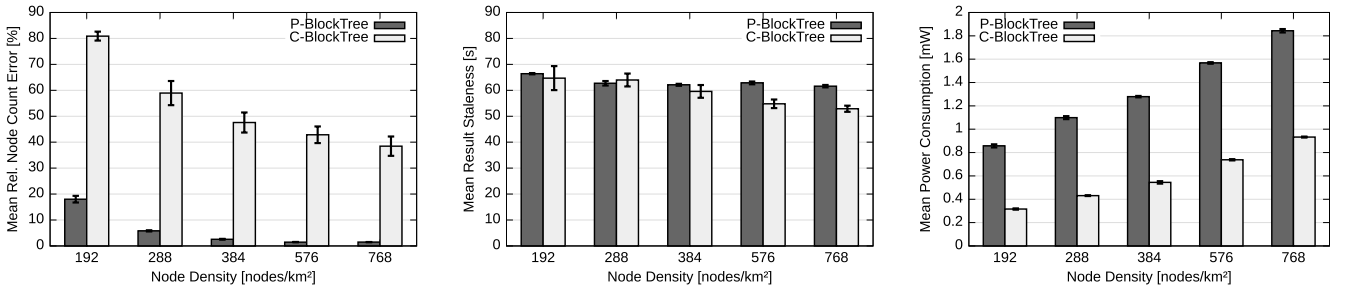
Figure 5(a) displays the mean relative node count error. Based on the results for 85, 765, and 2295m, it can be observed that the accuracy of C-BlockTree decreases for an increasing spatial network size. The deterioration results from the concentration of data at single blocks, which often fails if the blocks are sparsely populated or even empty. In larger areas, this problem accumulates, since the data is concentrated at multiple blocks over several steps. The negative impact of an arbitrary spatial network size becomes apparent by the low accuracy for 1250 and 2500m. The high error for 1250m results from the positioning of the highest responsibility block at the border of the map. This area is sparsely populated, thus, concentration and dissemination of data often fail in that responsibility block. For 2500m, the highest responsibility area of C-BlockTree does not cover the entire network but only a part of it. The remaining nodes try to concentrate the monitored data out of the modeled area, leading to the increased relative error. This problem can be denoted as *outer branches*. The high relative error for 255m results from an overestimation of active nodes. C-BlockTree does not purge the leaf information and hierarchy table in time. Thus, older values are propagated, even if a node left the block. The same effect can be observed for P-BlockTree, which attains its maximum relative error of 7.3% in the same scenario. Apart from this, the displayed results show that P-BlockTree does not suffer from an increasing spatial network size but provides accurate results even in large areas. P-BlockTree omits the collection of data at single blocks, but disseminates the data to federation of blocks so that sparsely populated blocks do not decrease the accuracy. Due to (i) the information exchange between any sectors at the same level and (ii) the omission of a concentrating block on top, the flexible topology of P-BlockTree even copes with arbitrary spatial network sizes.

Figure 5(b) shows that the staleness of results highly depends on the height of the hierarchy of C- and P-BlockTree. Given the maximum hierarchy levels in Table II, it can be observed that a larger spatial network size leads to an increased staleness, if the height of the hierarchy increases. In contrast, the staleness nearly remains constant for the same height. The high staleness of C-BlockTree for 1250m results from the positioning of the highest responsibility block in the sparsely populated area at the border of the map.

Given the utilized energy model [6], which assumes that the communication module consumes 808mW in the idle state, the results in Figure 5(c) display the additional mean power consumption, induced by each approach. Similar to staleness, the results show that an increasing spatial network size significantly raises the mean power consumption, if the height of the hierarchy increases as well. Moreover, it becomes apparent that the constantly high accuracy of P-BlockTree comes at the expense of increased power consumption, due to the redundant communication.

2) Varying the Number of Nodes: In the following, the effect of node density is considered. The number of nodes is set to 1200, 1800, 2400, 3600, and 4800 nodes, which results in a density of 192, 288, 348, 576, and 768 nodes/km^2 for an edge length of 2500m. In terms of accuracy, Figure 6(a) shows that a sparsely populated network leads to a larger relative error, whereas a higher node density is beneficial for the accuracy of results. Especially C-BlockTree suffers from sparsely populated networks. Due to empty or unreachable responsibility blocks, C-BlockTree considerably underestimates the number of nodes, which yields to the displayed high error. With an increasing density, the occurrence of such blocks decreases and leads to higher accuracy. The remaining error for C-BlockTree in the dense scenarios originates from outer branches due to the arbitrary spatial network size.

