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

Jérôme Detournay
Víctor Díez Rodríguez

Embedded and Intelligent Systems

Halmstad University, September 28, 2016

Jérôme Detournay, Víctor Díez Rodríguez: *An Approach for Receiver-Side Awareness Control in Vehicular Ad-Hoc Networks*, , Master's Programme in Embedded and Intelligent Systems, 120 credits, © September 2016

SUPERVISORS:

Alexey Vinel
Nikita Lyamin

LOCATION:

Halmstad

TIME FRAME:

September 2016

ABSTRACT

Vehicular Ad-Hoc Network ([VANET](#))s are a key element of Intelligent Transport System ([ITS](#))s. 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 thesis, we propose a stream-wise 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.

*It turns out that an eerie type of chaos can lurk just behind a facade of order
- and yet, deep inside the chaos lurks an even eerier type of order.*

— Douglas R. Hofstadter, *Metamagical Themas: Questing for the
Essence of Mind and Pattern*

ACKNOWLEDGEMENTS

We would like to thank our thesis supervisors Alexey Vinel and Nikita Lyamin for giving us the freedom to focus our thesis in the direction that we believed was the right one and for supporting us during this process.

Special thanks to Wojciech Mostowski for answering the GCDC emails, attending the meetings and dealing with the paperwork, to Maytheewat Aramrattana for hydrating and feeding us during the preparation week and to the rest of our friends of Team Halmstad for making this last year more entertaining.

CONTENTS

1	INTRODUCTION	1
1.1	Background	1
1.2	Intelligent Transport System	2
1.2.1	802.11p	3
1.2.2	Geonetworking and BTP	3
1.2.3	CAM and DENM	4
1.3	Grand Cooperative Driving Challenge	5
1.4	Related work	9
1.4.1	Related to communication protocols	9
1.4.2	Related to queuing and scheduling	11
1.5	Motivation	11
1.6	Focus of this work	13
2	METHODS	15
2.1	Overview of the Stream-wise Accumulating Priority Queue	15
2.2	Description of the algorithms	15
2.2.1	Traffic classification	15
2.2.2	Packet dispatching	20
3	IMPLEMENTATION	23
3.1	Implementation overview	23
3.2	Stream-wise Accumulating Priority Queue	23
4	RESULTS	27
4.1	Simulation results	27
4.1.1	Validation of vehicle classification algorithm	27
4.1.2	Evaluation of the Stream-wise Accumulating Priority Queue implementation	28
4.2	Examples with real data	33
5	CONCLUSIONS	37
A	APPENDIX	39
A.1	Coordinate conversion using lookup tables	39
	BIBLIOGRAPHY	41

LIST OF FIGURES

Figure 1	ITS station reference architecture	2
Figure 2	Frequency allocation to the 802.11p European standard	3
Figure 3	GeoUnicast	4
Figure 4	GeoBroadcast	4
Figure 5	Topologically-scoped broadcast	4
Figure 6	Structure of a CAM	5
Figure 7	Structure of a DENM	6
Figure 8	Platooning	6
Figure 9	Merging Scenario	7
Figure 10	Crossing Scenario	7
Figure 11	Emergency Scenario	8
Figure 12	Structure of an iCLCM	9
Figure 13	Proposed queuing in the ITS-G5 protocol stack	14
Figure 14	Stream-wise Accumulating Priority Queue Overview.	15
Figure 15	Accumulating Priority Queue and stream classification.	16
Figure 16	Closest distance calculation in different scenarios.	19
Figure 17	Relevance assignment.	19
Figure 18	Priority over time	21
Figure 19	Overview of the class diagram of the implementation	23
Figure 20	Detailed view of the Stream-wise Accumulating Priority Queue implementation	25
Figure 21	Simulation setup used for vehicle classification validation	27
Figure 22	Vehicle classification.	28
Figure 23	Relation between stream waiting time and data age	30
Figure 24	Waiting time of packets dispatched under low system load.	31
Figure 25	Waiting time of packets dispatched under high system load.	32
Figure 26	Distribution function of the waiting time for each class.	33
Figure 27	Distribution function of waiting time for streams	34
Figure 28	Waiting Time and message income/drop representation	34

Figure 29	Waiting time of packets generated in GCDC 2016 and dispatched under high system load.	35

LIST OF TABLES

Table 1	Spectrum ranges for GCDG 2016	8
Table 2	Final message classification	17
Table 3	Traffic classification according to message type	17
Table 4	Vehicle classification	20
Table 5	Distribution of vehicle classes used in the simulation	29
Table 6	Class accumulating factors used in simulation	29
Table 7	Simulation parameters used for the queue comparison.	31
Table 8	Comparison of packet drop rate for each class and amount of vehicles.	33
Table 9	Statistics generated by replaying GCDG 2016 logs through the SAPQ.	35
Table 10	Latitude degrees to kilometers conversion lookup table	39
Table 11	Longitude degrees to kilometers conversion lookup table	39

LIST OF ALGORITHMS

1	SAPQ put(BTTPacket)	24
2	SAPQ take()	24

ACRONYMS

AC	Awareness Control
CC	Congestion Control
BTP	Basic Transport Protocol
CAM	Cooperative Awareness Message
CACC	Cooperative Adaptive Cruise Control
CDF	Cumulative Distribution Function
CEN	Commission Européenne de Normalisation
DENM	Decentralized Environmental Notification Message
ETSI	European Telecommunications Standards Institute
GCDC	Grand Cooperative Driving Challenge
iCLCM	i-GAME Cooperative Lane Change Message
ISO	International Standardization Organization
ITS	Intelligent Transport System
ITS-S	ITS Station
MIO	Most Important Object
OSI	Open System Interconnection
PDU	Protocol Data Unit
APQ	Accumulating Priority Queue
SAPQ	Stream-wise Accumulating Priority Queue
FIFO	First-In-First-Out
VANET	Vehicular Ad-Hoc Network
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure

INTRODUCTION

1.1 BACKGROUND

VANETs are a key element of **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 **ITS** applications. This 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 might be vital 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 other 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 vastly 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 multilane highway. Broadcasting too many messages or with too much power in a busy road can cause the congestion of the channel. Broadcasting too few messages 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**) algorithms modify parameters of the application and communication protocol to prevent the congestion of the channel, while Awareness Control (**AC**) algorithms try to generate the minimum amount of traffic to provide a

certain level of awareness.

