Received August 2, 2021, accepted August 25, 2021, date of publication August 27, 2021, date of current version September 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3108346

On the Design and Implementation of a Real-Time Testbed for Distributed TDMA-Based MAC Protocols in VANETs

JINGBANG WU, LUYUAN ZHANG, AND YICHEN LIU
School of Computer Science and Engineering, Beijing Technology and Business University, Beijing 100048, China

Corresponding author: Jingbang Wu (wujingbang@btbu.edu.cn)

Supported by the Research Foundation for Youth Scholars of Beijing Technology and Business University (PXM2020_014213_000017) under Grant QNJJ2020-27.

ABSTRACT Rapid development and deployment of vehicular ad-hoc networks (VANETs) require an efficient and scalable media access control (MAC) protocol to support high-priority safety applications. To meet the requirements, researchers have proposed many distributed TDMA-based MAC protocols for VANET in recent years. Despite the superior performance shown in the simulations, few of the proposed protocols have been evaluated in the real-world environment. However, according to the measurement-based works, the extremely unstable wireless communication of VANETs in the real world challenges the performance of the MAC protocols. Hence it may deteriorate if the protocol is implemented in real vehicles. In this paper, we design and implement a real-time testbed for distributed TDMA-based MAC protocols in VANETs. The testbed utilizes the collaboration of hardware (FPGA) and software (Linux) to handle the high-priority and low-priority tasks, and high-accuracy GPS receiver for time synchronization. Results of the measurements show that the transmission delay of the high-priority packets is bounded and the TDMA access is functional.

INDEX TERMS TDMA, MAC, testbed, FPGA, vehicular ad hoc network.

I. INTRODUCTION

Vehicular ad-hoc networks (VANETs) have emerged as a core technology for the intelligent transport system (ITS) to support road safety, traffic management, and on-board infotainment applications, such as Internet access and multimedia streaming. To support the development of ITS, government agencies around the world such as Ministry of Industry and Information Technology of China (MIIT) and the United States Federal Communication Commission (FCC) have allocated a considerable amount of radio spectrum in the 5.8 GHz to 5.9 GHz band for LTE-V2X and Dedicated Short Range Communication (DSRC) to be exclusively used by ITS.

At the MAC layer, DSRC uses IEEE 802.11p [2], a contention based MAC protocol has been developed, which is based on the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) mechanisms. On the other hand, LTE-V2X makes use of Single-Carrier Frequency-Division Multiple Access (SC-FDMA), which divides the channel resources into two-dimensional resource grids (a time dimension and frequency dimension). Two resource allocation modes dedicated to LTE-V2X were introduced: Mode 3 and Mode 4 support direct vehicular communications but differ on how stations' resources are allocated. In Mode 3, vehicles are within the coverage of cellular network, and the channel resources are allocated by the base station. In Mode 4, channel resources are autonomously selected by the vehicles using the Sensing-Based Semi-Persistent Scheduling (SB-SPS) algorithm without the support of the base station.

In VANETs, vehicles need to broadcast the Basic Safety Messages (BSMs) [3] regularly, at least once in every 100 ms, to support safety-related applications such as pre-crash sensing warning, blind curve warning, intersection movement assist at a blind intersection, road works warning and so on [4], [5]. To ensure security, the MAC protocol should provide reliable broadcast services, including predictable bounded delay and high packet delivery rate (PDR). However, on the one hand, IEEE 802.11p does not satisfy the Quality of Service (QoS) requirements of the key applications of VANETs, particularly in heavy traffic conditions, due to the unbounded channel access delay [6]–[8]. In addition, in IEEE 802.11p [9], neither the RTS/CTS mechanism is

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Gao.

used nor the ACK message is sent to confirm the receipt of the message. Therefore, there is a hidden terminal problem and the exponential backoff scheme that cannot be used for broadcasting, and thus the PDR of the broadcasts is significantly affected [8]. On the other hand, LTE-V2X Mode 4 suffers similar problems as the 802.11p due to the poor performance of SB-SPS. Furthermore, the evolutionary standards IEEE 802.11bd and NR-V2X do not resolve the above problems [10].

