The Impact of Head of Line Blocking in Highly Dynamic WLANs

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

***Abstract*—Wireless LAN (WLAN) based networks can rely on the concept of Automatic Repeat Request (ARQ) to overcome temporary channel outages, such as those due to interference. However, the used acknowledgement scheme for unicast transmissions blocks other transmissions until a message has either been successfully transmitted or until the retry limit has been reached. We investigate this head of line blocking problem, its impact, and first steps towards a final solution. As a research methodology, we build upon an analytical model, computer simulations, and also experimentation in a lab environment. We consider the effects for a single node in a small and static network as well as studying the application layer performance in a highly dynamic network – we investigate Vehicular Ad Hoc Networks (VANETs) as a prime example – spanning several kilometers. Aside from considering the case of malicious users conducting easy active and passive attacks, we show that in highly dynamic networks this blocking behavior can (massively) be observed as local topologies change. Here, the discussed effects can lead to massive message loss and increased latencies in the order of seconds.**

***Index Terms*—IEEE 802.11, Vehicular Networking, EDCA, Unicast**

I. Introduction and Motivation

We investigate the problem of head of line blocking when using unicast transmissions in highly dynamic Wireless LAN (WLAN) environments. As a prime example, we focus on Vehicular Ad Hoc Networks (VANETs), even though all our investigations also apply to other application domains with highly dynamic topologies.

Wireless communication based on IEEE 802.11 WLAN has become the standard for establishing vehicular networks [1]. Building on IEEE 802.11, the U.S. DSRC/WAVE protocol stack [2] and the European ETSI ITS-G5 protocol suite [3] have been defined, both inheriting the physical and the MAC layer of IEEE 802.11.

Traditionally, the IEEE 802.11 WLAN MAC layer is designed to operate in the context of a Basic Service Set (BSS), a set of mobile nodes that have synchronized to use a common set of parameters [4]. However, joining such a network involves a message exchange procedure that has been found too time consuming for vehicular networks. Hence, the WLAN standard has been amended in IEEE 802.11p to allow operation in what was named "outside the context of a BSS" (OCB) mode [5]. This mode of IEEE 802.11p avoids the need for authentication to other nodes as well as the need to scan for, join, or associate to a BSS, either in infrastructure mode or in ad hoc mode. These modifications make IEEE 802.11 a reliable basis for vehicular communication: Commodity WLAN network interface cards [6] can be used with little modifications to the driver. The standard also supports low latency data transmission, which is crucial for most types of applications in vehicular networks. It is therefore the standard extension underlying both the aforementioned U.S. and European VANET standards.

One of the major tasks of the MAC is error control, i.e., to ensure reliable frame transmission by using retry mechanisms – at least for unicast transmissions. IEEE 802.11 (and thus, by extension, IEEE DSRC/WAVE and ETSI ITS-G5) can rely on a simple Automatic Repeat Request (ARQ) error control scheme: by default, any individually addressed frame will be kept as the head of the transmit queue until an acknowledgment (ACK) frame is received. If no ACK frame is received for a pre-defined duration, the frame is automatically retransmitted until an ACK frame is received, or until a retransmission limit is reached. This behavior leads to the so-called head of line (HOL) blocking problem: One frame delays the transmission of other frames. Moreover, the waiting time between retries of frames increases according to an exponential backoff algorithm to lower the channel load and, thus, stabilize the network, which increases the probability for a successful frame transmission. In summary, any transmit queue that is waiting for an ACK frame for a unicast frame is stalled – this queue will neither transmit broadcast frames, nor any other individually addressed frame to another node.

The head of line blocking effect has been identified in the early days of WLANs [7] and proposals have been made to create an alternative MAC layer that monitors the individual wireless stations of a BSS and maintains separate transmit queues. Such a MAC layer can then defer re-transmissions to *bad* stations until the estimated end of a (presumed) burst error. However, the key assumption of such proposals has always been that lost frames are due to collisions or burst errors in the channel. For the envisioned target scenario at that time, this was a very reasonable assumption as nodes were meant to maintain a relatively static topology. Still, even in static networks the effect is rather easy to trigger. All that is needed is to simply provoke a node to send a unicast data frame to a node that is not there. Alternatively, a node performing selective jamming of some unicast acknowledgements can trigger the same effect with potentially even bigger impact (see Section III-F). Finally, a completely passive denial of service attack can be realized by the destination of a unicast transmission: by simply not acknowledging another node's unicast transmission. Yet, the impact of the effect of head of

 The authors are with the Heinz Nixdorf Institute and Dept. of Computer Science, Paderborn University, Germany ({klingler,dressler,sommer}@ccs-labs.org).

line blocking has been considered to be no worse than reducing the attainable throughput over the wireless channel. This has led to the effect being widely ignored in standardization.

For safety applications and, more so, in highly dynamic environments such as VANETs, however, the effect of head of line blocking can be disastrous. When triggered, unnecessary delays of Cooperative Awareness Messages (CAMs) or Basic Safety Messages (BSMs) are possible, which negatively influences safety applications. Moreover, as we will show, the effect is frequently triggered in networks of high topology dynamics when nodes attempt unicast transmissions to other nodes – other cars, Internet Access Points (APs), or dedicated Roadside Units (RSUs) – that they wrongly consider to still be neighbors. Such frames will never be acknowledged, remaining at the head of the transmit queue of the sender until it expires.

Head of line blocking not only delays the unicast transmission in question but all packets waiting in the same queue if a node adheres to the Enhanced Distributed Channel Access (EDCA) system of WLAN [4]. Which queue is blocked depends on the Access Category assigned to the frame. Yet, with only four categories defined in WLAN, a large number of different applications will likely share a single queue. As a consequence, a single blocked queue impacts a multitude of related applications (for example, all safety applications). Without further information, a sender can also not readily determine whether the receiver suffers from interference that, indeed, keeps it from replying with ACKs, or whether ACKs are selectively suppressed, making passive attacks hard to detect reliably.

Even worse, the impact of head of line blocking is also long-lasting. In highly dynamic network topologies, the destination node is often simply no longer a neighbor and remains permanently unreachable, e.g., due to radio signal shadowing [8]. This causes the transmit queue to block until the maximum number of retries have been exceeded, wasting channel capacity, keeping other nodes from transmitting, and (even worse) keeping the same node from transmitting potentially safety-critical information.

Building upon our earlier work [9], we point out a way towards a general solution based on an investigation of the effects of head of line blocking in more breadth and depth: We expand our focus from specialized hardware and settings of VANET Field Operational Tests (FOTs) to that of regular commercial off-the-shelf WLAN adapters. We further expand the depth of our studies, investigating both more general metrics to study the true impact on the application layer as well as more specific metrics to investigate the reported effects. We also take great care to cross-validate every step in our investigation between analytics, computer simulations, and hardware experiments. We furthermore report on a completely new experimental and analytical study on the behavior of commodity WLAN cards under active attacks. In order to give some insights into possible algorithmic solutions to the problem, we also investigate a rather simple protocol that helps overcoming head of line blocking issues. We see this protocol as a basis on which future work can build upon to fully eliminate its negative effects in highly dynamic WLAN environments.

Our main contributions can be summarized as follows:

- We first investigate the impact of head of line blocking in a small and static network – analytically, in computer simulations, and in hardware experiments (Section III);
- we continue our study by evaluating the backoff behavior of commodity WLAN cards when subjected to selective jamming of acknowledgements, validating experimental results against analytical predictions (Section III-F);
- in a final set of experiments, we assess the macroscopic view in presence of head of line blocking for a full VANET application scenario of a highly dynamic network (Section IV).

II. Related Work

VANETs are a prime example of highly dynamic wireless networks and an emerging technology on the verge of wide scale real world deployment [1], making them a worthwhile sample use case of this broader category. Typical applications of VANETs range from safety, to traffic efficiency, and to comfort applications [10] each having different requirements for the underlying communication stack, e.g., delay, reliability, and goodput. To support this variety of application domains, several communication patterns have been found to be beneficial in VANETs [1], [11].

One of the most prominent communication patterns for safety messages is beaconing – the process of sending periodic 1-hop broadcasts of small status reports of vehicles [12]. This kind of information exchange (including current position, speed, and driving direction) is standardized in Europe in the ETSI ITS-G5 protocol stack [3] and in the U.S. in the IEEE DSRC/WAVE protocol suite [2]. These periodic CAM or BSM broadcasts, respectively, do not require any kind of acknowledgements, so they are not impacted by the head of line blocking problem discussed in this work.

Conversely, information exchange for comfort or efficiency applications commonly involves vehicles communicating with either a dedicated vehicle, an RSU, or a gateway. This connection-like oriented communication pattern often involves unicast routing over multiple hops [13], [11]. Many of those routing protocols have been originally developed for Mobile Ad Hoc Networks (MANETs); and part of them can be applied to VANETs as well. Also, aside from comfort and efficiency applications, the unicast communication principle is also used in the literature for specific VANET applications like Geocasting and platooning [14], [15]. Indeed, several detailed surveys on unicast routing protocols for VANETs can be found in the literature: Li and Wang [16] give an overview about different routing strategies and name popular routing protocols according to their communication type. Bernsen and Manivannan [17] classify and characterize available unicast routing protocols for VANETs and provide a qualitative comparison among those. Sichitiu and Kihl [18] focus on the taxonomy of VANET applications and study the requirements from an underlying network. This underlines the prevalence of the unicast communication pattern even protocol designs targeting highly dynamic networks.