1.2 INTELLIGENT TRANSPORT SYSTEM

Cooperative ITS is a way to share data between ITS Station (ITS-S) to improve the safety, the efficiency and comfort. To achieve a good understanding of the vehicle some rules and definition have to be done. In ITS, those standards are developed by different committee : ISO, CEN and ETSI.

Functionalities used by the ITS are defined by the ITS station reference architecture (ISO 21217), represent in figure 1. As explained in [2], the ITS station reference architecture is an extended version of the OSI model for layered communication protocol.

In figure 1, the Access block represents the two first layers of the OSI model, the Networking block layers three and four and the Facilities block layers five, six and seven. The application block handles the connection between different ITS-S. On the left part, the management module manages the communication inside the ITS-S. On the right side, the Security module provides security to the station.

Figure 1: ITS station reference architecture

In [2] and [3], they explain an ITSC station contains several functional elements. The router to interconnect two ITS element, the gateway to link two different OSI protocol stack and the host which contains at least the ITS station reference. Those functional elements are present in the ITS subsystem : personal ITS subsystem, vehicle ITS subsystems, roadside ITS subsystems, and central ITS subsystems.

Different protocols can be used with the ITS station reference architecture. Those use in this work are 802.11p for the access layer, Geonetworking and Basic Transport Protocol (BTP) for the network and transport layer and Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM) for the facilities layer and are describe in the next section. Details about other protocols can be found in [2], [4], [5], [6].

1.2.1 Access Layer - 802.11p

In [4] and [5], it is explained that 802.11p is a short range communication standard developed for ad hoc communication in V2V and V2I. It operates in the 5.9GHz range. In the European standard (ETSI), the frequency band is divided into three part. Figure 2 represent the division of the frequency band : ITS-G5B is from 5.855 to 5.865 Mhz and is used for the non safety C-ITS application. ITS-G5A is from 5.875 to 5.905 Mhz is for the safety and traffic efficiency. In ITS-G5A, the control channel (CCH) is the primary safety channel the two other are additional service channel (SCH 1 and 2). The third division ITS-G5D is for future C-ITS application.

Figure 2: Frequency allocation to the 802.11p European standard

1.2.2 Network and Transport Layer - Geonetworking and BTP

Geonetworking is a network protocol used for ad hoc wireless communication such as VANETs.

As explained in [7] and [8], Geonetworking can be split in two functions. The geographical addressing : Instead of addressing the peers by using their IP addresses the Geonetworking protocol used the geographical position. The geographical forwarding : A packet is sent by a vehicle to a specific location. The message is forwarded by all vehicles but only the vehicle inside the target area processes it.

The geographic routing consists of three different methods :

- The GeoUnicast : A packet is sent from a node to a specific destination. The message is forwarded by the different nodes until the destination is reached. i.e as shown in figure 3.
- GeoBroadcast : A packet is sent from a node to a destination area. It is forwarded by the different nodes until the destination. At the destination, the message will be broadcasted to all vehicles inside the area. i.e as shown in figure 4.

Figure 3: GeoUnicast

Figure 4: GeoBroadcast

- Topologically-scoped broadcast : A packet is broadcasted by the source and rebroadcasted by the n-hop neighbourhood. i.e as shown in figure 5. When a packet is sent to one hop neighbourhood it is called single hop broadcast.

Figure 5: Topologically-scoped broadcast

One of the transport protocols used on top of geonetworking is [BTP](#). As defined in [\[9\]](#) [BTP](#) is a end-to-end, connection-less transport service for [ITS](#) ad-hoc network.

1.2.3 Facilities layer - CAM and DENM

In [\[4\]](#) and [\[10\]](#), it is explained that CAM is a periodic message sent at a frequency between 1 and 10Hz. CAM contains information about the [ITS](#) station. The message is sent to aware and provides informa-

tion to the neighbourhood ITS station. As shown in figure 6, CAMs are divided into different containers. Three mandatory : the ITS Protocol Data Unit (PDU) header includes information about the protocol version, the message type and the ITS station ID. The basic container provides basic information as the station type and the geographical position. The high frequency container contains dynamic data such as the vehicle speed, heading, acceleration, ... And two optional containers : The low frequency container for static and low dynamic data, vehicle role or path history for example. And the special vehicle container used for specific senders such as public transport, special transport, ...

Figure 6: Structure of a CAM

In [4] and [11], it is described that DENM is an event-triggered message used to alert ITS station of an event, for example, a road work. The DENM message, as CAM, is divided into several containers shown in figure 7. Two mandatory : the ITS PDU header contains the protocol version, the message type and the ITS station ID. The management container provides the action ID, the reference time, the event position, ... And Three optional : situation container with field to describe the event, location container contains information about the speed, the heading of the event and A la carte container to add some specific content such as the lane position.

More details about the different container and fields can be found in the ETSI 302 637-2 [10] for CAM and 302 637-3 [11] for DENM.

1.3 GRAND COOPERATIVE DRIVING CHALLENGE

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. The first edition took place in 2011, where the main scenario was a Cooperative Adaptive Cruise Control (CACC) platooning . As shown in figure 8, by transmitting information (speed, position, accel-

Figure 7: Structure of a DENM

eration,...) cars are driving in a train of vehicles lead by the platoon leader, the blue car here.

Figure 8: Platooning

For [GCDC 2016](#) three scenarios were proposed : merging, crossing and emergency vehicle scenario.