To satisfy the stringent requirements of the key applications in VANETs, researchers have proposed many distributed Time Division Multiple Access (TDMA)-based MAC protocols for VANETs in recent years [11]–[21]. Although the simulation results of the protocols showed the superior performance, most of the existing research has not been experimentally validated in the real-world environment because of its high cost and the complexity required to deploy and maintain a vehicular testbed. Compared to the relatively ideal communication channel in the simulation environments, the wireless communications between vehicles or infrastructures (V2V and V2I) in the real world are extremely unstable, which impact the performance of the networking protocols [22]–[24]. Therefore, experimental validation in the real-world environment should be an important indicator in the protocol evaluations.

Omar *et.al.* designed and implemented a TDMA prototype system (VeMAC) for the protocol evaluation in the real-world environment on the basis of an ARM development board [25], [26], but they only performed simple functional tests on the prototype system, and there is no detailed performance evaluation. Some researchers have focused on the implementation of the TDMA system on the basis of the Linux operating system and commercial off-the-shelf hardware [27]–[30]. However, those protocols are not designed for the mobility of the vehicles. From the perspective of system implementation, the accuracy of time synchronization, time slot control, and the length of the guard interval between time slots will directly affect the performance of protocol. On the other hand, considering the evaluation of upper-layer protocols, the prototype system needs to provide a full network stack and program compatibility.

Utilizing a pure hardware platform can easily meet the real-time requirements of the system, such as the FPGA-based software-defined radio (SDR) [31] with some open-sourced implementation (e.g. OpenAirInterface [32] and srsRAN [33]), however, providing a full network stack and ensuring program compatibility is a very heavy and daunting task; a pure software (such as Linux) platform does not need to consider the implementation of the protocol stack and the compatibility of the program, but it is difficult to achieve real-time requirements. Therefore, it is an ideal solution to use a combination of software and hardware to ensure both requirements.

In this paper, we design and implement a real-time testbed for the evaluation of the TDMA-based MAC protocols in VANETs. The contributions of the paper and the main features of the testbed are listed as follows:

- We innovatively combine software (Linux) and hardware (FPGA) to design and implement a real-time testbed for the evaluation of the distributed TDMA protocols in VANETs. The testbed is able to provide real-time data delivery and full-stack networking capability.
- We implement the ADHOC MAC protocol [34] and the SATMAC protocol [35] in the FPGA layer, which only handles the critical tasks and passes the non-critical tasks to the Linux kernel through FIFOs.

The remainder of this paper is organized as follows. Section II describes the related work. Section III describes the overview of the testbed. Section IV describes the details of the system architecture and implementation. Section V evaluates the performance of the proposed testbed. Finally, we conclude this paper and discuss future work in Section VI.

II. RELATED WORK

Because IEEE 802.11p and LTE-V2X Mode 4 cannot provide reliable channel access (determined access delay, reliable broadcasting, etc.) [6], a lot of work has proposed non-competitive MAC protocols for VANETs. One of the mainstream solutions is based on the distributed TDMA [36]. The TDMA protocol divides time into frames, and then further into time slots. The vehicle autonomously applies for time slots based on neighbor information, and ensures that there is no time slot reuse within the two-hop neighbor range to prevent hidden terminal problem (time slot collision). On the one hand, there are access collisions and merging collisions in the allocation of time slots, resulting in a decrease in channel access quality [36]: neighbors within two hops occupy the same time slot when applying for a new time slot, which leads to an access collision; the rapid movement of the vehicle leads to a merging collision in the allocated time slot. On the other hand, the time slot allocation strategy greatly affects the channel utilization. Designing time slot control strategies to reduce collisions and improve channel utilization are two key issues in recent years.

A. DISTRIBUTED TDMA-BASED PROTOCOLS

Table 1 lists representative related protocols in recent years. According to the different goals, it can be roughly classified into two categories: improving channel access quality (reduce time slot collision) [11]–[21], and improving channel utilization [37]–[46].

