Cuckoos United: Extending Cuckoo Filters for Message Dissemination in Vehicular Networks

Touhid Hossain Pritom*, Simon Welzel† and Florian Klingler†
*Dept. of Computer Science, Paderborn University, Germany
†Dept. of Computer Science and Automation, TU Ilmenau, Germany
{touhid.pritom, welzel, klingler}@wnc-labs.org

***Abstract*—In Vehicular Ad Hoc Networks (VANETs), neighbor information of vehicles is an important prerequisite for many use cases ranging from intersection collision avoidance up to more complex applications like vehicular platooning. A prime use case of this neighbor information is message forwarding in larger scenarios. The conventional way of transferring this neighbor information in VANETs is beaconing – simple one-hop broadcasts periodically transmitted by each vehicle including position and mobility information. A key requirement for efficient beaconing protocols is to keep the size of beacons small to avoid channel congestion. One possible approach to reduce the beacon size is to transmit the information in a compressed form using a probabilistic data structure, like a Cuckoo Filter. In order to inform nodes at larger scenarios, recent works have shown that extending the beaconing approach with two-hop neighbor information is beneficial. In this paper, we employ such a beaconing scheme and use a two-hop neighbor table approach utilizing Cuckoo Filters for warning message dissemination. A core contribution of our work is the extension of standard Cuckoo Filters to support the union operation which is necessary for proper two-hop neighbor management. We compare our Cuckoo Filter approach against a naïve approach that transmits raw information for beaconing to evaluate the effectiveness of our system. Our results show that our Cuckoo Filter approach performs better than a naïve approach in terms of channel utilization and shows an increased number of covered two-hop neighbors for warning message dissemination.**

I. Introduction and Related Work

In Vehicular Ad Hoc Networks (VANETs), research has shown that maintaining one-hop and two-hop neighbor information is beneficial for performing operations like routing, clustering, and message dissemination [1]. A core principle in that area to exchange neighbor information is beaconing – the transmission of small one-hop broadcasts among vehicles. This has also been incorporated by standardization bodies resulting in standardized message formats, e.g., Cooperative Awareness Messages (CAMs) [2], [3]. For this purpose, a small beacon size is desired as it decreases the probability of frame collisions and generally contributes to a lower channel utilization allowing other applications to use wireless channel resources as well. In the past, many works investigated the frequency at which beacons are transmitted by vehicles to make efficient use of the wireless resources. Even standardization bodies addressed this concept, e.g. ETSI ITS-G5 Decentralized Congestion Control (DCC) [4], which adapts the beaconing interval based on channel conditions. From the perspective of message forwarding, many works investigated different concepts for routing in ad-hoc networks, e.g., classifying traffic based on their information [5] or investigating unicast routing concepts for vehicular networks [6].

In one of our previous works [1], we have shown that having two-hop neighbor information is beneficial for transmitting messages among two-hop neighbors by introducing Bloom Filters for neighbor table management. Bloom Filters for message dissemination are also studied in an earlier work by Bujari [7], where the explicit use and maintenance of two-hop neighbor information was not a focus of the work. Further, the work by Alzamzami and Mahgoub [8] proposes a geographic routing protocol in VANETs which demonstrates improved performance by selecting forwarders based on distance and link quality. Similarly, a routing protocol called Fast Multi-hop Broadcast Algorithm (FMBA) for VANETs is introduced by [9] to address the message dissemination delay by selecting the fastest vehicle in the transmission range to deliver the message to the destination. Another routing protocol called Direction Aware Best Forwarder selection (DABFS) is introduced by [10] for efficient transmission of warning messages. They consider the bidirectional nature of highways and present a direction-based greedy approach to select the best node for transmission. The work by Amador et al. [11] explores the shortcomings of ETSI ITS-G5 Contention-Based Forwarding (CBF) and proposes solutions that reduce the number of transmissions and improve message delivery. A recent work by Kartun-Giles et al. [12] focuses on analyzing the expected number of vehicles with a two-hop connection to a fixed roadside unit (RSU) in VANETs which can be beneficial for predicting densities of vehicular networks to optimize parameterization of networking protocols. One of our recent works explored the use of Cuckoo Filters for managing neighbor information in VANETs [13]. The Cuckoo Filter is a more recently developed probabilistic data structure that claims better space efficiency than traditional Bloom Filters for low target false positive ratios [14]. Based on our previous work, we are revisiting the Cuckoo Filter in the context of VANETs to build a novel message forwarding protocol in this paper. To achieve this, we investigate Cuckoo Filters for their ability to support message forwarding protocols based on two-hop neighbor tables, where we discover important shortcomings of standard Cuckoo Filters to support these type of applications. To address these shortcomings, we design, implement, and evaluate a union operation for Cuckoo Filters that utilizes a technique called *bit reservation* which is a prerequisite in our