The merging scenario took place on a motorway. Two lines of vehicles drive toward a roadwork on the left lane as shown in figure [9](#). When arriving at the competition zone vehicles receive a roadwork message from a RSU. In figure [9b](#), after receiving the message, the vehicles are pairing together and cars in the right lane make a gap. When the *safe-to-merge* message is received from the backward partner car, the left vehicle merges in the right lane. When the merging is done, the red vehicle joins platoon B as shown in figure [9c](#).

The second scenario is illustrated in figure [10](#). Three vehicles are coming from each side of a T-junction and the red car has to enter on the same road as the two other vehicles. As shown in figure [10a](#) the three cars are driving toward the crossing and begin to communicate after reaching the competitive zone. The grey and yellow car will manoeuvre in a way to let the red car enter the road as shown in figure [10b](#). Figure [10c](#) shows when the red vehicle has entered the road.

The last scenario is illustrated in figure [11](#), two platoon of cars are driving on a motorway. After a while, an emergency vehicle arrives behind the vehicles and sends an alert message. When the message is received cars make a gap to let the fire truck pass.

Figure 9: Merging Scenario

Figure 10: Crossing Scenario

Figure 11: Emergency Scenario

For the cooperative part, cars have to communicate and understand each other. And it is important that some specifications are established. The GCDC contest follows the [ITS](#) reference model explained in section [1.2](#) using Geonetworking, [BTP](#), ITS-G5, [CAM](#) and [DENM](#). But to perform correctly some little changes had to be done. Instead of sending CAMs at 10Hz, it was required by [\[12\]](#) that they were sent at 25Hz in order to reduce communication latency that could create safety risks and to reduce the safety distance between cars. Furthermore, [\[13\]](#) shows how CACC controllers for platooning applications can benefit from a higher message transmission rate. Also, a new type of message, i-[GAME](#) Cooperative Lane Change Message ([iCLCM](#)), was introduced. The channels shown in the spectrum table [1](#) were used. More details about the service channels (SCH1 and SCH2) and the control channel (CCH) can be found in [\[14\]](#).

Channel Type	Center frequency	Channel number	Channel spacing
G5-CCH	5900 MHz	180	10 Mhz
G5-SCH2	5890 MHz	178	10 Mhz
G5-SCH1	5880 MHz	176	10 Mhz

Table 1: Spectrum ranges for GCDC 2016

[iCLCM](#) is a periodic message sent at 25Hz. It was created to overcome the missing field in [CAM](#) and [DENM](#) for the merging scenario. The message is composed of a common [ITS PDU](#) Header and seven

other containers. The **iCLCM** header contains the generation delta time, high frequency container for information that needs to be sent frequently and low frequency container for data which does not need to be updated too often. The Most Important Object (**MIO**) container contains the neighbours and the most important object. Lane shows where the lane ID is set, pair ID contains the station ID of the pairing partners. Merge container contains all the information useful for the merging. And additional container **S** is used for the scenario execution.

Figure 12: Structure of an iCLCM

1.4 RELATED WORK

1.4.1 Related to communication protocols

Extensive research has been made in the field of communication in order to increase the channel utilization and reduce the amount of traffic required to provide cooperative driving application 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 [15], 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. M. Sepulcre et al. classify **CC** methods into reactive **CC** and proactive **CC**. Reactive **CC** uses information about the current sta-

tus 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. On the other hand, the control algorithm 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.

In [\[16\]](#), the authors show how the message generation rules used in the current ETSI standard are prone to create channel congestion in platooning applications. It exhibits the difficulty of designing an effective method for congestion and [AC](#) and how a transmission rate approach can be counterproductive.

In [\[17\]](#), it is shown how prioritization of multiple traffic classes on the same channel can improve the awareness level in cooperative application. Bohm et al. use the DENM dissemination delay and CAM up-to-dateness to evaluate the performance multiple combinations of priority levels.

A different approach is taken in [\[18\]](#), 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.

In [\[19\]](#) the authors 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 [\[20\]](#), 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 a relevance based on it to the received [CAMs](#).

1.4.2 *Related to queuing and scheduling*

Most research done regarding network queuing and scheduling is not specific to vehicular communication but to general IP 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 [21]. 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 presented and evaluated new policies that reduce the loss of low priority traffic while maintaining the optimal performance for high priority traffic.

Duplicate Scheduling with Deadlines (DSD)[22] 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 [23], 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. In [24], the authors 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.

1.5 MOTIVATION

Cooperative driving applications rely greatly on the performance of the wireless communication systems used to transmit information among the vehicles. While each vehicle is in control of the transmis-

sion 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 clear how to determine what information is useful for every concrete 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 channel in healthy conditions 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 know 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 the channel can carry, 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 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 [15], the number of vehicles in communication range in a multilane highway can go up to 300. Of these 300, the messages broadcasted by more than half of them will be almost irrelevant due to the distance and driving direction. In [25], it is estimated that, using the current standard on transmission rate, a vehicle will easily receive up to 500 CAMs 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 channel 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 is 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 [26], which serves the messages in a first-come-first-serve manner. 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.

As shown by Clark et al. in [27], the most expensive operations in communication protocols are not transfer control operations, but data manipulation. Specifically, they point out that data presentation can cost more than all other manipulations combined. In the ITS-G5 protocol stack, this is done by ASN.1 encoding and decoding. ITS-G5 uses ASN.1 Unaligned Packed Encoding Rules (UPER), which favors a reduced encoded size in detriment of slower encoding speed in order to save bandwidth during the transmission of the message. In fact, we observed that in the ITS-G5 stack implementation used in GCDC [26], the most expensive operation was the encoding and decoding of the CAM, DENM and iCLCM messages. Therefore, the best place to discard unnecessary messages is even before they have been completely decoded.