One step further, a number of designs explicitly target or, indeed, rely on unicast pattern and its acknowledgement

mechanism – both for supporting multi-hop routing and for single-hop transmissions. For example, approaches have been made to mitigate packet duplication of unicast routing protocols introduced by the two (hop-by-hop and end-to-end, respectively) recovery mechanisms of the MAC layer and the routing protocol [19]. Similarly, Han et al. [20] build on the MAC retry mechanism based on ACKs and design an improved retry mechanism based on NACKs (negative acknowledgements). It allows an application layer message to quickly and repeatedly be retransmitted until it is eventually (successfully) received by its indented destination. As a third example, Xie et al. [21] present a two-dimensional Markov chain model based on the IEEE 802.11 model of Bianchi [22] to investigate the delay of channel access using a stochastic road traffic model. Again, a central assumption is that unicast is needed for reliable VANET protocol design and that this implies retransmissions performed in the MAC layer.

However, when IEEE 802.11 was designed many years ago, the exponential backoff strategy for unsuccessful unicast communication triggered by lost acknowledgments was designed to solve channel congestion problems. The node topology was assumed to be relatively static, thus the most common causes for lost acknowledgements were assumed to be hidden terminal situations and, more importantly, an overloaded channel.

In our work we show that for VANETs this assumption does not hold anymore. Indeed, reliable unicast communication drastically lowers the performance of VANETs when unicast packets are sent to nodes that are out of range. We also show that this is a common occurrence in VANETs.

III. Small and Static Networks

We first investigate the impact of the discussed head of line blocking effect in a small and static network that mimics topology dynamics; *(A)* analytically, *(B)* in experiments with off-the-shelf and FOT-ready WLAN cards, and finally *(C)* in computer simulations.

A. Analytical Evaluation

In the following, without loss of generality, we focus on an OFDM PHY with 10 MHz bandwidth as specified in the current version of the IEEE 802.11 standard [4]. Following both the values (and the formalism) introduced in the standard, we adopt the following PHY timing parameters: $T_{\text{preamble}} = 32\,\mu\text{s}$, $T_{\text{signal}} = 8\,\mu\text{s}$, and $T_{\text{sym}} = 8\,\mu\text{s}$. MAC parameters are also set according to the standard, to $t_{\text{SIFS}} = 32\,\mu\text{s}$, $t_{\text{slot}} = 13\,\mu\text{s}$, and $t_{\text{rx\_delay}} = 49\,\mu\text{s}$. We further assume that the RTS threshold is set above the frame size, so that no RTS/CTS procedure is invoked, as well as (otherwise) empty EDCA queues and an idle channel.

The time to transmit data is calculated according to the PLME-TXTIME.confirm primitive described in the standard [4, Section 18.4.3]. When transmitting headers and payload of size l at 6 Mbit/s (thus $N_{\text{DBPS}} = 48$ bit), this time can be calculated as

$$t_{\text{tx}}(l) = T_{\text{preamble}} + T_{\text{signal}} + \left\lceil \frac{16 + l + 6}{N_{\text{DBPS}}} \right\rceil T_{\text{sym}}. \quad (1)$$

For a broadcast packet with a payload of $l = 2400$ bit, we calculate $t_{\text{tx}}(2400\,\text{bit}) = 448\,\mu\text{s}$. Similarly, for $l = 112$ bit, the size of an ACK frame, we obtain $t_{\text{tx}}(112\,\text{bit}) = 64\,\mu\text{s}$.

The frame exchange sequence for a reliable unicast transmission of a frame is: *send data, wait for a SIFS, send ACK*. Thus, the lower bound for the duration of a unicast transmission (which might be blocking a queue) is achieved if the channel has been idle for some time and the transmission is immediately acknowledged; it can be calculated as

$$t_{\text{best}} = t_{\text{tx}}(2400\,\text{bit}) + t_{\text{SIFS}} + t_{\text{tx}}(112\,\text{bit}) = 544\,\mu\text{s}. \quad (2)$$

If we now focus on the case of a node trying to send such a unicast frame to a node that does not exist, we have to factor in the time spent for retries, each waiting for an ACK that does not arrive within $t_{\text{ACK\_wait}}$, as well as the time spent in backoff. According to the standard [4, Section 9.3.2.8], $t_{\text{ACK\_wait}}$ can be calculated as

$$t_{\text{ACK\_wait}} = t_{\text{SIFS}} + t_{\text{slot}} + t_{\text{rx\_delay}} = 94\,\mu\text{s}. \quad (3)$$

Backoff times are set to k times t_{slot}, the number k being randomly drawn from a contention window (CW), which is initially set to CW_{min}; in the worst case, the maximum number is drawn each time. After each unsuccessful transmission (i.e., no ACK was received) the contention window size CW is updated to $2(CW + 1) - 1$, up to CW_{max}. Only when the packet is finally deleted from the transmit queue, CW is reset to CW_{min}. Thus, the upper bound of time spent for backoff alone during n attempts to transmit can be calculated as

$$t_{\text{CW}}(n) = t_{\text{slot}} \times \sum_{i=0}^{n-1} \min\Big\{2^i\big(CW_{\text{min}} + 1\big) - 1, CW_{\text{max}}\Big\}. \quad (4)$$

Similarly, as backoff values are drawn uniformly from the CW for every attempt, the mean value of the distribution of waiting times can be calculated by halving t_{CW}.

Before decrementing the backoff counter, the channel needs to be idle for at least the duration of an Arbitration Interframe Space (AIFS). If we assume the sender to be operating in OCB mode and using Access Category AC_BE, it will wait for AIFSN $= 6$ slots [4, table 8-106], resulting in

$$t_{\text{AIFS}} = t_{\text{SIFS}} + 6 \times t_{\text{slot}} = 110\,\mu\text{s}. \quad (5)$$

Default values suggested by the standard [4, pages 1623 and 2425] are: $CW_{\text{min}} = 15$, $CW_{\text{max}} = 1023$, and *dot11ShortRetryLimit* $= 7$ retransmission attempts. This configuration has been found to be beneficial to protocol operation in VANETs [23]. However, we discovered in our measurements that the used wireless cards perform 9 retransmission attempts if no ACK is received (independent of their configuration). Thus, we also use this value in the following calculations.

Taken together, the upper bound for the duration of a unicast transmission is achieved if the channel only just turned idle and the transmission remains unacknowledged for 10 attempts, each time choosing the maximum backoff time from the CW; it can be calculated as

$$\begin{aligned} t_{\text{worst}} &= 10\Big(t_{\text{AIFS}} + t_{\text{tx}}(2400\,\text{bit}) + t_{\text{ACK\_wait}}\Big) + t_{\text{CW}}(10) \\ &= 72\,742\,\mu\text{s}. \end{aligned} \quad (6)$$

Thus, each unicast sent to a node that no longer exists (whether sent in error or provoked maliciously) blocks any transmissions from the same queue for an average of around 40 ms and up to approx. 73 ms. Note that this effect is cumulative if multiple such frames are queued.

B. Experimental Study

We confirmed both the presence and the analytically derived duration of the blocking effect in real world experiments, as we will detail in the following.

As our device under test, we investigated an embedded system running Linux 3.18 based on [24] and outfitted with an Compex WLM200N5-23ESD miniPCI card using an Atheros AR9220 chipset with the ath9k driver. Due to its combination of chipset and driver, it can serve as a good specimen of typical WLAN cards. Moreover, it has already been used as a reliable basis for building an ITS prototype [6] for FOTs, as it allows tuning the radio to ITS frequencies in the 5.9 GHz band as well as using the IEEE 802.11p bandwidth of 10 MHz. In previous work [9] we confirm that the effect of head of line blocking is equally present in specialized equipment designed for ITS FOTs worldwide like the Cohda Wireless MK5.

We modified the Linux kernel to report how long each frame was delayed in a transmit queue (from entering into the queue to it being deleted). We then configured an independent virtual interface set to monitoring mode to record these statistics. Similarly, the receiver used information made available from a modified kernel to derive the number of backoff slots chosen for a frame. Additionally, an independent monitoring node with a wireless card tracked the channel load $\rho = t_{\text{busy}}/(t_{\text{busy}} + t_{\text{idle}})$ as the fraction of the time the wireless channel was sensed busy.

To investigate the interplay of communication modes, we ran a process on the device which periodically creates permutations of three independent types of messages to model three different applications sending traffic over the wireless channel. For simplicity we call these App 1, App 2, and App 3. The process enqueues three messages simultaneously and then waits for the transmission to conclude to saturate an otherwise clear channel. The ordering of messages is random. A sample ordering of messages looks like $(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)$.

We configured the physical layer according to the specifications used in the analytical study: a 10 MHz wide channel at 5.890 GHz, not using RTS/CTS, and transmitting at a rate of 6 Mbit/s. The MAC layer was configured to send packets using a TXOP value of 0 (one post-transmit backoff after every frame) and Access Category AC_BE (that is, an initial contention window size of 15 slots, a maximum contention window size of 1023 slots, and an AIFSN value of 6 slots).