IEEE International Conference on Computing, Networking and Communications (ICNC 2024)

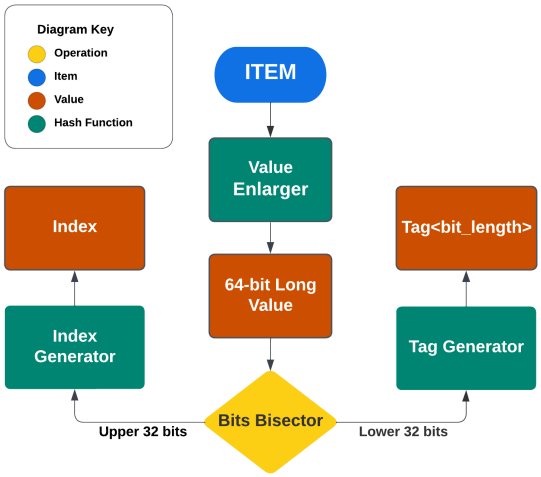


Figure 1. Cuckoo Filter insertion: High-level overview.

scenario for efficient message dissemination based on two-hop neighbor tables.

II. The Cuckoo Filter

The Cuckoo Filter is a set membership probabilistic data structure that is designed to use a limited amount of memory resources. After inserting items, the data structure can be queried for an item and will answer positively if the item has been previously inserted. If the item was not inserted, however, the data structure might still answer positively, resulting in a *false positive* answer. A false negative answer is not possible in standard Cuckoo Filters. The Cuckoo Filter is a condensed version of the Cuckoo Hash Table [15], however, instead of storing key-value pairs, the Cuckoo Filter only stores fingerprints - a bit string generated by hashing the original item followed by truncation [14]. The probability of false positive results increases with the compression of the data, resulting in a trade-off between accuracy and consumption of resources. A standard Cuckoo Filter[1] consists of a simple hash table structure - an array of buckets where each bucket can contain multiple entries, each being capable of storing one fingerprint. The minimum size of the fingerprint required depends on the desired false positive ratio ϵ. For a smaller value of ϵ, the size of the fingerprint needs to be increased. The value of ϵ also depends on the capacity of the filter and the number of items to be inserted into the filter. The standard Cuckoo Filter allows to perform three operations: *insert*, *look-up* and *delete* [14]. Figure 1 presents a high level overview of the insertion process. At the insertion process, the item is hashed to a larger value. Afterward, the index and the tag (fingerprint) are computed using the upper and lower 32 bits of the value, respectively.

III. Message Forwarding with Cuckoo Filters

A core principle for efficient message dissemination in highly mobile networks is to select neighboring nodes as forwarders. This could be achieved by optimizing message dissemination towards a specific geographic direction (*geocasting*) where nodes closest to that direction are selected to forward that message. Another possibility is to select forwarder nodes in a way to cover all or most of a node's two-hop neighbors.

To enable the Cuckoo Filter to be used for message dissemination in VANETs following this principle, it is important to combine the information from multiple Cuckoo Filters. In particular, to derive the minimum set of one-hop neighbors to be fitting forwarders to maximize the number of covered two-hop neighbors, we need to employ set operations on the Cuckoo Filters of each individual node. In our previous work [1] we have achieved this by using the union set-operation together with cardinality estimation on Bloom Filters. Due to the principle of Cuckoo Filters (see their limitations in Section III-B), a union operation like in Bloom Filters (bit-wise *or* operation) is not natively possible. Our main contribution of this paper is to introduce this set operation to obtain a union ($\cup$) of several Cuckoo Filters to enable sophisticated message dissemination protocols using this type of probabilistic data structure.

