Bloom Hopping: Bloom filter based 2-Hop Neighbor Management in VANETs

Florian Klingler *Student Member, IEEE*, Reuven Cohen *Senior Member, IEEE*,
Christoph Sommer *Member, IEEE* and Falko Dressler *Fellow, IEEE*

Abstract—Recent works have shown that it would be beneficial for nodes in wireless networks with very dynamic topology to maintain a list of 2-hop neighbors, namely, the neighbors of its neighbors. This is important, for example, for routing, clustering, and message dissemination to all the nodes in a given geographic vicinity. In this paper, we propose a scheme that uses Bloom filters for maintaining 2-hop neighborship information. Furthermore, we developed a novel 2-hop broadcast algorithm making use of the specific nature of our Bloom filter encoded neighbor information. We particularly focus on the Vehicular Ad Hoc Networks (VANETs) application scenario. Here, beaconing is a periodic broadcast of awareness messages by each vehicle to its immediate neighbors. A naïve approach would be to include all 2-hop neighbors in each beacon message, which, however, would work only for small or sparse scenarios. We show that our approach significantly reduces the length of the beacon messages, thereby keeping channel load and collision probability considerably lower than in the naïve scheme. We further demonstrate the application of our Bloom filter based 2-hop neighbor table for developing higher layer protocols and introduce a multi-hop broadcast protocol called Bloom Hopping.

1 Introduction

In Vehicular Ad Hoc Networks (VANETs), beaconing is a periodic dissemination of a control message by each vehicle (node) to its immediate (1-hop) neighbors. Such messages are needed for cooperative awareness and road traffic safety applications and have been standardized as Cooperative Awareness Messages (CAMs) [1]. Information included in a single beacon message usually contains a node's ID together with its current status, e.g., position, speed, and heading.

Recent works have shown that it would be beneficial for a node to maintain not only the list of its 1-hop neighbors, but also of its 2-hop neighbors; namely, the neighbors of its neighbors [2]. This is important, for example, in applications where a message has to be forwarded to all the nodes in a given geographic vicinity. Instead of using inefficient flooding, the originating node can broadcast the message and choose a subset of its 1-hop neighbors for forwarding. This subset is chosen such that it contains a minimum number of nodes to cover the 2-hop neighbors of the originator.

- *F. Klingler, C. Sommer and F. Dressler are with the Heinz Nixdorf Institute and the Dept. of Computer Science, Paderborn University, Germany, E-mail: {klingler, sommer, dressler}@ccs-labs.org.*
- *R. Cohen is with the Department of Computer Science, Technion—Israel Institute of Technology, Israel, E-mail: rcohen@cs.technion.ac.il.*

Each node can maintain a list of its 2-hop neighbors by including in its beacon messages the full list of all its 1-hop neighbors. However, this naïve approach would probably work only for small or sparse scenarios with very few vehicles being involved. It lacks scalability because the beacon size is directly proportional to the number of neighbors. As an example, suppose that the regular 6 Byte MAC addresses are used to uniquely distinguish nodes. On a 6-lane freeway, with a communication distance of 500 m, the expected number of 1-hop neighbors is 150. Thus, every beacon has to contain an expected amount of $150 \times 6\,\mathrm{Byte} = 900\,\mathrm{Byte}$ of neighbor information. The immediate result is not only a substantial beacon overhead, but – more importantly – reduced reliability of beacon messages due to hidden terminal effects and an increased beacon collision probability on the wireless channel.

In this paper, we propose a scheme that uses Bloom filters [3] for disseminating and maintaining 2-hop neighborship information. We show that, if used right, employing Bloom filters significantly reduces the length of the beacon messages, thereby keeping channel load and packet collision probability considerably lower than in a naïve scheme that incorporates full neighbor information into the beacons. However, the efficiency of Bloom filters comes with the cost of false positives.

We also address the following important decisions:

- What information should be included in each beacon.
- When to add or remove neighbors to/from the local 2-hop neighbor table.
- When to send neighbor information, and at which frequency, i.e., the beacon rate.
- How to select a subset of 1-hop neighbors for forwarding, to cover (with high probability) the whole set of 2-hop neighbors.

In addition to allowing efficient broadcast by maintaining the list of 2-hop neighbors, our scheme provides the basis for a variety of other applications that build on top of neighborship information: routing and clustering, for example. Some of these applications require each node to have not only 2-hop neighbor information, but N-hop information for some $N > 2$. Moreover, the usage of Bloom filters for neighbor tables provides a privacy preserving way due to their inherent nature to not store addresses but hashes [4].

Without loss of generality, our approach can also be adapted for $N > 2$ by using multiple Bloom filters indicating

different hop ranges, as well as for other types of networks, particularly if the network topology changes rapidly.

To summarize, our contributions are as follows:

- We analytically explore the properties of Bloom filter operations for probabilistic neighbor management (Section 3).
- We introduce a Bloom filter based 2-hop neighbor management scheme for dynamic wireless networks (Section 4).
- We demonstrate the applicability of our system and introduce a multi-hop broadcast protocol called Bloom Hopping (Section 5).
- We study the performance of our Bloom filter based neighbor management and the Bloom Hopping broadcast system (Section 6).

2 Related Work

Neighborship information is the underlying basis for routing, clustering, and message dissemination in mobile ad hoc networks. Yet, for highly dynamic topologies, this still constitutes a fundamental research problem, particularly if 2-hop neighbor information are needed. This particularly holds for highly dynamic wireless networks like VANETs, where protocols need to maintain as accurate as possible 1-hop [5]–[8] or 2-hop [9]–[12] neighbor information. Recently Dressler et al. [13] proposed a novel context aware and class based broadcast architecture for vehicular networks, which takes advantage of 1-hop and 2-hop neighbor information to support many different application domains for Intelligent Transportation Systems (ITS).

A typical approach is to rely on hello or beacon messages, which are also used for cooperative awareness, i.e., letting vehicles become aware of each other. Technically, we talk about small 1-hop broadcasts (e.g., CAMs) emitted periodically by each vehicle at 1–10 Hz. It was found that adaptive beaconing solutions are necessary in order to deal with congestion control [6], [14]. For example, Adaptive Traffic Beacon (ATB) [6] adapts the beacon interval according to the available channel capacity; thus, sending beacons as often as possible, but avoiding to overload the channel. This concept has been adopted by ongoing standardization activities: the ETSI ITS-G5 Decentralized Congestion Control (DCC) protocol [15] adapts the beacon interval according to a state machine resulting in different beaconing intervals (relaxed, active, and restricted state). It periodically measures the channel busy ratio b_t using a sampling process and performs transitions in the state machine accordingly. This allows the protocol to react to overloaded channels.

More recent works focus on application demands and safety metrics for selecting a fitting beacon interval [16], [17] in order to efficiently use the radio resources.

Pulsar [18] uses 2-hop piggybacking of congestion control information in order to maximize the overall beacon rate of nodes – again for vehicular safety applications. 2-hop information is also required by LIMERIC [19], [20] to adapt the beacon rate in order to provide fairness to all nodes.

Recently, a first work using Bloom filters for calculating connected dominating sets in vehicular networks has been presented [21]. This approach extends the work in [22] by using Bloom filter set-operations to lower the algorithm's complexity for finding a connected dominating set in an ad hoc network. In essence the authors reduce the algorithmic complexity of [22] from $\mathcal{O}(n^5)$ to $\mathcal{O}(n)$ by using two different Bloom filter operations: *union* (w/o loss of information), and *intersection* (w/ loss of information). Further the authors propose a fixed size beacon structure to incorporate a list of neighbors as well as a list of packet identifiers within a single Bloom filter. The evaluation of the algorithm shows that it is possible to reduce the beacon overhead compared to the original solution, even though the amount of received nodes and the retransmission overhead remains the same. Although the authors of [21] focus more on lowering the algorithmic complexity of the protocol, their approach is using the intersection of Bloom filters which causes loss of information, thus, increases the false positive error rate.

The use of Bloom filters has also been investigated for multi-hop message dissemination in general networks [23], [24] as well as in VANETs [2], [25]. A message dissemination protocol using Bloom filters without explicit 2-hop neighbor information has been proposed in [26]. In this approach, for each message to be forwarded a Bloom filter has to be included. A node rebroadcasts a message, if it can contribute to reach additional neighbors, and appends his local neighbor set in the Bloom filter which is included in that particular rebroadcast. This scheme is based on contention based forwarding, and thus limited in the usage within a general network protocol stack since other applications running on the same node may also rely on exact neighbor information. That is in contrast to our approach, since we focus on (a) efficient *neighbor management* algorithms to decide when to add which neighbors into our neighbor table, such that we can use that information to (b) efficiently choose fitting nodes from that neighbor table to rebroadcast our message. Our stack builds a basis for future applications to retrieve neighbor information.

TO-GO [2] uses Bloom filter encoded 2-hop information to allow probabilistic membership tests for selecting a set of forwarding nodes for multi-hop geocast. Considering routing and data discovery, different protocols have been proposed using Bloom filters in wireless sensor networks [27]–[29], yet, their performance is limited to rather stationary topologies. Moreover, Bloom filter structures are also used in wireless networks for routing in hierarchical topologies [30], or by using aggregated forwarding information for geographical routing [31].