1) IMPROVING CHANNEL ACCESS QUALITY (REDUCE TIME SLOT COLLISION)

In the distributed TDMA-based MAC protocol for VANETs, ADHOC MAC protocol is a pioneering work [34], which requests each node to independently select and apply for a time slot and periodically broadcast the slot occupancy status of its own and the one-hop neighbors through the

TABLE 1. Distributed TDMA protocols for VANETs.

Goals	Method	Protocol name	Date of publish
Improving channel access quality (reduce time slot collision)	Time slot grouping (driving status of vehicle)	VeMAC [11]	2013
		A-VeMAC [12]	2017
		MoMAC [13]	2018
		d-TiMAC [14]	2018
		MAC-AC [15]	2020
	Potential collision prediction	PTMAC [16]	2016
		Efficient TDMA [17]	2019
	Determine the cause of control packet loss	Enhanced D-TDMA [18]	2017
	Game theory	GAH-MAC [19]	2017
	Adaptive beacon frequency	ABC [20]	2021
	Signal strength detection	Lightweight TDMA [21]	2020
Improving channel utilization	Frame length adjustment	A-ADHOC [37]	2011
		ATSA [38]	2014
	Time slot sharing	SS-MAC [39]	2018
		Fine-Grained MAC [40]	2019
		ASTSMAC [41]	2019
	Competitive/non-competitive hybrid access	HER-MAC [42]	2016
		EMMAC [43]	2018
		TCGMAC [44]	2019
		BB-MAC [45]	2019
		SCMAC [46]	2020

frame information (FI) to achieve the non-collision access within two-hops. On the basis of the ADHOC MAC protocol, the protocols represented by [11]–[15] use time slot grouping to reduce the possibility of time slot conflicts: It divides the time slots in a frame into two or more groups, and assigns the time slot groups to vehicles in different driving states (driving directions, etc.). The problem is that its performance largely depends on how to group time slots to adapt to the uneven traffic flow, as well as the problem of vehicle status changes [47].

One idea to solve the problem of time slot grouping is to expand the node's state perception range, and adjust the time slot occupancy before the collision occurs [16], [17]. PTMAC [16] senses the neighbor's motion status and predicts potential collision within the communication range, and informs the conflicting party to adjust the time slot through a control message; Efficient TDMA [17] controls the sensing range by adjusting the node broadcast power to reduce packet collision. In addition, [18]–[21] respectively proposed to determine the cause of control packet loss to improve the accuracy of time slot conflict detection, based on game theory, adaptive beacon sending frequency, Signal strength detection scheme to reduce time slot collisions.

2) IMPROVING BANDWIDTH UTILIZATION

The TDMA protocols for VANETs usually use a semi-permanent time slot allocation strategy, that is, the node will continue to occupy the requested time slot until a collision occurs. For fairness considerations, most protocol default that a node can only apply for one time slot in a frame. Therefore, if the frame length is fixed, the channel utilization will decrease as the node density decreases. There are three main ways to improve channel utilization: dynamically adjust the frame length according to the node density (the number of time slots in a frame) [37], [38], time slot sharing [39]–[41], and hybrid accessing [42]–[46].

Due to the high dynamics of vehicles, the density of vehicles in a local area will frequently change, so the channel utilization rate has a direct relationship with the node density. On the basis of this perspective, some work has designed a adaptive frame length [37], [38]. The basic method is to set the frame length to only double or halve when adjusting, and each node broadcasts its own frame length to unify Frame length within the range of one-hop neighbors. The main problem is that the frame length reduction operation will inevitably cause many nodes to lose the time slots that they have applied for, causing a large number of access conflicts; in addition, the algorithm needs to use control packets to ensure that the frame lengths in the neighbors are consistent, which affects the algorithm, thus affecting the scalability of the protocol. In addition, the protocol represented by [39]–[41] designed a time slot sharing method, which allows security applications with different broadcast periods and meeting certain conditions to share the same time slot in the two-hop set to improve channel utilization.

