QQDCA: Adapting IEEE 802.11 EDCA for Unicast Transmissions at High Topology Dynamics

Gurjashan Singh Pannu, Florian Klingler, Christoph Sommer, and Falko Dressler
Heinz Nixdorf Institute and Dept. of Computer Science, Paderborn University, Germany
{pannu,klingler,sommer,dressler}@ccs-labs.org

***Abstract*—We present Multi-Queue Distributed Channel Access (QQDCA), an improved MAC scheme to mitigate the head-of-line (HOL) blocking problem observed in highly dynamic networks when Wireless LAN (WLAN) unicast (re-)transmissions are used. Focusing on Vehicular Ad Hoc Networks (VANETs) as a prime example, our novel MAC protocol can be used for all IEEE 802.11 based wireless networks having very dynamic topologies. In brief, failing unicast messages create a problem by blocking all remaining frames in the transmit queue until they are successfully acknowledged or the maximum retransmission counter expires. This HOL blocking has been identified as a rare or temporary phenomenon in the late 1990s already. However, it turned out that the problem is both much more frequent and persistent in VANETs due to their very dynamic nature. Here, it may take a substantial amount of time in which no other message can be sent. We present and discuss QQDCA to address this problem, analytically show its benefits, and evaluate the protocol in an extensive simulation study. QQDCA can reduce the delay induced by HOL blocking by up to 95 %.**

I. Introduction

Wireless communication based on IEEE 802.11 Wireless LAN (WLAN) has become the de facto standard for vehicular networking [1]. Standardization bodies all around the world developed protocol suites, e.g., the U.S. DSRC/WAVE and the European ETSI ITS-G5 stacks, which use IEEE 802.11p to support various application domains ranging from safety to efficiency to entertainment. Geonetworking protocol [2], which is an essential part of ETSI ITS-G5, uses unicast communication extensively. A fundamental part of the MAC is error control, i.e., to ensure reliable data transmissions by using retransmissions, in particular for unicast communication. IEEE 802.11 using standard Enhanced Distributed Channel Access (EDCA) provides a simple Automatic Repeat Request (ARQ) scheme for error control, in which every unicast frame, i.e., any frame addressed to an individual station, is retransmitted (potentially multiple times) if no Acknowledgment (ACK) is returned from its destination. Assuming overload or bad channel conditions to be the most common root cause, between these retransmissions, an exponential backoff procedure is invoked to lower channel utilization and stabilize the network. To achieve such retransmission, the frame is kept at the head of the transmit queue and, thus, blocks any other frames. This leads to the head-of-line (HOL) blocking problem, which has been studied in depth for WLAN [3]–[6].

It is important to distinguish between two possible cases for lost acknowledgements in highly dynamic wireless networks. First, for high channel utilization, the probability of frame (data or ACK) collisions increase, e.g., due to hidden terminals. Secondly, communication neighbors may disappear. This can happen when communicating nodes are at the fringe of their communication distance, or nodes having an asymmetric communication channel. In Vehicular Ad Hoc Networks (VANETs), the mobility of the vehicles further amplifies the problem. In such a case, it makes no sense to perform retransmissions of frames, and even worse, perform exponential backoff between those retransmissions.

In 2015, Klingler et al. [7] investigated the influence of HOL blocking on vehicular networks and showed that the problem is not just a theoretic one, but that it has significant impact on the performance of vehicular networks using unicast communication. In particular, experimental results show that each single failed unicast transmission can block the transmit queue for tens of ms, leading to delays of hundreds of ms for both broadcast and unicast frames. This also holds for EDCA as used in IEEE 802.11p, if unicasts and broadcasts use the same access category, i.e., the same queue at the MAC. Investigations further revealed that the root cause of many lost acknowledgements, and thus HOL blocking, in a usual VANET scenario [7] is not interference. Instead, the fluctuation of neighbors in such a network due to high mobility is the main cause for degraded performance.

In this paper, we propose an improved version of the EDCA scheme, named Multi-Queue Distributed Channel Access (QQDCA), to counter the severe performance degradation when unicast transmissions fail. The core idea is to provide multiple backup queues besides the main transmission queue. This way, every failing unicast frame can be re-queued into one of the backup queues in order to avoid HOL blocking in the main queue. We further propose a two-stage scheduling algorithm to process the main and the backup queues. We evaluated our QQDCA system in-depth both analytically as well as in an extensive simulation study. Our results show that we can substantially reduce the impact of HOL blocking with little to no adverse effects on regular operation.

Our main contributions can be summarized as follows:

- We present QQDCA, a novel extension to EDCA to mitigate the HOL blocking problem. QQDCA uses a set of backup queues as well as a two-stage scheduling approach to handle failing unicasts (Section III).
- We analytically evaluate the behavior of QQDCA to validate the system internals. This analysis already shows the huge potentials of QQDCA (Section IV).

- We performed an extensive simulation study to show the overall performance gain in different scenarios. This study confirms the capability of QQDCA to alleviate HOL blocking (Section V).

II. Related Work

The HOL blocking of transmit queues for unicast transmissions in the MAC layer is known from the late 1990s [3]. Several approaches have been investigated to mitigate this problem in different network types like commercial WLANs [5], [6], Mobile Ad Hoc Networks (MANETs) [8], and Flying Ad Hoc Networks (FANETs) [9]. Because of the unique characteristics of each network type, network-specific solutions can, in many cases, not be directly applied to another network type to solve the HOL blocking problem. For example, existing solutions did not take the characteristics of VANETs into account, e.g., very rapidly changing topologies which makes existing solutions not perfectly applicable for vehicular networking.