1.6 FOCUS OF THIS WORK

In this thesis, we will design, implement and analyze a message queuing system for VANETs. The aim of such system is to reduce the impact of bottlenecks in the message processing pipelines found in current cooperative applications running on limited computer hardware resources. This will be achieved by prioritizing messages based on the cooperative awareness requirements of safety-critical applications and by considering the firm real-time nature of these messages.

Figure 13 shows the location of the queuing system proposed with respect to the ITS-G5 protocol stack.

In chapter 2 we introduce Stream-wise Accumulating Priority Queue, a queuing system designed to fulfill the requirements defined in this introduction. In chapter 3 we describe the actual implementation of the queuing system over an already existing platform. In chapter 4 we evaluate the performance of the implemented Stream-wise Accumulating Priority Queue with both traffic generated using simulations and real traffic gathered during GCDC 2016.

Figure 13: Proposed queuing in the ITS-G5 protocol stack

METHODS

2.1 OVERVIEW OF THE STREAM-WISE ACCUMULATING PRIORITY QUEUE

The first step in the Stream-wise Accumulating Priority Queue (SAPQ) is separating every packet received in streams. Each stream will contain a certain type of message from a concrete vehicle. After the streams are defined, the SAPQ can be divided in three main subsystems: packet buffering, stream classification and priority assignment. The packet buffering subsystem stores the most recent packet received in each stream and drops old packets that have not been served. The stream classification subsystem assigns a class to each stream in order to give more importance to streams that contain more valuable information. This process is explained in detail in section 2.2.1. The priority assignment subsystem calculates the instantaneous priority of each stream using an APQ. Then, the packet contained in the buffer of the stream with the highest priority is served.

Figures 14 and 15 show an schematic view of the system.

Figure 14: Stream-wise Accumulating Priority Queue Overview.

2.2 DESCRIPTION OF THE ALGORITHMS

2.2.1 Traffic classification

SAPQ makes use of traffic classification in order to prioritize the messages received. We want to set apart messages that are relevant for

Figure 15: Accumulating Priority Queue and stream classification.

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 packet received using two different classification criteria: message type and vehicle relevance. Then, a general classification will be obtained by combining these two criteria as shown in table 2. Note that, in this case, streams with lower class will be assigned a higher priority and therefore a higher accumulating factor.

2.2.1.1 Message type classification

The first criterion is related with the type of information that the packet carries. This is directly related to the type of the message contained in the packet. As each type of message belongs to a well-know BTP port [28][12], it is straightforward to classify each packet based on the arrival port. Table 3 shows the classification according to message type.

Table 2: Final message classification

Vehicle Class \ Message Class	1	2	3
1	1	1	2
2	1	2	3
3	2	3	4
4	3	4	4

Table 3: Traffic classification according to message type

Message type	Port	Packet class
DENM	2002	1
CAM	2001	2
iCLCM	2010	3

2.2.1.2 Vehicle classification

In order to assign a class to each vehicle, we will use 2 parameters: current distance to the vehicle and the minimum distance to the vehicle. Using 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 vehicle to evaluate to approximate the time where the distance between both vehicles will be minimal and then calculate the minimum distance. This will allow us to tell whether a vehicle is coming towards us or driving away, and even anticipate a collision.

We define the position of a vehicle $\mathbf{p}_i$ and its velocity $\mathbf{u}_i$:

$$\mathbf{p}_i = (x_i, y_i) \quad \mathbf{u}_i = (v_i, w_i)$$

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:

$$\begin{aligned}
\mathbf{d}_{ij}(t) &= \mathbf{p}_j(t) - \mathbf{p}_i(t) \\
&= \mathbf{p}_j - \mathbf{p}_i + (\mathbf{u}_j - \mathbf{u}_i) \cdot t \\
&= (x_j - x_i + (v_j - v_i) \cdot t, y_j - y_i + (w_j - w_i) \cdot t)
\end{aligned}$$

And the distance:

$$\|\mathbf{d}_{ij}(t)\| = \sqrt{(x_j - x_i + (v_j - v_i) \cdot t)^2 + (y_j - y_i + (w_j - w_i) \cdot t)^2} \quad (1)$$

Then we can calculate the time to minimum distance by solving:

$$\begin{aligned}
\frac{\partial \|\mathbf{d}_{ij}(t)\|}{\partial t} &= 0 \\
\frac{\partial \|\mathbf{d}_{ij}(t)\|}{\partial t} &= \frac{2 \cdot (v_j - v_i) \cdot (t \cdot (v_j - v_i) + x_j - x_i) + 2 \cdot (w_j - w_i) \cdot (t \cdot (w_j - w_i) + y_j - y_i)}{2 \cdot \sqrt{(t \cdot (v_j - v_i) + x_j - x_i)^2 + (t \cdot (w_j - w_i) + y_j - y_i)^2}} = 0 \\
t_{min} &= \frac{x_i \cdot v_j + v_i \cdot x_j - v_j \cdot x_j + y_i \cdot w_j + w_i \cdot y_j - w_j \cdot y_j - v_i \cdot x_i - w_i \cdot y_i}{-2 \cdot v_i \cdot v_j + v_j^2 + w_j^2 - 2 \cdot w_i \cdot w_j + v_i^2 + w_i^2} \quad (2)
\end{aligned}$$

Figure 16 shows how this equations are applied to different scenarios. In the plots, $p(t_0)$ shows the position of each vehicle at the current time, along with the velocity vector. The points $p(t)$ show the position of each vehicle at the time when the distance between them is minimal. A negative t indicates that the minimum distance between both vehicles was in the past i.e. the vehicles are moving away from each other. Although counter-intuitive, giving higher relevance to vehicles moving away can be useful for applications that consist on following vehicles, such as platooning.

We will use the values of minimum distance and current distance to assess the relevance of the neighbouring vehicles. Figure 17 shows how the relevance of each vehicle is assigned depending on these values. Each relevance class will contain vehicles with the following characteristics:

- High Relevance: vehicles in risk of immediate collision, closest platoon members, vehicles overtaking our own vehicle, vehicles in the adjacent lanes.
- Medium Relevance: 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.

