A Case Study on Achieving Fair Data Age Distribution in Vehicular Communications

Xinhai Zhang[1], Xinwu Song[1], Lei Feng[1], Lei Chen[2], Martin Törngren[1]

Department of Machine Design[1]
KTH - Royal Institute of Technology, Stockholm, Sweden
Email: (xinhai,xinwu,lfeng,martint)@kth.se

Research Institute of Sweden, RISE VIKTORIA[2]
Lindholmspiren 3A, Gothenburg, Sweden
Email: lei.chen@ri.se

***Abstract*—In vehicular communication protocol stacks, received messages may not always be decoded successfully due to the complexity of the decoding functions, the uncertainty of the communication load and the limited computation resources. Even worse, an improper implementation of the protocol stack may cause an unfair data age distribution among all the communicating vehicles (the receiving bias problem). In such cases, some vehicles are almost locked out of the vehicular communication, causing potential safety risk in scenarios such as intersection passing. To our knowledge, this problem has not been systematically studied in the fields of vehicular communication and intelligent transport systems (ITS). This paper analyzes the root of the receiving bias problem and proposes architectural solutions to balance data age distribution. Simulation studies based on commercial devices demonstrate the effectiveness of these solutions. In addition, our system has been successfully applied during the Grand Cooperative Driving Challenge, where complicated scenarios involving platooning maneuvering and intersection coordination were conducted.**

I. Introduction

Cooperative driving and autonomous vehicles are two emerging trends driven by foreseen benefits such as improved transport efficiency and safety, reduced environmental impact, and enhanced productivity. Vehicular communication provides the foundation for cooperative driving, where sensor information and vehicle intent can be shared among vehicles for coordinated maneuvering [1, 2]. Two major standards for cooperative driving exist [1], namely the Dedicated Short-Range Communications (DSRC) [3] in the United States and the Cooperative Intelligent Transport Systems (C-ITS) standards [4] in the Europe. Both are based on IEEE 802.11p.

To accelerate the introduction of cooperative driving, the Grand Cooperative Driving Challenge (GCDC)[1] has been organized with competitions among international teams, each providing vehicles enabled for cooperative driving. The event, GCDC 2016 [5], took place in Helmond, the Netherlands at the end of May. The competition involved three scenarios: cooperative platoon merge (Fig. 1), cooperative intersection passing, and emergency vehicle passing. All scenarios require highly reliable vehicle communications. The KTH truck team in cooperation with Scania CV AB earned the 3rd place in the competition. Experiences with the vehicular communication and its implementation led to this paper.

Fig. 1: The platoon merging scenario of the GCDC 2016

Current research efforts on vehicular communication focus on standardization (e.g., [4, 3, 1]) and performance evaluation of the wireless links such as the media access control (MAC) mechanisms (e.g., [6, 7, 8, 9]). Less work can be found that analyzes the influence of the communication protocol implementation on the end-to-end (E2E) performance. Such analysis should consider not only the MAC level, but also the software behavior including scheduling, buffering and the time-consuming processing such as message decoding.

A vehicular communication subsystem essentially represents an open system as it is hard to estimate a realistic upper bound for the number of communicating vehicles, hence no upper bound for the required computation. Moreover, automotive embedded systems are highly resource constrained and cost sensitive [10] and also restricted by rigorous real-time requirements to guarantee for example stability and safety [11]. Therefore, it is a challenge to harness the openness of the vehicular communication subsystem with strictly restricted timing and computational budget to achieve highly reliable E2E communication performance.

This paper explores this challenge through a case study on developing the vehicular communication subsystem of the Scania truck for GCDC 2016. In this case study, the design of the communication subsystem targeted the generic cooperative driving scenario, where the number of communicating vehicles is unpredictable, compared to the limited number of vehicles in the GCDC competition. During the development, it was ob-

[1]GCDC official website: http://www.gcdc.net/en/

978-1-5090-5269-1/17/$31.00

served that the fairness of message reception was significantly affected by decoding congestion when the system was overloaded. This receiving bias problem is considered as a potential risk of safety because critical vehicles may be locked out of vehicular communication for an unacceptably long time. To the best of our knowledge, this problem has not been addressed in neither vehicular communication standards [4, 12, 3] nor previous GCDC participants [13, 14, 15, 16, 17, 18]. In this paper, we call this problem the *Receiving Bias* problem. Considering this problem, new requirements are analyzed for the communication subsystem and corresponding architectural solutions are also proposed and validated.

The rest of the paper is organized as follows: firstly, functional requirements for the communication subsystem are presented in Section II. According to these requirements, the software architecture of the communication subsystem is presented in Section III. The receiving bias problem is defined and analyzed in Section IV, which also proposes solutions to the problem. Performance metrics and analysis methods are presented in Section V and based on those, the performance of the proposed solution is studied and the results are presented in Section VI. Section VII concludes the paper and discusses potential future work.

II. Functional Requirements

In C-ITS standards, each intelligent unit within the whole transport system is defined as an ITS station (ITS-S) that can reside within e.g., cooperative vehicles, road side infrastructures, hand-held PDAs and control centers. Thus, each GCDC participant vehicle is associated with an individual ITS-S to communicate and cooperate with other participant vehicles (V2V) and specified roadside infrastructures (V2I). The cooperation should follow predefined interaction protocols to complete the tasks in all the GCDC competition scenarios.

A. Hardware Architecture of the ITS-S

To fulfill these requirements, the Scania truck was integrated with an ITS-S as shown in Fig. 2. The ITS-S contains two main Electronic Control Units (ECUs), namely the ITS-S router for vehicular communication and the ITS-S host for vehicle control. The ITS-S router exchanges messages with surrounding vehicles and roadside units wirelessly using the C-ITS communication protocol. The broadcast messages contain the vehicle state information such as speed, size and the occupied lane. To accomplish this, the ITS-S host sends the ego vehicle state information to the ITS-S router where the information is broadcast to the others. The received messages are transmitted to the ITS-S host through Ethernet to provide better awareness of the neighborhood. Given the information of the ego vehicle and its neighborhood, the ITS-S host controls the longitudinal behavior of the truck by sending control commands to its CAN bus. The ego vehicle state information is acquired from the CAN bus and an external Global Positioning System (GPS). For security reasons and also to protect the timing of messages on the CAN bus, the connection between the ITS-S host and the CAN bus had to be protected by a provided Scania CAN gateway.

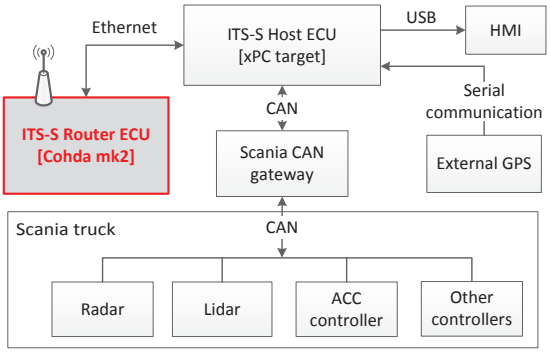

Fig. 2: Hardware architecture of the participant truck

This paper focuses on the software development of the ITS-S router. The hardware of this ECU uses the Cohda mk2 on board unit (OBU),[2] a commercial solution for wireless vehicular communication. It has a Freescale i.MX35 ARM-11 MCU @ 533MHz as the main processor, 64MB SDRAM as the main memory and all the necessary peripherals.