Besides infrastructure-less vehicular networks (which this paper focuses on), VANETs can also take advantage of infrastructure like Roadside Units (RSUs) to obtain a better network connectivity in sparsely connected scenarios. Silva et al. [32] summarize in their survey more than ten years of research in different disciplines of infrastructure-based VANET communication, ranging from access technologies to deployment strategies of infrastructure. Further, VANETs can also benefit from cellular communication like LTE-V2V, which also allows direct vehicle-to-vehicle communication as IEEE 802.11p does. Bazzi et al. [33] analytically compare the performance of IEEE 802.11p and LTE-V2V by taking into account different modulation and coding schemes as well as focusing on hidden terminals and the capture effect of IEEE 802.11p.

In this paper, we focus on vehicle-to-vehicle commu-

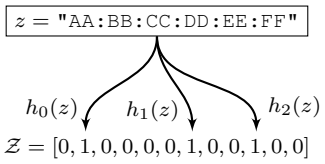


Fig. 1. Adding an element z (here, a MAC address) to an initially empty Bloom filter $\mathcal{Z}$ (of size $m = 12$ bit and using $k = 3$ hash functions).

nication, going one step further and not only propose Bloom filter based neighbor management but also study the properties of Bloom filters and common mathematical operations on them for 2-hop neighbor management. We show that the efficient use of Bloom filters not only decreases the overhead of transmissions, i.e., the beacon size and thus channel utilization, but actively improves the application layer performance. With the help of our proposed Bloom Hopping algorithm, we show how to make use of the specific nature of Bloom filter encoded 2-hop neighbor information to achieve very efficient multi-hop data dissemination. Our system also builds a fundamental basis for supporting a wider range of communication protocols with relevant 2-hop neighbor information.

3 Preliminaries

3.1 Bloom Filter

A Bloom filter, first introduced in 1970 by Burton Howard Bloom [3], is a space-efficient probabilistic data structure. It is used to (probabilistically) test whether an element is member of a set – with the possibility of false positives.

In detail, a Bloom filter $\mathcal{Z}$ consists of a bit array of m bits; all initialized to 0. k different hash functions map an element z to one of the m array positions in Bloom filter $\mathcal{Z}$ with a uniform distribution. For adding an element z (written as $\mathcal{Z} \leftarrow z$), the corresponding positions in $\mathcal{Z}$ are set to 1 as shown in Fig. 1. To simplify the notation, we write the insertion of all elements z in a set $\mathbb{Z}$ as $\mathcal{Z} \leftarrow \mathbb{Z}$.

To check whether an element is contained in the set, we feed it to all k hash functions to get again the corresponding k positions in the bit array $\mathcal{Z}$. If any of the bits at these positions is set to 0, the element is definitely not contained in the set. If all bits on these k positions are set to 1, the element is contained in the set, or, the bits were set when adding other elements resulting in a false positive detection error. False negatives are not possible in standard Bloom filters.

This type of filter does not allow the deletion of elements, because a bit within the bit array could have been set to 1 after the insertion of the element which is to be deleted. One approach could be to add a second Bloom filter, which contains deleted elements. Whenever an element is contained in the original Bloom filter and the one with deleted elements, we can assume that the element was deleted.

By design this procedure may cause the problem of false negatives since an element falsely queried in the Bloom filter for deleted elements causes a wrong decision whether an element is contained in the composite Bloom filter. An alternative approach are Counting Bloom Filters [34], which, however, can also suffer from false negatives [35].

In order to construct a Bloom filter, we have to know the Bloom filter size m and the number of hash functions k to use. We can derive a suitable size m (in bits) [24] from an expected insertion count n and acceptable false positive error rate p_{fp} as

$$m = -\frac{n \ln p_{\text{fp}}}{(\ln 2)^2} \,. \tag{1}$$

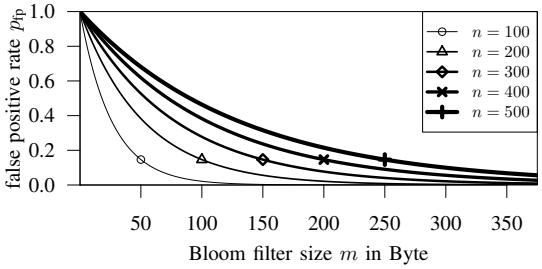


Fig. 2. False positive rate p_{fp} vs. Bloom filter size m for increasing numbers of inserted elements n.

In a vehicular scenario, this number n can be approximated by using traffic density estimation protocols [36], [37], or even by probabilistic data aggregation using Flajolet-Martin sketches [38], such that a fitting Bloom filter size can be determined. In [36], an overview of traffic density estimation approaches focusing on infrastructure-free vehicular networks has been presented. In this context, [37] was identified as a promising approach, which allows estimating the traffic density in a fully distributed manner even when only a low penetration rate of 30 % is available.

The optimal number of hash functions k [39] to minimize the false positive error rate for a given Bloom filter size and inserted element count can be derived as

$$k \approx \frac{m}{n} \ln 2 \,. \tag{2}$$

For a given Bloom filter size m, number of inserted elements n, and number of hash functions k, the false positive error rate p_{fp} for testing whether an element is member of this Bloom filter can be calculated [23] as

$$p_{\text{fp}} = \left[1 - \left(1 - \frac{1}{m}\right)^{kn}\right]^k \,. \tag{3}$$

In Fig. 2, we show the false positive rate as a function of the Bloom filter size and the number of inserted elements. For each Bloom filter size we calculated the optimal value of k according to Eq. (2). We can clearly see the steep increase of the false positive rate when the Bloom filter size gets small.

3.2 Operations on Bloom Filters

In order to compare Bloom filters, we need to introduce common operations performed on Bloom filters, which we use in the remaining part of this paper. Suppose we have two Bloom filters $\mathcal{A}$ and $\mathcal{B}$ using the same number of bits and the same hash functions. To generate a Bloom filter that represents $\mathcal{A} \cup \mathcal{B}$, we simply perform a $\vee$ (bitwise OR) operation on $\mathcal{A}$ and $\mathcal{B}$ to get the union of those two Bloom filters.

Similarly, the intersection of two Bloom filters $\mathcal{A} \cap \mathcal{B}$ is performed using a $\wedge$ (bitwise AND) operation on $\mathcal{A}$ and $\mathcal{B}$. Please note that this only approximates the intersecting Bloom filter [39], which is sufficient for large filters [40].

Finally, we show how to estimate the number of elements $|\mathbb{Z}|$ that have been inserted into a Bloom filter $\mathcal{Z}$. The cardinality of a Bloom filter can be approximated as

$$|\mathbb{Z}| \approx |\mathcal{Z}| = -\frac{m \ln(1 - \frac{\mathrm{c}(\mathcal{Z})}{m})}{k}\,, \tag{4}$$

where the function $\mathrm{c}(\cdot)$ counts the number of bits set to 1 within the Bloom filter [41]. In the special case of all bits set to 1 in a Bloom filter, the cardinality is not defined. For illustration purposes, we plot this as infinity in the following.

4 Neighbor Table Management

We start by describing the scheme that allows each vehicle to build and maintain a Bloom filter database of its 2-hop neighbors. To this end, each node x maintains a table with the identities of its 1-hop neighbors (how to determine which nodes are 1-hop neighbors is detailed in Section 4.1). Based on this table, node x generates a Bloom filter of contained identities and includes this Bloom filter in the beacons it periodically broadcasts using interval I. Moreover in each beacon we include the channel utilization $b_t = \frac{t_{\text{busy}}}{t_{\text{busy}}+t_{\text{idle}}}$ as the fraction of the time the wireless channel was sensed busy since the last sent beacon, i.e., $t-I$ and t. Further, each beacon includes the current interval I of the sending node to indicate at which time the receivers (most probably) can assume to get the next beacon, as well as the GPS position of the node. The neighbor table is then annotated with information about each 1-hop neighbor y, including its last broadcast Bloom filter, GPS position, distance, and time when the last beacon was received, as well as the last value of b_t.

The Bloom filters in the 1-hop neighbor table allow node x to make probabilistic decisions about its 2-hop neighbor set, by performing bitwise OR on all of the Bloom filter vectors. In this paper, we focus on what we believe is the most important application for acquiring 2-hop neighbor information, choosing a good set of 1-hop neighbors for forwarding, to cover all of the 2-hop neighbors. This "good" set should be as small as possible, but contain neighbors with which node x has a good wireless connectivity. The selection process is discussed in detail in Section 5.

4.1 Maintaining 1-Hop Neighbor Tables

In general, a node x may consider any beacon it receives as coming from a direct 1-hop neighbor for maintaining its list of neighbors $\mathbb{X}$ and the corresponding Bloom filter $\mathcal{X} \leftarrow \mathbb{X}$. This is the Bloom filter that will be sent periodically by node x; the received Bloom filter of node y at node x is called $\mathcal{X}_y$.

However, communication channels are sometimes asymmetric, that is, even though node x receives a beacon from node y, the same node y might not be able to receive messages from node x. This means that node x must verify that y receives its beacons before it can consider node y as an acknowledged 1-hop neighbor. To address this problem, when node x receives the beacon from y, it checks whether its own identity is contained in the Bloom filter sent by y. If yes, then there is a high probability (depending on the false positive probability of the Bloom filter) that bidirectional communication is possible; if not, then either node y has not yet received a beacon of x or the communication channel is asymmetric and the two nodes should not consider each other as a neighbor.