Figure 16: Closest distance calculation in different scenarios.

Figure 17: Relevance assignment.

- Low Relevance: all other neighbouring vehicles.

Table 4 describes the final vehicle classification using current distance and minimum distance.

Table 4: Vehicle classification

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

2.2.2 Packet dispatching

The packets received are dispatched using an [APQ](#) [23]. The [APQ](#) does not contain individual messages, but message streams with pending messages. 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. The streams are divided in multiple classes according to the procedure described in 2.2.1. Every class has assigned a accumulating factor b_c , which will determine the speed at which its priority increases. Then, the instantaneous priority of a stream $q_s(t)$ is defined by:

$$q_s(t) = (t - t_{arrival}) \cdot b_c \quad (3)$$

Where t is the current time and t_s is the arrival time of the first message received in the stream since the last time it was served. Thus, the streams that belong to a class with higher accumulating factor gain priority more rapidly and are served more often. The reason why t_s is defined in this way instead of using the arrival time of the most recent message is to prevent the loss of instant priority in the stream due to the arrival of newer information. Note that the stream with higher priority is the one with higher numerical value, as opposed to the usual convention where a smaller number means higher priority and, therefore, the stream that is served is the one with highest $q_s(t)$ at the moment of selection.

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

Figure 18 shows an example of the [APQ](#) described previously. The graph shows an [APQ](#) containing 3 streams, each with a different accumulating factor. The dotted line marks each time that the queue is

Figure 18: Priority over time

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 is served.

IMPLEMENTATION

3.1 IMPLEMENTATION OVERVIEW

This chapter describes the actual implementation of the [SAPQ](#) described in the previous sections. The implementation was made completely in Java in order to be easily integrated with the GeoNetworking stack used in GCDC that can be found in [26]. The main aspects that were considered for the implementation were the performance of the algorithms and the thread-safety of the data structures.

On the other hand, the implementation contains a great amount of instrumentation code used to gather the information necessary to evaluate the system. This information ranges from fine-grained data, such as packet-wise queue waiting time, to general statistics such as packet drop percentage for a given traffic class. Figure 19 shows a simplified class diagram of the system.

Figure 19: Overview of the class diagram of the implementation

3.2 STREAM-WISE ACCUMULATING PRIORITY QUEUE

The interface to the [SAPQ](#) consists of a constructor and two methods:

- 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. This is 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  $\leftarrow$  (BTPPacket.address, BTPPacket.port)
2: PacketBuffer[Key]  $\leftarrow$  BTPPacket
3: if Key in QueueElementMap then
4:   QueueElement  $\leftarrow$  QueueElementMap[Key]
5: end if
6: QueueElement.position  $\leftarrow$  BTPPacket
7: if not QueueElement.active then
8:   QueueElement.time  $\leftarrow$  now
9:   QueueElement.active  $\leftarrow$  true
10: end if
11: QueueElementMap[Key]  $\leftarrow$  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  $\leftarrow$  0
2: for QueueElement in QueueElementMap do
3:   QueueElementClass  $\leftarrow$  calculateClass(QueueElement)
4:   CurrentPriority  $\leftarrow$ 
     AccumulatingFactor[QueueElementClass] * (now -
     QueueElement.time)
5:   if CurrentPriority  $\geq$  MaxPriority then
6:     MaxPriority  $\leftarrow$  CurrentPriority
7:     MaxPriorityElement  $\leftarrow$  QueueElement
8:   end if
9: end for
10: MaxPriorityElement.active  $\leftarrow$  false
11: 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.

ture is that it is not necessary to update the classification of the low priority vehicles compared with the high priority vehicles.

If a cache miss is produced, the classification of the vehicle is calculated using the approach described in section 2.2.1. Furthermore, in order to reduce the computation time of the vehicle distance, conversion from the position in WGS 84 to the relative position of the vehicles in meters is be done using the lookup tables shown in tables 10 and 11.

Next, the message type class is obtained according to table 3. With these two classes, the final class is obtained following table 2. Using the final class, the proper accumulating factor for that 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 returned.

RESULTS

4.1 SIMULATION RESULTS

The main simulation software used in this work is PreScan which was provided by TASS International. PreScan is a simulation platform that integrates with MATLAB/Simulink used in the automotive industry for the development and testing for Advanced Driver Assistance Systems based on sensors. PreScan was used to generate the data necessary to run the simulations i.e. realistic vehicle information such as GPS position, speed and heading. The simulations were programmed in MATLAB for the validation of the vehicle classification and in Java for the evaluation of the [SAPQ](#), which allowed us to integrate the existing GeoNetworking stack and reuse the same infrastructure used in GCDC.

4.1.1 Validation of vehicle classification algorithm

PreScan was used to generate the data set required for the validation of the classification algorithm. Vehicle traces containing GPS latitude/longitude, speed and heading information were recorded. Figure 21 shows the setup used to record simulation data. Four vehicles approach a crossing nearly at the same time, generating multiple intersection pairs to analyze with our algorithm. The traces were processed by a MATLAB implementation of the classification algorithm which used the lookup table for coordinate conversion proposed in tables 10 and 11.

Figure 21: Simulation setup used for vehicle classification validation

Figure 22 shows the results of running the classification algorithm using the traces obtained from PreScan. The black dot represents

the position of our vehicle at a given moment during the simulation, while the other dots represent the position of other vehicles. The graph shows the range of the different classes in a realistic environment.

Figure 22: Vehicle classification.

4.1.2 Evaluation of the Stream-wise Accumulating Priority Queue implementation

The objective of this section is to show the benefits of using [SAPQ](#) in comparison with the traditional First-In-First-Out ([FIFO](#)) approach and provide a quantitative evaluation of the properties of our implementation in different scenarios.