Bhagwat et al. [3] were among the first ones to point out the HOL blocking problem in a WLAN connected through a base station to a wired network. They proposed to maintain a set of per destination transmit queues and a scheduler that keeps on monitoring the wireless link status for the selection of next destination for transmission. Frame loss on a queue results in deferring further transmissions until the channel has recovered again. The approach leads to improved performance by employing per destination queues. However, this system suffers from two main issues: First, the links which have a good channel quality get an unfair share of the bandwidth. Second, their channel scheduler is highly dependent upon the accuracy of the channel state predictor.

Having the correct measure of channel state on every link at any point of time is not feasible in reality. Fragouli et al. [4] proposed a feasible solution to measure the channel quality on-demand, i.e., at the time when there is a need to transmit frames on a specific link. To do this, they introduced a goodness metric to describe the quality of the wireless link between a sender and a receiver. This goodness metric is derived as a function of the number of Ready To Send (RTS) attempts necessary for a successful reception of a Clear To Send (CTS) frame. With an increasing number of RTS attempts on a link, its goodness metric keeps decreasing.

These approaches helped to alleviate HOL blocking in infrastructure based networks. While the base station is able to monitor the channel state via feedbacks on the control channel, a distributed approach is needed for applications in MANETs or VANETs. Wang et al. [10] proposed Opportunistic packet Scheduling and Media Access control (OSMA) to solve the HOL blocking problem in MANETs. This MAC also maintains per-destination transmit queues and relies upon a modified RTS/CTS handshake. The sending station selects a subset of candidate receivers for the next unicast transmissions and multicasts an RTS to them. The order in which the receivers are supposed to reply with a CTS is also included in the RTS. Each candidate receiver evaluates the channel state based on a physical layer assessment of the received RTS. The sent CTS is piggybacked with additional information about the channel state that is used by the sender to decide the best receiver for the next unicast frame. From the perspective of VANETs, this approach has a few drawbacks. One major drawback is to do with the modified RTS/CTS handshake. To accommodate all the CTS frames from candidate receivers for a single RTS transmission, the waiting time before the actual unicast data is transmitted is increased. Second, no queue is reserved for broadcasts, a majority of VANET network traffic [11].

Wang et al. [12] further enhanced OSMA by taking advantage of multi-user diversity, rate adaptation techniques, and frame bursting. The HOL blocking problem can also be tackled indirectly using techniques like cooperative relaying [13]. In cooperative relaying, the stations acting as a relay help in relaying the frames to the destination. This increases overall network throughput and reliability of unicast communication which, in turn, reduces the incidence of HOL blocking.

Rozner et al. [5] proposed a solution to the problem in WLANs using network coding techniques. They proposed a MAC protocol that maintains two queues: the first queue receives the frames from the upper layer and the second queue holds all of the failed unicast frames. Now, for retransmission from the second queue, several frames are coded together and transmitted in a single transmission. However, the selection of frames from the retransmission queue for coding becomes an NP-hard problem. To solve this problem, the protocol compares different heuristics for investigating the performance. However, for more dynamic VANETs, where the neighbor count for a vehicle could change very rapidly from a few vehicles to hundreds in congested networks, finding an approximation to this NP-hard problem is a non-trivial task.

To sum up, HOL blocking in the MAC layer has been identified several times in various types of networks. To the best of our knowledge, no solution has been presented to solve this problem in VANETs and the existing solutions cannot be applied directly to VANETs. In this paper, we present QQDCA to fill the missing gap and to alleviate the HOL blocking effects caused by failed unicast communication in VANETs.

III. QQDCA

In the following, we describe the key components of QQDCA. The complete protocol is described in Algorithm 1.

A. Transmit Queues

The traditional EDCA system was meant for providing simple Quality of Service (QoS) in wireless networks. It provides multiple Access Categorys (ACs). Each AC maintains different clear channel access parameters and has only one transmit queue. QQDCA replaces these single transmit queues by an overhauled queuing system as shown in Figure 1. It has one transmit queue called the Main Queue (MQ), which is equivalent to the queue in the traditional EDCA system. Additionally, it is also capable of creating on-demand (virtual) per-destination transmit queues called Backup Queues (BQs) for the failed unicast frames.

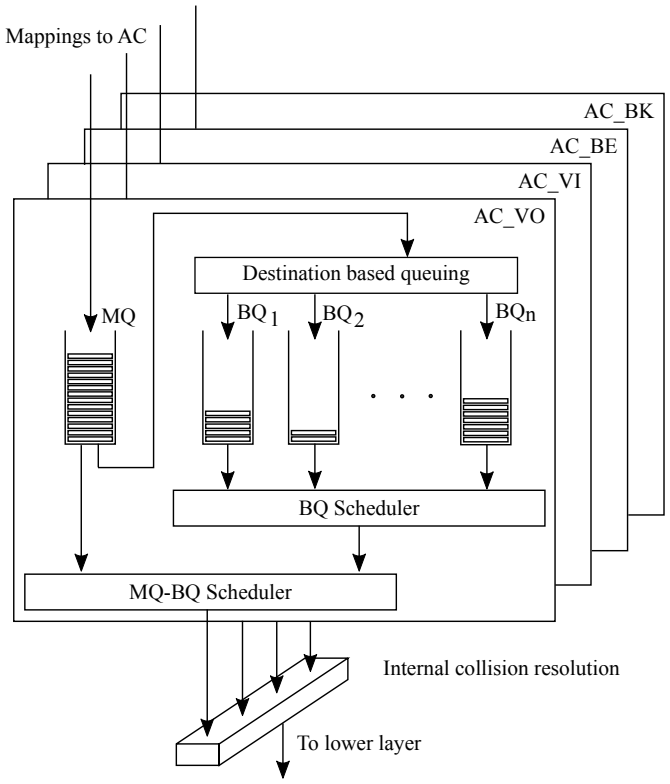

Figure 1. QQDCA architecture: The MQ receives all the frames from the upper layer. Failed unicast frames are moved to the BQ corresponding to the destination address. BQ Scheduler and MQ-BQ Scheduler collectively decide the queue from which next transmission takes place.