Mathematically, we can create a subset of acknowledged neighbors $\mathbb{X}'$ of neighbors $\mathbb{X}$ of node x as

$$\mathbb{X}' = \left\{ u \in \mathbb{X} : x \in \mathcal{X}_u \right\}. \tag{5}$$

This results in an accurate Bloom filter based neighbor table that only covers nodes that are able to communicate bidirectionally. For simplicity, in the following, we call only these neighbors 1-hop neighbors.

4.2 Maintaining 2-Hop Neighbor Tables

Our rationale to use 2-hop neighbor tables for VANETs is as follows: First, 1-hop information in a network does not give viable information about the topology of the underlying network. This can be improved by including, for example, the number of neighbors a node sees in the periodically exchanged beacons to infer about the network connectivity of the neighboring node. However, a receiver cannot derive to what degree the neighbor set of one of its neighbors differs from its own neighbor set. Having 2-hop neighboring information at hand, we can calculate the difference of the neighbor set and, thus, gain further information about whether this node could be a promising rebroadcast node.

4.2.1 Algorithm

We now extend the concept of our neighbor management approach to 2-hop neighbor tables. In our approach, each beacon contains a Bloom filter of a node's 1-hop neighbors. Thus, we can build a per node 2-hop neighbor table consisting of the respective Bloom filters. These Bloom filters consist of all 2-hop nodes that we know from our 1-hop neighbors. Because of interference, a node may not be aware of some 1-hop neighbors. Moreover, neighbor tables might be rather unstable, i.e., fluctuating entries because of lost beacon messages. This may lead to wrong neighbor tables: too many entries in the 2-hop neighbor table, and missing entries in the 1-hop neighbor table.

To address the problem of wrongly assigning nodes as 1-hop or 2-hop neighbors, we can take advantage of a node's position and other nodes' Bloom filters as well as their announced beacon time. Whenever we are going to remove a neighbor from the 1-hop neighbor table, we check whether its ID is included in the Bloom filter of other nodes, which are more distant to the node that we are about to remove and which have sent a beacon after the announced beacon time of the node we want to remove. Assuming node x is checking its neighbor table for stale entries and wants to remove node y from the neighbor table, we can decide whether to keep the neighbor y (but mark it as hidden) in three steps:

1) A node x considers which of its neighbors are already marked as hidden and which are not, storing them as

$$\bar{\bar{\mathbb{X}}} = \left\{ u \in \mathbb{X}' : u \text{ is hidden} \right\}, \tag{6}$$

$$\bar{\mathbb{X}} = \left\{ u \in \mathbb{X}' : u \text{ is not hidden} \right\} = \mathbb{X}' \setminus \bar{\bar{\mathbb{X}}}\,. \tag{7}$$

2) Node x calculates a set of closer neighbors $\mathbb{C}_{x,y} \subseteq \bar{\mathbb{X}}$, the set of all visible 1-hop neighbors of x which are equally or further away from node y than node x as

$$\mathbb{C}_{x,y} = \left\{ u \in \bar{\mathbb{X}} \ : \ \text{distance}(u,y) \geq \text{distance}(x,y) \right\}, \tag{8}$$

as well as a set of neighbors $\mathbb{D}_{x,y} \subseteq \mathbb{C}_{x,y}$ from which a beacon has been received after the announced beacon time of node y

$$\mathbb{D}_{x,y} = \left\{ u \in \mathbb{C}_{x,y} \ : \ \mathrm{t}_{\text{b.-received}}(u) \geq \mathrm{t}_{\text{b.-announced}}(y) \right\} \tag{9}$$

and a set of neighbors $\mathbb{E}_{x,y} \subseteq \mathbb{D}_{x,y}$ that have a lower or equal channel busy ratio b_t compared to node x

$$\mathbb{E}_{x,y} = \left\{ u \in \mathbb{D}_{x,y} \ : \ b_t(u) \leq b_t(x) \right\}. \tag{10}$$

This is essential to select only those neighbors $\mathbb{E}_{x,y}$ that have a reasonably high probability to include node y and do not suffer from a high amount of non-relevant 2-hop neighbors, which would increase the Bloom filter false positive rate.

3) Node x only deletes node y from its neighbor table if y is not in any of the resulting neighbors' Bloom filters, that is, if $\mathbb{F}_{x,y} = \{u \in \mathbb{E}_{x,y} : y \in \mathcal{X}_u\}$ and $\mathbb{F}_{x,y} = \emptyset$. Otherwise, it merely marks y as hidden. Please note that when broadcasting beacons, we omit such hidden neighbors from the Bloom filter.

We now can summarize all 2-hop neighbors in the Bloom filter $\mathcal{X}''$ as

$$\mathcal{X}'' = \bigcup_{u \in \mathbb{X}'} \mathcal{X}_u \,. \tag{11}$$

The main advantage of our solution is that it keeps the neighbor tables itself intact, avoiding the problems of wrongly annotating neighbors as 1-hop or 2-hop neighbors. Furthermore, we can compensate short lasting outages of beacons and do not need to remove this node from our neighbor table.

4.2.2 Time Complexity

The time-complexity of the proposed scheme for maintaining 2-hop neighbor tables basically depends on Bloom filter membership tests (Eq. (5)) and calculations of subsets of 1-hop neighbors (Eqs. (7) to (10)). The time-complexity of a Bloom filter membership test does not depend on the number of already inserted elements but only on the number of used hash functions k, which can be derived by Eq. (2). Usually k is small, e.g., $k = 37$ for a Bloom filter size $m = 300\,\text{Byte}$ and an element count $n = 45$. Therefore, to test whether a given element is with high probability part of the Bloom filter, the time-complexity is $\mathcal{O}(k)$ since k hash values need to be calculated. The same time-complexity also holds for inserting an element to the Bloom filter.

To decide whether a node should be deleted or just marked as hidden, the following time-complexities hold: The subset calculation of hidden nodes in Eq. (7), of closer neighbors in Eq. (8), and of nodes from whom a beacon was received after the announced beacon time in Eq. (9) fall each into $\mathcal{O}(n)$, where n denotes the number of 1-hop neighbors. This process can be optimized at runtime by performing all necessary calculations within one traversal of the 1-hop neighbor table. Moreover, for each node remaining in $\mathbb{E}_{x,y}$, we perform a membership test in the corresponding Bloom filter, where the traversal through the set can be stopped whenever a membership test was successful. Only in the worst case (when a node is being marked as hidden), the whole traversal through the list has to be performed.

The overall process needs to be repeated for all 1-hop neighbors leading to a time-complexity of $\mathcal{O}(k \times n^2)$ for n 1-hop neighbors and k hash functions. For a typical scenario of $n = 45$ 1-hop neighbors, a Bloom filter size of $300\,\text{Byte}$, and a number of $k = 37$ hash functions, we obtain $74\,925$ calculations of hash functions to be performed whenever we update the neighbor table, which is necessary whenever we send a beacon. For an Intel Atom® N450, which offers 1.6 MHash/s [42] employing the SHA256 hash function, this would lead up to 749 250 calculations per second and, thus, to a very high CPU utilization when using 10 Hz beaconing. However, as we show in Section 6.3, a beaconing rate of 10 Hz would overload anyhow the wireless channel, thus, using lower beaconing frequencies in the range of up to 1 s will also decrease the computation overhead. Moreover, Bloom filters can also take advantage of different and more inexpensive hash functions than SHA256, again lowering the computational effort. A typical optimization for Bloom filters [43] exploits the fact that k different hash functions can be replaced by just two hash functions as follows: $g_i(x) = h_1(x) + i \times h_2(x)$, where the index i denotes the hash function within k and $g_i(x)$ is the resulting hash function used by the Bloom filter. This way, the time-complexity of our algorithm degrades to $O(n^2)$. This set of computations can even be parallelized since there are no dependencies among traversals through the neighbor table.

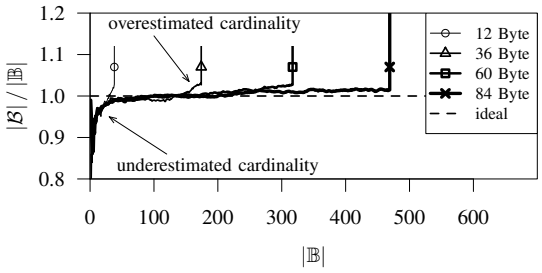


Fig. 3. Estimating the number of inserted elements in a Bloom filter. The line width depicts the Bloom filter size; the dashed line shows the ideal behavior.

4.3 Cardinality Estimation of Bloom Filters

We conducted several Monte-Carlo simulations to show the performance of cardinality estimation, i.e., to approximate the number of elements in a Bloom filter and the false positive rate (cf. Section 3.2). Our simulation setup consists of two Bloom filters $\mathcal{A}$ and $\mathcal{B}$ having the same bit length and using the same set of hash functions. We randomly fill both Bloom filters and made sure that ⅓ of the entries is added to both filters, i.e., both Bloom filters overlap to 50 %. We performed simulations for different Bloom filter sizes and repeated each experiment at least 100 times with different seeds to obtain statistically significant results.