4.1.2.1 Simulation design and 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 25Hz. 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.
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. Due to the link between the nature of the applications that make use of the messages and the processing time for each message type, it is difficult to give an estimation of the relation among the processing times of

the 3 types of messages. 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 of the circles corresponding to the current distance limit for each class. Table 5 shows this distribution.

Table 5: Distribution of vehicle classes used in the simulation

Class	Class limit (m)	Area (m ²)	Δ Area (m ²)	Relation	Probability
1	30	2827.43	2827.43	1	0.01
2	60	11309.73	8482.30	3	0.03
3	150	70685.83	59376.10	21	0.21
4	300	282743.33	212057.50	75	0.75

The accumulating factors for the simulation were set arbitrarily as shown in table 6. The factors were selected with the intention of exhibiting significant differences among the classes and not with optimality in mind.

Table 6: Class accumulating factors used in simulation

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

The parameters for the simulations are:

1. Number of consumer threads (T): number of Java threads that are concurrently extracting and processing packets from the queue.
2. Processing time (b): time that takes to process a single packet. CAM messages take 5 times as much for the reasons given above.

3. Number of vehicles (v): number of vehicles in communication range that are transmitting messages at the same moment.

Figure 23: Relation between stream waiting time and data age

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 23.

4.1.2.2 Comparison of Stream-wise Accumulating Priority Queue and First-In-First-Out queue

In this section, we will compare the performance of a regular **FIFO** queue with the **SAPQ**. To do so, we will analyze the time that each packet has to wait in the queue before being processed in both systems. The simulation platform described previously was used to simulate both queues under two different scenarios. In the first scenario, the system is not overloaded and has resources to process each packet. We will address this scenario as *low system load*. The second scenario is *high system load*, where the system does not have enough resources to process each packet and keep up with the reception rate. The parameters used for the simulation of each scenario are shown in table 7.

Figure 24 shows the performance in terms of packet waiting time of our system and a **FIFO** queue under low system load. It is easy to see that, after a warm-up period of about one second, 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, the **SAPQ** is able to dispatch high priority packets much faster than other packets. For example, most packets from

Table 7: Simulation parameters used for the queue comparison.

	Thread number	Processing time	Number of vehicles
Low system load	8	0.1	100
High system load	1	1	300

class 1 are dispatched under the 5 ms mark, even when there is an unexpected packet peak caused by the reception of a burst of DENM packets after 16000 ms of simulation run time.

Figure 24: Waiting time of packets dispatched under low system load.

However, it is in the *high system load* scenario where the SAPQ shows a completely different behaviour, as shown in figure 25. The FIFO queue buffers every packet, creating longer and longer waiting times when new packets are received and dispatching older messages each time. While our system has the same processing capacity in terms of messages per second, it drops packets containing old information and replaces them with fresh ones. After 1000 ms of simulation time, a burst of DENM messages is received. Because new DENMs are not received after that, they are not replaced with new ones and old information is dispatched. The difference between the FIFO queue and the SAPQ is that the latter only dispatches old packets when new ones from the same stream are not available.

4.1.2.3 Analysis of the waiting time distribution

As explained in [24], the performance of an APQ can be analyzed by looking at the Cumulative Distribution Function (CDF) of the waiting time. To evaluate the system, a set of data containing the waiting time of each packet was generated by simulating the system using different parameters.

Figure 25: Waiting time of packets dispatched under high system load.

Figure 26 shows plots from different sets of parameters. The set in 26b used 4 threads, 0.7 ms busy time and 100 vehicles which gave us the best performance. Indeed, the waiting to process 100% the class 4 messages is less than 20ms and 99% of the class 1 messages are processed before 27ms. In comparison with figure 26a, with 4 threads, 0.7 ms busy time and 300 vehicles, where the waiting time to reach 100% of a class 4 is above 800ms and not even 1% of the class 1 messages are processed within 25ms.

Table 8 shows how the packet drop rate and waiting time of each class changes with a different amount 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. On the other hand, for a class 4 stream the packet waiting time is 430 ms and the application would receive about 2 packets per second.

In figure 27, 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. In comparison, the CDF of DENM streams, which are not periodic, does not resemble the one of the classes where they belong. Figure 28 shows

Figure 26: Distribution function of the waiting time for each class.

Table 8: Comparison of packet drop rate for each class and amount of vehicles.

Vehicles	Class	Drop rate	$\bar{x}$ (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%	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

the relation between waiting time and packet drop for packets received at a rate of 25Hz .i.e. a reception period of 40ms. If 100% of the messages of the stream have a waiting time lower than the reception period, there is no packet drop. In the case of a stream with maximum waiting time higher than the reception period, the [SAPQ](#) will receive a new message before the previous one is dispatched and the oldest one will be dropped.

For example, in figure [28b](#) the maximum waiting time is 60ms. At time 0 ms a message arrives and is inserted in the queue, but at 40 ms a new message arrives. The [SAPQ](#) only keeps the newest package and drops the old one. In this case one message out of two is dropped, in the case of [28c](#) two out of three are dropped.

4.2 EXAMPLES WITH REAL DATA

In this section we will analyze the performance of the [SAPQ](#) by using real data gathered during [GCDG 2016](#). In particular, the log files

(a) CDF of the waiting time (DENMs not plotted) (b) CDF of the waiting time (only DENMs plotted)

Figure 27: Distribution function of waiting time for streams

Figure 28: Waiting Time and message income/drop representation

containing the messages sent during the last heat of the merging scenario were used to generate **CAM**, **DENM** and **iCLCMs**. With respect to the system configuration, the same setup described in tables 6 and 7 was used except for the number of vehicles, which in this case is determined by the vehicles that participated in the competition. In these examples, the complete **SAPQ** is executed, including the vehicle classification algorithm described in section 2.2.1.2.