Our *baseline* scenario consists of two experiments: In the first experiment (Exp 1), all three applications sent broadcast packets. In the second experiment (Exp 2), App 1 was changed to send unicast packets. As an alternative scenario (denoted as *ghost*) we changed App 3 to transmit data to a device that was no longer there and repeated the previous two experiments (as Exp 3 and Exp 4). For this, we manually inserted entries into the ARP tables of nodes, thus capturing the scenario of a neighbor having existed previously before moving out of reception range. This represents the case of a vehicle trying to send data to a former neighbor (which has been shown to happen frequently in a VANET [25]). We then build on this scenario to investigate the results of two more experiments using a different retry strategy (Exp 5 and Exp 6). An overview of all experiments and their configurations is shown in Table I.

Table I
OVERVIEW OF EXPERIMENTS

scenario	experiment	App 1	App 2	App 3
baseline	**Exp 1**	broadcast	broadcast	broadcast
	Exp 2	unicast	broadcast	broadcast
ghost	**Exp 3**	broadcast	broadcast	*lost unicast*
	Exp 4	unicast	broadcast	*lost unicast*
app. retry	**Exp 5**	broadcast	broadcast	*lost unicast*
	Exp 6	unicast	broadcast	*lost unicast*

C. Computer Simulation

We validate our results in the small and static network by cross-checking the analytical and experimental results against a computer simulation of the same scenario. We set up simulations in the *Veins* Open Source[1] vehicular network simulation framework [26] version 4.4. Along with many other models, Veins provides realistic channel access models based on IEEE 802.11p and IEEE 1609.4. We extended the IEEE 802.11p MAC layer in order to support unicast transmission according to the IEEE 802.11 HCF. As in the experiments, the MAC layer was configured to send packets using a TXOP value of 0 (one post-transmit backoff after every frame) and Access Category AC_BE (that is, an initial contention window size of 15 slots, a maximum contention window size of 1023 slots, and an AIFS value of 6 slots to match the settings used in the measurements).

In the simulation, all three types of messages used a payload length of 2400 bit to saturate an otherwise clear channel. We perform the same set of experiments as in the experimental study outlined in Table I.

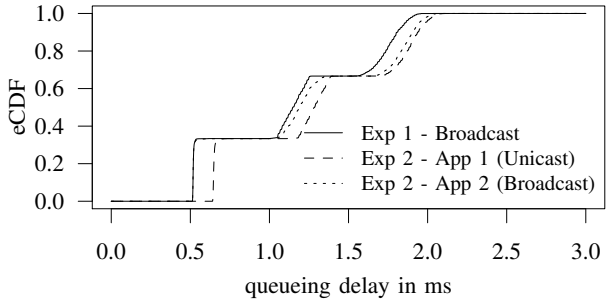
D. Results

Figure 1 illustrates the results of analytics, measurements, and computer simulations as empirical Cumulative Distribution Functions (eCDFs). Aside from little delays introduced by the software, queueing delays observed in the measurements agree very well with those in computer simulation, as well as with those predicted by analytics: The delays of frames fall into three clear categories according to how many (zero, one, or two) frames were queued before.

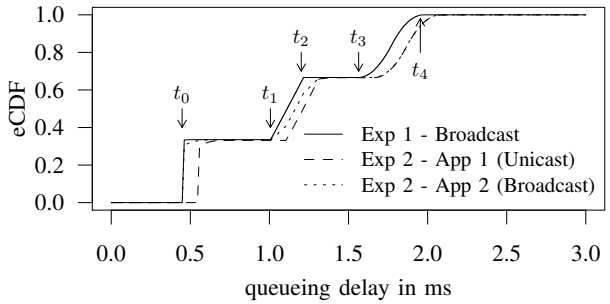
When sending only broadcast frames (*baseline* scenario, Exp 1) all data took either

$$t_0 = t_{\text{tx}}(2400\,\text{bit}) = 448\,\mu\text{s} \tag{7}$$

[1] http://veins.car2x.org

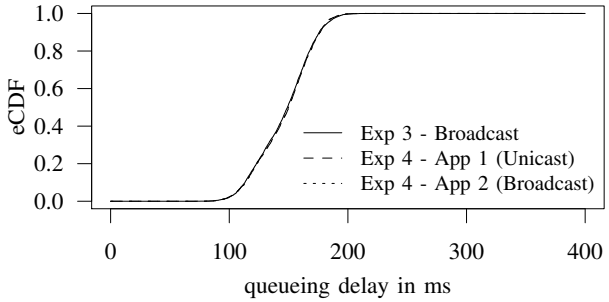


(a) measurements (using Atheros AR9220 based WLAN cards)

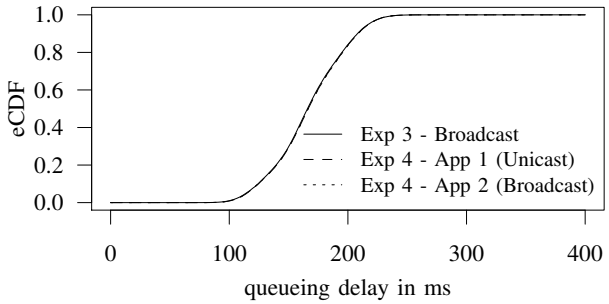


(b) simulation (using the *Veins* simulator)

Figure 1. TX queuing delay in the *baseline* scenario.



(a) measurements (using Atheros AR9220 based WLAN cards)



(b) simulation (using the *Veins* simulator)

Figure 2. TX queuing delay in the *ghost* scenario.

to send (if no frame was already queued) or it had to wait for data of one or two of the other applications to be sent.

When waiting for data of one application, this additional delay is characterized by a uniformly distributed random value of $\mathcal{U}(0, CW_{\text{min}})$ slots spent in post-transmit backoff, resulting in a uniformly distributed waiting time between

$$t_1 = t_0 + t_{\text{AIFS}} + 0 + t_{\text{tx}}(2400\,\text{bit}) = 1006\,\mu\text{s}, \quad (8)$$

$$t_2 = t_0 + t_{\text{AIFS}} + CW_{\text{min}} t_{\text{slot}} + t_{\text{tx}}(2400\,\text{bit}) = 1201\,\mu\text{s}. \quad (9)$$

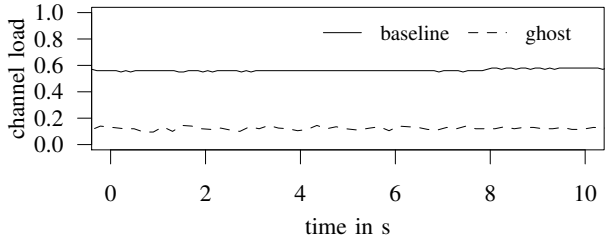
When waiting for data of two applications, the delay is characterized by the uniform sum distribution of both backoffs, with bounds of

$$t_3 = t_1 + t_{\text{AIFS}} + 0 + t_{\text{tx}}(2400\,\text{bit}) = 1564\,\mu\text{s}, \quad (10)$$

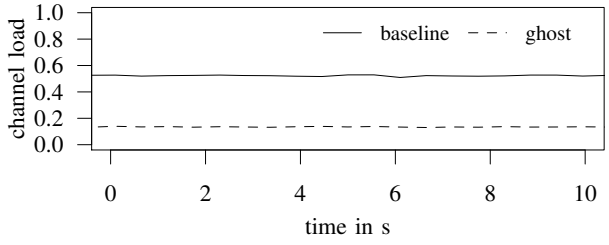
$$t_4 = t_2 + t_{\text{AIFS}} + CW_{\text{min}} t_{\text{slot}} + t_{\text{tx}}(2400\,\text{bit}) = 1954\,\mu\text{s}. \quad (11)$$

When changing App 1 to unicast (*baseline*, Exp 2), frames are delayed commensurate to the additional ACK frame that needs to be sent (and processed) – not just for App 1, which takes longer to send frames, but also for App 2 because of head of line blocking. Still, because unicast frames are almost immediately acknowledged, the head of line blocking effect is of no further consequence in this scenario. We thus next investigate a scenario where the intended receiver is not (or no longer) in the network.

Figure 2 illustrates the effects observed in the discussed alternative scenario (denoted as *ghost*, Exp 3 & 4) where App 3 was changed to transmit data to a device that was no longer there – thus reproducing the scenario of a neighbor having existed previously before moving out of reception range. It can readily be observed that, as predicted, the lost ACKs of App 3 transmissions had a catastrophic effect on the delay of App 1 and App 2 transmissions. In particular, no



(a) measurements (using Atheros AR9220 based WLAN cards)



(b) simulation (using the *Veins* simulator)

Figure 3. Channel load in the *baseline* and *ghost* scenarios.

appreciable difference can be observed between the impact on those applications that were generating broadcast frames (Exp 3 and Exp 4 App 2) and those that were generating unicast frames (Exp 4 App 1) – the reason being that both share a transmit queue with the frames generated by App 3. For both, lost acknowledgements cause head of line blocking, increasing the time they spent in the transmit queue until the head of line frame expires. Both broadcast and unicast frames were queued for a typical duration of 150 ms and delays could exceed 200 ms – well worse than the demands of many VANET applications [27], [28].