Frames received from upper layers are directly enqueued into the MQ. In contrast to the IEEE 802.11 standard, where the unicast retry attempts take place from the same transmit queue which causes persistent blocking of all trailing frames, the MQ in QQDCA removes all failed unicast frames right after sending to avoid any blocking of the following frames. The removed unicast frame is then enqueued into a BQ tagged with the destination address of the unicast frame. BQs are per-destination queues, i.e., each BQ can hold unicast frames targeted to a unique destination. A new BQ corresponding to a specific destination address is created if it does not exist at the time of a unicast failure. In order to avoid reordering of the frames on individual links, unicast frames from the MQ are also pushed to the BQ for the same destination without any transmission attempt if the corresponding BQ is not empty.

To sum up, each AC has one MQ and a variable number of BQs depending on the number of unicast failures to different destinations at any point of time. The presence of multiple transmit queues results in additional queue scheduling problems such as when to transmit from the MQ and when from one of the BQs – and which BQ to pick in the latter case.

B. Schedulers

To solve these problems, our QQDCA system is equipped with two schedulers, also shown in Figure 1.

The MQ-BQ Scheduler is responsible for deciding whether the frame to be transmitted comes from the MQ or from one of the BQs. We decided to use round robin scheduling, which gives alternating turns to both queue types, thus an equal share of the channel to both types of transmissions.

However, we suspect that because of round robin scheduling employed on the MQ-BQ Scheduler, the system may tend to stay in smaller contention windows. To elaborate more on this, let us consider that both the queue types have some frames enqueued and the last transmission took place on a BQ. According to the round robin policy, the next frame for transmission will come from the MQ. As the MQ can hold both unicast and broadcasts, there are two possibilities:

1) *Broadcast:* According to the backoff procedure defined in IEEE 802.11 standard, the broadcast transmission is always followed by resetting the contention window to its minimum. Thus, our system will reset the contention window to minimum.
2) *Unicast:* Here, we have to distinguish two cases: (a) the unicast is transmitted successfully and (b) the unicast transmission fails. In the former case, the contention window is again reset to minimum. The latter case results

Algorithm 1 QQDCA behavior when ready to transmit

Input: MQ$[n]$, BQ$_i[n]$: ordered lists of frames in all queues
execute Algorithm 2
if MQ-BQ Scheduler selects MQ **then**
  $Q \leftarrow$ MQ ▷ *set active queue to MQ*
else
  $i \leftarrow$ decision of BQ Scheduler
  $Q \leftarrow$ BQ$_i$ ▷ *set active queue to BQ$_i$*
end if
$p \leftarrow Q[0]$ ▷ *get HOL frame of selected queue*
transmit p, wait for completion
if p was broadcast **or** unicast of p was successful **then**
  reset contention window to minimum, backoff
  return
end if
double contention window, backoff
if Q is MQ **then**
  create BQ$_i$ for destination of p
  execute Algorithm 2
  return
end if
if retry limit reached **then**
  flush and delete BQ
end if

Algorithm 2 Move MQ frames to BQs if they exist

Input: MQ$[n]$, BQ$_i[n]$: ordered lists of frames in all queues
if $\exists$ any BQ$_i$ with destination of MQ[0] **then**
  enqueue(BQ$_i$, dequeue(MQ))
  execute Algorithm 2 again
end if

in doubling of the contention window.

As, thus, the majority of situations result in resetting the contention window to minimum, it becomes more common that the system operates with a small contention window. If road traffic is congested, there are many vehicles within each other's communication range. When all or some of them are stuck in lower contention window limits, the chances of collisions increase. We investigate this phenomenon in Section V.

The BQ Scheduler comes into play when the MQ-BQ Scheduler selects BQs as the next transmitting queue type. This scheduler finally selects one BQ for transmission. There is no limit on the number BQs that can exist in the system. In the worst case, there can be as many BQs as the total number of frames that are currently enqueued in the system – employing round robin scheduling among BQs in such a case would be very inefficient. Thus, each BQ is assigned a certain priority and priority based scheduling is used among BQs.

Each BQ maintains a priority value π_i. The higher the priority value, the higher is the preference given to the corresponding BQ. We use a simple algorithm to calculate the priority of a given BQ: For a BQ_i, whose HOL unicast frame had been queued in the system for t_i seconds and had r_i retry attempts, the priority value π_i is given by

$$\pi_i = (R+1) - r_i + t_i \ , \tag{1}$$

where R is the maximum number of allowed retry attempts. According to the IEEE 802.11 retransmission procedure, R can be 7 if the frame size is less than 3000 B, or 4 otherwise. From Equation (1) it is clear that priority π for a BQ increases as the queuing time for the HOL unicast frame increases and it decreases as the retry attempt count increases. This mechanism also prevents starvation of a BQ.