Table 9 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** were sent during the competition, which shifts the distribution to-

Table 9: Statistics generated by replaying GCDC 2016 logs through the SAPQ.

System load	Class	Packets	Drop rate	$\bar{x}$ (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

wards the higher 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 with 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 in low traffic scenarios.

Figure 29: Waiting time of packets generated in GCDC 2016 and dispatched under high system load.

Figure 29 shows the evolution of the waiting time of the same run on the *high system load* with real data. 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 probably 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 inferior classes and they must wait until they are finished.

CONCLUSIONS

In this thesis, we have looked not only into previous research work about congestion and awareness control in VANETs but also into generic queuing and buffering policies in order to avoid the reception of old messages caused by the congestion of the FIFO currently used and we have proposed a new system capable of dealing with this problem, SAPQ.

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. This concept is really interesting in our case. Indeed, by using the classes system 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 one point due to the accumulated priority.

During our research, we found that it is more interesting to classify and insert the packet in the queue before encoding and decoding it. Indeed, as we explain in section 1.5, the most expensive operation is the decoding and encoding of messages. Our application of SAPQ for VANETs uses information that is available before decoding the packet to classify it, 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 we integrated it with the Geonetworking stack used in CADC. By running the system, first with simulation data then with real data, we gathered the information necessary to evaluate and compare our system with a regular FIFO queue.

We concluded that SAPQ provides VANET receivers with the capability to prioritize Vehicle-to-Vehicle (V2V) messages based on the data contained in those messages and the relevance of the vehicle that sent them. 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 more resources available.

SAPQ enables the graceful degradation of the system due to high amount of V2V traffic 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 allows to run V2V communication systems with less resources and cheaper hardware. At the

same time, it reduces the time between the reception of a high priority packet and its processing, providing extra time for safety-critical cooperative applications such as collision avoidance.

However, there are still many aspects of the [SAPQ](#) that can be improved. As future work, it would be interesting to calculate the optimal proportion between the accumulating factors of the [APQ](#) for a given traffic situation, or even provide the system with the capability to adapt them for extreme conditions when necessary.

On the other hand, the vehicle classification could be improved by adding a higher degree of precision in the prediction of vehicle movement by considering the vehicle acceleration and curvature of the trajectory. However, this would carry a higher computational cost.

Finally, extensive in-vehicle testing of the [SAPQ](#) with multiple traffic conditions and applications making use of the system would be necessary.

APPENDIX

A.1 COORDINATE CONVERSION USING LOOKUP TABLES

Table 10: Latitude degrees to kilometers conversion lookup table

° to equator	$\Delta 1^\circ \text{ Lat (km)}$	Min $\Delta 1^\circ \text{ Lat (km)}$	Max $\Delta 1^\circ \text{ Lat (km)}$	$\epsilon \text{ Lat (km)}$	$\delta \text{ Lat (\%)}$
0°-5°	110.5764165	110.574304	110.5827384	0.006321875991	0.01
5°-10°	110.5932227	110.5827384	110.6077916	0.01456891067	0.01
10°-15°	110.6263369	110.6077916	110.6487207	0.02238376291	0.02
15°-20°	110.6747763	110.6487207	110.7043098	0.02953349211	0.03
20°-25°	110.7371008	110.7043098	110.7729044	0.03580357895	0.03
25°-30°	110.8114527	110.7729044	110.8524568	0.04100411387	0.04
30°-35°	110.8956089	110.8524568	110.9405844	0.04497548808	0.04
35°-40°	110.9870443	110.9405844	111.0346377	0.04759342543	0.04
40°-45°	111.0830045	111.0346377	111.1317777	0.04877318291	0.04
45°-50°	111.1805865	111.1317777	111.2290593	0.04847274383	0.04
50°-55°	111.2768258	111.2290593	111.3235206	0.04669483441	0.04
55°-60°	111.3687858	111.3235206	111.4122734	0.04348761445	0.04
60°-65°	111.4536487	111.4122734	111.4925926	0.03894392747	0.03
65°-70°	111.5288042	111.4925926	111.5620033	0.03319904532	0.03
70°-75°	111.5919324	111.5620033	111.6183593	0.02642690493	0.02
75°-80°	111.6410786	111.6183593	111.6599135	0.01883490674	0.02
80°-85°	111.6747174	111.6599135	111.6853748	0.01065742026	0.01
85°-90°	111.6918027	111.6853748	111.6938649	0.002062234154	0.00

Table 11: Longitude degrees to kilometers conversion lookup table

° to equator	$\Delta 1^\circ \text{ Long (km)}$	Min $\Delta 1^\circ \text{ Long (km)}$	Max $\Delta 1^\circ \text{ Long (km)}$	$\epsilon \text{ Long (km)}$	$\delta \text{ Long (\%)}$
0°-5°	111.2142449	111.3194908	110.8986955	0.3155493525	0.28
5°-10°	110.3734076	110.8986955	109.6393221	0.7340855187	0.67
10°-15°	108.6977552	109.6393221	107.5503973	1.147357932	1.06
15°-20°	106.1993113	107.5503973	104.6469309	1.552380344	1.46
20°-25°	102.8960584	104.6469309	100.9498614	1.946196946	1.89
25°-30°	98.81187035	100.9498614	96.48597408	2.32589627	2.35
30°-35°	93.97641589	96.48597408	91.28778845	2.688627443	2.86
35°-40°	88.42502821	91.28778845	85.39340913	3.031619081	3.43
40°-45°	82.19853561	85.39340913	78.84633471	3.352200902	4.08
45°-50°	75.3430475	78.84633471	71.69521941	3.647828083	4.84
50°-55°	67.90969077	71.69521941	63.99358419	3.916106584	5.77
55°-60°	59.95429347	63.99358419	55.79947036	4.154823113	6.93
60°-65°	51.53701061	55.79947036	47.17503922	4.361971384	8.46
65°-70°	42.72189485	47.17503922	38.18611329	4.535781556	10.62
70°-75°	33.57641029	38.18611329	28.90166355	4.674746746	13.92
75°-80°	24.17089404	28.90166355	19.3932468	4.777647243	19.77
80°-85°	14.57797105	19.3932468	9.734399956	4.843571094	33.23
85°-90°	4.871930005	9.734399956	0	4.871930005	100.00

