An Approach for Receiver-Side Awareness Control in Vehicular Ad Hoc Networks

Víctor Díez Rodríguez, Jérôme Detournay, Alexey Vinel, *Senior Member, IEEE*, and Nikita Lyamin

***Abstract*— Vehicular Ad hoc networks (VANET) are a key element of cooperative intelligent transport systems. One of the challenges in VANETs is dealing with awareness and congestion due to the high amount of messages received from the vehicles in communication range. As VANETs are used in critical applications, congestion on the receiver side caused by the buffering of the packets is a safety hazard. In this paper, we propose a streamwise queuing system on the receiver side and show how it improves the timeliness of the messages received and maintains the awareness of the system in a congestion situation.**

***Index Terms*— GCDC 2016, ETSI ITS-G5, vehicular communication, awareness control, priority queue.**

I. Introduction

VEHICULAR Ad-Hoc Network (VANET)s are a key element of Cooperative Intelligent Transport Systems (C-ITS). VANETs are meant to provide Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication capabilities in order to support a wide variety of C-ITS applications. These applications range from entertainment (e.g. Internet access, video streaming) to safety (e.g. collision avoidance) and efficiency (e.g. platooning). Cooperative driving applications rely on knowing the status of the surrounding vehicles. To make this possible, each vehicle broadcasts messages containing information such as its position, velocity, heading, etc. The messages can be broadcasted periodically or be generated based on certain events.

Multiple challenges arise in the operation of VANETs due to the wide variety of applications that make use of the communication systems and the highly dynamic topologies of the network. Each kind of application has very different Quality-of-Service requirements. While video streaming applications might need high bandwidth, for safety applications it is essential to guarantee low latency communication. Furthermore, the range of possible applications coexisting together makes it necessary to guarantee that low priority applications, such as entertainment applications, do not affect the performance of safety critical applications. The ETSI standard ITS-G5 [1] partly addresses this problem at the communication level by defining two classes of channels: control channel for safety critical data traffic and services channels for other traffic.

On the other hand, the number of vehicles in communication range can vary greatly depending on the traffic situation. For example, a rural road might contain only a few vehicles within communication range, while that number can be in the order of hundreds in a multi-lane highway. Broadcasting messages too frequently or with an exceedingly high transmission power on a busy road can cause congestion in the channel. Broadcasting messages rarely might not provide the awareness level required for some safety applications. It is important to optimize the use of the communication channel by maximizing the awareness level without causing congestion. Congestion Control (CC) and Awareness Control (AC) algorithms [2] modify application and communication parameters such as message transmission rate and transmission power to ensure efficient communication among neighbouring vehicles.

A. Motivation

Cooperative driving applications rely greatly on the performance of the wireless communication systems used to support information exchange among the vehicles. While each vehicle is in control of the transmission of its own information, the information received depends on the channel conditions and on the other users of the channel.

CC algorithms are used to avoid overloading the channel and AC algorithms are used to optimize the use of the channel by broadcasting the information that might be useful for other cooperative applications. However, it is not obvious how to determine what information is useful for every specific vehicle. Furthermore, most simple algorithms use only the status of the own vehicle as input, therefore they are not able to react directly to the changes in its surroundings. On the other hand, cooperative algorithms that use information about the other users will succeed in maintaining the congestion level in the channel while transmitting the maximum amount of information possible, but usually they require some adaptation time. On the other hand, AC on the receiver side could identify which information is useful for the own system at any given time.

The use of CC and AC algorithms allows to (1) maximize the amount of traffic that is delivered over the channel, and (2) provide a fair use of the channel for each user when the channel is not able to hold more traffic. While this is optimal from the channel utilization point of view, due to the broadcast nature of the V2V communication, it also entails

Manuscript received December 9, 2016; revised April 13, 2017 and June 21, 2017; accepted June 27, 2017. Date of publication October 2, 2017; date of current version March 28, 2018. The Associate Editor for this paper was N. van de Wouw. *(Corresponding author: Víctor Díez Rodríguez.)*

The authors are with the School of Information Technology, Halmstad University at Sweden, Halmstad, Sweden (e-mail: victor@diezrodriguez.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2017.2749966

that the amount of information that a vehicle receives per time unit will increase with the number of vehicles until it reaches the channel limit. Then, if we assume that not every vehicle in communication range is relevant for the purpose of a specific cooperative application, we can conclude that as the number of vehicles increases, the ratio of relevant information will decrease. As pointed out in [2], the number of vehicles in communication range in a multi-lane highway can go up to 300. Of these 300, the messages sent by more than half of them will be almost irrelevant due to the distance and driving direction. In [3], it is estimated that, using the current standard on transmission rate, a vehicle will easily receive up to 500 Cooperative Awareness Message (CAM)s per second, depending on the penetration rate of VANET technology. In the same document, the authors conclude that the large amount of information received by a vehicle will present a challenge in the future.

Thus, it is necessary to establish a reliable way to filter the useful messages from all the information received by the vehicle before it reaches the application layer. This becomes even more important when the applications are running on constrained hardware and with limited resources. Processing unnecessary messages means that the processing of essential information may be delayed, which is unacceptable for safety critical applications based on real-time data, such as collision avoidance and platooning. This kind of undesired behaviour was observed when running the ITS-G5 stack proposed at [4], which serves the messages in a First-In-First-Out (FIFO) manner, during the development work done by Team Halmstad for Grand Cooperative Driving Challenge (GCDC) 2016 [5]. In a FIFO queue, messages are buffered and served as soon as resources are available. When the system is overloaded, the messages served first are already old and, therefore, useless, while the new version of the same information waits on the back of the queue.

The contribution of this paper is threefold:

1) Stream-wise Accumulating Priority Queue (SAPQ), the method aiming to enhance Cooperative Awareness in ETSI ITS-G5 communications is proposed.
2) The performance of SAPQ is evaluated in typical mobility scenarios. It is shown that it significantly decreases the data-age of high priority messages while still keeping low data-age levels for low-priority message in both dense and sparse mobility scenarios.
3) Performance of the SAPQ is also verified on the measurement-based data obtained during GCDC competition.