In Fig. 3, we show how good the cardinality of a Bloom filter $|\mathcal{B}|$ can be approximated compared to the true amount

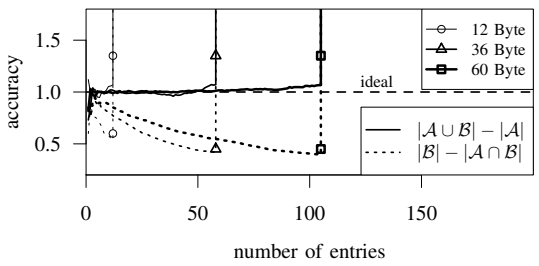


Fig. 4. Estimation accuracy for the number of additional entries provided by a certain Bloom filter normalized to the ground truth. The line width depicts the Bloom filter size; dotted and solid lines represent the two calculation options.

of elements $|\mathbb{B}|$ inserted in this particular Bloom filter. The approximation closely matches the ideal behavior until to the point where the Bloom filter is filled, i.e., almost all bits are set to 1. This provides us an upper bound of elements to be inserted in a Bloom filter of a given size.

In a second experiment, we investigated the applicability of calculating $\mathcal{A} \cap \mathcal{B}$. Results show that the fraction of false positives when building a Bloom filter from scratch containing exactly the intersecting elements is much lower than the Bloom filter gained from the intersection (data not shown due to length constraints).

The overall objective, however, is to assess the number of 2-hop neighbors that are not direct 1-hop neighbors. The quality of a specific Bloom filter to estimate $\text{diff}(\mathcal{A}, \mathcal{B})$, the number of entries in a foreign filter $\mathcal{B}$ that are not part of a local filter $\mathcal{A}$, can formally be expressed as

$$|\mathcal{A} \cup \mathcal{B}| - |\mathcal{A}| \stackrel{?}{=} \text{oracle} \stackrel{?}{=} |\mathcal{B}| - |\mathcal{A} \cap \mathcal{B}| \,. \tag{12}$$

Figure 4 shows the results normalized to the ground truth. The estimation using the union of the Bloom filters follows the ideal line until one of the two Bloom filters is completely filled, except for a slight overestimation when the Bloom filter is getting close to saturation. In contrast, the approach using the intersection significantly underestimates the number of new elements. We thus conclude that the use of

$$\text{diff}(\mathcal{A}, \mathcal{B}) = |\mathcal{A} \cup \mathcal{B}| - |\mathcal{A}| \tag{13}$$

to estimate the amount of additional neighbors is the most appropriate option.

5 Bloom Filter based Multi-Hop Broadcast

2-hop (N-hop) neighbor tables have a broad range of applications for improving the management of dynamic networks. In this section, we show how the specific Bloom filter based neighbor management protocol can be extended to also support efficient broadcast-based data dissemination, e.g., for warning messages or as the basis for general VANETs.

Our proposed multi-hop dissemination algorithm Bloom Hopping works as follows. We start with the idea proposed in [2], where every node x that receives or generates a packet to be broadcast nominates a forwarding set of nodes which then use a scheme similar to Contention-based Forwarding (CBF) [44]. However, instead of contending for a rebroadcast, our scheme x chooses as many 1-hop forwarders as necessary to reach all (or a sufficiently large subset of) its 2-hop neighbors. To this end, node x uses the Bloom filters maintained by the beacon protocol for neighbor management. Specifically, node x has to choose a set of re-broadcasters $\mathbb{R}_x$ as the minimum subset of visible 1-hop neighbors $\mathbb{X}'$ to cover all of its 2-hop neighbors.

The minimum set cover problem is *NP*-hard [45]. Thus, we decided to have node x using the set $\mathbb{R}_x$ by a greedy iterative process that selects a node u that has most new (uncovered) neighbors and has not yet been selected as a rebroadcast node. Formally, node x starts with an empty set of re-broadcasters $\mathbb{R}_x$ and a Bloom filter of already-covered 2-hop neighbors $\widehat{\mathcal{X}}''$, which is initialized to the symmetric 1-hop neighbors, i.e., $\widehat{\mathcal{X}}'' \leftarrow \mathbb{X}'$. It repeatedly chooses the best node u as

$$u = \arg\max_{u \in \mathbb{X}'} \left(\text{diff}(\widehat{\mathcal{X}}'', \mathcal{X}_u) \right) \tag{14}$$

and adds the chosen 1-hop neighbor u to $\mathbb{R}_x$ and its Bloom filter $\mathcal{X}_u$ to $\widehat{\mathcal{X}}''$, that is $\widehat{\mathcal{X}}'' \leftarrow (\widehat{\mathcal{X}}'' \cup \mathcal{X}_u)$. The process ends when all 2-hop neighbors (or a sufficiently large subset) are covered, as can be derived by comparing $\widehat{\mathcal{X}}''$ and $\mathcal{X}''$.

The set $\mathbb{R}_x$ now contains all 1-hop neighbors selected to rebroadcast the message. Since $\mathbb{R}_x$ is usually small (e.g., it is close to 2 in freeway scenarios), it is added to the broadcast message. In addition, to prevent collisions between rebroadcasting 1-hop neighbors, node x adds to the message the value of an artificial delay for each such neighbor. This delay could be made dependent on the actual link quality reported by that particular neighbor in the periodic beacons, or it could be derived by a more sophisticated approach like the following: For each pair of neighbors u, v which have in their Bloom filters no other neighbors in common but the node x, we select a zero rebroadcast delay. This has the advantage that the message gets forwarded very quickly without risking packet collisions at receiving nodes since the rebroadcast regions do not overlap on the neighbors. For all other rebroadcast nodes, a slightly different delay is chosen for rebroadcasting, to avoid possibilities of packet collisions.

In dense networks, e.g., in urban scenarios, node x can make the packet dissemination process more robust by adding more 1-hop neighbors to $\mathbb{R}_x$. If the cardinality of $\mathbb{R}_x$ is too large to include all chosen 1-hop neighbors, the addresses of these nodes can be replaced by a Bloom filter $\mathcal{R}_x \leftarrow \mathbb{R}_x$.

6 Performance in VANETs

6.1 Realistic Road Traffic and Network Simulation Setup

For all simulations, we used the vehicular networking simulation toolkit Veins [46], which couples the SUMO road traffic simulator with the network simulator OMNeT++. We used synthetic but very realistic road traffic modeled by SUMO in favor of road traffic traces since it allows us to easily control the scenario in terms of traffic density.

We first configured a six-lane freeway of which the network simulator used 5 km. We collected protocol performance metrics in a Region of Interest (ROI) of 3 km to avoid border effects. Road traffic was modeled as a mixture of 90 % cars and 10 % trucks by sampling from a distribution of five different vehicle types (two types of trucks and three

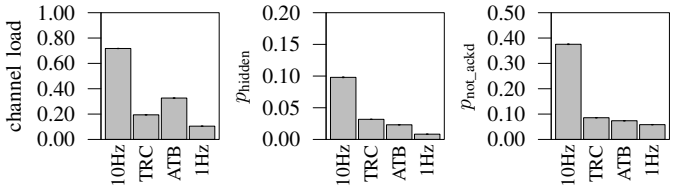
TABLE 1
Vehicular network simulation parameters

ETSI ITS-G5 TRC	
Minimum beacon interval	$I_{\text{min}} = 40\,\text{ms}$
Default beacon interval	$I_{\text{def}} = 500\,\text{ms}$
Maximum beacon interval	$I_{\text{max}} = 1\,\text{s}$
b_{min}	0.15
b_{max}	0.40
ATB	
Minimum beacon interval	$I_{\text{min}} = 100\,\text{ms}$
Maximum beacon interval	$I_{\text{max}} = 1\,\text{s}$
Interval weight w_{I}	0.75
Beacon message	
Packet size	200 Byte + Bloom filter
Bloom filter size	from 12 Byte to 350 Byte
MAC priority	AC_BE
	$\text{CW}_{\text{min}} = 15$, $\text{CW}_{\text{max}} = 1023$
	AIFSN = 6
Multi-hop message	
Packet size	300 Byte
MAC priority	AC_VO
	$\text{CW}_{\text{min}} = 3$, $\text{CW}_{\text{max}} = 7$
	AIFSN = 2
IEEE 802.11p PHY	
NIC bitrate	6 Mbit/s
NIC TX power	20 mW
Path loss model	freespace ($\alpha = 2.0$)
building obstacle shadowing	$\beta = 9\,\text{dB}$, $\gamma = 0.4\,\text{dB/m}$

types of cars modeling a variety of driving styles). We used two different road traffic densities of ∼43 and ∼148 veh/km to model low and high density traffic. Second, we conducted our simulations in a scenario similar to a Manhattan grid with a road traffic density of ∼400 veh/km^2 and four types of cars modeling a variety of driving styles and a ROI of 2.1 km^2 as well as buildings within the scenario. We used a warm-up period to fill the roads with vehicles and reach a steady state of neighbor table protocol operations, as well as to pre-populate 1-hop and 2-hop neighbor tables.

For beaconing, we selected three approaches as a baseline to compare the performance of our Bloom filter based approach. In particular, we used simple fixed period beaconing (at 1 Hz and 10 Hz), originally defined for sending CAM messages, Transmit Rate Control (TRC) [15] of the ETSI ITS-G5 DCC standard.