A. Neighbor Management

In VANETs, vehicles send beacons to exchange neighbor information. In our scenario, each beacon contains the sender vehicle's Medium Access Control (MAC) address (6 Bytes) and a Cuckoo Filter filled with the sender's one-hop neighbor's MAC addresses. In Figure 2 we show an example of such a scenario: The nodes B and C are the one-hop neighbors of node A. The green dotted circle represents the exemplary communication range of A. Whenever a vehicle receives a beacon, an entry containing the sender's MAC address and the Cuckoo Filter containing its one-hop neighbor's MAC addresses is added, or, if it already exists, updated. In order to check whether bidirectional communication is possible, node A looks up its own MAC address in the Cuckoo filter received from node B to check whether node B has previously received a beacon from node A. This way, node A can be sure with a high probability that node B can receive messages from node A. We consider a neighbor entry stale if six consecutive beacons are not received from the same sender. Concerning figure 2, the nodes D and E are the one-hop neighbors of either B or C which makes D and E the two-hop neighbors of A. The red and blue dotted circles again represent the exemplary communication range of B and C, respectively. We now merge the neighbor's one-hop Cuckoo Filters to create a unified filter.

B. Extending Cuckoo Filters To Facilitate Union Operation

The main reason why a standard Cuckoo Filter is unable to perform union operation is because comparison of fingerprints is not possible for different-sized filters. Fingerprints only then are identical when the generated fingerprint and index values are identical. Since the index calculation is dependent on the filter size, it is impossible to know the index value of a fingerprint in a different-sized filter without the knowledge of the original item. To address this limitation, we extend the

[1]https://github.com/efficient/cuckoofilter

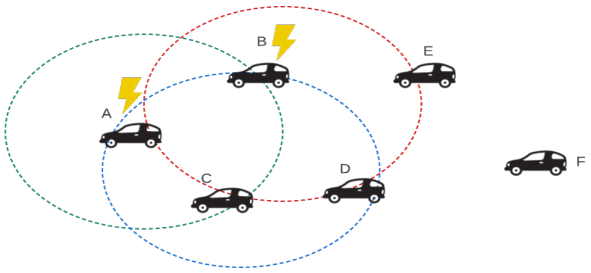


Figure 2. One-hop, two-hop neighbors and re-broadcast set for car A.

Cuckoo Filter with a bit reservation technique which allows us to create a union. First, the fingerprint stored in a filter needs to be made transferable from any filter to a filter called the $Union$ filter. For this, the corresponding index value has to be known. The $Union$ Filter capacity needs to be large enough to accommodate all the fingerprints that we obtain after the union operation. We transfer fingerprints received through one-hop filters into the Union Filter such that there exist no colliding fingerprints, resulting in a two-hop Union Filter. It is important to emphasize that a Union Filter is just a simple Cuckoo Filter that contains the fingerprints as a result of the union operation. Its main characteristic is that it has a fixed pre-defined size. In our extended Cuckoo Filter, when an item is inserted into a given filter along with the fingerprint and bucket index, an additional corresponding bucket index is also computed for the Union Filter. From the knowledge of the pre-defined size of the Union Filter, the index calculation for the Union Filter is trivial. This additional information regarding the Union Filter index is then stored at a particular portion of the fingerprint using the bit reservation technique. Afterwards, this index information of the Union Filter is transferred along with the fingerprint itself. The fingerprint has a predefined storage length (*bits-per-item)*. From these bits, m most significant bits are reserved for the storage of the Union Filter index. The remaining bits remain the same as the original fingerprint. The size of m needs to be large enough to match the maximum capacity of the Union Filter. When the fingerprint is transferred to the $Union$ filter, the bucket index value for the Union Filter of a particular fingerprint is extracted from the reserved m bits of the fingerprint. Figure 3 demonstrates a high-level presentation of the modified Cuckoo Filter.

For simplicity, we predicted the number of vehicles and the thereby required capacity of our Union Filter based on empirical observations. If the capacity is not predicted correctly and more items than there is capacity for are inserted, an overflow occurs.