In comparison to existing literature which mostly concentrates either on the IEEE 802.11p broadcast channel congestion control (CC) or sender side awareness control (AC), our focus is on the receiver side awareness provisioning. Our SAPQ approach is in line with the proposals from Breu and Menth [6], [7], but implies stream-level filtering in contrast to packet-level one, what enables fair service of the messages coming from different vehicles. In addition, we are the first who support our simulation modeling study with the experimental results obtained during the GCDC 2016 test of cooperative driving applications.

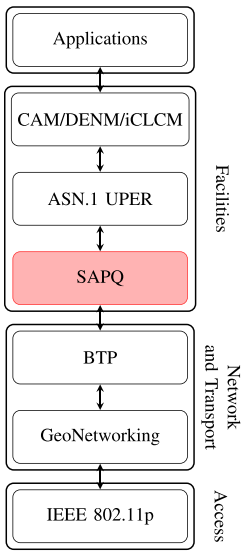

Fig. 1. SAPQ in the ITS-G5 protocol stack.

Figure 1 shows the position of SAPQ in the ETSI ITS-G5 protocol stack.

II. Background

A. Grand Cooperative Driving Challenge

GCDC is a European contest where teams from different European universities competed against each other for best performance in cooperative and autonomous driving. In 2016, the second edition of the competition was celebrated in Helmond, in the Netherlands. The teams were required to implement interaction protocols to perform complex maneuvers autonomously with the help of Vehicle-to-Vehicle (V2V) communication. The scenarios executed during the competition are explained in detail in [8].

The V2V communication used in the competition was based on ETSI ITS-G5. All the traffic was transmitted in the G5-CCH channel centered on 5.9 GHz, with a channel spacing of 10 MHz and a data rate of 6 Mbit/s. Decentralized Congestion Control (DCC) was not required during the competition and was not used during this work. GeoNetworking [9] and Basic Transport Protocol (BTP-B) [10] were used on the network and transport layers. Messages were encoded and decoded using ASN.1-UPER [11]. In the competition CAM [12], Decentralized Environmental Notification Message (DENM) [13] and i-GAME Cooperative Lane Change Message (iCLCM) were used to send vehicle status and maneuver information.

B. Related Work

Extensive research has been made in the communication area in order to increase the channel utilization and reduce the amount of traffic required to provide cooperative driving applications with enough information to maintain a minimum safety level. Most techniques described here work at the Medium Access Control layer by modifying parameters such as power level and transmission rate depending on the status of the channel and/or the vehicle itself.

In [2], a extensive survey of different CC and AC methods is made. Most algorithms modify the transmission rate and transmission power to obtain the desired awareness level and avoid congestion of the channel using metrics such as channel busy time ratio and channel load for CC, and latency, dissemination area and reliability for awareness control. Sepulcre *et al* [2]. classify CC methods into reactive CC and proactive CC.

Reactive CC uses information about the current status of the channel to modify the parameters of the communication, which means that they start working after the congestion has been produced and therefore are not suitable for safety critical applications. Proactive CC uses communication models to estimate the parameters that provide congestion-free communication, however it is shown that creating accurate estimates for every possible situation can be very difficult and require precise information about the applications that use the communication system. Furthermore, the control algorithm used can be either open loop, making them dependent on the accuracy of the system model, or closed loop, thus creating communication overhead to transmit the feedback data.

Lyamin *et al.* [14], show how the message triggering rules used in the current CAM ETSI standard are prone to create channel congestion in platooning scenario. It exhibits the difficulty of designing an effective method for CC and AC and how a transmission rate approach can be counterproductive.

In [15], it is shown how prioritization of multiple traffic classes on the same channel can improve the awareness level in cooperative applications. Bohm *et al.* [15] use the DENM dissemination delay and CAM up-to-dateness to evaluate the performance of multiple combinations of priority levels.

A different approach is taken in [16], where the authors propose the use of a collision-free MAC scheme in a Service Channel to provide reliable intra-platoon communication. The paper shows how the proposed mechanism performs better than the standard approach in terms of CAM up-to-dateness and allows for a higher CAM transmission rate and higher number of vehicles while keeping a reasonable DENM dissemination delay.

Barradi *et al.* [17] show how the use of strict priorities can improve the transmission delay of safety critical traffic. In contrast, the current IEEE 802.11p standard uses "soft" priorities which gives a relative advantage to a traffic class over the other, and can result in lower priority traffic being transmitted before higher priority traffic.

In [6], a function to calculate the relevance of the received CAMs is presented. This function estimates the potential of collision of a vehicle using the relative position and movement of the vehicles involved and assigns them a relevance level based on the received CAMs.

[7] evaluates several relevance estimation functions and describes a mechanism to buffer and select CAMs. The proposed mechanism uses a priority queue to select the messages with higher relevance. Relevance aging is applied in order to give preference to newer messages over older ones.

Most research done regarding network queuing and scheduling is not specific to vehicular communication but to general networks. The literature in this topic describes scheduling and buffering algorithms to handle traffic in networks that carry packets with very diverse requirements. Such algorithms are meant to be implemented in routing and switching equipment in order to provide different Quality-of-Service guarantees to the users of the network.

Buffering policies to reduce the number of high priority packets dropped are introduced in [18]. In this case, the difference between high priority and low priority traffic is the desired loss probability. The idea behind the protective algorithms presented in the paper is that a system carrying both low and high priority traffic should not lose more high priority packets than a reference system carrying only high priority traffic. At the same time, the authors present and evaluate new policies that reduce the loss of low priority traffic while maintaining the optimal performance for high priority traffic.

Duplicate Scheduling with Deadlines [19] is used to guarantee a low bounded delay in the transmission of packets sent in a connection carrying traffic with diverse requirements. Marking is used to differentiate between packets that require delivery with low delay from traffic that requires high throughput. Furthermore, the algorithm is designed not to give preference to a specific type of traffic, but to fulfill as best as possible the requirements of each type.