B. Requirements for the ITS-S Router

In this GCDC competition, there were three types of messages that needed to be transmitted over the air, namely the Cooperative Awareness Messages (CAMs), the Decentralized Environmental Notification Messages (DENMs) and the i-GAME Cooperative Lane Change Messages (iCLCMs). CAM and DENM are standard messages from C-ITS for cooperative driving while iCLCM is a separate message designed based on the scenario requirements of GCDC 2016. Brief descriptions of these messages are given as follows:

- **CAM** is a periodic message to broadcast basic information of the ego vehicle such as its position, speed, heading and shape.
- **DENM** is an event triggered message for emergent notifications such as road construction or hard brake.
- **iCLCM** is another periodic message defined only for GCDC 2016. It contains scenario specific information that is not included in CAM or DENM, e.g., the request and confirmation of platoon merge.

Due to the requirements from platoon controllers to perform complicated maneuvering such as platoon merge and intersection coordination, a frequency of 25 Hz (higher than the 10 Hz of the C-ITS standards) was used for sending CAMs and iCLCMs. Moreover, participants did not need to send DENMs. They only needed to receive DENMs from road side units to understand the current scenario.

The implementations of both the ITS-S host and router needed to follow the C-ITS layered reference architecture [12], which is an extension of the ISO OSI model [19]. The layers are illustrated at the top of Fig. 3. The application layer focuses

[2]CohdaWireless: http://cohdawireless.com/Products/Hardware.aspx

on the cooperation strategies. It was fully implemented on the ITS-S host. The facility layer provides application support and communication support. The communication support was implemented on the ITS-S router. Fig. 3 also presents the necessary functions of the software and maps them to corresponding layers by sharing the same background patterns.

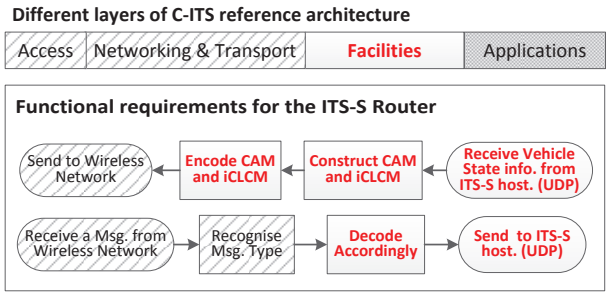

Fig. 3: Functional requirements of the communication system

The software on the ITS-S router implemented the C-ITS communication protocol stack. The access, networking and transport layers were implemented by directly reusing the commercial libraries provided by CohdaWireless, where limited modifications could be made. The facility layer is mainly responsible for encoding and decoding messages with Packed Encoding Rules (PER) [20]. This layer is in focus of this paper since the decoding function was found to be the computation bottleneck during the development. In our implementation, the C code of the PER encoding and decoding algorithms for different message types was automatically generated by the open source code generator, ASN.1 C,[3] with the data structure of the messages as input.

III. Software Architecture of the ITS-S Router

Based on the aforementioned requirements, the software on the facility layer for message receiving and sending was designed as shown in Fig. 4, where each light gray block represents a thread. Detailed explanation and motivation of this design is given in the rest of this section.

A message received by the facility layer will be decoded according to its type and transmitted to ITS-S host through UDP. On the ITS-S host, this message is used to update the local dynamic map and to make control decisions. The ITS-S router also periodically receives the vehicle state information from the ITS-S host to construct and broadcast CAMs and iCLCMs. Except the message receiving thread, all the other threads residing in the facility layer have the same priority and are FIFO scheduled. The message receiving thread is the interface (a callback function) from the lower layer. According to the architecture design of Cohda mk2, this thread has a higher priority than all the other threads in the higher layers.

On the receiving side, the message decoding functions were isolated into individual threads since they are time-consuming due to the inherent complexity of PER. Moreover,

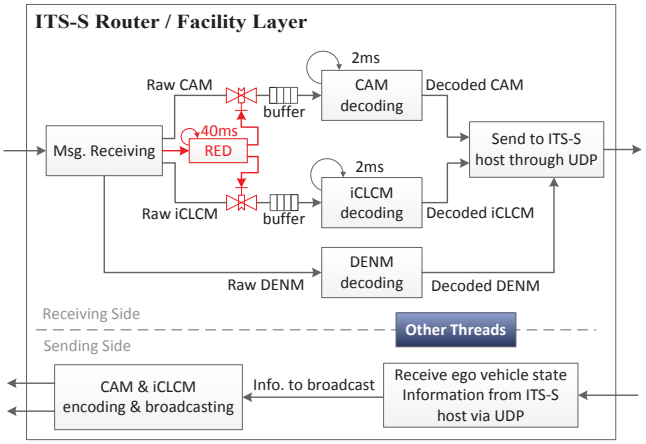

Fig. 4: Thread structure of the facility layer

the generated code for the encoding and decoding algorithms is immature for real-time applications. It does not conform to MISRA C.[4] For example, functions with unpredictable behaviors (e.g., malloc) are called iteratively without proper depth protection. This immaturity resulted in the unstable execution time of the decoding function. It was observed that in most of the cases, the execution time was less than 1 ms. However, in some cases, the execution time of one decoding even exceeded 20 ms. We have also tried to allocate the decoding functions in the message receiving thread. In that case, the whole system often crashed unpredictably, especially when communicating with many vehicles. The burstiness of the message receiving thread, which was time-consuming and high-priority, caused an unacceptable number of deadline misses in other critical system tasks.

Due to the open feature of the cooperative driving system, the upper bound for the communication load of one ITS-S router can hardly be estimated. The communication range specified by the ITS standard is from 300 to 500 meters according to different scenarios. On a busy road with 8 lanes, assuming one vehicle every 20 meters per lane, there can be approximately 400 vehicles within the communication range. Furthermore, 1) cooperative driving can reduce the distance between vehicles. 2) If roundabouts, intersections and overpasses are taken into account, this number can be even higher. 3) Road-side infrastructure and pedestrian ITS-Ss [12] can also increase the communication load. Therefore, it is difficult to design the ITS-S router based on the worst case. The final system has to be tolerant to the overload situation. To this end, the computational resource utilization of message processing tasks has to be restricted, especially the time-consuming decoding functions.

In order to restrict the computational resource for the decoding functions, the decoding threads were designed to be periodic (with period T_{dec}) and only one message is decoded within each period. It was observed that when T_{dec} was set to be 2 ms, for most of the time, all the threads

[3]ASN.1 C: http://lionet.info/asn1c/compiler.html

[4]MISRA: https://www.misra.org.uk/MISRAHome/tabid/55/Default.aspx

in the facility layer did not miss their deadlines. In this situation, the CPU demand for the decoding is restricted since the maximum number of decoded messages within a given time interval is deterministic. Even though unexpected increases of the decoding time might temporary compromise the schedulability of the threads in the facility layer, they would not crash the system. All the vehicles send CAMs and iCLCMs periodically (with T_{msg} denoting the period). Therefore, the maximum number of communicating vehicles that one ITS-S router can completely handle can be calculated as $n_{max} = \lfloor (T_{msg}/T_{dec}) \rfloor$. For example, in the competition, T_{msg} was prescribed to be 40 ms and T_{dec} was designed to be 2 ms as shown in Fig. 4. Therefore, $n_{max} = 20$. If there were more than 20 vehicles communicating at the same time, the system would be overloaded and not all the received messages could be decoded. n_{max} and T_{dec} should be determined by the computation power of the router and the timing performance of its software. Decreasing the decoding time by applying a more powerful processor or adding a new ECU particularly for message processing can enlarge n_{max} but cannot totally avoid the overload situation since the upper bound of the communication load can hardly be guaranteed.