For evaluating the performance of our Bloom Hopping protocol, every 100 ms we randomly select 10 vehicles uniformly distributed within the ROI to disseminate messages. We performed simulations for different Bloom filter sizes and show a subset of the most interesting results. For all simulation experiments, we performed at least 5 runs with different random seeds for simulating road and network traffic to obtain statistically significant results – some of the experiments indeed took close to half a day but calculated confidence intervals clearly show that the collected measurements are statistically sound. The most important simulation parameters are summarized in Table 1.



(a) Channel utilization (b) p_{hidden} (c) $p_{\text{not\_ackd}}$

Fig. 5. Beaconing performance for the high density freeway scenario (average value and 95 % confidence interval).

6.2 Performance Evaluation Using an Oracle

We investigated the following metrics to observe the protocol performance.

First, we focus on the beacon interval as well as the channel utilization to assess the impact of the additional neighbor information piggybacked in the beacons. Furthermore, we measure the fraction of nodes that we did not purge from the neighbor table as $p_{\text{hidden}} = \frac{|\bar{\mathbb{X}}|}{|\mathbb{X}'|}$ which can be used as an additional metric to measure channel quality. We also measure the fraction of not acknowledged neighbors of a node as $p_{\text{not\_ackd}} = 1 - \frac{|\mathbb{X}'|}{|\mathbb{X}|}$ to measure how symmetric the communication channel is. Following up to results published in [47], we evaluate the neighbor churn rate, which shows the fraction of deleted 1-hop neighbors per second due to lost beacons or because a neighbor moved outside the communication range. The churn rate helps to understand the dynamics of the network and, thus, the fluctuations in neighborship information.

Second, in order to assess the quality of the neighbor tables, i.e., the up-to-dateness of 1-hop and 2-hop entries, we compare the results to an oracle. Instead of using a simplistic oracle based on a unit disk model, we developed a more sophisticated method by taking advantage of an idealistic MAC and PHY ignoring packet collisions and delays caused by CSMA/CA and EDCA queues. A collision is defined to be a packet that could have been correctly decoded if there would not have been any interference. We use a 10 Hz beaconing scheme to populate our oracle with neighbors and create a database for each simulation run containing the 1-hop neighbors $\mathbb{O}'$, and 2-hop neighbors $\mathbb{O}''$ for each vehicle over time. In our simulation setup each vehicle has on average 44 one-hop and 41 two-hop neighbors (freeway low density); 169 one-hop and 159 two-hop neighbors (freeway high density); and 54 one-hop and 131 two-hop neighbors (Manhattan).

The fraction of missing 1-hop neighbors of a node x compared to the oracle is calculated as $p'_{\text{missing}} = \frac{|\mathbb{O}' \setminus \mathbb{X}'|}{|\mathbb{O}'|}$. The fraction of outdated 1-hop neighbors of a node x is the relative amount of unnecessary neighbors compared to the oracle as $p'_{\text{outdated}} = \frac{|\mathbb{X}' \setminus \mathbb{O}'|}{|\mathbb{X}'|}$. Similar metrics were recorded for 2-hop neighbors as p''_{missing} and p''_{outdated}. These metrics are collected every 100 ms after the warm-up period.

Finally, for multi-hop message dissemination, both the fraction of informed nodes and the channel utilization were recorded. For all results, we plot the average value together with the 95 % confidence interval.

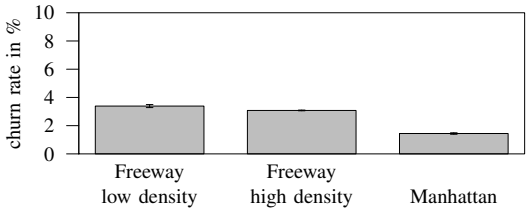


Fig. 6. Neighbor churn rate for different scenarios and traffic densities (average value and 95 % confidence interval).

6.3 Impact on Beaconing

In a first experiment, we investigate the impact of the Bloom filter approach on the beaconing performance using a fixed size Bloom filter of 240 Byte. In Fig. 5a, we plot the channel utilization for the high density freeway scenario. Both TRC and ATB are rather sensitive to the channel utilization and configure the beacon interval (data not shown due to space constraints) to higher values in the high density scenario, even though ATB is increasing the interval less prohibitive (resulting in lower delays). These results are in line with recent studies on beacon protocols and helped validate both the setup and the beacon protocol implementation [6], [8]. We can clearly see that 10 Hz beaconing strongly overloads the wireless channel in the high traffic density scenario. For the Manhattan scenario we observe qualitative similar results, but in general with lower beacon intervals and channel utilization due to obstacle shadowing (data not shown). In particular ATB choses a slightly larger beacon interval than TRC to cope with the network dynamic caused by obstacle shadowing and hidden terminals.

We expect neighbor information being less up-to-date for high vehicle density; particularly for TRC and even more critical for 10 Hz beaconing. Thus, we assess the metrics p_{hidden} and $p_{\text{not\_ackd}}$, which focus on measuring the channel conditions in Figs. 5b and 5c, again using a Bloom filter size of 240 Byte and the high density freeway scenario. As can be seen, the performance of 10 Hz beaconing in the high density scenario is poor compared to all other beaconing protocols. The cause is the highly overloaded channel leading to unstable neighbor tables. Also TRC performs worse compared to ATB and 1 Hz beaconing; this is because for TRC the protocol state machine has rather large steps between the available beacon intervals. This means that a neighbor might be dropped due to a quick change in the beacon configuration. ATB does not suffer from this problem, thus, p_{hidden} and $p_{\text{not\_ackd}}$ stay at a lower value. This of course also holds for 1 Hz beaconing. Overall, these metrics are rather sensitive to the channel load. With lower channel load, the TRC results are much better (e.g., for the low density scenario), still TRC suffers from a higher number of hidden neighbors due to the operation of the protocol state machine. For the Manhattan scenario the results show similar effects: both TRC and 10 Hz beaconing suffer from a higher value of p_{hidden} compared to ATB and 1 Hz beaconing. ATB keeps low values as it carefully adjusts the beacon interval according to the channel state. In general, $p_{\text{not\_ackd}}$ is even more sensitive to channel congestion as well as to oscillating beacon frequencies.

To assess the number of deleted neighbors per second, we show the results for the low and the high density freeway scenario as well as the Manhattan scenario in Fig. 6 using 1 Hz beaconing and a fixed size Bloom filter of 240 Byte. In essence, we observe around 3 % and 3.3 % deleted neighbors per second for the low and high density freeway scenario, respectively. For the Manhattan scenario, we observe around 1.5 % deleted neighbors per second, which is caused by the slower driving speed of the vehicles compared to the Freeway scenario.

6.4 Bloom Filter based Neighbor Table Management

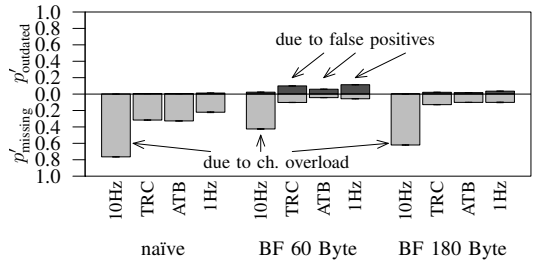
We now concentrate on the performance of the Bloom filter based approach for neighbor management. We start investigating the capabilities of the beacon protocols to maintain the neighbor tables. In particular, we show the results of outdated and missing entries compared against an oracle. In addition, we compare our results with a naïve baseline neighbor management protocol that exchanges 2-hop neighbor information using a list of identifiers without the usage of Bloom filters. The beacon size of this baseline approach grows linear with the amount of neighbors; we call this method *naïve*. For the Bloom filter, we show results for two different filter sizes, one being too small, thus, suffering from false positives as well as a fitting (for the particular scenario) Bloom filter.

In Fig. 7, we show the results for the high density freeway and the Manhattan scenario. Intuitively, we expect larger and more accurate neighbor tables the more frequently we exchange information, i.e., the smaller the beacon interval gets. As can be seen in the results, this hypothesis cannot be confirmed. In high density scenario, the load on the channel is so high that no continuous update is possible due to collisions. Our observation is in line with findings on beaconing approaches published elsewhere, e.g., in [8].

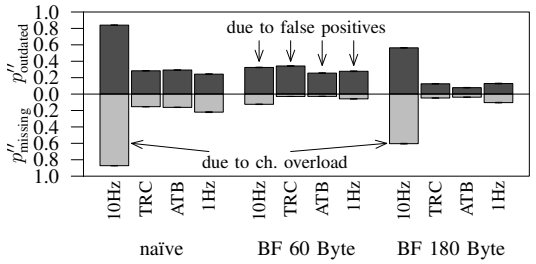
The overall outdated and missing fraction is a good indicator of how well the protocols are able to maintain the neighbor tables compared to the oracle. If the Bloom filters are too small, the outdated ratio increases due to false positives. This is not only valid for 2-hop neighbors, but also influences the fraction of outdated 1-hop neighbors (cf. Eq. (10)). On the other hand, a too big Bloom filter directly increases the channel load (and, thus, the collision probability). Thus, a compromise has to be found between the Bloom filter size and the false positive rate.