A further consequence of this is revealed when examining

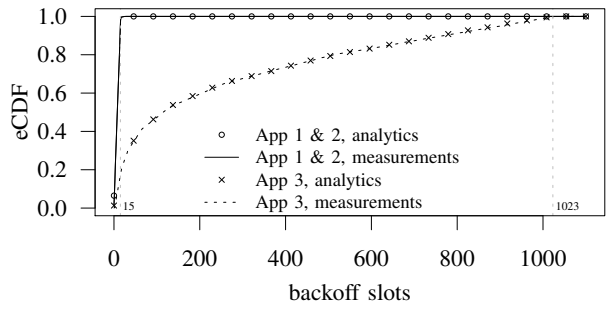
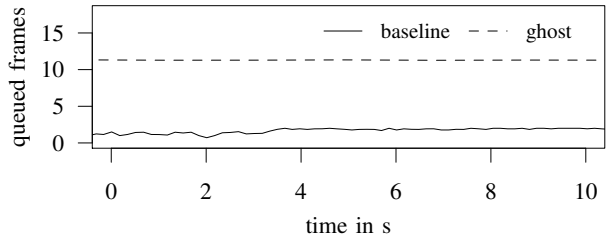
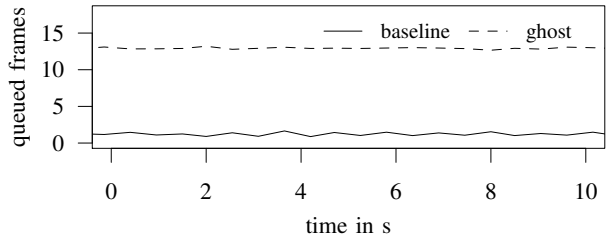


Figure 4. Distribution of chosen backoff slots for each application in the *ghost* scenario; results for hardware measurements and analytical calculations.



(a) measurements (using Atheros AR9220 based WLAN cards)



(b) simulation (using the *Veins* simulator)

Figure 5. Number of queued frames in the *baseline* and *ghost* scenarios.

channel load: In Figure 3 we see that, as expected, in the baseline scenario the applications manage to saturate the channel. Conversely, looking at the results for the *ghost* scenario, in both computer simulation and actual measurements lost acknowledgements cause the channel load to drop down to values of around 15 % to 20 %.

The reason for this is evident from Figure 4, where we plot the chosen backoff slots for each application and compare these measurements to Monte Carlo simulations following the analytical derivations. As predicted by the analytics, the contention window in measurements increases according to the exponential backoff algorithm when no acknowledgement is received for App 3.

This is reflected in the queue utilization, shown in Figure 5. As expected, in both computer simulations and actual measurements queue fill levels in the baseline scenario stay near zero, as traffic is non-bursty and the offered load is below the capacity of the channel. However, queue levels increase massively in the ghost scenario, caused by HOL blocking of packets retransmitted due to lost acknowledgements. This gives a good impression of the negative impact of failed unicast transmissions on the networking performance and points to potential dangers of this effect for bursty traffic.

All in all, our experiments illustrate the grave effect that head of line blocking, provoked by unicast frames addressed to a former neighbor, has on broadcast frames' delay.

E. Towards a Solution

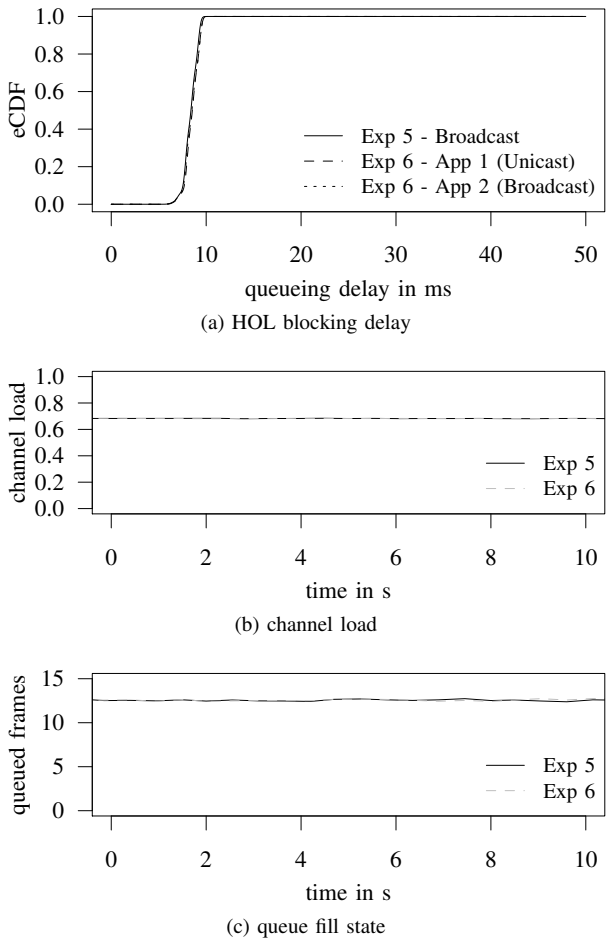
In the previous sections we show the negative impact of failed unicast transmissions on networking performance. One approach to avoid this is to lower the number of retransmissions at the MAC for failed unicast packets. To maintain the same level of reliability, retries then need to be taken care of by the application layer. Here we present such an application-based retransmission approach which avoids HOL blocking for failed unicast transmissions, but still retransmits those packets.

For each transmitted unicast packet, we do the following: If the transmission was successful (we received an acknowledgement after a SIFS) we delete the packet from the queue head. If the transmission was unsuccessful (we do not receive an acknowledgement within $t_{\text{ACK\_wait}}$) we keep the contention window at CW_{min} and do not perform any immediate retransmission. Instead, we reinsert the packet at the tail of the queue (i.e., after all other queued packets for that particular access category). If the retransmission count for that particular packet exceeds the configured maximum number of retransmissions, the packet is dropped.

We achieve this by setting the maximum number of retries for failed unicast transmissions at the MAC layer to zero but still waiting for an acknowledgement to be received. Instead of going into exponential backoff when missing an acknowledgement, we retransmit the packet at the application layer. Intuitively this will lead to lower delays for the remaining queued packets as no retransmissions on the MAC layer are performed which would cause HOL blocking.

We evaluate this approach using the scenario denoted as *app. retry* (Exp 5 & 6) in Table I. It is otherwise identical to Exp 3 & 4: We again have three different applications which generate messages in random order. App 3 sends frames as unicast to a station that does no longer exist, thus no acknowledgements will be received for those packets.

In Figure 6a we show the delay that packets spent in the EDCA queue. Because head of line blocking is effectively circumvented by the presented approach, no appreciable difference can be seen between the unicast and broadcast experiments. Compared to the default approach (of keeping the frame at the queue head until its retransmit count is exceeded) the delay of all frames is now tremendously lowered to around 8.5 ms (down from the approx. 150 ms shown in Figure 2). As a consequence, the presented approach also brings the channel load (shown in Figure 6b) to values comparable to those measured in the baseline scenario with no lost acknowledgements (see Figure 3). Similarly, as every failed transmission is still retried as often as in the *ghost* scenario, the queue occupancy remains comparable, as shown in Figure 6c. Yet, if lost acknowledgements are not a consequence of an absent receiver and a retry is actually warranted, additional delays are likely incurred by the frame taking a round trip through the application layer. Another side effect of this scheme is that packets get reordered, thus this approach most probably negatively influences higher layer transport protocols like TCP, requiring special attention [29].



(a) HOL blocking delay

(b) channel load

(c) queue fill state

Figure 6. Simulation: application layer retransmissions (using the *Veins* simulator). Note that all lines are overlapping.

Still, due to its simplicity, this approach can serve as a baseline for comparisons and, we believe, a basis for future work.

F. The Special Case of Active Attacks

In this section we evaluate the impact of failed unicast transmissions on the goodput of traffic flows by actively jamming MAC acknowledgements. In the past, similar experiments were performed [30] by using custom developed hardware and it was found that different vendors for 802.11b wireless cards experience different performance caused by non-standard compliant backoff procedures. We instead focus on the impact of lost acknowledgements on the backoff behavior of unicast transmissions, and on the degree to which the channel load and goodput are affected.

In the following experiments we build upon the work of Vanhoef and Piessens [31] which present firmware extensions for popular USB WiFi sticks based on the ath9k-htc driver to accomplish low-layer attacks. The goal of the jammer is to stop decoding a frame while it is on the air, and immediately start sending a custom short frame – thus causing interference at a receiver. With high probability this will lead to a wrong Frame Check Sequence (FCS) of the original frame on the receiver, which then cannot further process this frame. As we measure a minimum delay of around 126 µs from the first Byte up to the point where we are able to start sending our jamming frame, this is far too long to reliably jam MAC acknowledgements of 112 bit size. We solved this issue by detecting the frame for which we expect an acknowledgement to be sent, perform busy waiting up to the point where we expect the acknowledgement, and then send our short jamming frame. All these processes are performed on the firmware of the USB WiFi dongle, configured using a command sent to the firmware. This command includes parameters to define the length and modulation and coding scheme of the jam signal, which frames to detect, and the offset between frame detection and sending the jam signal.

Our scenario consists of a sender and a receiver, each equipped with a wireless card of type Compex WLM200N5-23ESD using an Atheros AR9220 chipset with the ath9k driver and running Linux 3.18. Building on the work of Lisovy et al. [24] we configured them in OCB mode on channel 178 (5890 MHz). Due to a lack of 10 MHz channel bandwidth support of our jamming system, we use 20 MHz channel bandwidth for all nodes. Keeping $N_{\text{DBPS}} = 48$ bit results in a bit rate of 12 Mbit. Further, the MAC uses contention window settings of $CW_{\text{min}} = 15$ and $CW_{\text{max}} = 1023$ slots, as well as an AIFSN value of 6. RTS/CTS operation is disabled. The short and long retry limits are configured to 7 and 4 retries, respectively; however in the experiment we found that the wireless card performs 20 retries – and that it does not invoke exponential backoff – for the case that ACKs are received but cannot be decoded, as we detail in Section III-F.