Another mainstream method is the hybrid the competition and non-competition accessing strategy [42]–[46]. The basic method is to divide the time equally into frames, and then further divide the proportion of fixed or adjustable competition and non-competition parts; nodes can apply for time slots in the non-competition part, and use CSMA to send burst data in the competition part. On the one hand, the existing problem is the adjustment of the proportions of the two parts; on the other hand, because the frame length itself is fixed, the existence of the competing part will cause insufficient time slots in areas with dense nodes, causing security problems. SCMAC [46] solves the above problems by splitting the time slot into contention and non-contention parts, however, due to further restrictions on the length of the time slot, longer data packets cannot be sent.

B. LTE-V2X MODE 4

In order to better support public safety applications, 3GPP released a Device-to-Device (D2D) communication mode in 2015 and defined a new device-to-device link called Sidelink, which allows LTE devices establish direct communication with another device without support of the base station [48]. On the basis of D2D, 3GPP proposed a LTE-V2X LTE Release 14 [49] at the end of 2016 to support V2X communications.

LTE-V2X operates in a time and frequency division manner. The channel resources are divided in time by frames that are 10 *ms* long. Each frame is then divided into ten subframes of duration 1 *ms*. LTE-V2X includes the station mode (Mode 3) and the adhoc mode (Mode 4) [50]. In Mode 4, the allocation of resource blocks does not require the participation of the base station. In order to support that, Release 14 [49] defines an interface named "PC5" on Sidelink to support Sidelink communication. On this basis, [49] proposes a Sensing-Based Semi-Persistent Scheduling (SB-SPS) algorithm. The main feature of the SB-SPS is to access the channel at a fixed interval. The protocol has two parts: channel sensing and resource selection (reselection). When a node enters the network, it first listens to the channel for a period of time (usually 1000 subframes). After that, it collects the available RBs in the time period from T_1 to T_2 after the current time point, according to the received control message and the measured Sidelink signal strength (S-RSSI). Then, the node selects RBs with a fixed transmission interval according to the needs of its own application, and sets a resource usage counter; when the counter drops to 0, the node will decide whether to continue to use the previously occupied RBs with a probability of p, or reselect a new resource.

In recent years, researchers have focused on the evaluation and improvement of the LTE-V2X Mode 4 (abbreviated as Mode 4). Molina-Masegosa *et.al.* evaluated the performance of Mode 4 in city street scenarios [51]. Results show that the hidden terminal problem exists in Mode 4, and the probability of having a packet collision is high. Lopez studied the impact of node density, packet size and the power threshold of the collision detection on the packet reception rate, channel utilization, neighbor perception rate and delay of Mode 4 in OMNeT++ [52]. Results show that the performance advantage of the SB-SPS algorithm in Mode 4 compared with the random occupancy algorithm is not obvious in most cases, and even slightly worse in a few cases. Bazzi *et.al.* analyzed and optimized the channel utilization of Mode 4 when data packets of different lengths exist in the network [53]. Nabil *et.al.* studied the impact of the number of available subchannels, the interval of channel accessing and the reselection probability on the packet delivery rate of Mode 4 [54]. Eckermann *et.al.* proposes an open sourced Mode 4 simulator based on NS-3, and studies the two indicators of the packet reception rate and the packet interval reception [55]. In general, SB-SPS in Mode 4 cannot yet achieve satisfactory performance [55]–[59].

III. OVERVIEW

A. KEY PROBLEMS

The purpose of implementing such a testbed is to evaluate various kinds of distributed TDMA-based protocol for VANETs in real-world environments. Therefore, compatibility of different protocols and the convenience of implementation of different protocols are two key problems which are taken into consideration. As shown in Fig. 2, we first define the core modules which can be adapted to different protocol implementations with minor modifications.