The fraction of outdated 1-hop neighbors is quite low for all protocols (cf. Figs. 7a and 7c). This comes from the fact that entries are removed from the list when there was no beacon received up until the next announced interval and no further away node with lower or equal channel utilization includes this node in the Bloom filter. In particular, when a too small Bloom filter is used, false positives causes a node to not delete 1-hop neighbors leading to a higher amount of outdated neighbors. We observed similar results in the low density freeway scenario (data not shown).

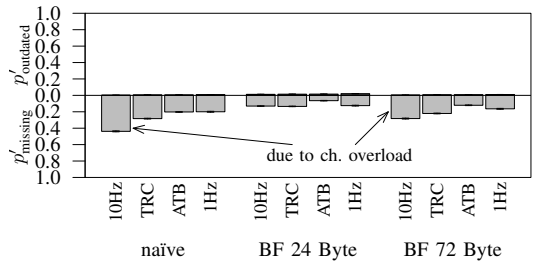
The fraction of missing entries, however, is higher for high vehicle density and particularly for the naïve approach. We also see that in the high density freeway scenario the 60 Byte Bloom filter experiences a lower missing 1-hop ratio compared to the 180 Byte version due to a smaller beacon size. This is due to packet collisions, thus, missing updates. Having a closer look at 1 Hz beaconing in the high density



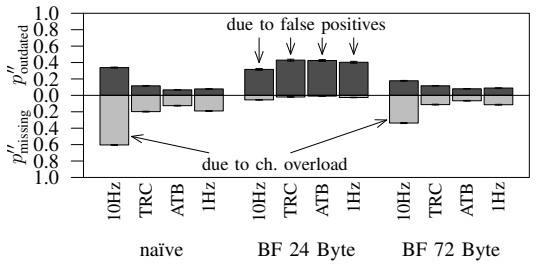
(a) Freeway: Fraction of missing and outdated 1-hop neighbors.



(b) Freeway: Fraction of missing and outdated 2-hop neighbors.



(c) Manhattan: Fraction of missing and outdated 1-hop neighbors.



(d) Manhattan: Fraction of missing and outdated 2-hop neighbors.

Fig. 7. Neighbor ratios (1-hop and 2-hop) for the high density freeway and Manhattan scenario: Naïve solution compared to our approach using Bloom filters with different sizes (average value and 95 % confidence interval).

freeway scenario, the number of missing 1-hop neighbors is lowered by 54 % when using a Bloom filter size of 180 Byte compared to the naïve approach. In the Manhattan scenario, the results for 1 Hz beaconing change to a decrease of 18 % for the missing 1-hop neighbors for a Bloom filter size of 72 Byte.

The results for 2-hop neighbor information behave similarly (cf. Figs. 7b and 7d), but the outdated neighbors ratio is higher since dissemination time accumulates over 2 hops. Moreover, a too small Bloom filter size increases the amount of outdated information due to false positives when querying the Bloom filters. The impact of the channel load becomes even more visible: the naïve approach for 10 Hz beaconing completely fails due to the congested channel. When the communication channel is completely overloaded (naïve 10 Hz beaconing), even our approach to mitigate the wrong assignment of 1-hop neighbors as 2-hop neighbors fails. Here the high value of p''_{outdated} is caused by 1-hop neighbors reporting nodes as 2-hop neighbors which we normally could reach within 1-hop when no permanent channel congestion occurs. For 1 Hz beaconing, we observe a decrease of 53 % of missing and a decrease of 47 % of outdated 2-hop neighbors when using a Bloom filter size of 180 Byte compared to the naïve approach. In the Manhattan scenario, the results for 1 Hz beaconing change to a decrease of 40 % for the missing 2-hop neighbors when using a Bloom filter size of 72 Byte, while the fraction of outdated neighbors (due to false positives) slightly increases by 14 % compared to the naïve approach.

In all scenarios, we clearly see the potential of using Bloom filters: with a fitting Bloom filter size of 60 Byte (low density freeway), 180 Byte (high density freeway), 72 Byte (Manhattan), the amount of missing and outdated information is very small for ATB, TRC, and 1 Hz beaconing.

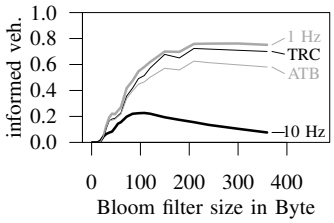
6.5 Bloom Hopping Performance

To measure the performance of message dissemination using Bloom Hopping, we record the fraction of 2-hop nodes that receive the message and monitor the observed wireless channel utilization. We performed a parameter study for different Bloom filter sizes m and plot the results for all underlying beacon protocols providing the needed neighborship information. Further we compare our Bloom Hopping protocol against the naïve approach in which we compute our forwarding set similar to Section 5 without using Bloom filters.

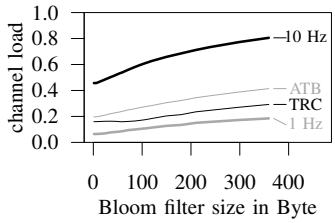
In Figs. 8 and 9, we show the results for the high density freeway and the Manhattan grid scenario, respectively. We can see that Bloom Hopping works best using 1 Hz beaconing, reaching about 80 % of all 2-hop nodes in all scenarios (cf. Figs. 8a and 9a). The missing 20 % come from the fact that increasing load on the wireless channel (cf. Figs. 8b and 9b) lowers the communication distance due to interference. These nodes have no chance receiving the message in reality. Similar results can be observed in the low density freeway scenario.

For the high density freeway scenario, we can further see that when using Bloom Hopping and 1 Hz beaconing, we reach around 12 % more 2-hop neighbors (cf. Fig. 8c) compared to the naïve approach. At the same time, we save around 43 % channel load (cf. Fig. 8d) compared to the naïve approach, again using 1 Hz beaconing. The results for the Manhattan scenario are comparable (cf. Figs. 9c and 9d).

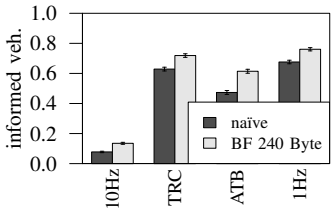
When using TRC or ATB, we notice a slight performance degradation in the high density freeway scenario. This is due to the increasing channel load, which leads to outdated or inaccurate neighbor information (cf. Section 6.4). Worst results have been collected for 10 Hz beaconing, which simply overloads the wireless channel in all scenarios. As can be seen in Figs. 8a and 9a, the Bloom filter size has a great



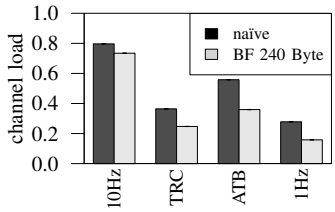
(a) Fraction of informed 2-hop neighbors vs. Bloom filter size.



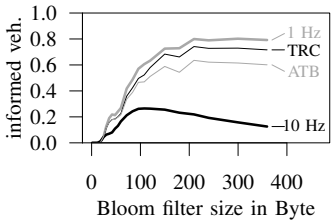
(b) Wireless channel utilization vs. Bloom filter size.



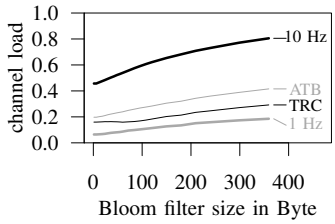
(c) Fraction of informed 2-hop neighbors using a fitting Bloom filter size.



(d) Wireless channel utilization using a fitting Bloom filter size.

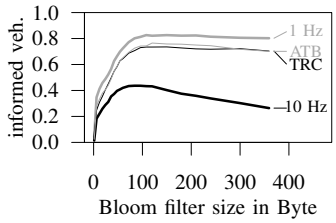


(e) Fraction of informed 2-hop neighbors vs. Bloom filter size incorporating non-symmetric nodes.

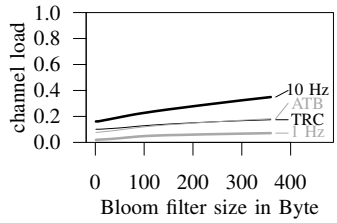


(f) Wireless channel utilization vs. Bloom filter size incorporating non-symmetric nodes.

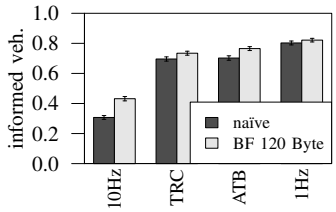
Fig. 8. Fraction of informed 2-hop neighbors and channel utilization for the high density freeway scenario using different Bloom filter sizes.



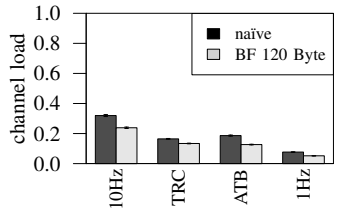
(a) Fraction of informed 2-hop neighbors vs. Bloom filter size.



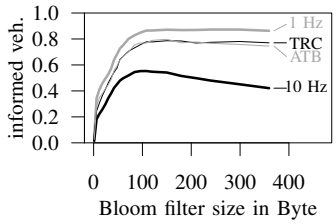
(b) Wireless channel utilization vs. Bloom filter size.