Decoding congestion is commonly caused by message burst, which means that many messages are received over a relatively short period. To solve this burstiness, received messages are buffered before being decoded. The maximum length of the buffers l_B should equal to n_{max} to fully utilize the computation budget for message decoding. If $l_B < n_{max}$, some messages have to be discarded even though the processor still has time to decode them. If $l_B > n_{max}$, when the system is overloaded, un-decoded messages in the buffer will lead to accumulative delay for message processing.

Assuming all the threads are schedulable, the periodicity of the decoding threads guarantees an upper bound for the computation load of the router during run-time. However, it gives rise to an unexpected phenomenon, which is defined as the receiving bias problem in the following section. The Random Early Discard (RED) [21] thread is used to solve this problem.

IV. The Receiving Bias Problem

When the number of communicating vehicles n exceeds $l_B = n_{max}$, the buffer before decoding will overflow and consequently some messages have to be discarded. Our observation from simulation showed that most of the discarded messages were sent from the same set of vehicles no matter the buffer being a queue (FIFO) or a stack (FILO). It implied that information from those vehicles was updated much less frequently than that of the others. In other words, the communication with those vehicles was more or less "locked out" from the application layer.

As observed, the bias became even worse when the vehicle was running in the following situations:

- The period of all the message sending is constant.
- The networking & Transport layers of the ego vehicle can successfully receive all messages from all the communicating vehicles.
- The decoding threads never miss their deadlines.

A. The Reasons

Since CAMs and iCLCMs are received and decoded following the same procedure, only the receiving bias problem of CAMs is analyzed in the rest of this section. Assuming the schedulability of all the threads on the ITS-S router, including the decoding threads, the message flows of CAMs can be modeled as presented in Fig. 5. A CAM is discarded from the buffer iff the buffer is full when the CAM is received by the facility layer. In order to investigate the reason of the receiving bias problem, the buffer utilization (i.e., the number of messages in the buffer) and the message reception have been modeled as a function of time.

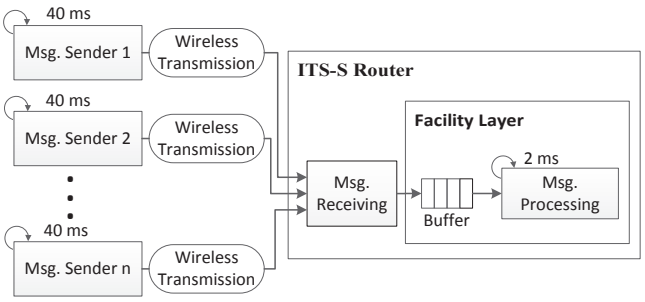

Fig. 5: Message flows of vehicular Communication

One CAM is taken from the buffer for decoding every T_{dec}. Assuming that it is taken at the very beginning of each T_{dec}, the buffer utilization $u_B(j)$ at the time instance $(T_{dec} \times j)$ can be analyzed as:

$$u_B(j) = min\Big(max\big(u_B(j-1) + n_{rx}(j) - 1, 0\big), l_B\Big) \quad (1)$$

where $n_{rx}(j)$ represents the number of received messages within the jth T_{dec}, i.e., $\big(T_{dec} \times (j-1), T_{dec} \times j\big]$. $n_{rx}(j)$ can be calculated as:

$$n_{rx}(j) = \sum_{k=1}^{n} b_k^{msg}(j) \quad (2)$$

where $b_k^{msg}(j) \in \mathbb{B}$ denotes whether a message from vehicle k is received within the jth T_{dec}. Each vehicle sends a CAM every T_{msg} and T_{dec} should be much smaller than T_{msg}. Let i be the index of T_{msg}. The jth T_{dec} can be either entirely placed within the ith T_{msg} or partly placed at the beginning of the ith T_{msg} and the rest at the end of the $(i-1)$th T_{msg}. Let $t_{i,k}^{msg}$ denote the receiving time of $msg_{i,k}$ (sent from vehicle k at the ith T_{msg}). When $j > 1$:

$$b_k^{msg}(j) = \begin{cases} 1 & \text{if} \quad T_{dec} \times (j-1) < t_{i,k}^{msg} \leq T_{dec} \times j \\ 1 & \text{if} \quad T_{dec} \times (j-1) < t_{i-1,k}^{msg} \leq T_{dec} \times j \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$\text{with} \begin{cases} i & = \lfloor \frac{T_{dec} \times j}{T_{msg}} \rfloor \\ t_{i,k}^{msg} & = T_{msg} \times i + \delta_k + \xi_{i,k} \end{cases}$$

where $T_{msg} \times i + \delta_k$ denotes the sending time instance of $msg_{i,k}$. Since not all the vehicles send CAMs at the same time, δ_k refers to the offset of the sending time of messages from vehicle k within each T_{msg}. δ_k is fixed during run time. $\xi_{i,k}$ represents the jitter time elapsed from the generation of $msg_{i,k}$ to its reception by the facility layer of the ego vehicle. Therefore, $\xi_{i,k}$ is mainly determined by the wireless transmission delay D^t. Besides D^t, the wireless transmission also introduces a message loss ratio lr_k^{tr}. Physically, it is difficult to guarantee that all the broadcast messages can be received successfully (i.e., $lr_k^{tr} = 0$) and timely (i.e., $D^t = 0$) due to 1) the increasing rate error caused by obstacles [8], distance and speed difference [9] between the vehicles; 2)the congestion at MAC layer [6] and 3) the failures in the communication systems or other interference to the wireless transmission. Further analysis of lr_k^{tr} and D^t can be found in [6]. Moreover, as prescribed by the C-ITS standards [4], the sender side congestion control methods may reduce the sending frequency in particular situations, e.g., when there are no big changes in the position and speed of the ego vehicle. For simplicity, this is also considered as a change of lr_k^{tr}.

According to Eq. (2) and (3), $n_{rx}(j)$ changes periodically, without considering the impact of $\xi_{i,k}$ and lr_k^{tr}. Eq. (1) reveals that given l_B, $u_B(j)$ is determined by $n_{rx}(j)$. Therefore, $u_B(j)$ is also periodic regardless of the wireless transmission. The least common multiple of T_{msg} and T_{dec} is their hyper-period. Within each hyper-period, the CAMs from the same set of vehicles are discarded due to buffer overflow, causing unfair message discard.