In [20], Kleinrock presents a time-dependent priority queue. In this queue, the priority of each element increases linearly with the time that it has been waiting in the queue. The class of the element determines how fast the priority increases. At any given time, the first element of the queue is the one that has the highest priority. Kleinrock showed that by tuning the factors at which the priority increases, it is possible to obtain any ratio of mean waiting time in the queue for elements of different classes. Stanford *et al.* [21] refer to that queue as Accumulating Priority Queue (APQ) and provide a method to calculate the distribution of the waiting time for each class in a multiclass APQ.

III. Stream-Wise Accumulating Priority Queue

In this section we present SAPQ, a filtering method thought to enhance Cooperative Awareness in ETSI ITS-G5.

A. Overview

The first step in the SAPQ is separating every packet received into streams. Each stream will contain a concrete type of message from a specific vehicle i.e. each message stream s is defined by a tuple (ID, M), where ID identifies a vehicle unequivocally and M the type of messages that the stream carries.

Streams are stored separately in single message buffers. Each buffer contains only the last message received in each stream. This means that if a message has not been dispatched, the arrival of another message in the same stream will drop the message currently stored. This guarantees that even in case of system overload, when it is not possible to serve every message received, the queue will not dispatch messages containing outdated information.

Streams are separated into discrete classes according to the relevance of the information that they contain. This process is explained in detail in section III-B. When the queue is polled,

TABLE I
FINAL STREAM CLASSIFICATION

Vehicle Class \ Message Class	1 (DENM)	2 (CAM)	3 (iCLCM)
1	1	1	2
2	1	2	3
3	2	3	4
4	3	4	4

the instantaneous priority of each stream is calculated using an APQ as described in section III-C. The packet contained in the stream with the highest priority is served.

By maintaining separate streams for each vehicle and message type, it is possible to guarantee that only the newest information will be forwarded to the higher layers. Stream prioritization is necessary to select the most relevant information and provide fair access to each stream.

B. Stream Classification

SAPQ makes use of stream classification in order to prioritize the messages received. The focus of stream classification is to set apart messages that are relevant for our own vehicle from those that do not have much utility to the applications running in our system. Furthermore, some messages may carry safety critical information that must be processed as soon as possible. The difficulty of doing this classification is due to two main causes:

- The queuing system must be application-agnostic: while we might know which kind of applications are currently being developed and used, it is hard to predict future necessities. SAPQ should not rely in data provided by applications in higher layers. Instead, the information used in the message classification should be general to all cooperative driving applications.
- A highly dynamic environment: a major characteristic of VANETs is their high mobility, which causes great variation in the network environment. Not only the radio conditions are affected, but also the relative importance of the vehicles due to the changes in position and speed. Therefore, it is necessary to make short-term estimations to reduce the reaction and adaptation time to these changes.

To cope with these two issues, our system will classify each stream based on two different criteria: type of the messages contained in the stream and relevance of the vehicle that sent the message. Then, a general classification will be obtained by combining these two criteria as shown in table I. Note that streams with lower class have higher priority.

The message class depends on the type of information that the stream carries. As each type of message belongs to a well-known BTP port [22], [23], it is straightforward to classify each message received based on the packet arrival port.

Similarly to how it is done in [24], we will use 2 parameters to assign a class to each vehicle: current distance and minimum distance to the vehicle. The current distance to the vehicle will let us filter out the vehicles that are far away and do not require high priority without doing complex calculations. For vehicles in a certain radius, we will use the current speed and heading of our vehicle and the other vehicle to estimate the minimum distance between both at any given time. This will allow us to tell whether a vehicle is coming towards us or driving away, and even anticipate a collision.

TABLE II
VEHICLE CLASSIFICATION

Current Distance	Minimum distance	Relevance	Class
< 30 m	< 15 m	High	1
< 60 m	< 30 m	Medium	2
< 150 m	> 30 m	Low	3
>= 150 m	—	—	4

We define the position of a vehicle $\mathbf{p_i}$ and its velocity $\mathbf{u_i}$:

$$\mathbf{p_i} = (x_i, y_i) \quad \mathbf{u_i} = (v_i, w_i)$$

In order to simplify the estimation of the position of the vehicle, we assume linear motion with constant speed. Then, the position in function of the time for each vehicle will be given by the following formulas:

$$\mathbf{p_i}(t) = \mathbf{p_i} + \mathbf{u_i} \cdot t$$

Then, the displacement between the two vehicles will be:

$$\begin{aligned}\mathbf{d_{i,j}}(t) &= \mathbf{p_j}(t) - \mathbf{p_i}(t) \\ &= \mathbf{p_j} - \mathbf{p_i} + (\mathbf{u_j} - \mathbf{u_i}) \cdot t \\ &= (x_j - x_i + (v_j - v_i) \cdot t, y_j - y_i + (w_j - w_i) \cdot t)\end{aligned}$$

And the distance:

$$\begin{aligned}&||\mathbf{d_{i,j}}(t)|| \\ &= \sqrt{(x_j - x_i + (v_j - v_i) \cdot t)^2 + (y_j - y_i + (w_j - w_i) \cdot t)^2}\end{aligned} \tag{1}$$

We can calculate the time to minimum distance t_{min} by solving:

$$\frac{\partial ||\mathbf{d_{i,j}}(t)||}{\partial t} = 0$$

Then, the current distance would be determined by:

$$||\mathbf{d_{i,j}}(t_0)||$$

And the minimum distance by:

$$||\mathbf{d_{i,j}}(t_{min})||$$

Threshold values for current distance and minimum distance are used to separate neighbouring vehicles into discrete classes. Table II shows the threshold values used for the evaluation of the system. These values were selected by considering factors such as the width of road lanes and common road scenarios. Each relevance class will contain vehicles with the following characteristics:

- High Relevance: vehicles with a current distance lower than 30 meters that might be closer than 15 meters to our vehicle in another moment. These are vehicles in risk

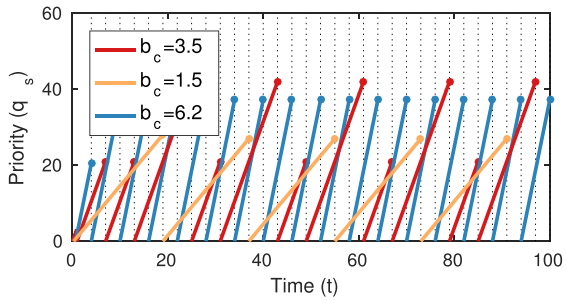

Fig. 2. Priority over time.

of immediate collision, closest platoon members, vehicles overtaking our own vehicle, vehicles in the adjacent lanes.