We further augment the priority scheduling by increasing the priority of BQs of destinations that might be back in radio contact. For every received frame, e.g., a periodically sent Cooperative Awareness Message (CAM) as defined in the ETSI ITS-G5 standard [14], the priority is further increased by 0.4 units. While such a received broadcast does not guarantee a successful transmission in the other direction (i.e., the channel might be asymmetric), it is still a good indicator.

Lastly, upon a successful transmission of a unicast frame Equation (1) resets the priority to yield the channel to a potential unicast frame to another destination. If, on the other hand, all retries failed, the BQ is flushed completely and the MAC notifies higher layers about the permanent failure.

IV. Analytical Evaluation

In this section we study the queuing times of the frames in a very simple scenario. For the queuing time calculations, we consider OFDM PHY parameters corresponding to 10 MHz channels as specified in the IEEE 802.11 standard [15]. We also assume that the optional virtual carrier sensing mechanism via RTS/CTS handshake is disabled. We further assume each data frame contains 3200 bit application data and a 256 bit header summing up to a frame size of 3456 bit. The time to transmit the data frame t_{data} can be calculated as

$$t_{\text{data}} = T_{\text{preamble}} + T_{\text{signal}} + \left\lceil \frac{16 + l + 6}{N_{\text{DBPS}}} \right\rceil T_{\text{sym}} \tag{2}$$

$$= 32\,\mu\text{s} + 8\,\mu\text{s} + \left\lceil \frac{16 + 3456 + 6}{48} \right\rceil 8\,\mu\text{s} = 624\,\mu\text{s}. \tag{3}$$

For an acknowledgement (112 bit), this sums up to $t_{\text{ACK}} = 64\,\mu\text{s}$. The waiting time for ACK is calculated as

$$t_{\text{ACKwait}} = t_{\text{SIFS}} + t_{\text{slot}} + t_{\text{rx-delay}} \tag{4}$$

$$= 32\,\mu\text{s} + 13\,\mu\text{s} + 49\,\mu\text{s} = 94\,\mu\text{s}, \tag{5}$$

where $t_{\text{rx-delay}}$ represents the minimum waiting time for an ACK to be 'on the air'. All lengths are taken from the IEEE 802.11 standard.

Now, suppose we enqueue three frames in the MQ of `AC_BE`: First, two unicasts U_1 and U_2, then one broadcast B_1. Further, the destination of U_1 shall have moved out of the communication range, so all its transmission attempts fail.

As a baseline, let us consider the three frames to be transmitted by the regular IEEE 802.11 EDCA system. First, U_1 will be transmitted a total of 8 times before any other transmission is attempted. The backoff window starts with a size of 15 and increases with every failed transmission. Starting from the time when the system waits for an Arbitration Interframe Space (AIFS) until the first transmission (including time for sending the frame, waiting for an ACK, and time spent for backoffs) it takes, on average (using CW/2),

$$\begin{aligned} t_{U_1} = 8\big(&t_{\text{AIFS}} + t_{\text{data}} + t_{\text{ACKwait}}\big) + \\ &\big(7.5 + 15.5 + ... + 511.5\big) t_{\text{slot}} = 26\,436\,\mu\text{s}. \end{aligned} \tag{6}$$

Afterwards, the system resets the contention window to the minimum and waits for channel access. Thus, sending U_2 takes

$$t_{U_2} = t_{\text{AIFS}} + 7.5 t_{\text{slot}} + t_{\text{data}} + t_{\text{SIFS}} + t_{\text{ACK}} = 927.5\,\mu\text{s}. \tag{7}$$

Now, at time 27 363.5 µs, the contention window is again set to the minimum and B_1 is transmitted. Thus, sending B_1 takes

$$t_{B_1} = t_{\text{AIFS}} + 7.5 t_{\text{slot}} + t_{\text{data}} = 831.5\,\mu\text{s} \tag{8}$$

and concludes at time 28 195 µs.

For comparison, let us consider the same frame order to be transmitted by our QQDCA system. Following the protocol in Algorithm 1, the example scenario progresses as follows. Initially, all the frames are enqueued in the MQ. The MQ-BQ Scheduler selects the MQ for the first transmission. Following the AIFS and one backoff, the first failed transmission of U_1 concludes, on average, at

$$t_{U_1} = t_{\text{AIFS}} + 7.5 t_{\text{slot}} + t_{\text{data}} + t_{\text{ACKwait}} = 925.5\,\mu\text{s}. \tag{9}$$

Now, a new BQ is created and U_1 is moved to the BQ. The contention window is doubled to 31 and the system enters backoff. The round robin MQ-BQ Scheduler schedules a BQ for the next transmission. The BQ Scheduler selects the only

BQ available, which is holding U_1. So, the first retry for U_1 takes, on average,

$$t'_{U_1} = t_{\text{AIFS}} + 15.5t_{\text{slot}} + t_{\text{data}} + t_{\text{ACKwait}} = 1029.5\,\mu\text{s}. \quad (10)$$

Thus, the first retry for U_1 concludes at time 1955 µs. The contention window is doubled again to 63. This time, the MQ-BQ Scheduler selects the MQ for the next transmission and transmits U_2. Successfully transmitting U_2 takes, on average,

$$t_{U_2} = t_{\text{AIFS}} + 31.5t_{\text{slot}} + t_{\text{data}} + t_{\text{SIFS}} + t_{\text{ACK}} = 1239.5\,\mu\text{s}. \quad (11)$$