Let dr_k denotes the ratio of discarded messages from vehicle k due to buffer overflow at the ego vehicle. The receiving bias problem can be interpreted as the unfair distribution of the dr_k on al the communicating vehicles. dr_k and lr_k^{tr} compose the total CAM loss ratio lr_k for vehicle k as showen in Eq. (4) where m_k^{dis} and m_k^{rec} refer to the total number of discarded and received CAMs by the facility layer from vehicle k.

$$\begin{aligned} lr_k &= 1 - (1 - lr_k^{tr}) \times (1 - dr_k) \\ \text{where} \quad dr_k &= \frac{m_k^{dis}}{m_k^{rec}} \end{aligned} \tag{4}$$

As analyzed above, when the ITS-S router is overloaded ($n > l_B$), the unfair distribution of dr_k is attributed to the periodicity of n_{rx} in Eq. (2). lr_k^{tr} and $\xi_{i,k}$ will stochastically affect this periodicity and thereby mitigate the receiving bias problem. If the decoding function cannot finish before its deadline in some periods, T_{dec} will be enlarged in that period and consequently affects the corresponding $n_{rx}(j)$. However, as long as the variances of lr_k^{tr}, $\xi_{i,k}$ and the decoding execution time are within certain ranges, the receiving bias problem cannot be totally avoided. The influence of $\xi_{i,k}$ and T_{dec} on the receiving bias problem is analyzed in Section VI.

The reason of the receiving bias problem is very similar to that of the TCP global synchronization problem [22]. Even though these two problems have different consequences, they are both attributed to the deterministic discarding of messages when the buffer is temporarily full. For the receiving bias problem, the sender side congestion control methods can somehow mitigate the unfairness by varying the sending periods. However, they cannot guarantee a total avoidance of the problem. One exceptional situation is where all the vehicles are accelerating at the same time. In this situation, no sender side congestion control methods can be triggered. Therefore, a corresponding solution is necessary on the receiver side.

B. Implemented Solution Based on RED

As discussed before, the receiving bias problem has a similar reason of the TCP global synchronization problem. A standard receiver side solution for the TCP global synchronization is active buffer management with Random Early Discard (RED) [21]. In this method, before received packets get enqueued, they are actively discarded according to a dynamically calculated discard probability. This discard probability is calculated based on the average buffer utilization and the prescribed buffer utilization threshold.

Inspired by this method, a look-ahead RED method was proposed to solve the receiving bias problem. Owning to the periodicity of the messages, the behaviors of the senders are much more predictable than those in general TCP communication. Additionally, the capacity of message processing is known. Therefore, our RED does not close the loop by monitoring the buffer utilization. Instead, it is open-loop and feed-forward. It adjusts the discard probability p_d according to an estimated upcoming communication load. Since each communicating vehicle sends messages every 40 ms, without considering the transmission loss ratio, n messages will be received within any 40 ms time window. Therefore, the RED thread in Fig. 4 periodically estimates how many messages will be received in the next 40 ms. Let $\hat{m}_{i+1}$ be the estimated number of received messages in the $(i+1)$th period. It determines the discard rate p_d for the RED method. In our implementation, $\hat{m}_{i+1} = m_i$ where m_i denotes the measured number of received messages within the ith period. Other estimation model can also be used here and may probably provide better performance. However, the detail of the estimation model is not the focus of this paper.

Given $\hat{m}_{i+1}$, p_d is calculated in the RED thread as shown in Eq. (5) to guarantee that only n_{max} CAMs will be buffered for decoding. The calculated p_d is sent to the message receiving thread where each received message has the probability p_d to be actively discarded before being buffered. Since messages are discarded without knowing their senders, p_d is equal for all the vehicles.

$$p_d = \begin{cases} 0 & \text{if} \quad \hat{m}_{i+1} \leq n_{max} \\ \frac{\hat{m}_{i+1} - n_{max}}{\hat{m}_{i+1}} & \text{if} \quad \hat{m}_{i+1} > n_{max} \end{cases} \tag{5}$$

This RED based solution can only guarantee a fair distribution of the message discard ratio dr_k, which still increases with the number of communicating vehicles n. In this case, the system may become unsafe when dr_k is high, since vehicle k can be critical e.g., the one in front of the ego vehicle.

This safety issue can be solved by designing the system based on a sufficient and reasonable upper bound of the

number of communicating vehicles (e.g., 1000 vehicles). However, this over-provisioning approach has an unacceptable cost and is virtually unnecessary for the following reasons. First, not all the vehicles are equally important. dr_k should be guaranteed to be zero only for the critical messages (i.e., the messages from safety critical vehicles). For the non-critical messages, a proper dr_k should be acceptable. In addition, a certain dr_k for the non-critical messages will also decrease the computation load on the application layer. Furthermore, the worst case communication load can be rarely reached since sender side congestion control methods will be triggered in most of the busy traffic scenarios. For example, during traffic jams, all the vehicles are moving slowly [23]. If plenty of vehicles are platooning together, the speeds of all these vehicles should be relatively stable [2]. In both of these cases, the message frequency will be reduced according to C-ITS standards. Therefore, the design of the ITS-S router should not be based on the worst case.

C. Proposed Solution Considering Safety

If critical vehicles can be identified in all the scenarios and the number of the critical vehicles is limited, a compromising solution is to guarantee strict timing requirements for the critical messages and to guarantee fairness for the non-critical vehicles. Fairness is important for the non-critical vehicles because "locked-out" vehicles will lose the chance to be marked as critical timely.

The criticality of a message cannot be identified on the facility layer, where only MAC address and an approximate position of its sender are known. More useful information to judge the criticality is only available on the application layer. Therefore, a cross-layered decoding congestion control mechanism is necessary to implement this solution. As shown in Fig. 6, the application layer suggests the facility layer with a fixed and relatively small number of critical vehicles (actually their MAC addresses). For example, in the highway scenario, the vehicles within the perception range of the ego vehicle can be marked as critical. On the facility layer, the decoding of the critical messages uses a separated buffer that is not managed by the RED. The size of the buffer for critical messages should be n^C_{max}, which is the maximum number of critical vehicles in any scenarios. The RED method is only employed for the non-critical messages. The decoding thread decodes non-critical message only when the buffer for the critical messages is empty, so as to guarantee the timing performance for the critical messages. In this case, this software architecture guarantees safety and also enables the tradeoff between QoS and cost.

When the system starts up, the application layer does not have enough information to identify critical vehicles. If all the messages are non-critical, the complete RED based solution introduced in Section IV-B has to be employed until the facility layer receives critical vehicle information from the application layer.

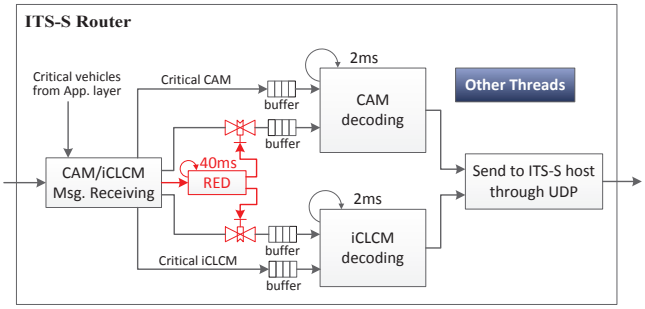

Fig. 6: Proposed software architecture for message reception

Though the cross-layered decoding congestion mechanism was not adopted in our ITS-S router, its performance is analyzed in the next section.