- Medium Relevance: vehicles with a current distance between 30 and 60 meters that might be between 30 and 15 meters to our vehicle in another moment. This includes medium distance vehicles in the adjacent lanes, vehicles approaching an intersection, closest vehicles from other directions in a highway, vehicles coming from the acceleration lane.
- Low Relevance: vehicles with a current distance greater than 60 meters that might not get closer than 30 meters or our vehicle in another moment. This comprehends all other neighbouring vehicles.

Determining the optimal number of classes and the threshold values is out of the scope of this paper.

C. Priority Calculation

The priority of a stream is calculated using an APQ [20]. In an APQ, the priority of an element at a given instant is the product of the accumulating factor of the element and the time that it has been waiting in the queue. The accumulating factor of a stream is given by its class, and it determines the speed at which its instantaneous priority increases. The first element of the queue is the one with highest instantaneous priority when the queue is polled. Therefore, higher accumulating factor means higher priority. The instantaneous priority of a stream $q_s(t)$ is defined by:

$$q_s(t) = (t - t_s) \cdot b_c \tag{2}$$

Where (b_c) is the accumulating factor of the stream, t is the current time and t_s is the arrival time of the oldest message received in the stream since the last time that it was served.

Figure 2 shows an example of the APQ described previously. The graph shows an APQ containing 3 streams, each with a different accumulating factor. The dotted lines in the background mark each time that the queue is polled. The stream that is served is the one with highest accumulated priority at that moment. Note that the stream enters the queue again just right after it is served.

D. Implementation

Figure 3 shows the class diagram of the main components in our implementation of SAPQ.

The interface to the SAPQ consists of a constructor and two methods:

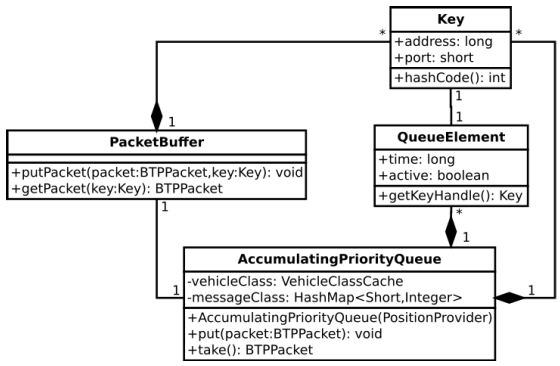

Fig. 3. Class diagram of the SAPQ implementation.

- The constructor takes a PositionProvider, which is used to obtain the current position of our own vehicle necessary for the calculations of the vehicle classification.
- *put()* (algorithm 1) inserts a BTPPacket in the queue. The information that is already contained in the BTP packet is used for its classification i.e. the origin address of the packet, used to identify each vehicle unequivocally, and the BTP destination port, used to identify the type of message.

Algorithm 1 SAPQ Put(BTPPacket)

```
1: Key ← (BTPPacket.address, BTPPacket.port)
2: PacketBuffer[Key] ← BTPPacket
3: if Key in QueueElementMap then
4:   QueueElement ← QueueElementMap[Key]
5: end if
6: QueueElement.position ← BTPPacket
7: if not QueueElement.active then
8:   QueueElement.time ← now
9:   QueueElement.active ← true
10: end if
11: QueueElementMap[Key] ← QueueElement
```

- *take()* (algorithm 2) returns the BTPPacket with highest priority at the instant of the call and removes it from the queue. It is a blocking method and therefore, if the queue is empty, it will block until there is a packet available.

Algorithm 2 SAPQ Take()

```
1: MaxPriority ← 0
2: for QueueElement in QueueElementMap do
3:   Class ← calculateClass(QueueElement)
4:   ActiveTime ← (now − QueueElement.time)
5:   Factor ← AccumulatingFactor[Class]
6:   CurrentPriority ← Factor ∗ ActiveTime
7:   if CurrentPriority ≥ MaxPriority then
8:     MaxPriority ← CurrentPriority
9:     MaxPriorityElement ← QueueElement
10:  end if
11: end for
12: MaxPriorityElement.active ← false
13: return PacketBuffer[MaxPriorityElement.key]
```

When a BTP Packet is added to the queue, a Key made of the packet address and BTP port is created. This Key will be used by different elements of the system to identify the multiple packet streams. Then, the packet is introduced into the PacketBuffer which contains the last packet received from each stream that has not been dispatched. Buffering the packets independently from the queuing makes it possible to modify the buffer size for each stream. However, we will always use a buffer size of 1 in order to reduce the data age of the packets that are dispatched.

At the same time, a QueueElement identified by the stream of the key is created. This QueueElement contains the position of the vehicle that sent the packet and the packet arrival time. QueueElements are the main structure used to determine the stream with highest priority. In order to save resources, QueueElements are created only with arrival of the first packet of the stream. Subsequent arrivals reuse the same object and set an *active* flag to show that there are packets pending from that stream.

The *active* flag is also used to indicate whether it is necessary to update the arrival time of the stream. Note that updating the arrival time while there are packets pending in the stream would reduce the instant priority of that stream, and therefore the arrival time is only updated when the *active* flag is not set.

Calling the *take()* method extracts a buffered BTPPacket from the stream with highest priority. At this moment, the priority of every active stream is calculated using the information in the QueueElements. First, the vehicle class is determined by checking the vehicle class cache. The cache hits if the classification for the vehicle has been updated recently. The amount of time that the classification remains valid can be configured separately for each class.

