Poster: Cuckoo Filters for Two-Hop Neighbor Management in Vehicular Networks

Simon Welzel*, Falko Dressler† and Florian Klingler*
*Dept. of Computer Science, TU Ilmenau, Germany
†School of Electrical Engineering and Computer Science, TU Berlin, Germany
{welzel, klingler}@wnc-labs.org, dressler@ccs-labs.org

***Abstract*—Neighbor management in vehicular networks comes with the risk of unnecessarily overloading the wireless channel, particularly when two-hop neighbor information is required. A possible solution to this challenge is the use of probabilistic data structures. In our previous work, we explored the benefits of using Bloom filters for maintaining this neighbor information showing promising results. In this paper, we now evaluate the usage of a additional probabilistic data structure, the Cuckoo Filter, which is advertised as a superior alternative to Bloom filter. We assess the performance of the Cuckoo approach in a vehicular networking scenario and find that it does not meet these expectations. In fact, it may lead to worse performance in specific configurations.**

I. Introduction and Related Work

The exchange and management of neighbor information is important for many vehicular networking applications such as cooperative awareness. This is commonly done in the form of neighbor tables, which are filled and maintained by utilizing beaconing [1]. However, not only the direct neighbors (one-hop neighbors) are of interest, but also the neighbors of my neighbors, the so-called *two-hop neighbors* [2]. A naïve approach would be to disseminate a list of my one-hop neighbors with every transmitted beacon. However, this may lead to a large amount of data to be exchanged periodically.

A potential solution to this problem is to use probabilistic data structures, which allow for more space-efficient data storage [3]. However, they introduce some challenges and risks: As they are non-iterable, inserted data can only be queried, which limits their usage for certain use cases. Additionally, they often come with the risk of false positives, which is typically happening due to hash collisions. Popular probabilistic data structures for membership tests are the *Bloom Filter* [4] and the *Cuckoo Filter* [5]. While both allow for the insertion and query of elements, the Cuckoo Filter also allows for the deletion of elements. The Cuckoo Filter is advertised as offering better space efficiency in certain configurations [5].

Klingler et al. [3] showed that using Bloom Filters allow for better performance, improving the two-hop neighbor set in comparison to a naïve approach transmitting raw neighbor information. In this work, we study the question to which extent the potentially better performing Cuckoo Filter plays out its higher performance in a typical vehicular networking scenario in comparison to the traditional Bloom Filter. Performance metrics considered are the channel utilization, packet delivery ratio and the size of the two-hop neighbor set as well as the quality of the two-hop neighbor set.

II. Cuckoo Filter Approach

The *Cuckoo Filter* is a set membership probabilistic data structure [5]. It allows for insertion, query, and deletion of items. The items are hashed, and a *fingerprint* of the hash is saved in a so-called bucket. Buckets are stored in two lists, and each fingerprint has two locations where it can be stored. If the primary location, determined by the element's hash, is already fully occupied, a random element is kicked out of the bucket and inserted into its alternative location. To query an element, the filter looks for the fingerprint in the corresponding bucket locations, which it determines by re-hashing the item. An element can be deleted by removing its fingerprint, without the risk of false negatives for future queries. From a performance perspective, the Cuckoo Filter is supposed to require fewer bits per element to maintain the same false positive rate [5].

On the other hand, a *Bloom Filter* uses a bit array to *store* items. An inserted element is hashed by a set of hash functions. The resulting hashes determine the indices of the bit array that are set to one when inserting the element. To query an element, the filter hashes it again and looks up the resulting indices in the bit array; only if all of them return a one, the item is contained in the filter [4]. Item deletion is not supported in the initial version of a Bloom Filter.

We want to find out whether Cuckoo Filters can be a viable or even preferred alternative over Bloom Filters for two-hop neighbor management in a typical vehicular networking context. Starting with ideas from Klingler et al. [3], a node broadcasts its identifier, position and a representation of its current (one-hop) neighbor table via beaconing – periodic one-hop broadcasts transmitted using wireless communications. Whenever beacons from nodes in reception range are received, the neighbor table is populated: For every received beacon, a neighbor table entry is created, containing identifier, position and the representation of that node's one-hop neighbors. Neighbor table entries are removed if they are older than two seconds. Only neighbors for which a symmetric communication link exists (symmetric neighbors) are considered.