Figure 6(b) displays the impact of node density on the staleness of monitoring results. Based on the outcome, it can be observed that (i) both approaches are able to serve more nodes with fresh results and (ii) a higher density even accelerates the collection and dissemination of data. The



(a) Mean result accuracy in terms of node count (b) Mean staleness of the provided results (c) Mean power consumption for communication

Figure 6. Comparative evaluation for different node densities

receiver-based position unicast of C-BlockTree leverages the increasing density to select nodes that maximally shorten the distance to the target. This effect leads to a reduced number of hops and becomes apparent by the constantly decreasing staleness. For P-BlockTree, the positive impact is not that strong, because a density above 288 nodes/km^2 already suffices to select well located nodes, which maximally reduce the number of hops to reach all nodes. On the other hand, the results in Figure 6(c) show that a higher node density leads to an increased mean power consumption. Although both approaches rely on receiver-based contention schemes, which try to reduce the number of transmissions while being robust, the higher node density increases the probability that a node receives the same information several times.

C. Detailed Analysis of P-BlockTree

The detailed evaluation addresses robustness and examines the influence of a varying node mobility, mean session length, and message loss on the accuracy of P-BlockTree. To examine accuracy, P-BlockTree monitors the node density, as proposed in [9]. In addition, the node density is used to evaluate the provisioning of location-aware monitoring results. Therefore, a node uses the monitored number of nodes per hierarchy to calculate the node density for the covered blocks. To measure the accuracy, the relative error between the calculated and the effective density is averaged over all covered blocks. At H0, the mean relative error consists of the node's current block. At H1, it is averaged over the three neighboring blocks. At H2, the result is averaged over the 12 blocks from the three neighboring sectors and so on. For each set of experiments one parameter is varied, while the remaining ones are set to the default values (cf. Table I). Each experiment is simulated five times. Box plots are used to summarize the periodically captured results of all nodes. The whiskers are set to the 2.5th and 97.5th percentile, the box and the line inside display the first and third quartile as well as the median.

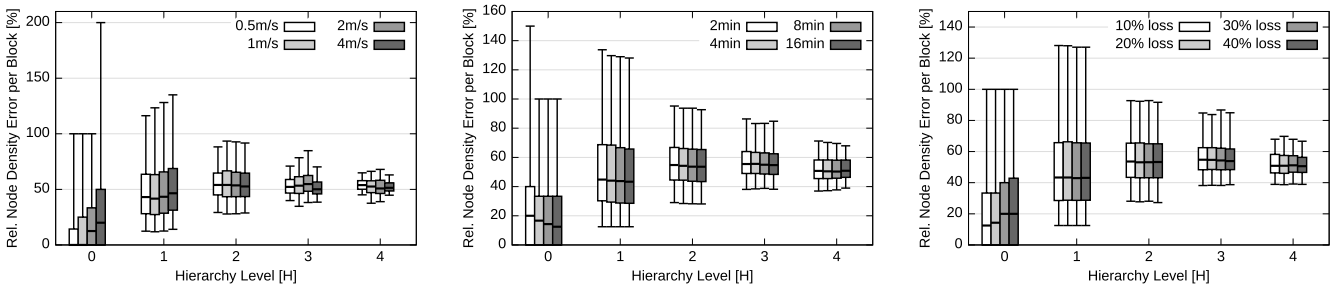
The three plots in Figure 7 outline that P-BlockTree delivers location-aware monitoring results. The approach provides an accurate view on a node's current block, whereas the aggregated monitoring information from distant places leads to a decreasing accuracy for the estimated node density in distant blocks. Moreover, the results highlight that the three parameter variations have little impact on the results per level and that P-BlockTree is robust, as detailed below. The low variability of accuracy at higher levels, which becomes apparent by shrinking boxes and whiskers, originates from averaging the relative error per node over 48 (at H3) and 192 (at H4) blocks.

Figure 7(a) reveals that P-BlockTree is able to handle an increasing mobility. For 0.5 and 1m/s more than 50% of the results from the own block are correct. Nevertheless, the results for 2 and 4m/s indicate that fast moving nodes decrease the accuracy, since they are not captured, while traversing a block. At higher levels, this trend flattens, as it takes longer to traverse a sector of a higher level. During this time, P-BlockTree is able to capture the node. The results in Figure 7(b) underline that P-BlockTree handles even short lived nodes. At H0, the decreasing mean session time down to a value of 4min has no influence on 75% of the results. At H1 the attenuated trend can still be observed, but mitigates at higher levels. Finally, Figure 7(c) shows that even an increasing unreliable communication medium does not influence the performance of P-BlockTree. The higher levels benefit from the high data and node redundancy, because if one transmission fails, a further transmission is started by a hesitating node. In contrast, this redundancy does not exist at H0, because all nodes must exchange their individually measured data. If one message is dropped, no other node can replace the missing information.

VI. Conclusion

BlockTree is a novel decentralized monitoring approach designed for MANETs. It implements the concept of location-aware monitoring to provide detailed information about neighboring blocks and summarized views on distant places.