If a cache miss is produced, the classification of the vehicle is calculated using the approach described in section III-B. Furthermore, in order to reduce the computation time of the vehicle distance, the transformation from the position in WGS 84 to the relative position of the vehicles in meters is done using lookup tables. Although using lookup tables entails certain error in the calculation, this is bounded by the number of entries in the table. Therefore, by using a static lookup table, it is possible to achieve $O(1)$ lookup time and arbitrary accuracy in the transformation.

Next, the message type class is combined with the vehicle class to obtain the final class according to table I. The accumulating factor for the final class is obtained and the instant priority of the stream is calculated by multiplying it by the time that the QueueElement has been active. Then, the Key of the stream with highest priority is used to extract the corresponding packet from the PacketBuffer, the *active* flag in the QueueElement is unset, and the packet is served.

IV. Evaluation

This section presents the performance evaluation of SAPQ in both sparse and dense traffic scenarios. We also show how SAPQ significantly outperforms basic ITS-G5 FIFO approach and provide a quantitative evaluation of the properties of our implementation in both scenarios. Furthermore, we demonstrate that SAPQ supports short packet waiting time in real-traffic scenario by evaluating its performance using the real gathered during GCDC 2016 competition.

TABLE III
Distribution of Vehicle Classes Used in the Simulation

Class	Radius (m)	A_c (m^2)	ΔA_c (m^2)	$p(c)$
1	30	2827	2827	0.01
2	60	11309	8482	0.03
3	150	70685	59376	0.21
4	300	282743	212057	0.75

A. Setup

In order to establish the base for the simulation scenario, some assumptions were made:

1) Each vehicle generates both CAM and iCLCM at a rate of 25 Hz, as required during GDCD 2016. Additionally, DENM bursts are generated randomly with a 0.05 probability i.e. all the cars will transmit DENMs at the same time with a 0.05 chance. In total, each vehicle will transmit 50 messages per second plus DENM bursts.
2) Processing a message takes a fixed amount of processor time, and each message type takes a different amount of time. Processing includes both decoding and application-level use of the information contained in the message. The variety of applications that can make use of the ETSI ITS-G5 stack makes it difficult to give a precise estimation of the processing time of an arbitrary message. However, during GCDC it was observed that processing CAMs took considerably more time than the rest of the messages. In the simulations, we will assume that processing of a CAM takes 5 times more than the other two types of message.
3) Vehicles have a static class. In order to simplify and have more control over the parameters of the simulation, each vehicle has an assigned class. The number of vehicles in each class follows a distribution calculated using the areas (A_c) of the circles corresponding to the current distance limit for each class. Table III shows this distribution. In particular, the probability of a vehicle being in a class ($p(c)$) is given by the relation between the increment of area (ΔA_c) for that class and the total area i.e. the area of the largest class.

$$p(c) = \frac{A_c - A_{c-1}}{A_{total}} = \frac{\Delta A_c}{A_{total}}$$

The accumulating factors for each class were set arbitrarily as shown in table IV. The factors were selected with the intention of showing significant differences among the classes and not with optimality in mind.

The parameters for the simulations are:

1) Number of consumer threads: number of Java threads that are concurrently extracting and processing packets from the queue.

TABLE IV
CLASS ACCUMULATING FACTORS USED IN SIMULATION

Class	Accumulating factor (b_c)
1	8
2	4
3	2
4	1

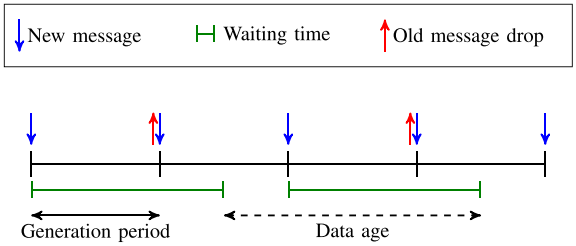

Fig. 4. Relation between packet waiting time and data age.

TABLE V
SIMULATION PARAMETERS USED FOR THE QUEUE COMPARISON

System load	Threads	Processing (ms)	Vehicles
Low	8	0.1 (0.5 for CAMs)	100
High	1	1 (5 for CAMs)	300

2) Processing time: time that takes to process a single packet. CAM messages take 5*b, as explained previously.
3) Number of vehicles: number of vehicles in communication range that are transmitting messages at the same moment.

The main metric recorded during the simulations was the waiting time of a stream in the queue i.e. the time that takes from the reception of a new packet in the stream until a packet from that stream is dispatched to the upper layers. In the following sections, we will use the waiting time values of the individual streams and the waiting time of a class as a whole to evaluate the performance of our system. It is important to note that the waiting time of each stream also correlates with the data age perceived by the application. For example, in the case of periodic messages, the data age will be bounded by the waiting time plus the message generation period as shown in figure 4. In the same figure, it is possible to see that no messages will be dropped as long as the waiting time is smaller than the generation period.

B. Comparison With FIFO

The parameters used for the simulation of each scenario are shown in table V.

Figure 5 shows the performance in terms of packet waiting time of our system and a FIFO queue under low system load. Both systems perform almost identically in the *low system load* scenario by dispatching every packet under the 30 ms mark. The FIFO queue lacks of any kind of traffic classification and handles every packet in the same manner. On the other hand,

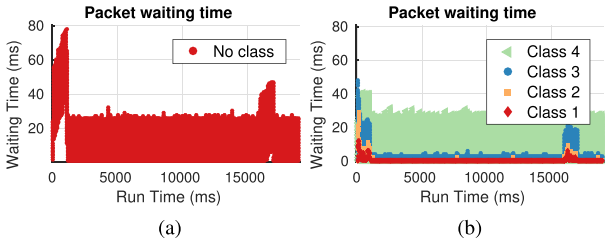

Fig. 5. Waiting time of packets dispatched under low system load. (a) FIFO. (b) SAPQ.