Specifically, the core modules consist of four parts: (1) The protocol state machine. Protocol should maintain a state machine to control each process of the protocol. The state machine is composed of at least three basic states: listening state, waiting state and working state. On top of the basic states, protocol-specific states can be added as required. (2) Status maintainer of the neighbor and time slot. Due to the absence of the base station, each node should maintain the status of its one-hop or two-hop neighbor and time slots in a frame to avoid the hidden terminal problem. (3) The construction and resolving of the control packets. In order to maintain the neighbor status, each node should periodically broadcast part of its neighbor information. Note that, different protocols may have different strategies of broadcasting neighbor information, but the main content is similar. (4) The controller of the channel access and the time synchronization. In the TDMA-based protocols, each node should acquire time slots to access the wireless channel, and the timing of the slot is critical. It depends not only on the accuracy of time synchronization, but also on whether the occupied time slots are free in their two-hop neighbors.

In addition to the core modules, the interaction of the control information between the MAC implementation and the radio is also one of the key problems in the implementation of the testbed prototype, especially the handling of device interrupt request (IRQ).

Due to the stringent time synchronization and the real-time processing requirement of the protocol, we implement the

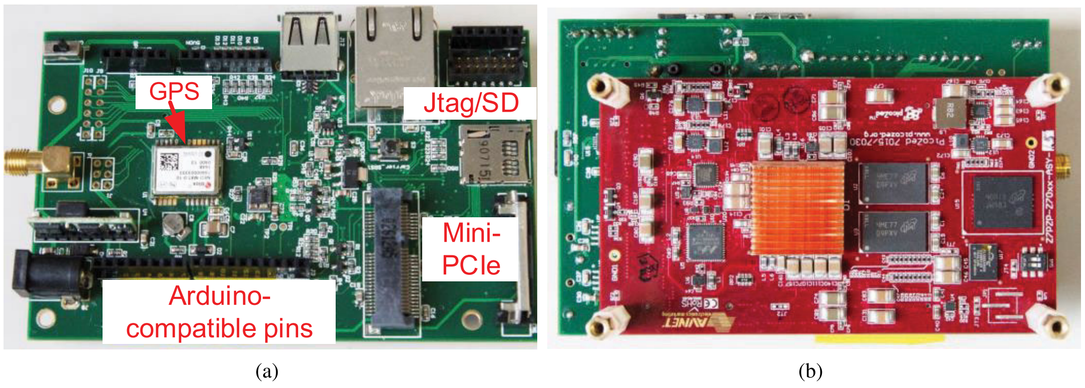


FIGURE 1. **The development board of the testbed prototype. (a) front view, (b) back view.**

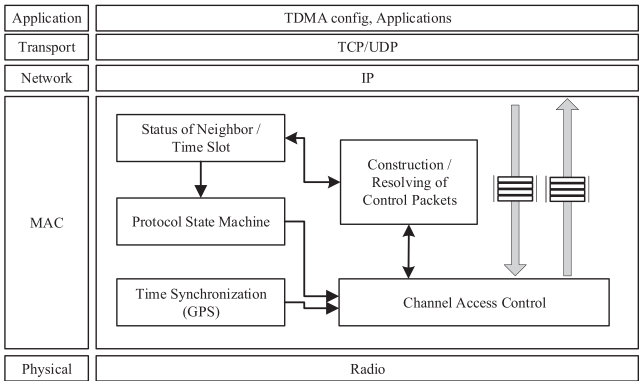

Application
TDMA config, Applications
Transport
TCP/UDP
Network
IP
MAC
Status of Neighbor / Time Slot
Construction / Resolving of Control Packets
Protocol State Machine
Time Synchronization (GPS)
Channel Access Control
Physical
Radio

FIGURE 2. **Core modules of the proposed system.**

TDMA-based protocol (lower MAC layer) in the FPGA, and establish interfaces between the Linux OS and the protocol implementation.

B. IMPLEMENTED PROTOCOLS

The proposed testbed implements two protocols: ADHOC MAC [34] and SATMAC [35].