Simulations have shown that especially P-BlockTree scales with the spatial network size and provides accurate results with a maximum mean relative error of 7.3% in terms of the node count. Considering the impact of node density, the approach works in sparsely populated scenarios as well, while accuracy and freshness of the results even benefit from an increasing density. The good performance of P-BlockTree comes at the expense of a higher average power consumption for an increasing spatial network size or node density. Due to the redundant information exchange, the average power consumption is twice as high as for C-BlockTree, which operates on a more organized hierarchy with clearly defined tasks per block. On the other hand, the results reveal that C-BlockTree highly depends on the node density as well as on the spatial network size. The results indicate that the concentration of data at single blocks only works in dense



(a) Impact of the max. movement speed on the mean relative error at different levels

(b) Impact of the mean session length on the mean relative error at different levels

(c) Impact of the message loss probability on the mean relative error at different levels

Figure 7. Detailed analysis of location-aware monitoring with P-BlockTree

scenarios, while the highest level of the hierarchy must be populated with nodes to avoid outer branches.

The detailed analysis of P-BlockTree outlines the characteristics of location-aware monitoring. It shows that the provided information enables a detailed and accurate view on the vicinity, whereas the status of distant places can be approximated, given the provided summarizing views. Through its robust design, P-BlockTree handles the presence of fast moving as well as short-lived nodes, which hardly influence the accuracy of the results. Moreover, the obtained results reveal that P-BlockTree even copes with an error-prone communication medium, which drops 40% of sent messages.

For future work it is planned to improve BlockTree in such a way that, e.g., the Update Interval varies between the hierarchy levels to improve the accuracy at lower levels, while saving resources at higher levels. In parallel, it is planned to implement BlockTree on Android smartphones to show the feasibility of the approach in practice.

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] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, “Wi-Fi in Ad Hoc Mode: A Measurement Study,” in *IEEE PerCom*, 2004.

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

[3] A. A. V. Castro, G. D. M. Serugendo, and D. Konstantas, “Hovering Information – Self-Organizing Information that Finds its Own Storage,” *Autonomic Communication*, pp. 111–145, 2009.

[4] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui, “Exploring Millions of Footprints in Location Sharing Services,” in *ICWSM*, 2011.

[5] E. Cohen and S. Shenker, “Replication Strategies in Unstructured Peer-to-Peer Networks,” in *ACM SIGCOMM*, 2002.

[6] L. M. Feeney, “An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks,” *Mob. Netw. Appl.*, vol. 6, no. 3, pp. 239–249, 2001.

[7] 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.

[8] K. Graffi, D. Stingl, J. Rueckert, A. Kovacevic, and R. Steinmetz, “Monitoring and Management of Structured Peer-to-Peer Systems,” in *IEEE P2P*, 2009.

[9] J. Hamada, A. Uchiyama, H. Yamaguchi, S. Kusumoto, and T. Higashino, “Self-Estimation of Neighborhood Density for Mobile Wireless Nodes,” in *Ubiquitous Intelligence and Computing*. Springer, 2009, vol. 5585, pp. 178–192.

[10] M. Hooijberg, “Lambert’s Conformal Conical Projection,” in *Practical Geodesy*. Springer Berlin Heidelberg, 1997, pp. 133–156.

[11] 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.

[12] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based Computation of Aggregate Information,” in *IEEE FOCS*, 2003.

[13] D. Kostoulas, D. Psaltoulis, I. Gupta, K. P. Birman, and A. Demers, “Active and Passive Techniques for Group Size Estimation in Large-Scale and Dynamic Distributed Systems,” *Journal of Systems and Software*, vol. 80, no. 10, pp. 1639–1658, 2007.

[14] 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.

[15] S.-B. Lee, S. H. Y. Wong, K.-W. Lee, and S. Lu, “Content Management in a Mobile Ad Hoc Network: Beyond Opportunistic Strategy,” in *IEEE INFOCOM*, 2011.

[16] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, “A Scalable Location Service for Geographic Ad Hoc Routing,” in *ACM MobiCom*, 2000.

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

[18] 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.

[19] R. Makhloufi, G. Bonnet, G. Doyen, and D. Gaïti, “Decentralized Aggregation Protocols in Peer-to-Peer Networks: A Survey,” in *IEEE MACE*, 2009.

[20] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju, “Floating Content: Information Sharing in Urban Areas,” in *IEEE PerCom*, 2011.

[21] P. Pantazopoulos, I. Stavrakakis, A. Passarella, and M. Conti, “Efficient Social-aware Content Placement in Opportunistic Networks,” in *IEEE WONS*, 2010.

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

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

[24] D. Stingl, C. Gross, J. Rückert, L. Nobach, A. Kovacevic, and R. Steinmetz, “PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems,” in *IEEE HPCS*, 2011.

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

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

[27] R. van de Bovenkamp, F. Kuipers, and P. Van Mieghem, “Gossip-based Counting in Dynamic Networks,” in *IFIP NETWORKING*, 2012.

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

[29] 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.