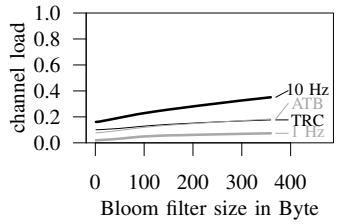
(c) Fraction of informed 2-hop neighbors using a fitting Bloom filter size.



(d) Wireless channel utilization using a fitting Bloom filter size.



(e) Fraction of informed 2-hop neighbors vs. Bloom filter size incorporating non-symmetric nodes.



(f) Wireless channel utilization vs. Bloom filter size incorporating non-symmetric nodes.

Fig. 9. Fraction of informed 2-hop neighbors and channel utilization for the Manhattan scenario using different Bloom filter sizes.

impact on the fraction of received nodes, as well as on the channel utilization. At the beginning, the amount of received neighbors increases with increasing size of Bloom filter, thus decreasing false positives. After reaching a maximum at around 240 Byte (high density freeway) and 120 Byte (Manhattan) the performance degrades due to increased channel load if the size of Bloom filters increase.

When we modify the Bloom Hopping algorithm outlined in Eq. (14) to include also non-symmetric nodes $\mathbb{X}$ instead of only symmetric nodes $\mathbb{X}'$ as calculated in Eq. (5), we observe very similar results for the fraction of informed vehicles and the channel load as shown in Figs. 8e and 8f for the freeway scenario. A non-symmetric link occurs when node x can receive messages from node y but not vice versa, e.g., due to interference. Intuitively, we would expect a much lower rate of informed vehicles when nodes with a asynchronous wireless link would be selected to retransmit a particular message but cannot receive the message due to interference. However, in our case the Bloom Hopping algorithm solves this issue as only those neighbors get selected to rebroadcast messages that gain additional uncovered 2-hop neighbors. As a node that suffers from high interference and thus observes a lower amount of decodable messages, it has a limited overview of neighboring nodes and, therefore, announces only a lower amount of direct 1-hop neighbors within its Bloom Filter. Therefore, a node which announces a higher amount of neighbors can gain more to cover all 2-hop neighbors and, thus, gets chosen in favor to a node only having a very small number of neighbors. Only in very rare cases, nodes with a non-symmetric wireless link can gain a very small contribution to reach further nodes, which can be observed in the Manhattan scenario shown in Figs. 9e and 9f.

Qualitatively comparing our results to other Bloom filter based message dissemination approaches like [21], we clearly see the following advantage of our system: The lowered channel utilization gained by transmitting a much smaller Bloom filter instead of a large list of neighbors leads to an increased number of informed vehicles for the Bloom Hopping message dissemination protocol as can be seen in Figs. 8 and 9. This is in contrast to existing work (e.g., [21]), where only the overhead of beacon transmissions (i.e., the channel utilization) is lowered, but no better performance of the application is achieved. Further, as outlined in Section 4.3, the intersection operation on Bloom filters causes loss of information and, thus, is not recommended as we show in Fig. 4. Yet, using the no longer recommended 10 Hz beaconing, which completely overloads the channel, shows an astonishing performance improvement when using our Bloom filter based neighbor management.

In summary, we can say that our Bloom Hopping approach increases the fraction of informed nodes and at the same time lowers the channel utilization.

7 Conclusions

We presented a novel probabilistic 2-hop neighbor management approach using Bloom filters for application in dynamic wireless networks. Compared to alternative solutions, the use of Bloom filters provides best scalability of the system that comes at the cost of a small false positive rate. We analytically explored the Bloom filter properties for this application field and determined best suited Bloom filter sizes to keep this false positive rate marginally small. This also helps to prevent overload on the wireless channel. We also explored the capabilities of our solution to build a fundamental basis for higher layer protocols. As an example, we designed the multi-hop broadcast protocol Bloom Hopping, which very efficiently selects forwarders using the Bloom filter encoded neighbor information.

We performed an extensive simulation study to assess the performance of the developed Bloom filter based neighbor management as well as the Bloom Hopping protocol. For the 2-hop neighbor management, we compared our Bloom filter based solution with an oracle. For this, we carefully explored theoretical reachability in the wireless network ignoring interference and delays at the MAC protocol but using a realistic path loss model. We clearly see that the difference to the oracle is very small given that the Bloom filter size has been adequately chosen for the application scenario. We further demonstrated that the Bloom filter approach can easily be tied to typical beacon protocols given that they are able to provide simple congestion control. We further found that higher beaconing rates not necessarily lead to better and more accurate neighborship information, which might be considered counter intuitive. The load on the wireless channel has been confirmed to be the most fundamental limit. Furthermore, Bloom filters as used in our Bloom Hopping protocol help to improve information dissemination and decrease channel utilization significantly. This clearly shows the applicability of the Bloom filter based neighborship information for higher layer protocols.

We want to emphasize that even though we investigated our approach in a very specific application domain, namely Vehicular Ad Hoc Networks (VANETs), the concept can be applied also to other networking scenarios. Without loss of generality it can be said, the Bloom filter based 2-hop neighborship management as well as the Bloom Hopping algorithm can be of benefit in all types of networks exhibiting a very dynamic topology.

References

[1] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, Nov. 2014.

[2] K. Lee, U. Lee, and M. Gerla, "Geo-Opportunistic Routing for Vehicular Networks," *IEEE Communications Magazine*, vol. 48, no. 5, pp. 164–170, May 2010.

[3] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[4] O. Abumansoor, A. Boukerche, B. Landfeldt, and S. Samarah, "Privacy Preserving Neighborhood Awareness in Vehicular Ad Hoc Networks," in *14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2011), 7th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet 2011)*, Miami, FL: ACM, Jul. 2011, pp. 17–20.

[5] O. K. Tonguz, N. Wisitpongphan, and F. Bai, "DV-CAST: A distributed vehicular broadcast protocol for vehicular ad hoc networks," *IEEE Wireless Communications*, vol. 17, no. 2, pp. 47–57, Apr. 2010.

[6] C. Sommer, O. K. Tonguz, and F. Dressler, "Traffic Information Systems: Efficient Message Dissemination via Adaptive Beaconing," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 173–179, May 2011.

[7] F. J. Ros, P. M. Ruiz, and I. Stojmenovic, "Acknowledgment-Based Broadcast Protocol for Reliable and Efficient Data Dissemination in Vehicular Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 33–46, Jan. 2012.

[8] C. Sommer, S. Joerer, M. Segata, O. K. Tonguz, R. Lo Cigno, and F. Dressler, "How Shadowing Hurts Vehicular Communications and How Dynamic Beaconing Can Help," *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1411–1421, Jul. 2015.

[9] K. Lee, J. Haerri, U. Lee, and M. Gerla, "Enhanced Perimeter Routing for Geographic Forwarding Protocols in Urban Vehicular Scenarios," in *IEEE Global Telecommunications Conference (GLOBECOM 2007), 2nd IEEE Workshop on Automotive Networking and Applications (AutoNet 2007)*, Washington, DC: IEEE, Nov. 2007, pp. 1–10.

[10] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14–25, Jan. 2002.

[11] A. Khan, I. Stojmenovic, and N. Zaguia, "Parameterless broadcasting in static to highly mobile wireless ad hoc, sensor and actuator networks," in *22nd IEEE International Conference on Advanced Information Networking and Applications (AINA 2008)*, Okinawa, Japan, Mar. 2008, pp. 620–627.

[12] H. A. Omar, W. Zhuang, and L. Li, "VeMAC: A TDMA-Based MAC Protocol for Reliable Broadcast in VANETs," *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1724–1736, Sep. 2013.

[13] F. Dressler, F. Klingler, C. Sommer, and R. Cohen, "Not All VANET Broadcasts Are the Same: Context-Aware Class Based Broadcast," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 17–30, Feb. 2018.

[14] E. Van de Velde and C. Blondia, "Adaptive REACT protocol for Emergency Applications in Vehicular Networks," in *32nd IEEE Conference on Local Computer Networks (LCN 2007)*, Dublin, Ireland: IEEE, Oct. 2007, pp. 613–619.

[15] ETSI, "Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part," ETSI, TS 102 687 V1.1.1, Jul. 2011.

[16] D. Barbieri, I. Thibault, D. Lister, A. Bazzi, B. M. Masini, and O. Andrisano, "Adaptive beaconing for safety enhancement in vehicular networks," in *15th International Conference on ITS Telecommunications (ITST 2017)*, Warsaw, Poland: IEEE, May 2017.

[17] M. Segata, F. Dressler, and R. Lo Cigno, "Jerk Beaconing: A Dynamic Approach to Platooning," in *7th IEEE Vehicular Networking Conference (VNC 2015)*, Kyoto, Japan: IEEE, Dec. 2015, pp. 135–142.

[18] T. Tielert, D. Jiang, Q. Chen, L. Delgrossi, and H. Hartenstein, "Design Methodology and Evaluation of Rate Adaptation Based Congestion Control for Vehicle Safety Communications," in *3rd IEEE Vehicular Networking Conference (VNC 2011)*, Amsterdam, Netherlands: IEEE, Nov. 2011, pp. 116–123.

[19] G. Bansal, J. Kenney, and C. Rohrs, "LIMERIC: A Linear Adaptive Message Rate Algorithm for DSRC Congestion Control," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 9, pp. 4182–4197, Nov. 2013.