V. Performance Analysis Methods

In order to analyze the receiving bias problem, and to evaluate the performance of the implemented solution (introduced in Section IV-B) and the proposed solution (introduced in Section VI-C), this section firstly introduces and analyzes the performance metrics. These metrics were measured for each solution on a simulation model that considered scheduling, buffering and transmission delays.

A. Communication Performance Metrics

Llatser [6] selected four metrics to evaluate vehicular communication performance from the standardization perspective. The ones related to the implementation of the communication subsystem were chosen with minor modification to evaluate the performance of our ITS-S router and to quantify the severity of the receiving bias problem. They are **communication processing (CP) delay** and **data age** as defined as follows:

- The **CP delay** for message sending D^{txP} refers to the time elapsed from the message generation on the ITS-S host to its transmission to the wireless networks. The **CP delay** for message receiving D^{rxP} measures the time span between the message reception from the air to its reception by ITS-S host.
- The **data age** A_k reflects the freshness of the information received from vehicle k. During run time, A_k equals to the time elapsed since the generation of the last successfully received message from vehicle k by the application layer of the ego vehicle [7].

The CP delay is affected by the computation power of the ITS-S router and its software implementation, e.g., the efficiency of the message decoding algorithm and the software architecture. The data age is determined by the transmission time of the message over the air, the CP delay, the transmission period of the message and the number of lost messages.

Since the decoding threads are periodic, the scheduling interference from message decoding to the other functions is mostly fixed. Therefore, the CP delay for message sending D^{txP} will not be affected by the uncertain communication load and it is therefore not the focus of this paper.

In order to analyze the impact of the implementation on the CP delay $D^{rxP}(i)$ for any received CAM msg_i and the worst case $D^{rxP}(i)$ for critical messages, correlated to our implementation, $D^{rxP}(i)$ can be decomposed as:

$$D^{rxP}(i) = D^{rec}(i) + D^{buf}(i) + D^{dec}(i) + D^{wait}(i) \quad (6)$$

where $D^{rec}(i)$ refers to the execution time of the message receiving thread to pop a message msg_i up from physical layer to facility layer. $D^{buf}(i)$ denotes the time that msg_i stays in the buffer. $D^{dec}(i)$ is the execution time of the CAM decoding thread to decode msg_i and $D^{wait}(i)$ represents the time that the decoding thread waits in the scheduling queue. According to the experimental data from the real implementation, D^{rec} is assumed to be constant for all the messages. As mentioned before, D^{dec} is not stably less than T_{dec} due to the immature implementation. In order to achieve a generic analysis of this problem, the rest of this section ignores this immaturity and assumes that all the threads are schedulable.

The worst case D^{wait} is predictable with the method in [24]. If the critical messages are buffered separately, since the critical messages are decoded before the non-critical messages, the worst case buffering time $WCD^{buf}(i)$ for an critical message msg_i happens when msg_i is at the end of the queue and one non-critical message is being decoded. Therefore,

$$WCD^{buf}(i) = T_{dec} \times (1 + n^C) \quad \text{If } msg_i \text{ is critical}$$

where n^C is the maximum number of critical vehicles in current scenario. The size of the buffer for critical messages is n^C_{max}, which is the maximum n^C for any scenarios. In each period, all the critical messages have to be decoded. Therefore, the design should guarantee $T_{dec} \times (1 + n^C_{max}) < T_{msg}$. In this case, the worst case CP delay for all the critical messages is predictable.

Data age A_k is a comprehensive metric considering transmission delay, CP delay and message loss. Therefore, A_k is the main metrics used in this paper to evaluate the communication performance and to quantify the severity of the receiving bias problem. According to its definition, during off-line analysis, $A_k(i)$ of msg_i is measured from the "birth" of msg_i to its "death" [7]. The "birth" refers to the time instance when msg_i is generated in vehicle k and the "death" refers to the instance when the application layer receives the next CAM msg_j from vehicle k. Therefore, $A_k(i)$ can only be measured for received messages and is ranged as:

$$A_k(i) \in [D(i), D(j) + T_{msg} \times (1 + m_l)] \quad (7)$$

where $D(i)$ represents the entire delay of msg_i from its generation to its reception by the application layer of the ego vehicle. It equals to the sum of the CP delay D^{txP} from the sending vehicle k, the wireless transmission delay D^t [6] and the receiving CP delay of the ego vehicle D^{rxP} as $D(i) = D^{txP}_k + D^t(i) + D^{rxP}(i)$. D^{txP}_k and $D^t(i)$ compose the jitter $\xi_{i,k}$ in Eq. (3). m_l in Eq. (7) refers to the number of lost messages from vehicle k after the reception of msg_i and before the next successful message reception. Average data age $A^{mean}_k = \mathbf{E}(A_k(i))$ for the communication with vehicle k is defined as the mean value of data ages of all the received messages from vehicles k.

The concerned metrics regarding data age are the average A^{mean}_k among all the vehicles $\mathbf{E}(A^{mean}_k)$ and its standard division (STD) $\sigma(A^{mean}_k)$. A^{mean}_k reflects the average communication performance, and $\sigma(A^{mean}_k)$ reflects the receiving fairness to all the communicating vehicles. A large $\sigma(A^{mean}_k)$ implies that the communication with some particular vehicle(s) is much worse than the communication with others. Therefore, $\sigma(A^{mean}_k)$ is used to evaluate the severity of the receiving bias problem. For the critical vehicles, worst case data age $WCA_C = max(\{A_k(i)|k \in C\})$ should be considered where C is the set of all the critical vehicles.

As analyzed in section IV-A, δ_k, $\xi_{i,k}$ and T_{dec} have stochastic influences on the communication performance and the receiving bias problem. In order to study these influences, a simulation based analysis was conducted in the next section.

B. Simulation Model

Restricted by our experimental equipments, at most 20 communicating vehicles can be simulated during the experiments with real systems. With this setting, the receiving bias problem can hardly be observed. Instead, simulation based studies were conducted to analyze the communication performance and the receiving bias problem based on the following assumptions:

- Since DENM is rarely used, the processing of DENM is out of consideration,
- Context switch takes no time,
- The execution time of all the threads are constant except the decoding threads,
- Other unknown interference due to incomplete configuration can be simulated by a periodic noise thread,
- For all the communicating vehicles, $lr^{tr}_k = 0$ so as to only focus on the influence of the software behavior.

The concepts introduced in [25] were used to establish the simulation model considering timing, scheduling, buffering, and uncertainties from wireless transmission and execution times of different tasks. Detailed description and validation of this simulation model is provided in Appendex A.

VI. Experimental Results and Discussion

According to the analysis in the previous sections, the following questions still remain to be verified.

- To what extent will the receiving bias problem $\sigma(A^{mean}_k)$ be mitigated by the uncertainties from the transmission jitters $\xi_{i,k} = D^{txP}_k + D^t(i)$, the decoding execution time $D^{dec}(i)$ and the scheduling?
- To what extent will the RED based solution improve the fairness $\sigma(A^{mean}_k)$?
- Will the type of buffer (queue or stack) affect the communication performance ($\mathbf{E}(A^{mean}_k)$ and $\sigma(A^{mean}_k)$)?
- For the proposed solution considering critical vehicles, will the worst case performance of the critical messages be guaranteed regardless of the communication load?