Further we use a dedicated node to continuously measure the channel load. Finally, our jamming node is equipped with a Netgear WNDA3200 USB WiFi dongle, configured similarly to the sender and receiver.

On the application layer we use the *iperf* tool in UDP mode to saturate the channel and measure goodput on the wireless link between the receiver and the sender. The packet length on iperf is set to $l_{\text{payload}} = 800$ Byte (6400 bit), which results in a total number of 6912 bit to be transmitted over the air including UDP header (64 bit), IP header (160 bit), LLC header (64 bit), and IEEE 802.11 header (224 bit).

We investigate 3 scenarios: In the *baseline* scenario, no jamming is performed and all acknowledgements are received. In the *jammed* scenario, acknowledgements are selectively jammed. In the *std* scenario, acknowledgements are selectively jammed, but we assume the hardware to follow the standard (7 retries, exponential backoff).

We first calculate the expected goodput of unicast packets on a wireless link for the given packet size, both for successful and failed transmissions.

We use the notation and derivations introduced in Section III-A, substituting t'' for t and T'' for T to indicate the use of timing parameters for 20 MHz channel bandwidth. In more detail, we use $T''_{\text{preamble}} = 16\,\mu\text{s}$, $T''_{\text{signal}} = 4\,\mu\text{s}$, $T''_{\text{sym}} = 4\,\mu\text{s}$, $t''_{\text{SIFS}} = 16\,\mu\text{s}$, $t''_{\text{slot}} = 20\,\mu\text{s}$, and $t''_{\text{rx\_delay}} = 25\,\mu\text{s}$.

Based on these, the transmission time of a frame of 6912 bit is calculated analogous to Equation (1) to be $t''_{\text{tx}}(6912\,\text{bit}) = 600\,\mu\text{s}$. Similarly, the time to transmit an acknowledgement frame of 112 bit length is $t''_{\text{tx}}(112\,\text{bit}) = 32\,\mu\text{s}$.

The arbitration time for which the channel has to be idle to start decrementing the backoff counter is derived analogous to

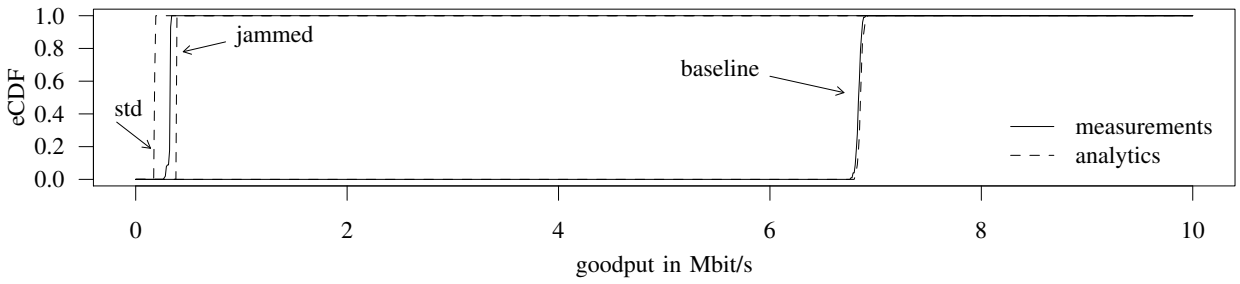


Figure 7. Distribution of measured goodput μ, compared with analytical predictions.

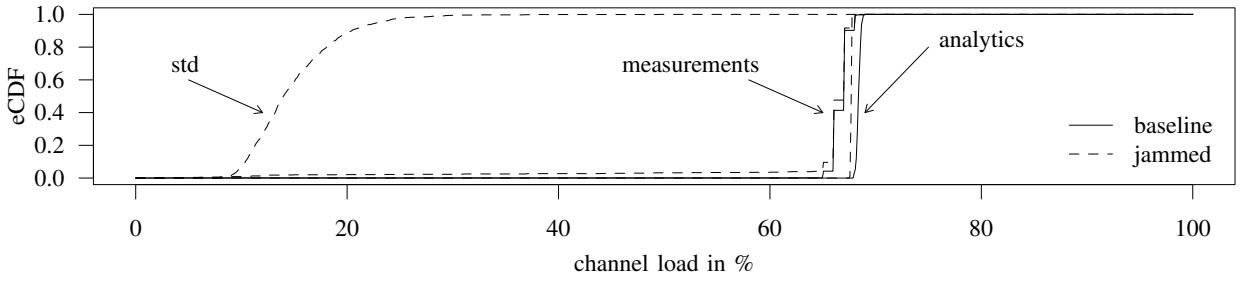


Figure 8. Distribution of measured channel load ρ, compared with analytical predictions.

Equation (5) as $t''_{\text{AIFS}} = 136\,\mu\text{s}$.

After the transmission of a frame ends, a node waits $t''_{\text{ACK\_wait}} = 61\,\mu\text{s}$, calculated analogous to Equation (3), to detect the PHY-RXSTART.indication which defines the point in time when a signal starts being decoded. If this indication is detected the node waits until its completion and checks whether the received frame is a valid acknowledgement for the data frame sent before. If no indication is detected, or if the frame is not a valid acknowledgement to the data frame, the exponential backoff procedure shall be invoked as outlined in the standard [4, Section 9.3.2.8].

In the *baseline* scenario (no jamming), the average time for channel access and transmitting a frame is calculated as

$$\begin{aligned} t_{\text{busy}} &= t''_{\text{tx}}(6912\,\text{bit}) + t''_{\text{tx}}(112\,\text{bit}) = 632\,\mu\text{s} \\ t_{\text{idle}} &= t''_{\text{AIFS}} + \frac{1}{2}\,CW_{\text{min}}\,t''_{\text{slot}} + t''_{\text{SIFS}} = 302\,\mu\text{s}. \end{aligned} \tag{12}$$

The average goodput is derived using the packet size without lower layer headers as

$$\mu_{\text{baseline}} = \frac{l_{\text{payload}}}{t_{\text{busy}} + t_{\text{idle}}} \simeq 6.85\,\text{Mbit/s}. \tag{13}$$

The channel load is derived as

$$\rho_{\text{baseline}} = \frac{t_{\text{busy}}}{t_{\text{busy}} + t_{\text{idle}}} \simeq 67.7\,\%. \tag{14}$$

In the *jammed* scenario (that is, assuming the nonstandard behavior of our hardware), the values change to

$$\begin{aligned} \bar{t}_{\text{busy}} &= 21\Big(t''_{\text{tx}}(6912\,\text{bit}) + t''_{\text{tx}}(112\,\text{bit})\Big) = 13\,272\,\mu\text{s} \\ \bar{t}_{\text{idle}} &= 21\Big(t''_{\text{AIFS}} + t''_{\text{SIFS}} + \frac{1}{2}t''_{\text{slot}}CW_{\text{min}}\Big) = 6342\,\mu\text{s} \\ \mu_{\text{jammed}} &\simeq 0.326\,\text{Mbit/s} \\ \rho_{\text{jammed}} &\simeq 67.7\,\%. \end{aligned} \tag{15}$$

For the *std* scenario (that is, expected reaction to jamming), we instead derive

$$\begin{aligned} \hat{t}_{\text{busy}} &= 8\Big(t''_{\text{tx}}(6912\,\text{bit}) + t''_{\text{tx}}(112\,\text{bit})\Big) = 5056\,\mu\text{s} \\ \hat{t}_{\text{idle}} &= 8(t''_{\text{AIFS}} + t''_{\text{SIFS}}) + \frac{1}{2}t''_{\text{CW}}(8) = 31\,696\,\mu\text{s} \\ \mu_{\text{std}} &\simeq 0.174\,\text{Mbit/s} \\ \rho_{\text{std}} &\simeq 13.76\,\%. \end{aligned} \tag{16}$$

We now compare our analytical calculations to real world experiments using iperf to send 800 Byte UDP packets on the wireless channel. We perform the experiment both in a baseline scenario without using a jammer and while jamming acknowledgements. We track the number of chosen backoff slots for each transmission by modifying the Linux kernel running on the receiver. As all interframe spaces and signal timings are known, the only unknown value is the time spent for decreasing the backoff counter to zero, which can be calculated by subtracting all known timings from the timespan between receiving the current frame, and the previously frame. However, we have to note that this only works when continuously generating packets on the sender side, i.e., fully saturating the wireless channel, as we do with iperf.

We measured the distribution of chosen backoff values by the sender. As could be expected, in the baseline scenario the backoff values are uniformly distributed over the range of CW_{min} (though we observed a slight offset of almost one slot, likely due to measurement inaccuracies). When jamming the acknowledgement of each unicast frame we expected that the exponential backoff scheme is invoked, thus increasing the contention window. However, the chosen backoff values do not increase, but are similar to the baseline scenario. This is due to somewhat surprising behavior of the wireless card: We observed that when it received incomplete (jammed) acknowledgements, the sender performed 20 retries of each frame. Further, it never

increased the contention window. The result of not doubling the contention window in case no valid acknowledgement is received is in line with the backoff behavior of different Atheros chipsets [20]. Note that (as shown in Section III-B) when no acknowledgement is received and the channel is idle during the ACK duration, the Atheros AR9220 chipset behaves as expected and performs exponential backoff.