1) THE ADHOC MAC PROTOCOL

The ADHOC MAC protocol proposed by Borgonovo *et al.* is a distributed TDMA-based MAC protocol for VANETs. In the protocol, each node maintains a synchronized time, and splits the time into frames consisting of a constant number of fixed duration time slots, e.g. each frame has 100 time slot, and the length of each time slot is 1 *ms*. Each node should acquire one slot per frame exclusively in their two-hop neighbor set (THS) to prevent hidden terminal. Each node must broadcast the Frame Information (FI) in their time slot to report the status of all the slots in the previous frame, and then each node is able to know the slot occupancy information within the THS and acquire a free slot. In the FI, there are N vectors specifying the status of the N time slots. The slot status can be either "BUSY by vehicle i" or "FREE": if vehicle i has broadcasted a packet in a slot, the status of this slot is "BUSY by vehicle i"; otherwise, it is "FREE". Based on the FI contained in the packet received in each slot of a frame, each vehicle marks each slot as "RESERVED", *i.e.*, the slot is acquired by a vehicle, or "AVAILABLE", *i.e.*, the slot is not used by any vehicle.

Two types of packet collisions can occur on a time slot [60]: access collision and merging collision. An access collision occurs when two or more nodes within the same THS attempt to acquire the same time slot, and the probability of access collisions is directly related to the node density and the frame length. A merging collision occurs when two or more nodes using the same time slot become members of the same THS due to node mobility.

Node x is assumed to have been powered on and needs to acquire a time slot, the protocol is initialed by "LISTEN" state. It listens to FI from the neighbors and updates the information of each time slot when a new FI is received. After the period of default frame length, node x can determine the status of each slot. Then, it enters "WAIT" state, randomly selects a free slot, and sends an FI in the slot reporting that it has occupied the slot. In the next frame, only when the node does not receive any conflicting FI (no collision happens), can it successfully occupy the time slot, and enters "WORK" state.

Currently, the basic strategy of most of the state-of-the-art distributed TDMA-based MAC protocols for VANETs is similar to the ADHOC MAC protocol [11]–[21]. Therefore, The implementation of the ADHOC MAC can be used as a basic set on which other protocols can be further implemented.

2) THE SATMAC PROTOCOL

On the basis of the implementation of the ADHOC MAC protocol, we further implement the SATMAC protocol [35] in the testbed prototype. Compared with the former, there are mainly the following three differences in SATMAC:

- A more complex and better performance strategy for the maintenance of the time slot state. The FI packet carries more information than that of the ADHOC MAC protocol, and correspondingly, there are more processing procedures for resolving an FI packet.
- SATMAC proposed a potential packet collision detection scheme based on the accurate time slot usage of

the two-hop neighbors and the rough information of the three-hop neighbors. A time slot adjustment approach is designed to not only avoid the packet collision but also conduct the frame length adjustment if needed. After a node successfully acquires a time slot, the node can relocate it in the following frames when needed. Two situations may trigger such time slot adjustments. First, once a node detects a potential collision in its time slot, it relocates it to another time slot to prevent the packet collision. Second, once a node finds that the node density in its three-hop neighbor set falls below a threshold, it relocates its time slot to the first half of the frame to prepare for the reduction of the frame length.

- SATMAC proposed an adaptive frame length method to fit various node densities in VANETs. Each node can adjust its frame length only based on the locally maintained information, and the frame length among neighbors may not be consistent.In general, SATMAC doubles the frame length when the local node density is high and halves it when the density is low. As such, the adaptive frame length can reduce the probability of slot collisions as well as prevent the starving problem when the node density is high. In addition, it can enhance the channel utilization when the node density is low.

IV. SYSTEM DESIGN AND IMPLEMENTATION

A. HARDWARE

The proposed testbed is built up based on the Xilinx Zynq-7015 SoC, which integrates a dual-core Central Processing Unit (CPU) and an FPGA, and an NEO-M8T timing GPS module from u-blox. The hardware connections are shown in Fig. 4. The CPU and the FPGA are connected through an Advanced eXtensible Interface (AXI), and the configuration space of both the FPGA subsystem and the NIC are mapped in the memory. We mount two radios in the prototype: Sub-1GHz (433 MHz) radio and 5.9 GHz Radio. The 5.9 GHz band is widely used in the ITS field around the world. The testbed uses the Atheros AR9462 Network Interface Card (NIC) for the 5.9 GHz band, and we have also tested the Atheros AR9382. The Sub-1GHz radio is a cheap and mature solution, which has been used in ITS applications in Japan [61]. Compared with the 802.11p channel, the Sub-1GHz channel has better diffraction capability and broader transmission range, but with lower transmission rate [24].