U_2 is dequeued from the MQ at time 3194.5 µs. The system resets the contention window to minimum. MQ-BQ Scheduler and BQ Scheduler schedule the BQ for a second retry of U_1, which fails again. This takes, on average,

$$t''_{U_1} = t_{\text{AIFS}} + 7.5t_{\text{slot}} + t_{\text{data}} + t_{\text{ACKwait}} = 925.5\,\mu\text{s}. \quad (12)$$

Now, at $t = 4120\,\mu$s, the contention window is doubled again to 31. This time, the MQ-BQ Scheduler selects the MQ and transmits B_1. Sending B_1 takes, on average,

$$t_{B_1} = t_{\text{AIFS}} + 15.5t_{\text{slot}} + t_{\text{data}} = 935.5\,\mu\text{s}. \quad (13)$$

At time 5055.5 µs, the only frame present in the system is U_1. As the MQ is empty, the schedulers always select BQ for the transmissions and, after another 5 retries, the failed U_1 is deleted from the system.

In QQDCA, on average, the queuing time for the successful unicast U_2 is 3194.5 µs and for the broadcast B_1 is 5055.5 µs. For the broadcast B_1, this is an improvement of 82.07 %.

V. Performance Evaluation

To evaluate the performance of QQDCA in comparison to regular IEEE 802.11p EDCA, we modeled the system in the popular vehicular network simulation toolkit Veins [16]. We validated that the BQs in QQDCA are created and destroyed as described in the protocol and no frame reordering occurs.

A. Metrics

We selected the following metrics for comparing our QQDCA to a regular EDCA system.

The *Broadcast Queuing Time* is calculated for every broadcast frame; it indicates the total time that this frame spent in the transmit queue. It is thus one of the most interesting metrics for evaluating the impact of the HOL blocking problem.

The *End-to-end Delay* is calculated for every successful unicast transmission; it indicates the time difference between frame generation by an upper layer and its successful reception at that same layer. It thus includes the queuing time and the transmission time. For our QQDCA system, this metric also helps evaluating if any of the BQ queues is being starved.

The *Channel Utilization* is calculated every 1 s by each node; it indicates the fraction of this time for which the wireless channel was sensed busy by this node.

The *Broadcast Lost Ratio* is calculated for every broadcast; it indicates the fraction of vehicles in communication range which could not decode this broadcast.

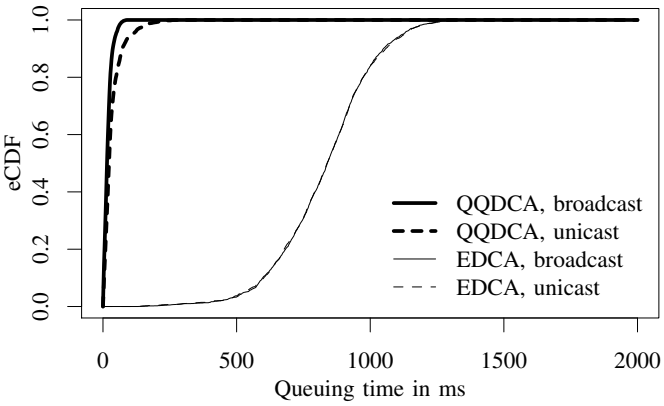

Figure 2. eCDF showing the queuing time of both broadcast and unicast in QQDCA and IEEE 802.11p EDCA in system validations.

The *Unicast Failure Ratio* is calculated for all unicast transmissions; it indicates the fraction of frames that were not transmitted successfully.

B. Small Scale System Validation

We first simulated a very simple scenario in which an application produces bursts of 10 frames every 0.1 s. The generated frames were of three types: broadcast, unicast (50 % loss probability), and unicast (100 % loss). The frame type for each frame was selected uniformly at random.

The eCDF of measured queuing times for both broadcast and unicast frames is shown in Figure 2. There is no discernible difference between both types of frames when relying on standard EDCA, confirming that both suffer similarly from HOL blocking. Conversely, QQDCA successfully moves failed unicast frames to a BQ, confirming that the additional queueing time they spend there does not influence queuing times of broadcast frames – nor that of other unicast frames, leading to overall much lower queueing times. The mean queuing times for successful unicasts and broadcasts in IEEE 802.11p using EDCA compared to our improved QQDCA were as follows: Unicast frame queuing time improved from 829 ms to 33 ms, that of broadcast frames to 18 ms – an improvement of 96 % and 97 %, respectively.

C. Scenario for Large Scale Experiments

We also conducted simulations in the Luxembourg city scenario [17]. An overview of this scenario – a well-frequented part of the city spanning approximately 2 km^2 with roads of varying length and size as well as with many buildings blocking radio communication – is shown in Figure 3. We use free space path loss model along with obstacle model [18] to realize shadowing effects on the radio signal due to buildings. We also conducted simulations in additional highway and Manhattan Grid scenarios, but do not discuss them in this paper; in brief, they confirm our findings.

In all scenarios, each vehicle has three or four applications running (cf. Figure 4): App0 maintains a neighbor table, App1 is sending broadcasts, App2 is sending unicasts to direct

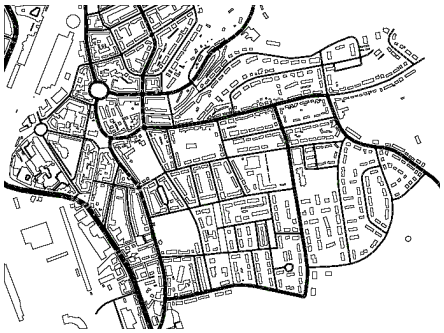

Figure 3. The Region of Interest (ROI) of approx. 2 km^2 in the Luxembourg city scenario.