The behavior of not performing exponential backoff between retransmission of frames when the acknowledgement is jammed can also be seen in the goodput metric. In Figure 7 we show the measurements together with Monte Carlo simulations following the derivations of Section III-F. The experiment matches our calculations, for both the baseline scenario with around 6.85 Mbit/s and the jamming scenario without performing exponential backoff with around 326 kbit/s. Moreover, we also plot the expected goodput when performing exponential backoff using 7 retries (denoted *std*) which goes towards 174 kbit/s.

Finally, we also measure the channel load for both scenarios. In Figure 8 we see that our measurements again fit our analytical findings. The channel load does not change significantly between the baseline scenario and the jamming scenario. This confirms that the tested wireless card does not perform the exponential backoff procedure for retransmissions required by IEEE 802.11 when an acknowledgement cannot be decoded.

IV. Large and Dynamic Networks

In order to investigate the impact of the discussed effects in a highly dynamic network, we conducted a computer simulation of a large, realistic VANET. The setup consisted of a large number of nodes running a typical protocol, which could be toggled between using broadcast or unicast communication. We now coupled the Veins simulation with the microscopic road traffic simulator SUMO (version 0.25.0) to model realistic road traffic.

As a common baseline and representing a prototypical VANET scenario [1], we configured a freeway scenario with a length of 7 km. In the interest of avoiding any border effects we performed network simulation in the center 5 km of the scenario. The 1 km border served to let the vehicles speed up and use realistic mobility patterns. We configured three different traffic densities on the freeway: 18 vehicles/km represented a very low traffic density scenario (characterized by few available neighbors and, thus, very challenging topology dynamics as we will show); 55 vehicles/km represented off-peak traffic on the freeway (low density, an in-between case); 169 vehicles/km represented high density traffic on a busy freeway (characterized by a very high number of neighbors and, thus, a challenge in terms of channel load as we will show). Road traffic was modeled by sampling from a distribution of five different vehicle types (two types of trucks, and three types of cars) modeling different kinds of drivers.

We collect results within a Region of Interest (ROI) of 3 km in order to not be influenced by border effects. The simulation warm-up period is configured to be 289 s to let the freeway get filled with vehicles, and another 11 s are used for the networking protocols to get into a steady state. Only after these 300 s we start to collect results. For all results, we plot the mean value together with the 95 % confidence interval (please note that these intervals are sometimes very small). We repeat each simulation setup at least 50 times with different seeds for the pseudo random number generator in order to get statistically significant results.

To show the impact of an application employing reliable unicast communication in a VANET we use a simple neighbor management process informing a Geocasting protocol which is tasked with disseminating information among vehicles. Such information might be as general as traffic reports or as specific as knowledge about certain active road works. Without loss of generality, we abstract a certain amount of information into what we call *information items*, each of which is simply represented as an opaque block of bits.

For the neighbor management process, each vehicle broadcasts a beacon at a frequency of 1 Hz and maintains a 1-hop neighbor table. Whenever vehicle v receives such a beacon from another vehicle u it adds u to the neighbor table $\mathbb{N}$. If two successive beacons of a vehicle are lost (here, after 2 s) this vehicle is removed from the neighbor table. This is performed right before information from a neighbor table is used.

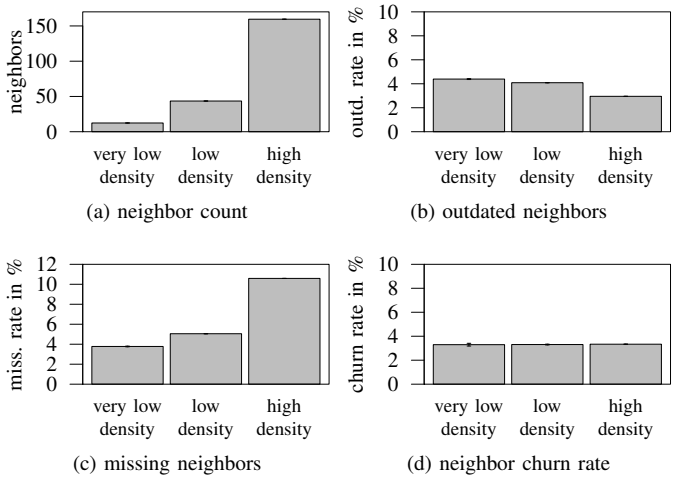
The Geocasting protocol builds on information in these neighbor tables. Its main building blocks are: First, maintaining a knowledge base consisting of arbitrary entries with geographic constraints and their expiration time. Second, exchanging of knowledge base digests among neighbors, as well as the requesting and receiving of entries from these knowledge bases. In more detail, the protocol works as follows:

Whenever a vehicle v discovers a new neighbor u, it makes a probabilistic decision whether to inform this neighbor about information stored in the knowledge base. With a probability of $p = 1/(\text{new neighbors per s})$, node u will be informed of the active events stored in the knowledge base of v. In this case v sends a small *digest* including fingerprints of all available events in the knowledge base, up to the maximum frame size.

When node u receives a digest, it responds with a *data request* including fingerprints of interesting information, called missing entries: An event is marked as missing if the distance between u and the entry's destination position is lower than the distance between v and the entry's destination position, or if the vehicle is driving towards the destination direction. In other words, a node only selects an entry as missing if it is closer to its destination position than the node which offers the entry, or the vehicle is driving towards the destination.

A node v which receives a data request from u constructs and sends a *data packet* to u containing all information which was marked as missing by u, again limited by the maximum frame size. This data packet can be overheard by all other neighbors using a monitor interface connected to the transceiver. When this data is received by any node w the knowledge base gets updated. Next, w iterates over all neighbors n in its neighbor table $\mathbb{N}$; then, for each neighbor it takes a probabilistic decision with $p = |\mathbb{N}|^{-1}$ whether to send a digest to node n.

In our simulation, we generate new information items at a rate of 4 Hz in the knowledge base of vehicles at each end of the ROI. Further we present simulation results for information item generation rates of 1 Hz and 10 Hz. The information items' destination position is at the opposite end of the ROI, meaning



(a) neighbor count (b) outdated neighbors

(c) missing neighbors (d) neighbor churn rate

Figure 9. Neighbor table performance for different traffic densities.

that each has to be disseminated through the whole network. After a simulation has reached a steady state, we track each information item as it traverses the network. We record the delay each node measures from generation of this information item until reception. The simulation was configured to collect results for 5 s.

Neighbor beacons use a different EDCA queue for transmission than the digest packets, data request packets, and data packets in order to not influence each other in terms of head of line blocking as outlined in Section III. In our simulations we have chosen AC_BE with an AIFS value of 6 slots for neighbor beacons, and AC_BK with an AIFS value of 9 slots for the rest. The CW_{min} and CW_{max} values for both EDCA queues are configured to be 15 and 1023 slots respectively. The number of retries for reliable communication is set to 7 retransmissions. The maximum frame size was configured to be 1024 Byte, an information item in the knowledge base takes 64 Byte, and a digest takes 8 Byte per entry.

A. Neighbor Management and Topology Dynamics

An important metric to evaluate the neighbor table maintenance and topology dynamics is the set of 1-hop neighbors known to each node – more specifically, the correctness of this information. We compare the neighbor maintenance process against an oracle. This oracle calculates the neighbor information according to a unit disk model. For the distances of nodes to be treated as 1-hop neighbors we use the 99 % quantile of distances of successful frame transmissions. Thus, we are able to calculate the fraction of missing and outdated neighbors which represents the quality of the maintained neighbor information. Finally, we evaluate the neighbor churn rate, meaning how many 1-hop neighbors were deleted from the neighbor table per second – due to lost beacons or because the node was not in range anymore. This allows us to quantify the stability of neighbor tables.

In our simulation we observe a mean value of around 12, 44 and 160 neighbors for each vehicle for the very low, low and high density scenario respectively (Figure 9a). Of course, the amount of neighbors is no indicator of how accurate this information is. We therefore investigate the rate of outdated (Figure 9b) and missing (Figure 9c) neighbors compared to an oracle and measure around 4 % outdated and 5 % missing information for the low density scenario. In the high density scenario we measure around 4 % outdated and 11 % missing information respectively. In the very low density scenario the value stayed at around 4 % for the outdated neighbor fraction and went slightly below 4 % for the missing neighbor information. To investigate the reasons for these observations in more detail, we turn to the mean churn rate of neighbors (Figure 9d). We note that the value remains constant for all three traffic densities, indicating that the neighbor management process does not cause channel congestion.

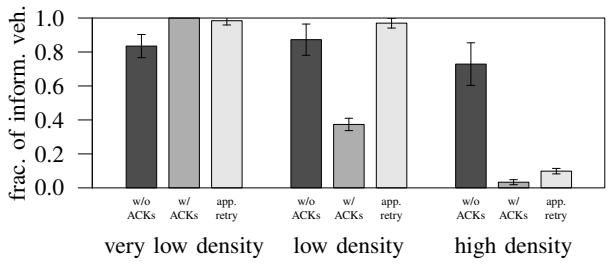
Taken together, all results indicate that – even in this simple freeway setting with no corners and no buildings – the network topology dynamics are non-negligible.