This section presents the simulation based experiments to answer these questions. The first experiment (Section VI-A) studied the influence of transmission delay $\xi_{i,k}$ on the receiving bias problem. The second experiment (Section VI-B) verified the implemented RED based solution and compared the performance between stack buffer and queue buffer. This experiment also examined the influence of the variant $D^{dec}(i)$ on the receiving bias problem $\sigma(A_k^{mean})$. The third experiment (Section VI-C) simulated the proposed solution considering critical vehicles.

In all of these experiments, the considered communication load n was ranged from 15 to 30 communicating vehicles. In each simulation, the number of communicating vehicles was fixed. 20 simulations were conducted for each number of communicating vehicles. After each experiment, the average values of the concerned matrices among the 20 simulations were plotted. Each simulation ran for 200 seconds simulation time. δ_k was pre-assigned for each simulation following the uniform distribution $\mathcal{U}(0, T_{msg})$. Within each simulation, δ_k was fixed for vehicle k. Other important perimeters were the same as the implementation such as T_{msg} =40 ms, $T_{dec} = 20$ ms and $l_B = 20$ for both CAM and iCLCM.

A. Analysis of The Receiving Bias Problem

In order to analyze the influence of $\xi_{i,k}$ on $\sigma(A_k^{mean})$, in this experiment, $D^{dec}(i)$ was assumed to be fixed and all the threads were schedulable. In each simulation, $\xi_{i,k}$ was independent with each other and followed gamma distribution $\Gamma(\alpha, \beta)$. When $\xi_{i,k} = 0$, some of the vehicles were totally "locked out", and $\sigma(A_k^{mean})$ was extremely high. Five combinations of α and β were adopted in this experiment to investigate the influence of the variance of $\xi_{i,k}$ on $\sigma(A_k^{mean})$. In all the simulations, $\mathbf{E}(\xi_{i,k}) = 2$ ms. $\sigma(\xi_{i,k})$ changed in different simulations.

Seen from the result in Fig. 7, the receiving bias problem was obvious and inevitable when the RED based solution was not adopted. A big variance of the transmission delay can mitigate the receiving bias problem, which however cannot be totally avoided.

B. Performance of the RED Based Solution

This experiment compared the performance between the implementations with and without RED in terms of $\mathbf{E}(A_k^{mean})$ and $\sigma(A_k^{mean})$. Stack buffer and queue buffer were also compared on the implementation with RED. In order to figure out the influence of $D^{dec}(i)$ on $\sigma(A_k^{mean})$, this experiment assumed that $\xi_{i,k} = 0$ and $D^{dec}(i)$ in the simulation followed the same distribution as the measured ones.

Figure 8 shows the result of the comparison. Despite the unstable decoding time, without the employment of RED, the receiving bias problem was very severe because of the huge $\sigma(A_k^{mean})$, when the router was overloaded. This problem became increasingly worse as n increased. With the help of RED, the average data age ($\mathbf{E}(A_k^{mean})$) was improved, and the receiving bias problem ($\sigma(A_k^{mean})$) was significantly

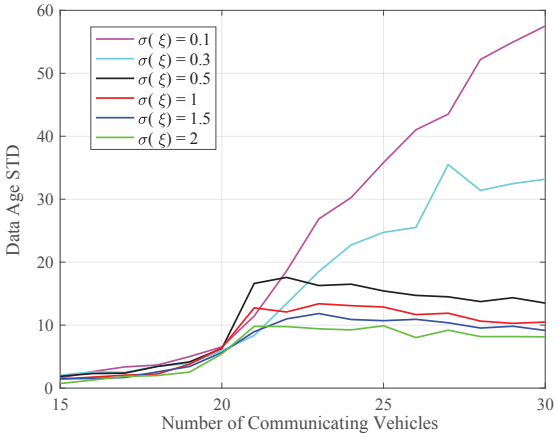

Fig. 7: Influence of transmission delay on the fairness

mitigated. Even though the average data age gradually increased with n due to the restriction from computation power, the fairness of message reception ($\sigma(A_k^{mean})$) was no longer sensitive to the growth of n. Besides, Fig. 8 also reveals that the stack buffer brought better average data age than the queue buffer, however, it introduced more unfairness.

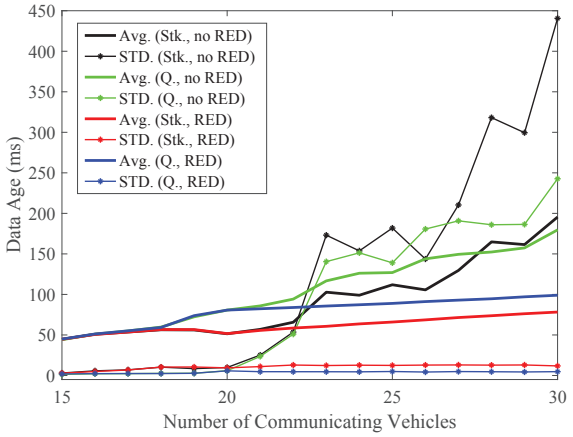

Fig. 8: Average value and STD of data age of all the communicating vehicles in different configurations

As illustrated in Eq. (7), data age is mainly determined by CP delay $D^{rxP}(i)$, transmission delay $\xi_{i,k}$ and message loss ratio lr_k. In this experiment, $\xi_{i,k} = 0$ and the transmission loss ratio $lr_k^{tr} = 0$ hence $lr_k = dr_k$ according to Eq. 4. Since almost all the lost messages were discarded by RED, the main reason for the different data ages between the implementations with different buffer types was CP delay. As shown in Fig. 9, queue buffer introduces higher but fairer CP delay. This is consistent with the observation on data age shown in Fig. 8.

C. Performance of The Proposed Solution

In this experiment, the number of critical vehicles was assumed to be $n^C = 6$. Both critical and non-critical messages

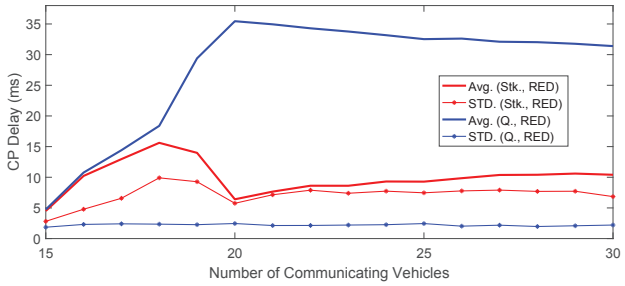

Fig. 9: Average value and STD of CP delays of all the received messages using different buffer types

were buffered in queues. The length of the queues for critical CAMS and iCLCMs equaled to n^C. For the non-critical messages $l_B = n_{max} - n^C = 14$. $D^{dec}(i)$ was assumed to be fixed and all the threads were schedulable. $\xi_{i,k}$ was still assumed to follow gamma distribution $\Gamma(40, 0.05)$ with $\mathbf{E}(\xi_{i,k}) = 2$ and $\sigma(\xi_{i,k}) = 0.1$. Fig. 10 illustrates the average data age of the messages from critical and non-critical vehicles and the worst case data age of critical vehicles.