Fig. 1 shows the front and back view of the development board of the testbed prototype, which consists of core board and the IO board. The core board is the Picozed development board produced by AVNET, including the Zynq-7015 FPGA core, 1 GB memory and 4 GB flash memory. Based on Picozed's IO expansion requirements, we designed IO boards, mainly including GPS module, Mini-PCIe slot (connected to 5.9GHz network card), Arduino expansion IO (connected to Sub-1GHz radio), RJ-45, USB-2.0, UART port, and other interfaces.

TABLE 2. Testbed components and parameters.

CPU+FPGA	Xilinx Zynq-7015 SoC Dual-core CPU @ 667 MHz
DRAM	1 GB
Radio	Atheros AR9462 @ Channel 178, 10 MHz bandwidth
Transmission Rate	12 Mbps
Communication Range	5 m
GPS Receiver	u-blox NEO-M8T
Operation System	Standard Linux (Kernel version 4.4)
Slot Duration	1 ms
UGV Speed	0.8 m/s
Battery	12 V, 5600 mAh
Other Communication Modules	Zigbee and Bluetooth

In addition to the internal structure and hardware of the prototype system, a key issue in the experiment is the control of the trajectory of the nodes. In related works, a common practice is to use multiple cars as network node. As shown in Fig. 3, in the propose testbed, we use the Unmanned Ground Vehicle (UGV) to carry each testbed prototype. Each UGV is solely controlled by an open source autopilot software: Ardupilot [62]. The Ardupilot can control the UGV to follow the user-defined trajectories at a constant speed on the basis of the GPS signal and the digital compass. Furthermore, we limit the transmitting power of each testbed prototype to control the communication range of each node to fit the mobility of the UGV.

Table 2 shows the components and parameters of the testbed. The Zigbee and Bluetooth are the management interfaces, through which we can send commands to all nodes simultaneously and easily access the system console of each node. The 5.9GHz radio runs in Outside the Context of a BSS mode (OCB) on channel 178 (5.885 GHz to 5.895 GHz band), and its data transmission rate is fixed at 12 Mbps[1]; the data rate of the 433 MHz radio is set to 100 Kbps. We have built six copies of the prototype, and each is mounted on a UGV as shown in Fig. 3d.

B. ARCHITECTURE

From the perspective of the network layer, the overall architecture of the proposed system is shown in Fig. 5. Our approach follows the TCP/IP model and the layer structure in the Linux kernel. As described in Section IV-A, we installed two radios in the system. The two radios can be used separately in the upper layer as different interfaces.

On the basis of the original network layer implementation in the Linux kernel, we insert an FPGA subsystem between the MAC layer and the physical layer. The ath9k is an open sourced driver for the Atheros NIC, responsible for the data interaction between OS and NIC. Because the Linux system is not an OS that pursues real-time performance, the maximum response delay of the interaction is unbounded (mainly the IRQ responding delay). Therefore, in order to provide the

[1]The 12 Mbps is one of the mandatory supported bit rates by the IEEE 802.11p.