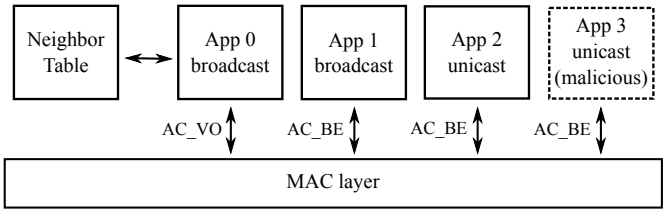

Figure 4. Experiment Setup – App0 periodically sends beacons to establish a neighbor table. App1, App2, and App3 share the same AC (AC_BE) to study the HOL blocking effects. App3 maliciously generates unicast with destinations not in communication range.

neighbors, and an optional App3 is sending unicasts to non-existing destinations. App3 thus simulates a malicious user or application (on-board or off-board), which triggers frames causing HOL blocking intentionally; we use it to study the robustness of QQDCA against explicitly causing HOL blocking by exploiting the denial of service vulnerability of IEEE 802.11p [7]. The baseline experiment is without App3, where the blocking effect is mainly due to frame collisions or outdated entries in the neighbor table.

The packet generation interval was 0.3–2 s. Like in the small scale system validation, all apps collectively generate a total of 10 frames in each of these generation intervals. The order in which these apps generate frame is uniformly distributed.

For maintaining the neighbor table that is used for targeting unicast frames, App0 periodically (every second) sends beacons using access category `AC_VO`. Beacon reception triggers the storing of a new neighbor. Since App0 sends beacon every second, any entry which becomes older than 2 seconds in the neighbor table is considered stale and is deleted. For the time period when an entry in neighbor table is stale but not yet removed, any unicast transmission to that destination will fail.

On an average, approx. 500000 packets were transmitted in a single simulation run of 30 s. We repeated each experiment 20 times with different seeds for statistical validation and identification of any possible outliers. All simulation parameters are summarized in Table I. The error bars in the plots indicate 5th and 95th percentiles. We also calculate confidence intervals which are extremely small.

Table I
SIMULATION PARAMETERS.

Path loss model	Free space ($\alpha = 2.0$)
Obstacle shadowing model	Buildings (cf. [18])
Frequency	5.89 GHz
Data rate	6 Mbit/s
Transmit power	20 mW
CCA threshold	−65 dBm
Receiver sensitivity	−89 dBm
PHY model	IEEE 802.11p
MAC models	IEEE 802.11p EDCA and QQDCA
Transmit queue size	60
AC beaconing	AC_VO
AC app data	AC_BE
App data size	3200 bit
Beacon data size	2400 bit
Beacon rate	1 Hz
Unicast retries	7
Simulation time	30 s

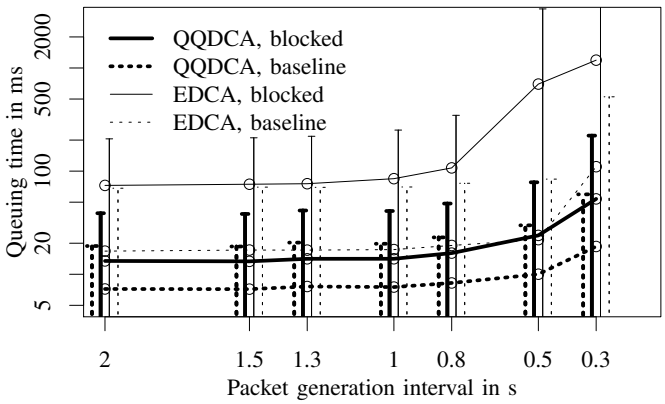

Figure 5. Queuing time of the broadcast frames.

D. Broadcast Queuing Time

In Figure 5, we plot the mean queuing time for all broadcast messages. The 'baseline' and 'blocked' labels in the graph represent absence and presence of the malicious app, respectively. A first obvious observation is that QQDCA shows a huge improvement in the broadcast queuing time compared to the standard IEEE 802.11p EDCA. This is because the failed unicast frames are removed from the MQ and enqueued into the BQ corresponding to the destination address. This avoids HOL blocking on the MQ, thereby permitting the MQ to transmit the trailing broadcast frames quickly.

In the blocking case, we observe relatively higher queuing times compared to that of the baseline. This is due to the continuously failing unicasts causing frequent retransmissions. For the smallest frame generation interval of 0.3 s in the blocking case, the mean queuing time of IEEE 802.11p EDCA is 1190 ms whereas QQDCA shows only 53 ms leading to a substantial improvement of 95 %. For the same interval in baseline, the queuing times are 110 ms and 18 ms for IEEE 802.11p EDCA and QQDCA, respectively – an improvement of 83 %.

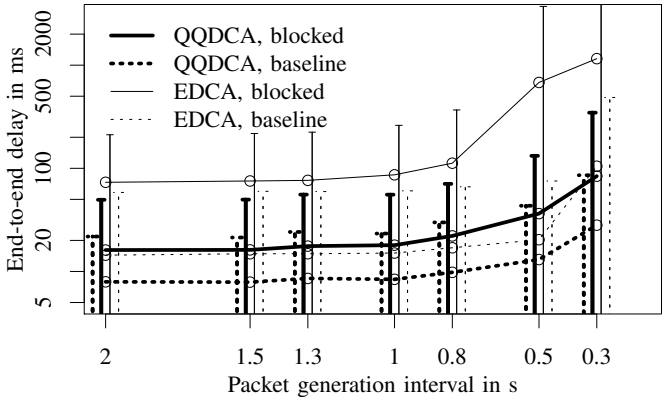

Figure 6. End-to-end delay for the unicast frames.