B. Application Performance

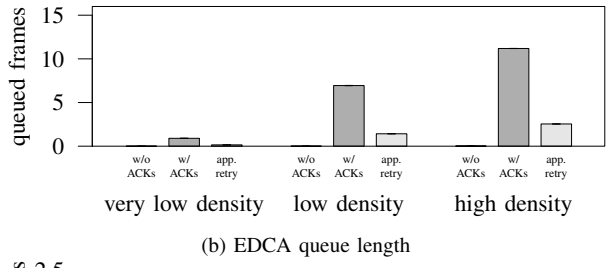
We show results for the Geocasting application using three different configurations for application layer messages (neighbor beacons are always sent as broadcasts): (*a*) messages sent using broadcast, meaning a node performs no retries and immediately goes into post transmit backoff after transmission of a frame (denoted as *w/o ACKs*); (*b*) messages sent using reliable unicast as defined by the IEEE 802.11 HCF (denoted as *w/ ACKs*); and (*c*) messages sent as broadcast frames, but using our application-based retransmission algorithm outlined in Section III-E (denoted as *app. retry*). Note that all configurations – broadcast, unicast, and our application-based retransmission algorithm – allow overhearing of information, thus a node can overhear unicast packets not designated to it (alike to running an additional interface in monitoring mode), handing their information up to the application layer.

For the Geocasting application the premier metric to gauge efficiency (and efficacy) of protocol performance and error control is the fraction of informed nodes for a specific information item. Besides that, also the delay for receiving nodes to get this information plays an important role. Further we report the number of frames queued in the EDCA queue assigned for Geocasting traffic as well as the queuing time of frames. Finally, we also evaluate the used channel capacity by measuring the channel load.

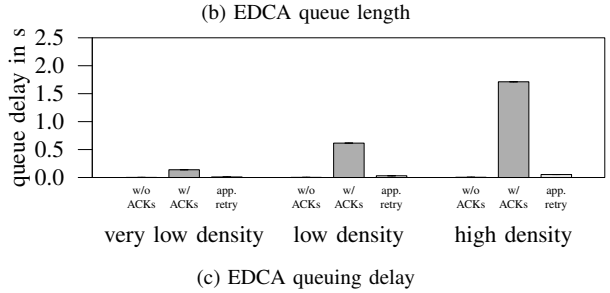
In Figure 10a, we show the fraction of vehicles that received a particular information item for three configurations: very low, low and high road traffic densities by using a information item generation frequency of 4 Hz. Not knowing about the effects discussed in these manuscript, one might expect a higher rate of informed nodes when using a communication mode with ACKs than without, that is, when using the reliable unicast communication mode of WLAN. The results, however, indicate the exact opposite for the low density and high density scenario: The fraction of informed vehicles drops to approximately half in the low density scenario, to one twentieth in the high density scenario. When substituting our very simple *app. retry* approach (see Section III-E) success varies depending on the node density: At low node densities, the approach indeed performs better than one without retries. At high node



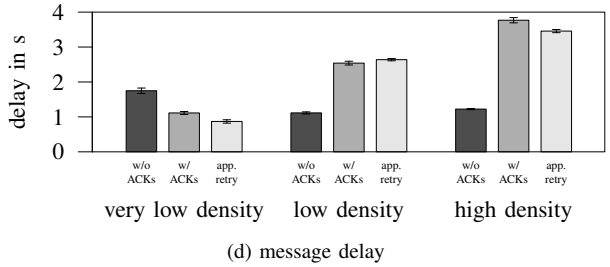
(a) fraction of informed vehicles



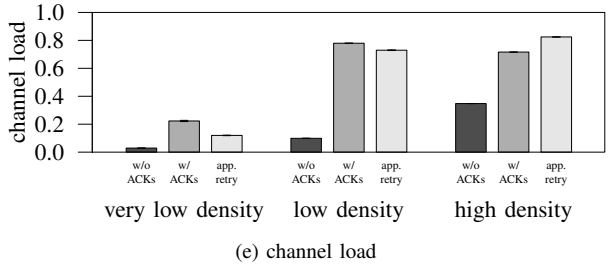
(b) EDCA queue length



(c) EDCA queuing delay

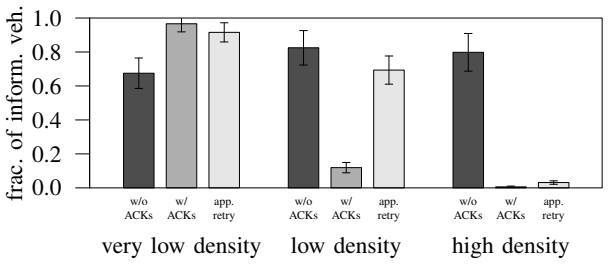


(d) message delay

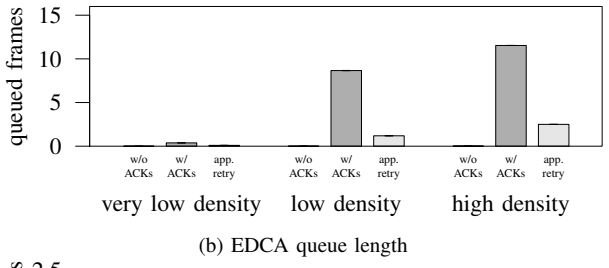


(e) channel load

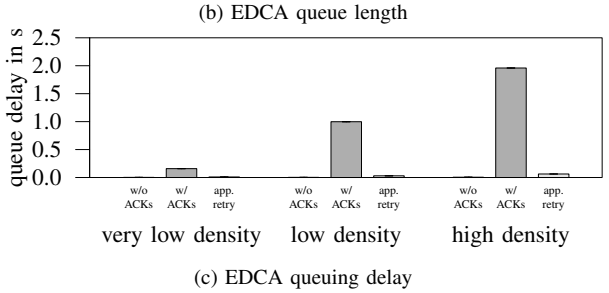
Figure 10. Performance of the Geocasting app for three different traffic densities and a message generation interval of 4 Hz.



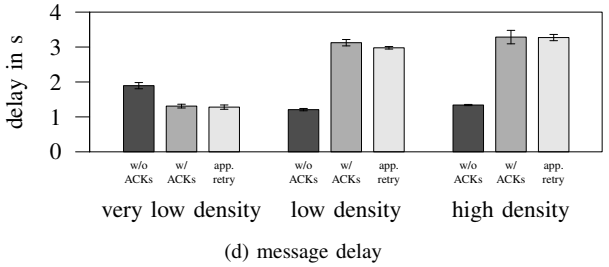
(a) fraction of informed vehicles



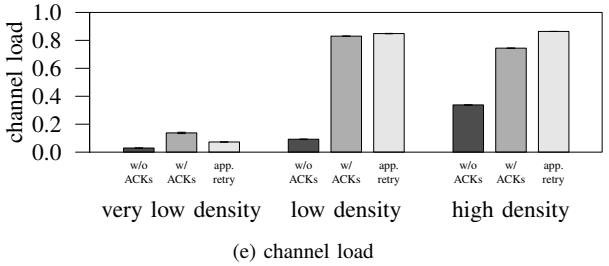
(b) EDCA queue length



(c) EDCA queuing delay



(d) message delay



(e) channel load

Figure 11. Performance of the Geocasting app for three different traffic densities and a message generation interval of 10 Hz

densities, however, not retrying transmissions at all results in vastly better application performance. However, when using a very low traffic density scenario, retransmissions of both, the reliable unicast communication mode of WLAN and our simple *app. retry* approach will increase the number of informed vehicles compared to a pure broadcast based communication principle. The results are comparable when using a information item generation rate of 10 Hz as shown in Figure 11a and 2 Hz as shown in Figure 12a.

To investigate the reasons for these results, we take a closer look at the queues. Figure 10b reveals head of line blocking as the reason for the low application performance when using the reliable unicast communication mode and a information item generation rate of 4 Hz. Because of the topology dynamics, the destination of a message is sometimes no longer there to reply (see Section IV-A), so frames waiting for ACKs both fill and block the queues. With increasing traffic density and information item generation rate, the number of queued frames also increase like seen in Figure 11b for a rate of 10 Hz and Figure 12b for a rate of 2 Hz.

Figures 10c, 11c and 12c illustrate what this means for message age for different information generation rates and

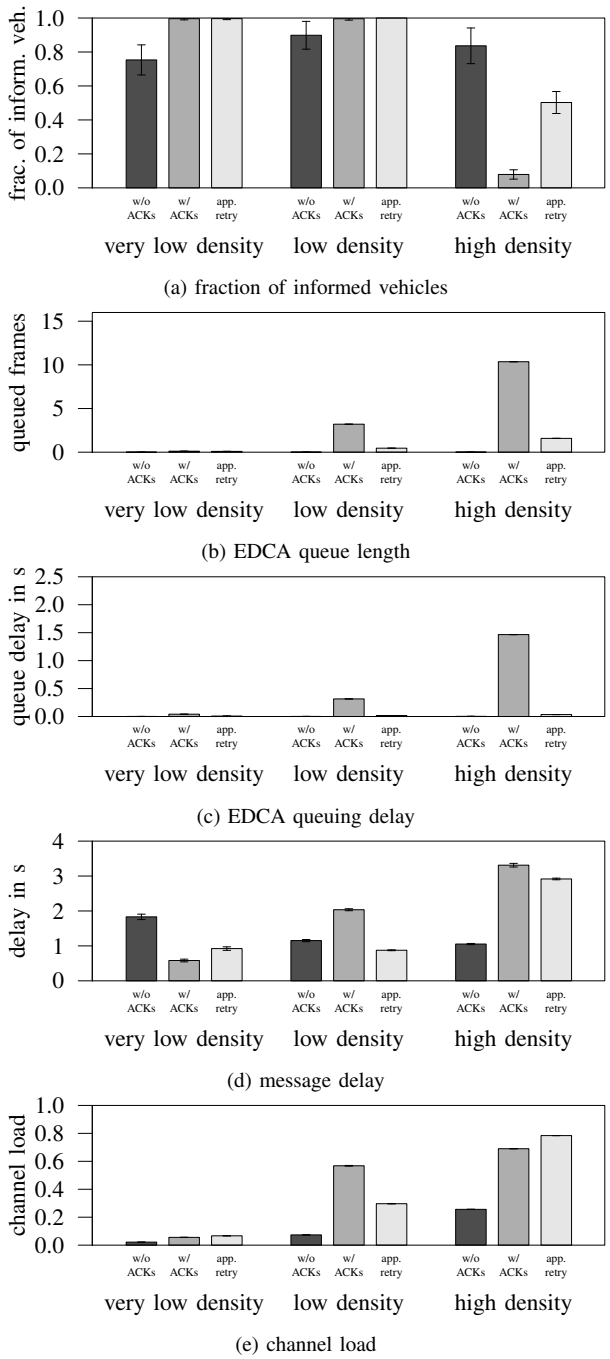


Figure 12. Performance of the Geocasting app for three different traffic densities and a message generation interval of 2 Hz.