C. Two-Hop Neighbor Table Construction

In Algorithm 1 we show how the two-hop neighbor set is constructed. Each entry of the neighbor table consists of a node's ID and a Cuckoo Filter containing all one-hop neighbors of that node. Initially, Algorithm 1 deletes a target vehicle's self-ID and the IDs of its direct one-hop neighbors from all the Cuckoo Filters, leaving them with the overlapping

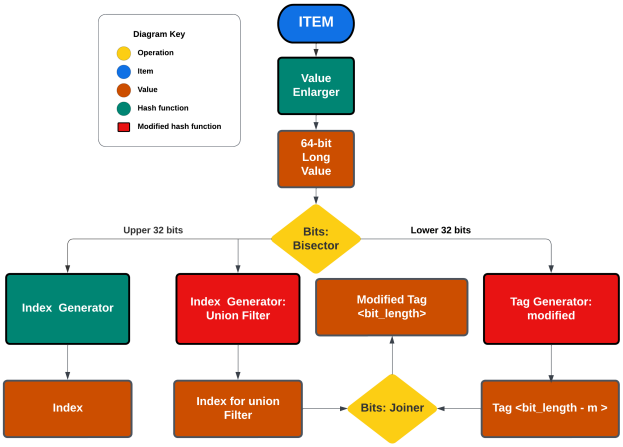


Figure 3. Modified Cuckoo Filter for union operation.

fingerprints of only two-hop neighbors. At this point, the Algorithm 1 transfers all the fingerprints to the Union Filter, skipping duplicates. A limitation of this algorithm is that it ignores the fact that two different items might generate a colliding fingerprint and bucket index. The algorithm does not differentiate whether this is due to the same item or a different item, leading to a more inaccurate cardinality estimation. This limitation, however, is just a matter regarding the parameterization of the Cuckoo Filters. Particularly, using longer fingerprints (increasing the size of Cuckoo Filters) lowers the impact of that limitation.

Algorithm 1 Two-hop neighbor filter construction

Require: A neighbor table where each entry contains the sender's ID and a Cuckoo Filter of its one-hop neighbors

// delete self-ID and one-hop neighbor's IDs
for all $CuckooFilter \in neighborTable$ **do**
  if $CuckooFilter$ contains self-ID or any direct one-hop neighbor IDs **then**
    Delete self-ID and direct one-hop neighbor IDs from the $CuckooFilter$
  end if
end for
//now all the filters in the neighbor table only contain the two-hop neighbor IDs
for all $CuckooFilter \in neighborTable$ **do**
  Extract $UnionFilter$ index from fingerprint
  Transfer all the fingerprints to the $UnionFilter$ that do not have a collision
end for
return $UnionFilter$

IV. Re-Broadcaster Selection

With the two-hop neighbor information available, we propose a greedy approach to select a small subset of one-hop neighbors as re-broadcasters to disseminate a message covering two-hop neighbors. We define $\mathbb{U}$ as the set of visible one-hop neighbors,

c_u as the one-hop Cuckoo Filter of node c where $c \in \mathbb{U}$, as well as f'' as the two-hop Cuckoo Filter, and $\mathbb{R}$ as the selected set of re-broadcasters. Visible neighbors are neighbors with bidirectional communication.

Algorithm 2 Re-broadcaster selection

Require: An empty $\mathbb{R}$, $c_u \forall c \in \mathbb{U}$ and f''.

1: **for** *all* c_u **do**
2: **if** $(c \in \mathbb{U}) \notin \mathbb{R}$ and c_u has maximum number of overlapping fingerprints in f'' **then**
3: **Insert** c into $\mathbb{R}$
4: **Delete** all matching fingerprints of c_u from f''
5: **Repeat** the process until f'' is empty
6: **end if**
7: **end for**
8: **return** $\mathbb{R}$

In Algorithm 2 we select the subset of one-hop neighbors to disseminate the warning message to all two-hop neighbors. While a warning message is disseminated, $\mathbb{R}$ is also sent along so whenever receiving a warning message only the member vehicle of the $\mathbb{R}$ set re-broadcast it. The set $\mathbb{R}$ is sent in raw format (thus, as a list of neighbor IDs) as it usually is of negligible size. In our example in Figure 2, the algorithm will select B as the member of the $\mathbb{R}$ set since it covers the maximum two-hop neighbors of A. Hence only A and B would broadcast the warning message.