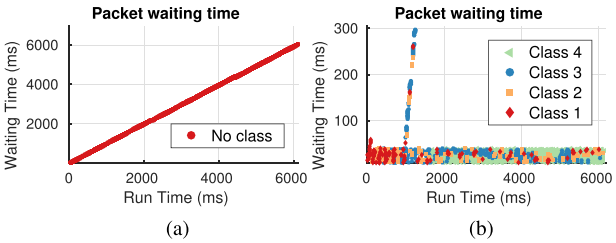

Fig. 6. Waiting time of packets dispatched under high system load. (a) FIFO. (b) SAPQ.

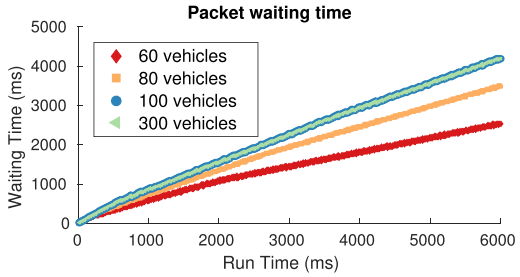

Fig. 7. Waiting time in the FIFO queue as the number of vehicles increases.

the SAPQ is able to dispatch high priority packets much faster than other packets.

However, it is in the *high system load* scenario where the SAPQ shows a completely different behaviour, as shown in figure 6. The FIFO queue buffers every packet, creating longer and longer waiting times when new packets are received. While our system has the same processing capacity in terms of messages per second, it drops packets containing old information in favor of dispatching the last packet available for each stream.

C. Distribution of Message Waiting Time

As explained in [21], the performance of an APQ can be analyzed by looking at the Cumulative Distribution Function (CDF) of the waiting time. To be able to appreciate the capacity of SAPQ to adapt to scenarios with different amount of vehicles, the other parameters were fixed to 4 threads and 0.7 ms of packet processing time. These values are within the ones selected for the high and low system load, and describe a medium system load scenario. Under this conditions, the FIFO queue becomes saturated as the number of vehicles approaches 80, as shown in figure 7.

Figure 8 shows the plot of the CDF of the waiting time for packets dispatched with SAPQ in scenarios with 100 and

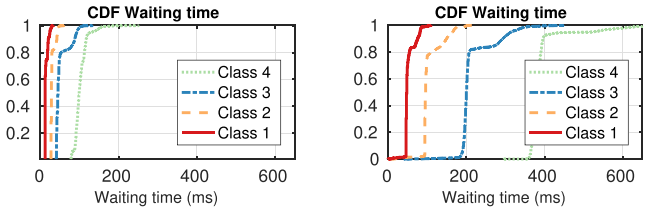

Fig. 8. CDF of the waiting time by class using SAPQ. (a) 100 vehicles. (b) 300 vehicles.

TABLE VI
COMPARISON OF PACKET DROP RATE FOR EACH CLASS AND AMOUNT OF VEHICLES

Vehicles	Class	Drop (%)	$\bar{x}$ (ms)	σ (ms)
300	1	51.08	58.76	26.80
	2	69.94	116.60	53.01
	3	84.23	246.02	91.06
	4	90.74	430.91	118.62
100	1	0.00	15.48	5.12
	2	18.07	31.43	7.36
	3	54.44	55.99	19.95
	4	67.84	105.11	18.85

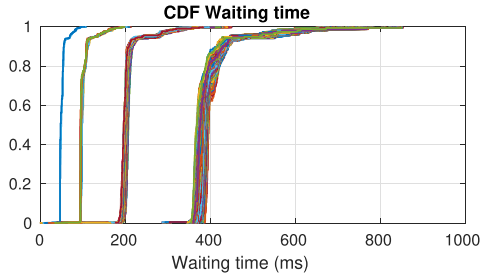

Fig. 9. CDF of the waiting time by stream using SAPQ with 300 vehicles (DENMs not plotted).

300 vehicles. It is easy to see that the waiting time for the scenario with 300 vehicles is longer than the one with 100. For example, 80% of the messages of class 3 are dispatched within 50 ms in figure 8a, while in figure 8b the same percentage requires above 200 ms.

Table VI shows how the packet drop rate and waiting time of each class changes with different amounts of vehicles. For a class 1 stream in a scenario with 300 vehicles, the packet drop rate is around 51%. However, the waiting time for a packet is 58 ms on average which means that, when accounting for a packet generation period of 40 ms, the maximum data age is 98 ms.

In figure 9, we show the CDF of each stream. It is interesting to see that the streams containing periodic messages (CAM and iCLCM) show the same CDF individually than the whole class as a group. Therefore, SAPQ guarantees that every stream from each class is polled equally and it is given its fair share of resources.

D. Examples With Real Data

In this section we will analyze the performance of the SAPQ by using real data gathered during GCDC 2016. The log files containing the messages sent during the last heat of the merging scenario were used. In this heat, 10 vehicles are changing their speed and relative position while sending CAMs, DENMs and iCLCMs. With respect to the system configuration, the same setup described in tables IV and V was used except for the number of vehicles. In these examples, the complete SAPQ is executed, including the vehicle classification algorithm described previously.

TABLE VII
STATISTICS GENERATED BY REPLAYING GCDC 2016 LOGS THROUGH THE SAPQ

System load	Class	Packets	Drop (%)	$\bar{x}$ (ms)	σ (ms)
High	1	849	54	10.49	2.47
	2	3600	66	17.66	5.54
	3	3448	53	33.25	7.81
	4	2102	62	59.77	14.37
Low	1	849	0.35	0.059	0.24
	2	3600	0.39	0.051	0.23
	3	3448	0.23	0.050	0.23
	4	2102	0.18	0.055	0.26

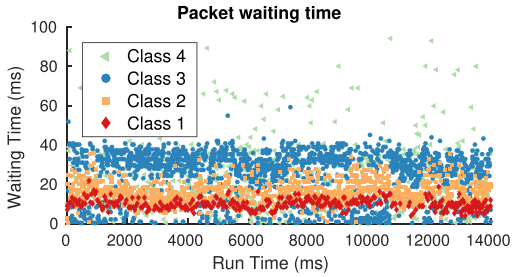

Fig. 10. Waiting time of packets generated in GCDC 2016 and dispatched under high system load.