BIBLIOGRAPHY

- [1] European Telecommunications Standards Institute, "ES 202 663 - V1.1.0 - 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," *ETSI Standard*, vol. 0, pp. 1–27, 2009.
- [2] European Telecommunications Standards Institute, "EN 302 665 V1. 1.1: Intelligent Transport Systems (ITS), Communications Architecture," *European Standard (Telecommunications Series)(September 2010)*, 2010.
- [3] N. Parrado and Y. Donoso, "Congestion based mechanism for route discovery in a V2I-V2V system applying smart devices and IoT," *Sensors*, vol. 15, no. 4, pp. 7768–7806, 2015.
- [4] A. Festag, "Cooperative Intelligent Transport Systems standards in Europe," *IEEE Communications Magazine*, vol. 52, pp. 166–172, December 2014.
- [5] F. Domingos Da Cunha, A. Boukerche, L. Villas, A. C. Viana, and A. A. F. Loureiro, "Data Communication in VANETs: A Survey, Challenges and Applications," Research Report RR-8498, INRIA Saclay ; INRIA, Mar. 2014.
- [6] L. Chen and C. Englund, "Cooperative ITS EU standards to accelerate cooperative mobility," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 681–686, IEEE, 2014.
- [7] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements," TS 102 636-1 V1.1.1, ETSI, March 2010.
- [8] J. Kuperus, "Geonetworking solutions in V2V infrastructures," 2010.
- [9] European Telecommunications Standards Institute, "TS 102 636-5-1 V1. 1.1 Intelligent Transport Systems (ITS)," *Vehicular communications*, 2011.
- [10] European Telecommunications Standards Institute, "EN 302 637-2 V1.3.1-Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service," *ETSI, Sept*, 2014.

- [11] European Telecommunications Standards Institute, "EN 302 637-3 V1.2.2-Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specification of decentralized environmental notification basic service," *ETSI*, Nov, 2014.
- [12] J. T. van de Sluis, O. T. Baijer, L. V. Chen, H. V. Hoang Bengtsson, L. I. Garcia-Sol, and P. I. Balague, "i-GAME D3.2 Proposal for extended message set for supervised automated driving," 2016.
- [13] C. Lei, E. van Eenennaam, W. Klein Wolterink, J. Ploeg, G. Karagiannis, and G. Heijenk, "Evaluation of CACC string stability using SUMO, Simulink, and OMNeT++," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, p. 116, 2012.
- [14] European Telecommunications Standards Institute, "TS 102 724 V1. 1.1 Intelligent Transport Systems (ITS); Harmonized Channel Specifications for Intelligent Transport Systems operating in the 5 GHz frequency band," *Intelligent Transport Systems*, 2012.
- [15] M. Sepulcre, J. Mittag, P. Santi, H. Hartenstein, and J. Gozalvez, "Congestion and awareness control in cooperative vehicular systems," *Proceedings of the IEEE*, vol. 99, pp. 1260–1279, July 2011.
- [16] N. Lyamin, A. Vinel, and M. Jonsson, "Does ETSI beaconing frequency control provide cooperative awareness?," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, pp. 2393–2398, IEEE, 2015.
- [17] A. Böhm, M. Jonsson, and E. Uhlemann, "Co-existing periodic beaconing and hazard warnings in IEEE 802.11 p-based platooning applications," in *Proceeding of the tenth ACM international workshop on Vehicular inter-networking, systems, and applications*, pp. 99–102, ACM, 2013.
- [18] 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 *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 545–552, Oct 2013.
- [19] M. Barradi, A. S. Hafid, and J. R. Gallardo, "Establishing strict priorities in IEEE 802.11p WAVE vehicular networks," *GLOBE-COM - IEEE Global Telecommunications Conference*, pp. 0–5, 2010.
- [20] J. Breu and M. Menth, "Relevance estimation of cooperative awareness messages in VANETs," in *Wireless Vehicular Communications (WiVeC), 2013 IEEE 5th International Symposium on*, pp. 1–5, June 2013.

- [21] I. Cidon, R. Guerin, and A. Khamisy, "On protective buffer policies," in *INFOCOM '93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future, IEEE*, pp. 1051–1058 vol.3, 1993.
- [22] P. Hurley, J. Y. Le Boudec, P. Thiran, and M. Kara, "ABE: Providing a low-delay service within best effort," *IEEE Network*, vol. 15, no. 3, pp. 60–69, 2001.
- [23] L. Kleinrock, "A delay dependent queue discipline," *Naval Research Logistics Quarterly*, vol. 11, no. 3-4, pp. 329–341, 1964.
- [24] D. A. Stanford, P. Taylor, and I. Ziedins, "Waiting time distributions in the accumulating priority queue," *Queueing Systems*, vol. 77, no. 3, pp. 297–330, 2014.
- [25] J. Breu, A. Brakemeier, and M. Menth, "A quantitative study of Cooperative Awareness Messages in production VANETs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–18, 2014.
- [26] A. Voronov, J. De Jongh, D. Heuven, and A. Severinson, "Implementation of ETSI ITS G5 GeoNetworking stack, in Java: CAMDENM / ASN.1 PER / BTP / GeoNetworking," June 2016. DOI 10.5281/zenodo.55650.
- [27] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," *ACM SIGCOMM Computer Communication Review*, vol. 20, no. 4, pp. 200–208, 1990.
- [28] European Telecommunications Standards Institute, "EN 302 636-5-1 V1.2.0 - Basic Transport Protocol," vol. 0, pp. 1–16, 2013.

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se