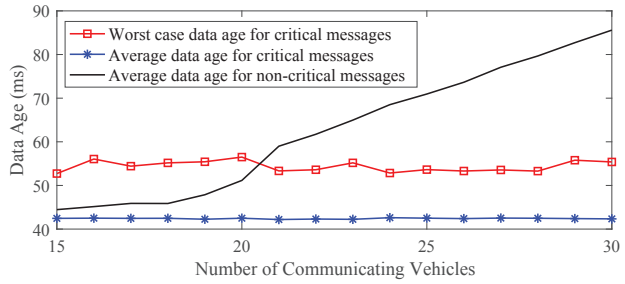

Fig. 10: Performance comparison between critical and non-critical messages

Fig. 10 shows that the average and the worst case data age for critical messages were not affected by the number of communicating vehicles. When the system was overloaded ($n > 20$), the average data age of non-critical messages increased as the number of communicating vehicles increased.

VII. Conclusion and Future Work

Despite extensive efforts on improving the communication performance at the wireless links, the decoding congestion on the facility layer is found to be a potential bottleneck for E2E communication performance in vehicular communication. Due to the limited computation budget, decoding is designed to be periodic. While this design limits the computation load, it introduces the receiving bias problem, leading to extremely unfair data age distribution among messages from different vehicles. This significantly deteriorates the E2E communication performance, introducing risky situations. This paper, among the first of such works, analyzes this problem and presents architectural solutions. A RED based mechanism is introduced to actively manage the buffer before decoding to mitigate the unfair data age distribution. Separated buffers can also be used for critical messages to guarantee their worst case performance. Simulation studies were conducted to verify the solutions. In addition, the RED based solution was successfully applied in a Scania truck, with which the KTH truck team won the third prize during the GCDC 2016.

GCDC 2016 involved a small number of vehicles. There were only ten vehicles participating the competition. Including the two pace cars to supervise the competition, there were in total twelve ITS-Ss involved in the vehicular communication. In addition, messages could also be lost during wireless transmission. Therefore, the receiving bias problem rarely occurred during the competition. So far, this problem was mostly discovered in simulation and the proposed solutions have not been fully verified in practice. Therefore, it is necessary to have a co-simulation considering the behaviors on both software level and lower levels, such as MAC, to achieve deeper investigation of this problem. In addition, realistic testing involving a larger number of vehicles can be considered to test the performance of potential solutions.

As analyzed, the receiving bias problem only occurs on the bottleneck layer, whose throughput is smaller than that of the layer below. In our case, the facility layer is the bottleneck layer since the decoding function is time consuming. If time consuming algorithms are adopted in the application layer for decision making, the bottleneck may be lifted up to the application layer. If the bottleneck layer cannot be easily identified during design time, the receiving bias problem should be considered on all the potential bottleneck layers.

Besides IEEE 802.11p, vehicular communication is also one of the application areas of the fifth generation of cellular communications (5G). 5G brings increased bandwidth allowing much more information to be exchanged between more vehicles with even more stringent E2E latency. The new features will further push the bottleneck to computational-intensive functions such as decoding. As analyzed in Section IV-A, the stringent latency will further deteriorate the fairness if RED is not employed. Therefore, the receiving bias problem is inevitable and calls for sophisticated solutions.

In addition, the following suggestions are proposed from our experience as future work: (1) It is necessary to have an analytical timing model for the entire vehicular communication process considering all the stochastic uncertainties. (2) A more sophisticated RED method can be proposed taking feedback information into account such as the buffer utilization and the individual loss ratio of the messages from each communicating vehicle. (3) The standardization institute should consider the receiving bias problem and specify corresponding requirements to the implementation of the protocol stack. Fixed number of critical vehicles should be defined for all the cooperative driving scenarios. Strict E2E timing requirements should be specified for the critical messages.

References

[1] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, “Enhancements of V2X commu-

nication in support of cooperative autonomous driving," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 64–70, dec 2015.

[2] S. Behere, M. Törngren, and D.-J. Chen, "A reference architecture for cooperative driving," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1095–1112, nov 2013.

[3] J. B. Kenney, "Dedicated Short-Range Communications (DSRC) Standards in the United States," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, jul 2011.

[4] A. Festag, "Cooperative intelligent transport systems standards in europe," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 166–172, dec 2014.

[5] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff, "The Grand Cooperative Driving Challenge 2016: boosting the introduction of cooperative automated vehicles," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 146–152, aug 2016.

[6] I. Llatser, A. Festag, and G. Fettweis, "Vehicular communication performance in convoys of automated vehicles," in *2016 IEEE Inter. Conf. on Communications (ICC)*. IEEE, may 2016, pp. 1–6.

[7] A. Vinel, L. Lan, and N. Lyamin, "Vehicle-to-vehicle communication in C-ACC/platooning scenarios," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 192–197, aug 2015.

[8] R. Meireles, M. Boban, P. Steenkiste, O. Tonguz, and J. Barros, "Experimental study on the impact of vehicular obstructions in VANETs," in *2010 IEEE Vehicular Networking Conf.* IEEE, dec 2010, pp. 338–345.

[9] J. Yin, T. ElBatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty, "Performance evaluation of safety applications over DSRC vehicular ad hoc networks," in *Proceedings of the first ACM workshop on Vehicular ad hoc networks - VANET '04*. New York, New York, USA: ACM Press, 2004, p. 1.

[10] S. Chakraborty, M. A. Al Faruque, W. Chang, D. Goswami, M. Wolf, and Q. Zhu, "Automotive Cyber-Physical Systems: A Tutorial Introduction," *IEEE Design & Test*, vol. 33, no. 4, pp. 92–108, aug 2016.

[11] P. Axer, C. Rochange, M. Sebastian, R. V. Hanxleden, R. Wilhelm, W. Yi, R. Ernst, H. Falk, A. Girault, D. Grund, N. Guan, B. Jonsson, P. Marwedel, and J. Reineke, "Building timing predictable embedded systems," *ACM Trans. on Embedded Computing Systems*, vol. 13, no. 4, pp. 1–37, mar 2014.

[12] ETSI EN 302 665 V1.1.1, "Intelligent Transport Systems (ITS); Communications Architecture," Tech. Rep., 2010.

[13] A. Geiger, M. Lauer, F. Moosmann, B. Ranft, H. Rapp, C. Stiller, and J. Ziegler, "Team AnnieWAY's Entry to the 2011 Grand Cooperative Driving Challenge," *IEEE Trans. on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1008–1017, sep 2012.

[14] K. Lidstrom, K. Sjoberg, U. Holmberg, J. Andersson, F. Bergh, M. Bjade, and S. Mak, "A Modular CACC System Integration and Design," *IEEE Trans. on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1050–1061, sep 2012.

[15] R. Kianfar, B. Augusto, A. Ebadighajari, U. Hakeem, J. Nilsson, A. Raza, R. S. Tabar, N. V. Irukulapati, C. Englund, P. Falcone, S. Papanastasiou, L. Svensson, and H. Wymeersch, "Design and Experimental Validation of a Cooperative Driving System in the Grand Cooperative Driving Challenge," *IEEE Trans. on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 994–1007, sep 2012.