Table VII shows some queue statistics generated by running a 15-second fragment of the log, with an average of 668 messages per second. The main difference between the simulations and the real data is the distribution of the amount of messages in each traffic class. This is due to the scenarios of the competition, where a small number of vehicles is concentrated in a small area. Furthermore, many DENM where sent during the competition, which shifts the distribution towards the higher priority classes in comparison with a normal environment. For the *high system load* setup, we can appreciate that while the packet drop rate does not change drastically between classes, the mean waiting time ($\bar{x}$) is much lower for the higher classes.

On the other hand, running the logs on a *low system load* setup shows that both the packet drop rate and mean waiting time are negligible and the performance of the system is not affected negatively by the SAPQ.

Figure 10 shows the evolution of the packet waiting time for the same log on the *high system load*. We can see how the majority of class 1 packets are dispatched within a lower waiting time than both class 2, 3 and 4. However, it is interesting to see that some class 2, 3 and 4 packets are dispatched even faster than the class 1 packets. This behaviour was not observed previously in the simulations and it is possibly due to the phase shift of the packet arrival for

each vehicle, in contrast with the simulation where the packets arrived in a synchronized manner. We can conjecture that class 2 and 3 packets arrive when the consumer threads are idle and therefore they are dispatched immediately. When class 1 packets arrive, the consumer threads are busy processing packets from low priority classes and they must wait until they are finished.

V. Conclusion

In this paper, we have looked not only into previous research work about congestion and awareness control in VANETs but also into generic queuing and buffering policies. Aiming to avoid the reception of old messages caused by the congestion of the FIFO queue currently used, we have proposed a new system capable of dealing with this problem, SAPQ. SAPQ enhances Cooperative Awareness in ETSI ITS-G5 communication by prioritizing and discarding the messages based on the relevance of the vehicle that transmitted the message and the information that is contained in that type of message.

SAPQ is based on the existing APQ, which makes use of traffic classification and assigns a priority to each class. Each element in the queue accumulates priority over time depending on his class. The most important packets will get a higher accumulating factor and will be dispatched more often, but even if a packet has low priority it will be dispatched at some point due to the accumulated priority.

We demonstrate that classifying and inserting the packet in the queue before decoding it may decrease packet waiting time in the queue, thereby, decreasing overall end-to-end delay. Encoding and decoding of messages are the most expensive operations, excluding processing at application layer. Our application of SAPQ for VANETs uses information that is available before decoding the packet, therefore saving processing resources that otherwise would be spent in decoding unnecessary messages.

To analyze the viability and performance of our method, we developed a system implemented in Java and further integrated it with the ITS-G5 protocol stack used in GCDC. By running the system, first with simulation data and then with real data, we gathered the information necessary to evaluate and compare our system with a regular FIFO queue.

We conclude that SAPQ provides VANET receivers with the capability to prioritize V2V messages based on the data contained in the messages exchanged and the relevance of the vehicle where the message is originated. Furthermore, information from upper layers or decoding the messages is not necessary, which reduces both the complexity and the resources required to run the system that makes use of it. Using a SAPQ reduces the waiting time of the packets that belong to a high priority class and reduces the number of low value packets that are dispatched to the system when there are no resources available. Furthermore, by organizing the queue in a stream-wise manner as opposed to the packet-wise queue seen in similar systems [7], SAPQ is able to fairly serve messages coming from multiple vehicles.

SAPQ enables the controlled smart degradation of the system under high V2V traffic load by dropping unnecessary packets and prioritizing the most important ones. In contrast, a regular FIFO queue would saturate and fail in the same conditions. This potentially allows to run V2V communication systems with less resources and a cheaper hardware. At the same time, it reduces the time between the reception of a high priority packet and its processing, decreasing accordingly end-to-end delay, which is, no doubts, beneficial for safety-critical C-ITS applications.

However, there are still many aspects of the SAPQ that can be improved. As a future work, we are aiming to find the optimal value of the accumulating factors of the APQ for a given traffic situation, and derive an adaptive model with the capability to adjust them for extreme conditions when necessary.

The proposed vehicle classification should be improved by adding a higher degree of accuracy in the calculation of vehicle distance and prediction of vehicle trajectories. Additionally, The consequent increase in computational costs could be mitigated by reusing the calculated values in other components of the system.

Requirements of C-ITS applications, such as the ones presented in [25], must be further analyzed to guarantee that the proposed approach is applicable to all safety critical situations. Finally, extensive in-vehicle testing of the SAPQ in different traffic conditions and applications that make use of the system would be necessary.

References

[1] *Intelligent Transport Systems (ITS); European Profile Standard for the Physical and Medium Access Control Layer of Intelligent Transport Systems Operating in the 5 GHz Frequency Band*, document ES 202 663-V1.1.0, ETSI, Sophia Antipolis, France, 2009, pp. 1–27.

[2] M. Sepulcre, J. Mittag, P. Santi, H. Hartenstein, and J. Gozalvez, "Congestion and awareness control in cooperative vehicular systems," *Proc. IEEE*, vol. 99, no. 7, pp. 1260–1279, Jul. 2011.

[3] J. Breu, A. Brakemeier, and M. Menth, "A quantitative study of cooperative awareness messages in production VANETs," *EURASIP J. Wireless Commun. Netw.*, vol. 2014, no. 1, p. 98, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1186/1687-1499-2014-98

[4] A. Voronov, J. De Jongh, D. Heuven, and A. Severinson, "Implementation of ETSI ITS G5 GeoNetworking stack, in Java: CAM-DENM/ASN.1 PER/BTP/GeoNetworking," Tech. Rep., Jun. 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.55650, doi: 10.5281/zenodo.55650.

[5] M. Aramrattana *et al.*, "Team halmstad approach to cooperative driving in the grand cooperative driving challenge 2016," *GCDC Special Issue ITS*, 2017.