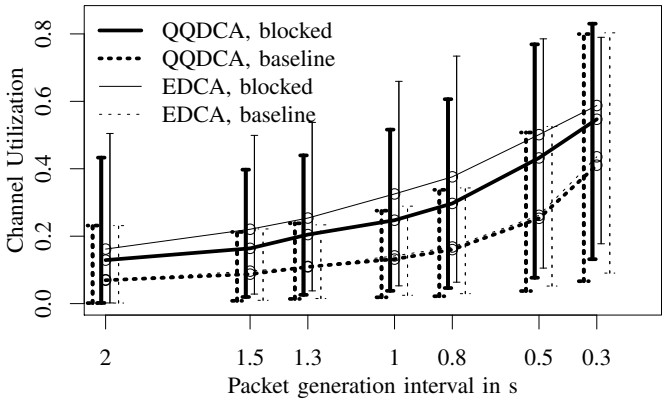

Figure 7. Channel utilization.

E. End-to-end Delay

Figure 6 plots the mean end-to-end delay for the successfully transmitted unicast frames. The HOL blocking affects not only broadcast frames but also all unicast frames in transmit queue. Therefore, the highest end-to-end delays are observed for the unicast frames in the blocking case for IEEE 802.11p EDCA. In QQDCA, the failed unicast frames are moved to per-destination BQs. Therefore, poor quality of the link between source and one destination does not affect the trailing unicast frames targeted to other destinations.

For the smallest frame generation interval of 0.3 s in the blocking case, the mean end-to-end delay of IEEE 802.11p EDCA is 1156 ms whereas QQDCA shows only 83 ms – a 92 % improvement. For the same interval in baseline, the mean end-to-end delays are 104 ms and 28 ms for IEEE 802.11p EDCA and QQDCA, respectively – an improvement of 73 %. We can also conclude that priority scheduling by the BQ Scheduler helps avoid starvations.

F. Channel Utilization

Studying channel utilization gives us an overall picture of the scenario. Under high channel utilizations, the number of collisions increases, thereby reducing the overall efficiency of the network. Figure 7 plots the channel utilization for the studied cases. Comparing blocking with baseline, we see typically higher channel utilization for the blocking case. This is because of the presence of the malicious app: its generated unicast frames always need retransmissions. This increases the total number of transmissions and, hence, contributes to the channel load.

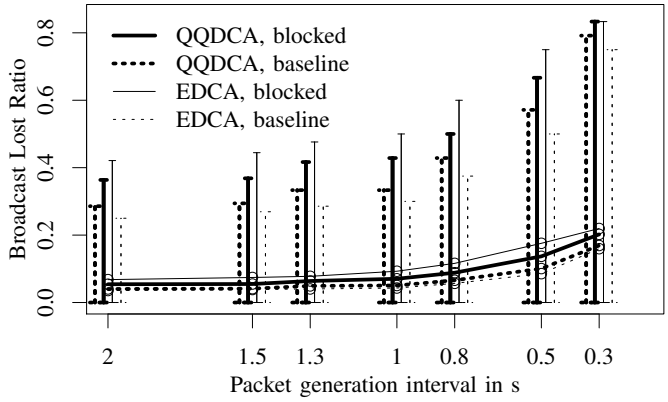

Figure 8. Broadcast lost ratio.

The IEEE 802.11p EDCA standard shows a higher channel utilization compared to QQDCA in the blocking case. This is because QQDCA flushes the BQ, removing all frames waiting for the same destination, once the retry limit for a unicast frame is reached. In the baseline case, where the frame loss is due to interference or vehicles moving out of communication range, the channel utilization is similar for both systems.

G. Broadcast Lost Ratio

Figure 8 illustrates that, for the blocking case, we observe that the broadcast lost ratio is lower for QQDCA. This can be related to the observed channel utilization.

For the baseline, we observe a reversal of this trend. In this specific constellation, QQDCA exhibits slightly worse performance than regular IEEE 802.11p EDCA. This increase in broadcast lost ratio is because of round robin scheduling being employed by the MQ-BQ Scheduler in QQDCA. As discussed before, the system tends to stay in smaller contention windows by resetting the contention window to minimum. With several vehicles in an area – and all of them tending to stay in lower contention windows – the probability of collisions increases. For the IEEE 802.11p EDCA standard, the system tries to retransmit repeatedly for 7 times (always doubling the contention window), thus increasing the probability of longer random backoffs before the next channel access. While such longer backoffs cause HOL blocking on the only available transmit queue, they also lower broadcast lost ratios. Thus, slightly more so than in regular EDCA, for QQDCA it is recommended that lost broadcast frames be recovered where needed, e.g., using rebroadcasting techniques [19] of emergency messages. Further, modern higher layer standards [20] commonly mandate that some frames be retransmitted irrespective of success or failure to reach higher frame delivery rates.