[16] J. Mårtensson, A. Alam, S. Behere, M. A. A. Khan, J. Kjellberg, K.-Y. Liang, H. Pettersson, and D. Sundman, "The Development of a Cooperative Heavy-Duty Vehicle for the GCDC 2011: Team Scoop," *IEEE Trans. on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1033–1049, sep 2012.

[17] M. R. I. Nieuwenhuijze, T. van Keulen, S. Oncu, B. Bonsen, and H. Nijmeijer, "Cooperative Driving With a Heavy-Duty Truck in Mixed Traffic: Experimental Results," *IEEE Trans. on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1026–1032, sep 2012.

[18] L. Guvenc, I. M. C. Uygan, K. Kahraman, R. Karaahmetoglu, I. Altay, M. Senturk, M. T. Emirler, A. E. Hartavi Karci, B. Aksun Guvenc, E. Altug, M. C. Turan, Ö. S. Tas, E. Bozkurt, Ü. Ozguner, K. Redmill, A. Kurt, and B. Efendioglu, "Cooperative Adaptive Cruise Control Implementation of Team Mekar at the Grand Cooperative Driving Challenge," *IEEE Trans. on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1062–1074, sep 2012.

[19] ISO/IEC 7498-1, "Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model," Tech. Rep., 1994.

[20] I. T. Union, "Information technology - ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)," Tech. Rep., 2002.

[21] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[22] S. Vegesna, "Per-Hop Behavior: Congestion Avoidance and Packet Drop Policy," in *IP Quality of Service*, John Kane, Ed. Indianapolis: Cisco Press, 2001, ch. 6, pp. 98–112.

[23] T.-Y. Wu, M. S. Obaidat, and H.-L. Chan, "QualityScan scheme for load balancing efficiency in vehicular ad hoc networks (VANETs)," *Journal of Systems and Software*, vol. 104, pp. 60–68, jun 2015.

[24] M. Joseph, "Finding Response Times in a Real-Time System," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, may 1986.

[25] D. Henriksson, A. Cervin, and K.-E. Årzén, "Truetime: Simulation of control loops under shared computer resources," in *Proceedings of the 15th IFAC world congress*. Elsevier, 2002.

APPENDIX

A. Simulation Model Description

The basic behavior of the simulation model is illustrated in Fig. 11. The scheduler monitors the state of the CPU and decides which thread to run according to the FIFO scheduling algorithm. All the threads in Fig. 11 share the same priority except the message receiving (wireless Rx) thread whose priority is higher. The running thread needs to execute its internal message transmission and informs the CPU about its execution time for the CPU to update its state. If the running thread needs to trigger some other threads, it needs to inform the Internal Events block about when to release these threads. Periodic threads trigger themselves after their execution.

Algorithm 1 uses the execution of the CAM decoding thread as an example to illustrate how threads are executed in this simulation. The Internal Events block manages the release times of all the threads. When the simulation time reaches the time to release a thread, this thread will be pushed into the scheduling queue by the scheduler. The noise thread simulates the interference from the unknown threads implemented by CohdaWireless below the facility layer and all the other unknown threads. The External Events block simulates the message receptions from the wireless networks and the ITS-S host through UDP. Within each simulation step, at most one external event and one internal event can be sent to the scheduler.

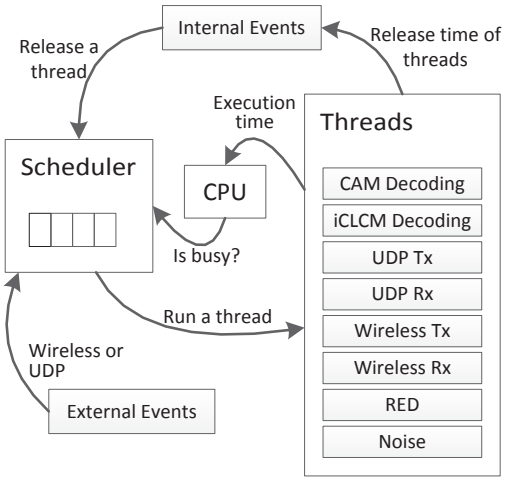

Fig. 11: Basic behavior of the simulation code

The CP delay $D^p(i)$ is measured from the time when msg_i is generated in the External Events block to the time when it is completely sent to ITS-S host. Since $lr_k^{tr} = 0$, message loss ratio $lr_k = dr_k = m_k^{dis}/m_k$ where m_k is the total number of messages sent by vehicle k. Data age $A_k(i)$ is considered for each message received by ITS-S host. Since the transmission delay $D^t = 0$, $A_k(i)$ is measured from the time when msg_i is generated from the External Events block to the time when the next message from vehicle k is completely transmitted. If the simulation time is long enough, the sum of data ages for a particular vehicle k should approach the simulation time.

Algorithm 1: CAM decoding thread

Data: CAM decoding buffer: Buf_{CAM};
Curret time: t_{now}; `/* A global variable, updated in each simulation step */`
Result: Release time of the thread $UdpTx$: R_{udpTx};
Next release time of this thread: R_{dec};
Execution time of the decoding: D^{dec};
begin
The CAM to decode $msg_i = Buf_{CAM}.pop()$;
if $msg_i \neq NULL$ **then**
Generate an execution time for this CAM decoding: D^{dec};
else
$D^{dec} = 0$;
$R_{udpTx} = t_{now} + D^{dec}$;
$R_{dec} = t_{now} + max(T_{dec}, D^{dec})$;

B. Simulation Model Validation

Since the CP delay is difficult to measure. Message loss ratio was used to validate our simulation model against the software implementation. In the experiments, another Cohda MK2 was used to simulate all the communicating vehicles. Within each 40 ms, n pre-encoded CAMs and n iCLCMs were sent from this Cohda MK2 where n is the number of vehicles it simulated. Our ITS-S router received and processed these messages and logged the actual loss ratios for different n. These loss ratios were compared with the values produced by the simulation model. RED was employed on both the implementation and the simulation model. The results of the comparison is illustrated in Fig. 12 with the maximum difference less than 1.3%. Due to the limitation of Cohda MK2, 20 vehicles is the maximum for one Cohda MK2 to simulate. The message decoding time in the simulation has the same distribution of the measured ones.

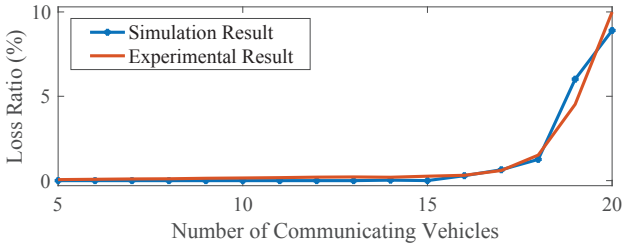

Fig. 12: Comparison between the simulated loss ratio and the tested loss ratio

As analyzed in the paper, the message loss ratio is mainly affected by the timing behavior of the software. Therefore, the result of the comparison implies that this simulation model can reflect the behavior of the final implementation of our ITS-S router.