III. Evaluation

To assess the performance of our Cuckoo Filter approach, we compare it against the Bloom Filter approach and a naïve

approach. The naïve approach sends its one-hop neighbors as a plain list of neighbor entries. For the naïve approach, each neighbor entry is considered to be the size of a MAC address (6 Byte). The total length of the beacons sent by the naïve approach is not fixed, but depends on the exact number of identifiers stored. In contrast, the size of the beacons sent by the Bloom and Cuckoo approaches is fixed, independent from the exact amount of identifiers stored. The size of the Bloom Filter is determined by capacity (element count) and desired false positive ratio, whereas the size of the Cuckoo Filter is determined by capacity and fingerprint length.

A. Simulation Setup

We conduct simulations using Veins, which relies on OMNeT++ and SUMO. A freeway scenario consisting of four lanes in each direction of 5km length with bidirectional traffic is used. The average vehicle density is 810 vehicles per kilometer. As communication technology, IEEE 802.11p is used. We configured an MCS of QPSK-$1/2$, which results in a PHY data rate of 6 Mbit/s. As transmit power we chose 1 mW in order to provoke challenging communication scenarios to better evaluate the quality of our probabilistic neighbor approach. Further, we use the free space path loss model with a path loss coefficient $\alpha = 2.0$ and for communication a center frequency of 5.89 GHz at 10 MHz channel bandwidth. Beacons are transmitted at a frequency of 2 Hz. As the Cuckoo Filter does not allow to set a desired false positive ratio, the filters are parameterized in a way that they result in a very similar and comparable beacon size. We then use the false positive ratio to measure the information quality.

B. Results

The average beacon lengths for Naïve, Bloom and Cuckoo approach are 6980, 2347, and 2364 Bits, respectively. While Bloom and Cuckoo approach achieved an average channel utilization of around 37%, the naïve approach resulted in a channel utilization of 74%, as shown in Figure 1. This leads to a substantially lower packet delivery ratio (results not shown due to space constraints) for the Naïve approach (76%) when compared to Bloom and Cuckoo approaches (both 93%).

Figure 2 shows the achieved average false positive ratio (FPR) across all filters. The false positive ratio is determined by querying a filter with all vehicle identifiers in the simulation, counting the number of false positive queries and dividing that number by the number of vehicles in the simulation. While the Bloom approach achieved a false positive ratio of only around 1.7%, the Cuckoo approach achieved 17.3%.

In an additional experiment (results not shown due to space constraints) we evaluated a configuration of the Bloom and Cuckoo approaches that uses severely undersized filters, resulting in occupation ratios of up to 100%. We found that the Bloom Filter showed a very high false positive ratio. This is due to the fact that almost all entries in the bit table will be set to binary one if the filter is fully occupied, not allowing for any queries to be negative anymore. The Cuckoo Filter did not show a comparably high false positive ratio. However, as the Cuckoo

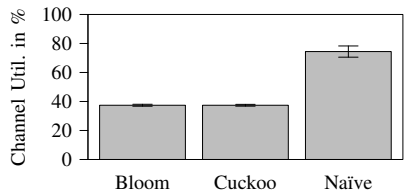

Figure 1. Average channel utilization for each protocol.

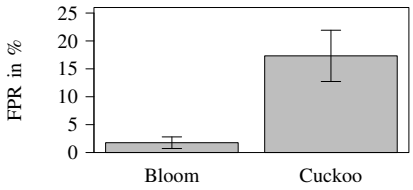

Figure 2. Average false positive ratio (FPR) for Bloom and Cuckoo approach.

Filter *kicks out* a fingerprint from a fully occupied bucket when a new fingerprint is added to that bucket, fingerprints got removed from the filter, resulting in negative queries even though queried elements had been inserted in the first place. Both approaches, Bloom and Cuckoo, proved to be unusable if they are undersized.

IV. Conclusion and Future Work

In this paper, we examined the usage of Cuckoo Filters for two-hop neighbor management in the vehicular context. Our results show that probabilistic data structures have the potential to decrease the beacon size, reducing the channel utilization and ultimately increasing the two-hop neighbor set size. Summarizing our findings, the Cuckoo Filter approach does not show an improvement in any of the performance metrics in our configuration when compared to the Bloom Filter approach in the context of vehicular two-hop neighbor management. A possible explanation for this behaviour is that the Cuckoo Filter could only show its superiority in larger filter configurations.

To find out if and where there is a crossover point, we plan to conduct a parameter study. Also, as traffic densities vary temporally and position-based, a configuration algorithm that calculates optimal parameters based on environmental factors such as traffic density or location may further improve the performance.

References

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

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

[3] 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.

[4] 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.

[5] 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.