traffic densities. As discussed in Section III, the effect of blocked queues when using reliable unicast communication is cumulative, yielding messages that queue for durations on the order of seconds before they are finally sent. This further exacerbates the effects of even moderate topology dynamics. As our simple *app. retry* approach does not suffer from head of line blocking, it works around this problem.

Yet, Figures 10d, 11d and 12d reveal for the low and high density scenario that, while retries might be able to spread a message further through the network, they increase the average delay with which network participants receive new information in the network. Only in the very low density scenario the delay of broadcast based communication not using frame retries is slightly higher due to the application protocol behavior: when new neighbors are detected they will be informed about information items available in a vehicles' knowledge base as outline in Section IV.

Figures 10e, 11e and 12e illustrate one of the reasons – and the reason why the simple *app. retry* approach no longer works in high density networks: All investigated forms of retries increase the channel load. Indeed, in the case of high density networks, the channel load increases to more than the network can handle. Thus, while retries have a certain value for improving the reliability of communications even in networks of high topology dynamics, blindly retrying every failed transmission multiple times leads to few potential gains at the cost of massively increased channel load.

V. Conclusion

We studied the WLAN mechanism for reliable unicast communication in networks of high topology dynamics. Taking VANETs as a common example, we demonstrated that this mechanism frequently causes head of line blocking because of missing ACK frames. We investigated this effect using analytical calculations, measurements on hardware, as well as computer simulations.

We showed that head of line blocking can be disastrous for applications with low latency requirements: a lost ACK frame can block other frames for an average of 40 ms and a single unicast application can block twice as many broadcast applications on the same node for more than 200 ms. We also showed that even moderate topology dynamics can cause the higher layer protocol performance of a system to suffer massively from head of line blocking. We demonstrated an increase of channel load to the point of overloading the channel, massive message loss, and increased delays on the order of seconds – all due to the interplay of network topology dynamics and the reliable unicast communication mechanism. We further showed that head of line blocking is easy to trigger using active or passive attacks – and that such attacks can be doubly effective against commodity WLAN cards: Due to non-standard behavior of such cards, the provoked drop in goodput of a network under attack is not accompanied by the expected drop in channel load.

As a way towards a solution, we also investigated a simple approach that avoids triggering head of line blocking by relegating the task of retrying failed transmissions from the MAC layer to higher layers. We showed that even this simple approach can reduce delays by an order of magnitude, though the retries will still cause substantial channel load. We conclude that wireless networks of high topology dynamics can benefit from adapted retry mechanisms for reliable unicast communication.

References

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

[2] "Wireless Access in Vehicular Environments (WAVE) – Multi-channel Operation," IEEE, Std 1609.4-2010, Feb. 2011.

[3] "Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band," ETSI, EN 302 663 V1.2.1, Jul. 2013.

[4] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, Std 802.11-2012, 2012.

[5] "Wireless Access in Vehicular Environments," IEEE, Std 802.11p-2010, Jul. 2010.

[6] S. Laux, G. S. Pannu, S. Schneider, J. Tiemann, F. Klingler, C. Sommer, and F. Dressler, "OpenC2X – An Open Source Experimental and Prototyping Platform Supporting ETSI ITS-G5," in *8th IEEE Vehicular Networking Conference (VNC 2016), Demo Session*. Columbus, OH: IEEE, Dec. 2016, pp. 152–153.

[7] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," in *15th IEEE Conference on Computer Communications (INFOCOM 1996)*. San Francisco, CA: IEEE, Mar. 1996, pp. 1133–1140.

[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] 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.

[10] F. Dressler, H. Hartenstein, O. Altintas, and O. K. Tonguz, "Inter-Vehicle Communication – Quo Vadis," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 170–177, Jun. 2014.

[11] E. Schoch, F. Kargl, M. Weber, and T. Leinmüller, "Communication Patterns in VANETs," *IEEE Communications Magazine*, vol. 46, no. 11, pp. 119–125, Nov. 2008.

[12] 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*, 2017, available online: http://dx.doi.org/10.1109/TNET.2017.2763185.

[13] S. Yousefi, M. S. Mousavi, and M. Fathy, "Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives," in *6th International Conference on ITS Telecommunications (ITST 2006)*, Chengdu, China, Jun. 2006, pp. 761–766.

[14] M. Kihl, M. Sichitiu, T. Ekeroth, and M. Rozenberg, "Reliable Geographical Multicast Routing in Vehicular Ad-Hoc Networks," in *Wired/Wireless Internet Communications*. Springer, 2007, pp. 315–325.

[15] A. Böhm, M. Jonsson, K. Kunert, and A. Vinel, "Context-Aware Retransmission Scheme for Increased Reliability in Platooning Applications," in *6th IFIP/IEEE International Workshop on Communication Technologies for Vehicles (Nets4Cars 2014-Spring)*. Offenburg, Germany: Springer, May 2014, pp. 30–42.

[16] F. Li and Y. Wang, "Routing in Vehicular Ad hoc Networks: A Survey," *IEEE Vehicular Technology Magazine*, vol. 2, no. 2, pp. 12–22, Jun. 2007.

[17] 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.

[18] M. L. Sichitiu and M. Kihl, "Inter-Vehicle Communication Systems: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 2, pp. 88–105, 2008.

[19] L. Urquiza-Aguiar, C. Tripp-Barba, and Á. R. Muir, "Mitigation of packet duplication in VANET unicast protocols," *Elsevier Ad Hoc Networks*, vol. 52, pp. 63–77, Dec. 2016.

[20] B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller, "Maranello: Practical Partial Packet Recovery for 802.11," in *7th USENIX Conference on Networked Systems Design and Implementation (NSDI 2010)*, San Jose, CA, Apr. 2010, pp. 205–218.

[21] Y. Xie, I. W.-H. Ho, and L. F. Xie, "Stochastic Modeling and Analysis of Unicast Performance in 802.11p VANETs," in *10th International Conference on Information, Communications and Signal Processing (ICICS)*. Singapore: IEEE, Dec. 2015.

[22] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, Mar. 2000.

[23] R. Reinders, M. Van Eenennaam, G. Karagiannis, and G. Heijenk, "Contention window analysis for beaconing in VANETs," in *7th International Wireless Communications and Mobile Computing Conference (IWCMC 2011)*. Istanbul, Turkey: IEEE, Jul. 2011, pp. 1481–1487.

[24] R. Lisovy, M. Sojka, and Z. Hanzálek, "IEEE 802.11p Linux Kernel Implementation," Industrial Informatics Research Center, Czech Technical University, Technical Report, Dec. 2014.

[25] W. Alasmary and W. Zhuang, "Mobility impact in IEEE 802.11p infrastructureless vehicular networks," *Elsevier Ad Hoc Networks*, vol. 10, no. 2, pp. 222–230, Mar. 2012.

[26] 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.

[27] M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. Lo Cigno, and F. Dressler, "Towards Inter-Vehicle Communication Strategies for Platooning Support," in *7th IFIP/IEEE International Workshop on Communication Technologies for Vehicles (Nets4Cars 2014-Fall)*. Saint-Petersburg, Russia: IEEE, Oct. 2014, pp. 1–6.

[28] T. K. Mak, K. P. Laberteaux, and R. Sengupta, "A Multi-channel VANET Providing Concurrent Safety and Commercial Services," in *2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2005)*. Cologne, Germany: ACM, Sep. 2005, pp. 1–9.

[29] E. Blanton and M. Allman, "On making TCP more robust to packet reordering," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, pp. 20–30, Jan. 2002.

[30] G. Bianchi, A. Di Stefano, C. Giaconia, L. Scalia, G. Terrazzino, and I. Tinnirello, "Experimental assessment of the backoff behavior of commercial IEEE 802.11b network cards," in *26th IEEE Conference on Computer Communications (INFOCOM 2007)*. Anchorage, AK: IEEE, May 2007, pp. 1181–1189.

[31] M. Vanhoef and F. Piessens, "Advanced Wi-Fi Attacks Using Commodity Hardware," in *30th Annual Computer Security Applications Conference (ACSAC 2014)*. New Orleans, LA: ACM, Dec. 2014, pp. 256–265.

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.

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 also serves 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.

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.