[6] J. Breu and M. Menth, "Relevance estimation of cooperative awareness messages in VANETs," in *Proc. IEEE 5th Int. Symp. Wireless Veh. Commun. (WiVeC)*, Jun. 2013, pp. 1–5.

[7] J. Breu and M. Menth, "Efficient selection of VANET messages for series vehicles," in *Proc. 8th Int. Workshop Resilient Netw. Design Modeling (RNDM)*, Sep. 2016, pp. 274–280.

[8] C. Englund *et al.*, "The grand cooperative driving challenge 2016: Boosting the introduction of cooperative automated vehicles," *IEEE Wireless Commun.*, vol. 23, no. 4, pp. 146–152, Aug. 2016.

[9] *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*, document TS 102 636-4-1-V1.1.1, ETSI, Sophia Antipolis, France, 2011.

[10] *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-Part 1: Basic Transport Protocol*, document EN 302 636-5-1-V1.2.0, ETSI, Sophia Antipolis, France, 2013.

[11] *ITU-T X.691 ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER)*, International Telecommunication Union, 2015.

[12] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, document EN 302 637-2 V1.3.1, ETSI, Sophia Antipolis, France, Sep. 2014.
[13] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specification of Decentralized Environmental Notification Basic Service*, document EN 302 637-3 V1.2.2, ETSI, Sophia Antipolis, France, Nov. 2014.
[14] N. Lyamin, A. Vinel, and M. Jonsson, "Does ETSI beaconing frequency control provide cooperative awareness?" in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 2393–2398.
[15] A. Böhm, M. Jonsson, and E. Uhlemann, "Co-existing periodic beaconing and hazard warnings in IEEE 802.11p-based platooning applications," in *Proc. 10th ACM Int. Workshop Veh. Inter-Netw., Syst., Appl.*, 2013, pp. 99–102.
[16] A. Böhm, M. Jonsson, and E. Uhlemann, "Performance comparison of a platooning application using the IEEE 802.11p MAC on the control channel and a centralized MAC on a service channel," in *Proc. IEEE 9th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2013, pp. 545–552.
[17] M. Barradi, A. S. Hafid, and J. R. Gallardo, "Establishing strict priorities in IEEE 802.11p WAVE vehicular networks," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–6.
[18] I. Cidon, R. Guerin, and A. Khamisy, "On protective buffer policies," in *Proc. 12th Annu. Joint Conf. IEEE Comput. Commun. Soc., Netw., Found. Future (INFOCOM)*, vol. 3. 1993, pp. 1051–1058.
[19] P. Hurley, J. Y. L. Boudec, P. Thiran, and M. Kara, "ABE: Providing a low-delay service within best effort," *IEEE Netw.*, vol. 15, no. 3, pp. 60–69, May 2001.
[20] L. Kleinrock, "A delay dependent queue discipline," *Naval Res. Logistics Quart.*, vol. 11, nos. 3–4, pp. 329–341, 1964. [Online]. Available: http://dx.doi.org/10.1002/nav.3800110306
[21] D. A. Stanford, P. Taylor, and I. Ziedins, "Waiting time distributions in the accumulating priority queue," *Queueing Syst.*, vol. 77, no. 3, pp. 297–330, Jul. 2014.
[22] *Basic Transport Protocol*, document EN 302 636-5-1 V1.2.0, European Telecommunications Standards Institute, Sophia Antipolis, France, 2013, pp. 1–16.
[23] J. T. van de Sluis, O. T. Baijer, L. V. Chen, H. V. H. Bengtsson, L. I. Garcia-Sol, and P. I. Balague, "i-GAME D3.2 Proposal for extended message set for supervised automated driving," Tech. Rep., 2016.
[24] N. Cenerario, T. Delot, and S. Ilarri, "A content-based dissemination protocol for vanets: Exploiting the encounter probability," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 771–782, Sep. 2011.
[25] M. Sepulcre and J. Gozalvez, "On the importance of application requirements in cooperative vehicular communications," in *Proc. 8th Int. Conf. Wireless Demand Netw. Syst. Services*, Jan. 2011, pp. 124–131.

Víctor Díez Rodríguez received the B.Sc. degree in computer science from the University of Burgos, Spain, in 2014, and the M.Sc. degree in embedded and intelligent systems from Halmstad University at Sweden, Sweden, in 2016. He was a part of the Team Halmstad during the Grand Cooperative Driving Challenge in 2016. He is currently a Consultant Engineer with Segula Technologies, Sweden. His main interests include vehicular communication and software engineering.

Jérôme Detournay received the M.Sc. degree in embedded and intelligent systems from Halmstad University at Sweden, Sweden, in 2016. In 2016, he was a part of the Team Halmstad for the Grand Cooperative Driving Challenge. His interests include communication and networking.

Alexey Vinel (M'07–SM'12) received the bachelor's and master's (Hons.) degrees in information systems from the Saint Petersburg State University of Aerospace Instrumentation, Saint Petersburg, Russia, in 2003 and 2005, respectively, and the Ph.D. degree in technology from the Institute for Information Transmission Problems, Moscow, Russia, in 2007, and the Tampere University of Technology, Tampere, Finland, in 2013. He is currently a Professor of data communications with the School of Information Technology, Halmstad University at Sweden, Halmstad, Sweden. He has been involved in research projects on vehicular networking standards, advanced driver assistance systems, and autonomous driving. He has been an Associate Editor for the IEEE COMMUNICATIONS LETTERS since 2012.

Nikita Lyamin received the B.S. degree and M.S. (Hons.) degrees in telecommunications from the Siberian State University of Telecommunications and Information Sciences, Novosibirsk, Russia, in 2011 and 2013, respectively. He is currently pursuing the Ph.D. degree with the School of Information Technology, Halmstad University at Sweden, Halmstad, Sweden. His current research interests include vehicular ad-hoc networks, ETSI ITS-G5 vehicular communications, C-ACC/Platooning performance evaluation, and security in C-ACC/Platooning.