H. Unicast Failure Ratio

Finally, we study the unicast failure ratio, which serves as another reliability metric for the system. The plot is shown

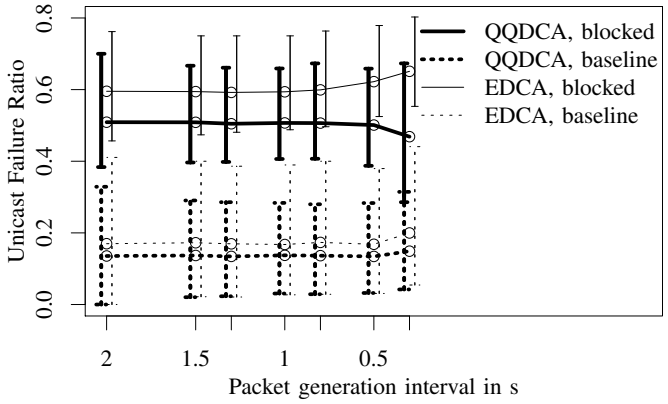

Figure 9. Unicast failure ratio.

in Figure 9. In general, the observed unicast failure ratio for QQDCA is lower compared to IEEE 802.11p EDCA. There are two opposing factors contributing to this behavior. The first one is the same as discussed for the broadcast lost ratio, i.e., the nodes tend to stay in lower contention window values in QQDCA. This factor contributes to higher collisions and consequently higher failures. However, the flushing of the BQ once the retry limit has reached for its HOL unicast frame prevents the system from trying to retransmit unicasts to the destinations which have already moved out of the communication range – thus yielding better performance of our QQDCA system.

VI. Conclusions

In this paper, we presented Multi-Queue Distributed Channel Access (QQDCA), which enhances IEEE 802.11p Enhanced Distributed Channel Access (EDCA) with overhauled transmit queuing to solve the head-of-line (HOL) blocking problem on the transmit queues.

QQDCA is capable of creating on-demand backup transmit queues per-destination to cover failed unicast frames. Such failed unicast frames are dequeued from the Main Queue (MQ) and enqueued into a per-destination Backup Queue (BQ) before any retransmissions. This results in the MQ never getting blocked due to unicast failures, thereby making our QQDCA robust to unicast failures, which are prominent in Vehicular Ad Hoc Network (VANET) scenarios.

We compared our QQDCA system to IEEE 802.11p EDCA in different situations and in a variety of scenarios. Under the evaluated conditions, our results show that the queuing time of broadcasts can be reduced by 83–95 %. At the same time, the unicast end-to-end delay can be reduced by 73–83 %. This improvement comes at very little cost in terms of performance and can be realized completely in software as all the queues are handled in the upper MAC. In summary, we can conclude that by employing QQDCA at the MAC layer, we can alleviate the HOL blocking problem without having any negative impacts on system performance.

References

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

[2] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality," ETSI, Tech. Rep. 302 636-4-1 V1.2.1, Jul. 2014.

[3] 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, Mar. 1996, pp. 1133–1140.

[4] C. Fragouli, V. Sivaraman, and M. B. Srivastava, "Controlled multimedia wireless link sharing via enhanced class-based queuing with channel-state-dependent packet scheduling," in *17th IEEE Conference on Computer Communications (INFOCOM 1998)*, San Francisco, CA, Mar. 1998.

[5] E. Rozner, A. P. Iyer, Y. Mehta, L. Qiu, and M. Jafry, "ER: Efficient Retransmission Scheme for Wireless LANs," in *13th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2017)*. New York, NY: ACM, Dec. 2007.

[6] C.-S. Hwang and J. M. Cioffi, "Opportunistic CSMA/CA for achieving multi-user diversity in wireless LAN," *IEEE Transactions on Wireless Communications*, vol. 8, no. 6, Jun. 2009.

[7] 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.

[8] L. B. Jiang and S. C. Liew, "An adaptive round robin scheduler for head-of-line-blocking problem in wireless LANs," in *IEEE Wireless Communications and Networking Conference (WCNC 2005)*. New Orleans, LA: IEEE, Mar. 2005.

[9] V. Kolar, S. Tilak, and N. B. Abu-Ghazaleh, "Avoiding head of line blocking in directional antenna [MAC protocol]," in *29th IEEE Conference on Local Computer Networks (LCN 2004)*. Tampa, FL: IEEE, Dec. 2004.

[10] J. Wang, H. Zhai, and Y. Fang, "Opportunistic packet Scheduling and Media Access control for wireless LANs and multi-hop ad hoc networks," in *IEEE Wireless Communications and Networking Conference (WCNC 2004)*. Atlanta, GA: IEEE, Mar. 2004.

[11] 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, to appear.

[12] J. Wang, H. Zhai, Y. Fang, and M. C. Yuang, "Opportunistic media access control and rate adaptation for wireless ad hoc networks," in *IEEE International Conference on Communications (ICC 2004)*. Paris, France: IEEE, Jun. 2004.

[13] P. Liu, Z. Tao, S. Narayanan, T. Korakis, and S. S. Panwar, "CoopMAC: A Cooperative MAC for Wireless LANs," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 340–354, 2007.

[14] 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.

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

[16] 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.

[17] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: 24 Hours of Mobility for Vehicular Networking Research," in *7th IEEE Vehicular Networking Conference (VNC 2015)*. Kyoto, Japan: IEEE, Dec. 2015.

[18] C. Sommer, D. Eckhoff, R. German, and F. Dressler, "A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Environments," in *8th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2011)*. Bardonecchia, Italy: IEEE, Jan. 2011, pp. 84–90.

[19] N. Wisitpongphan, O. K. Tonguz, J. S. Parikh, P. Mudalige, F. Bai, and V. Sadekar, "Broadcast Storm Mitigation Techniques in Vehicular Ad Hoc Networks," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 84–94, Dec. 2007.

[20] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specification of Decentralized Environmental Notification Basic Service," ETSI, EN 302 637-3 V1.2.0, Aug. 2013.