[20] B. Cheng, A. Rostami, M. Gruteser, H. Lu, J. B. Kenney, and G. Bansal, "Evolution of Vehicular Congestion Control Without Degrading Legacy Vehicle Performance," in *17th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2016)*, Coimbra, Portugal: IEEE, Jun. 2016.

[21] K. Na Nakorn, Y. Ji, and K. Rojviboonchai, "Bloom Filter for Fixed-Size Beacon in VANET," in *79th IEEE Vehicular Technology Conference (VTC2014-Spring)*, Seoul, Korea: IEEE, May 2014, pp. 1–5.

[22] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," in *3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM 1999)*, Seattle, WA: ACM, Aug. 1999, pp. 7–14.

[23] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, Jan. 2005.
[24] A. Marandi, M. F. Imani, and K. Salamatian, "Practical Bloom filter based epidemic forwarding and congestion control in DTNs: A comparative analysis," *Elsevier Computer Communications*, vol. 48, pp. 98–110, Jul. 2014.
[25] Y.-T. Yu, M. Gerla, and M. Y. Sanadidi, "Scalable VANET Content Routing Using Hierarchical Bloom Filters," *Wireless Communications and Mobile Computing*, vol. 15, no. 6, 1001–1014, Apr. 2015.
[26] A. Bujari, "A Network Coverage Algorithm for Message Broadcast in Vehicular Networks," *ACM/Springer Mobile Networks and Applications (MONET)*, Apr. 2016.
[27] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree Bloom: Scalable Opportunistic Routing with ORPL," in *11th ACM Conference on Embedded Networked Sensor Systems (SenSys 2013)*, Rome, Italy: ACM, Nov. 2013.
[28] F. Angius, M. Gerla, and G. Pau, "BLOOGO: BLOOm Filter Based GOssip Algorithm for Wireless NDN," in *13th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2012), 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications (NoM 2012)*, Hilton Head, SC: ACM, Jun. 2012, pp. 25–30.
[29] K. Lin and P. Levis, "Data Discovery and Dissemination with DIP," in *6th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2008)*, St. Louis, MO: IEEE, Apr. 2008, pp. 433–444.
[30] A. Reinhardt, O. Morar, S. Santini, S. Zöller, and R. Steinmetz, "CBFR: Bloom Filter Routing with Gradual Forgetting for Tree-structured Wireless Sensor Networks with Mobile Nodes," in *13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2012)*, San Francisco, CA: IEEE, Jun. 2012.
[31] P.-H. Hsiao, "Geographical Region Summary Service for Geographical Routing," *SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 4, pp. 25–39, Oct. 2001.
[32] C. M. Silva, B. M. Masini, G. Ferrari, and I. Thibault, "A Survey on Infrastructure-Based Vehicular Networks," *Hindawi Mobile Information Systems*, Aug. 2017, Article ID 6123868.
[33] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "On the Performance of IEEE 802.11p and LTE-V2V for the Cooperative Awareness of Connected Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 419–10 432, Nov. 2017.
[34] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.
[35] D. Guo, Y. Liu, X. Li, and P. Yang, "False negative problem of counting bloom filter," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 651–664, Mar. 2010.
[36] T. Darwish and K. A. Bakar, "Traffic density estimation in vehicular ad hoc networks: A review," *Elsevier Ad Hoc Networks*, vol. 24, no. A, pp. 337–351, Jan. 2015.
[37] L. Garelli, C. Casetti, C.-F. Chiasserini, and M. Fiore, "Mob-sampling: V2V Communications for Traffic Density Estimation," in *73rd IEEE Vehicular Technology Conference (VTC Spring 2011)*, Budapest, Hungary: IEEE, May 2011.
[38] C. Lochert, B. Scheuermann, and M. Mauve, "A probabilistic method for cooperative hierarchical aggregation of data in VANETs," *Elsevier Ad Hoc Networks*, vol. 8, no. 5, pp. 518–530, Jul. 2010.
[39] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for Bloom filters," *Distributed and Parallel Databases*, vol. 28, no. 2-3, pp. 119–156, Dec. 2010.
[40] M. C. Jeffrey and J. G. Steffan, "Understanding Bloom Filter Intersection for Lazy Address-set Disambiguation," in *23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2011)*, San Jose, CA: ACM, Jun. 2011, pp. 345–354.
[41] S. J. Swamidass and P. Baldi, "Mathematical Correction for Fingerprint Similarity Measures to Improve Chemical Retrieval," *Journal of Chemical Information and Modeling*, vol. 47, no. 3, pp. 952–964, 2007.
[42] K. J. O'Dwyer and D. Malone, "Bitcoin Mining and its Energy Footprint," in *25th IET Irish Signals & Systems Conference (ISSC 2014)*, Limerick, Ireland: IET, Jun. 2014.
[43] A. Kirsch and M. Mitzenmacher, "Less hashing, same performance: Building a better Bloom filter," *Random Structures & Algorithms*, vol. 33, no. 2, pp. 187–218, May 2008.
[44] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, "Contention-based forwarding for mobile ad hoc networks," *Elsevier Ad Hoc Networks*, vol. 1, no. 4, pp. 351–369, 2003.
[45] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 5th, ser. Algorithms and Combinatorics 21. Berlin Heidelberg: Springer, 2012.
[46] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011.
[47] F. Klingler, F. Dressler, and C. Sommer, "IEEE 802.11p Unicast Considered Harmful," in *7th IEEE Vehicular Networking Conference (VNC 2015)*, Kyoto, Japan: IEEE, Dec. 2015, pp. 76–83.

Florian Klingler (klingler@ccs-labs.org) received his BSc and MSc in Computer Science from the University of Innsbruck, Austria, in 2010 and 2012, respectively. In 2012 he was a visiting Research Assistant with the group of Jiannong Cao at the Department of Computing at the Polytechnic University (PolyU) in Hong Kong. He is currently working towards his Ph.D. in the Distributed Embedded Systems Group of Falko Dressler at Paderborn University, Germany. His research focuses on protocols for adaptive wireless networks in highly dynamic scenarios, with applications in vehicular communications.

Reuven Cohen (rcohen@cs.technion.ac.il) received the B.Sc., M.Sc. and Ph.D. degrees in Computer Science from the Technion—Israel Institute of Technology, completing his Ph.D. studies in 1991. From 1991 to 1993, he was with the IBM T.J. Watson Research Center, working on protocols for high speed networks. Since 1993, he has been a professor in the Department of Computer Science at the Technion. He has also been a consultant for numerous companies, mainly in the context of protocols and architectures for broadband access networks. Reuven Cohen has served as an editor of the IEEE/ACM Transactions on Networking and the ACM/Kluwer Journal on Wireless Networks (WINET). He was the co-chair of the technical program committee of Infocom 2010 and headed the Israeli chapter of the IEEE Communications Society from 2002 to 2010.

Christoph Sommer (sommer@ccs-labs.org) is Assistant Professor (Juniorprofessor) of computer science at Paderborn University, Germany. Since 2017 he is heading the Cooperative Mobile Systems group of CCS Labs at the Heinz Nixdorf Institute and the Dept. of Computer Science. He received his Ph.D. degree in engineering (Dr.-Ing., with distinction) and his M.Sc. degree in computer science (Dipl.-Inf. Univ.) from the University of Erlangen in 2011 and 2006, respectively. In 2010, he was a visiting scholar with the research group of Ozan K. Tonguz in the Electrical and Computer Engineering Department at Carnegie Mellon University (CMU). In 2012, he was a visiting scholar with the research group of Mario Gerla in the Computer Science Department at the University of California, Los Angeles (UCLA). Until 2014, he was a postdoctoral research fellow with the Computer and Communication Systems Group at the University of Innsbruck. Before being appointed a professor, he was Akademischer Rat in the Distributed Embedded Systems group at Paderborn University. Since 2011, he serves as a member of the ACM/Springer Wireless Networks (WINET) editorial board. Since 2016, he serves as area editor for Elsevier Computer Communications (COMCOM). His research is focused on questions regarding traffic efficiency, safety, and security aspects of Car-to-X communication in heterogeneous environments. He also authored the textbook Vehicular Networking, published in 2014 by Cambridge University Press.

Falko Dressler (dressler@ccs-labs.org) is full professor of computer science and chair for Distributed Embedded Systems at the Heinz Nixdorf Institute and the Dept. of Computer Science, Paderborn University. Before moving to Paderborn, he was a full professor at the Institute of Computer Science, University of Innsbruck and an assistant professor at the Dept. of Computer Science, University of Erlangen. He received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. Dr. Dressler is associate editor-in-chief for Elsevier Computer Communications as well as an editor for journals such as IEEE Trans. on Mobile Computing, IEEE Trans. on Network Science and Engineering, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has been guest editor of special issues in IEEE Journal on Selected Areas in Communications, IEEE Communications Magazine, Elsevier Ad Hoc Networks, and many others. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM, and many others. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. Dr. Dressler is a fellow of the IEEE as well as a senior member of the ACM, and member of the german computer science society (GI). He has been serving on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include adaptive wireless networking, self-organization techniques, and embedded system design with applications in ad hoc and sensor networks, vehicular networks, industrial wireless networks, and nano-networking.