V. Evaluation

We evaluate our Cuckoo approach against a naïve approach where beacons use the actual MAC address (6 Bytes) as the vehicle IDs and do not use any data compression mechanisms like Cuckoo Filters. We further evaluate the model against an oracle based on a unit disc model where we approximated the maximum communication distance based on an empirical approach. A core metric to assess the efficiency of our approach is the channel utilization, which expresses the fraction of time the wireless channel is occupied. To evaluate the Cuckoo approach's performance, we assess its effectiveness in disseminating messages by comparing it with the naïve approach.

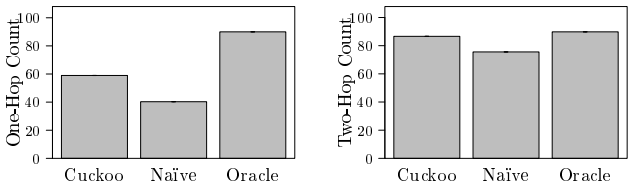
A. Simulation Setup

We use the discrete event simulator OMNeT++ [16], the traffic simulation tool SUMO [17] and the vehicular network simulation framework Veins [18], using IEEE 802.11p as the communication technology. To avoid border effects, we use a warm-up period as well as a region of interest for the result recording.

In Table I we show the main simulation parameters. We specifically selected the timing and location for data collection to ensure the data accurately reflects areas with uniform vehicle traffic density. The warning messages are transmitted three times by the original sender to compare the effectiveness and the efficiency of the message dissemination protocol using the two approaches. We chose a reservation of 5 bits for maintaining a Union Filter with table size 32 since it is required to represent the index in the range $0, 1, \ldots, 31$ (see Table I). In a standard Cuckoo Filter, each bucket can hold four fingerprints hence a Union Filter with a table size of 32 can contain at most 128 fingerprints. A filter with this capacity turns out to be a sufficient fit for our scenario.

Simulation Parameters	Value
Freeway length	5000m
No. of lanes per direction	4
No. of cars per lane	250
Simulation time	50s
Transmission power	2mW
Data rate	6Mbit/s
Center frequency	5.89GHz
Channel bandwidth	10MHz
Path loss coefficient α	2.0
Beaconing frequency	10Hz
Oracle comm. range	170m
Fingerprint Size	12 bits
Reserved bits	5
Union Filter table size	32

Table I
Simulation parameters for high-density.



(a) Number of one-hop neighbors (b) Number of two-hop neighbors

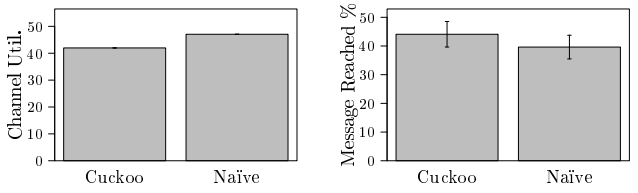
Figure 4. Neighbor count of our Cuckoo and naïve approach, and the oracle.

B. Results

All the result plots of this paper are shown with error bars marking 95% confidence interval for 30 simulation runs for statistical significance.

In Figure 4a we show the comparison of the one-hop neighbor count against the oracle. As can be seen, the number of one-hop neighbors of the Cuckoo approach is closer to the one of the oracle's approach in contrast to the naïve approach. Still, the one-hop neighbor counts of the Cuckoo and the naïve approach differ substantially from the oracle as the oracle does not model channel effects such as interference and frame collisions. Since the Cuckoo approach reduces the beacon size, the beacons require less time for transmission and thus the frame collision chance is reduced.

Figure 4b shows the results for the two-hop neighbor count. It is apparent that the Cuckoo approach better approximates the oracle than the naïve approach. However, it should be noted that the difference between the Cuckoo and the naïve approach is not as drastic for the two-hop count as it is for the one-hop count. This is because when a neighbor is not correctly identified as a one-hop neighbor (e.g., due to interference or



(a) Channel utilization

(b) Percentage of covered two-hop neighbors

Figure 5. Cuckoo Filter message forwarding performance in comparison to the oracle.

frame collisions), there is a high chance that it might be falsely identified as a two-hop neighbor instead. Again, this is mainly caused due to channel congestion and interference in general.

Figure 5a compares the average channel utilization between our Cuckoo and the naïve approach. The figure shows that the Cuckoo approach achieves lower overall channel utilization than the naïve approach which is caused by the smaller beacon sizes when using Cuckoo Filters. This allows our approach to preserve channel resources to be used by other applications.

Figure 5b illustrates the percentage of warning messages received by the Cuckoo and naïve approach. It can be seen that the Cuckoo approach has a tendency for a higher mean received percentage than the naïve approach. This means, that using Cuckoo Filters could be a promising approach for designing efficient message dissemination protocols for two-hop neighbor management. Due to the space efficiency of Cuckoo Filters, even more sophisticated approaches, e.g., extending our concept to n-hop neighbor dissemination could be possible.

VI. Conclusion

In this paper, we investigated probabilistic neighbor management using the Cuckoo Filter. To employ efficient message dissemination among two-hop neighbors utilizing neighbor tables in highly dynamic scenarios, we extended the Cuckoo Filter to allow for a union set operation. To assess the performance of our approach, we benchmarked the message dissemination approach against a naïve approach and a ground truth by comparing performance metrics to an oracle. The evaluation based on in-depth simulation studies shows promising results which could make Cuckoo Filters an alternative to other probabilistic data structures like Bloom Filters for certain applications.

References

[1] F. Klingler, R. Cohen, C. Sommer, and F. Dressler, "Bloom Hopping: Bloom filter based 2-Hop Neighbor Management in VANETs," *IEEE Transactions on Mobile Computing (TMC)*, vol. 18, no. 3, pp. 534–545, Mar. 2019.

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

[3] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service," ETSI, TS 102 637-2 V1.1.1, Apr. 2010.

[4] 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.

[5] 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.

[6] J. Bernsen and D. Manivannan, "Unicast routing protocols for vehicular ad hoc networks: A critical comparison and classification," *Elsevier Pervasive and Mobile Computing*, vol. 5, no. 1, pp. 1–18, Feb. 2009.

[7] A. Bujari, "A Network Coverage Algorithm for Message Broadcast in Vehicular Networks," *ACM/Springer Mobile Networks and Applications (MONET)*, Apr. 2016.

[8] O. Alzamzami and I. Mahgoub, "Link utility aware geographic routing for urban VANETs using two-hop neighbor information," *Ad Hoc Networks*, vol. 106, p. 102 213, 2020.

[9] A. Bujari, M. Conti, C. De Francesco, and C. E. Palazzi, "Fast multi-hop broadcast of alert messages in VANETs: An analytical model," *Ad Hoc Networks*, vol. 82, pp. 126–133, 2019.

[10] S. Haider, G. Abbas, Z. H. Abbas, and T. Baker, "DABFS: A robust routing protocol for warning messages dissemination in VANETs," *Computer Communications*, vol. 147, pp. 21–34, 2019.

[11] O. Amador, M. Urueña, M. Calderon, and I. Soto, "Evaluation and improvement of ETSI ITS Contention-Based Forwarding (CBF) of warning messages in highway scenarios," *Vehicular Communications*, vol. 34, p. 100 454, 2022.

[12] A. P. Kartun-Giles, K. Koufos, X. Lu, and D. Niyato, "Two-Hop Connectivity to the Roadside in a VANET Under the Random Connection Model," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 5508–5512, 2023.

[13] S. Welzel, F. Dressler, and F. Klingler, "Cuckoo Filters for Two-Hop Neighbor Management in Vehicular Networks," in *14th IEEE Vehicular Networking Conference (VNC 2023), Poster Session*, Istanbul, Turkey: IEEE, Apr. 2023, pp. 155–156.

[14] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo Filter: Practically Better Than Bloom," in *10th ACM International on Conference on Emerging Networking Experiments and Technologies (CoNEXT 2014)*, Sydney, Australia: ACM, Dec. 2014.

[15] R. Pagh and F. F. Rodler, "Cuckoo Hashing," *J. Algorithms*, vol. 51, no. 2, pp. 122–144, May 2004. [Online]. Available: https://doi.org/10.1016/j.jalgor.2003.12.002.

[16] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08, Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[17] D. Krajzewicz, G. Hertkorn, C. Feld, and P. Wagner, "SUMO (Simulation of Urban MObility); An open-source traffic simulation," Jan. 2002, pp. 183–187.

[18] C. Sommer, D. Eckhoff, A. Brummer, et al., "Veins: The Open Source Vehicular Network Simulation Framework," in *Recent Advances in Network Simulation*. 2019.