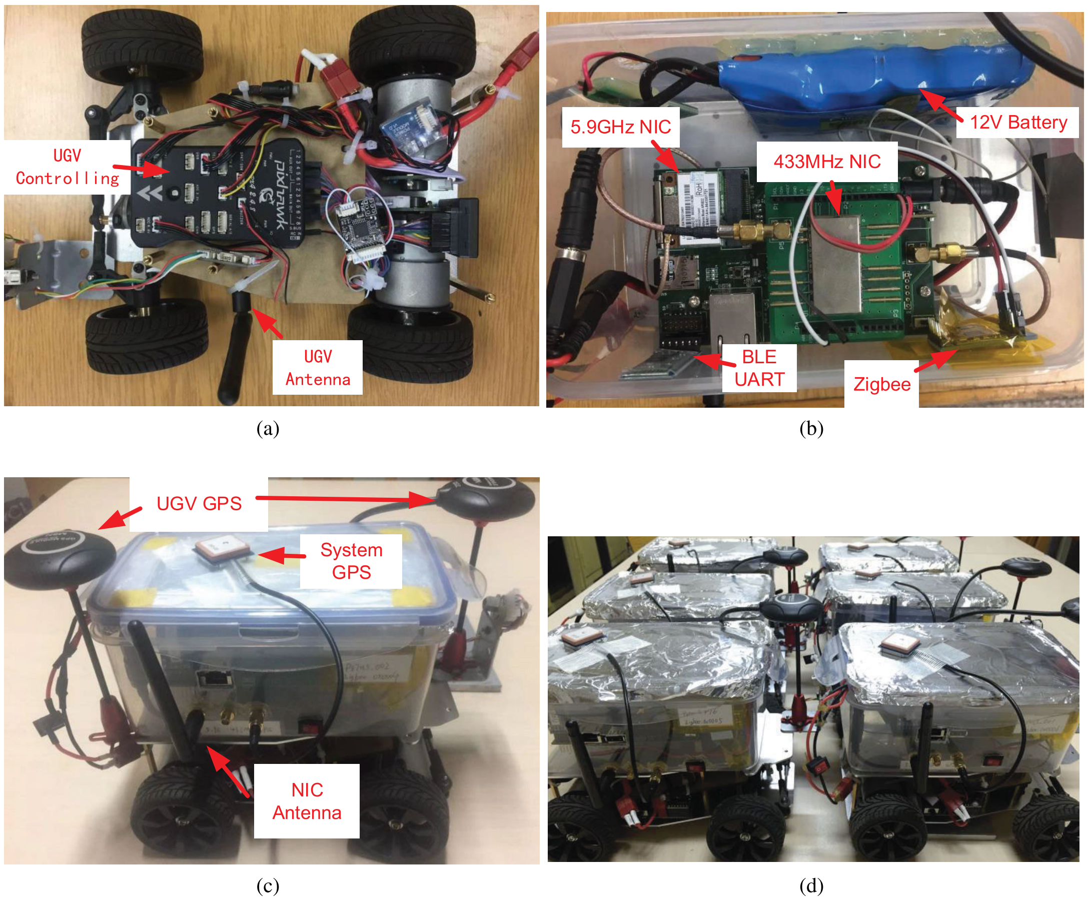


FIGURE 3. (a) Top view, (b) Inside view, (c) Side view, (d) The prototypes mounted on UGVs.

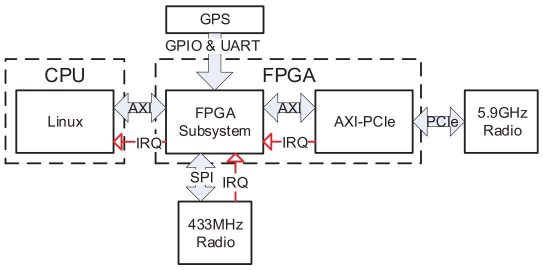


FIGURE 4. Hardware connections.

real-time data delivery for high-priority packets, the FPGA subsystem handles the critical tasks, including the transmitting (Tx) and receiving (Rx) of the high-priority packets. The non-critical tasks will be passed to the Linux OS.

The FPGA subsystem contains the timer module for time synchronization, the Hierarchical Processing Module (HPM) for handling the interactions between Linux OS and NIC, the core modules for the controlling of the time slots and the configuration module. A revised ADHOC MAC protocol and an SATMAC protocol are implemented in the FPGA, including the protocol state machine and the transmission/process of the FI packet.

The PCIe memory-mapped interface maps the NIC's configuration and control registers to system memory, and the packets are accessed on the basis of the Tx/Rx descriptors through the Enhanced Direct Memory Access (EDMA). When a packet has to be sent, the MAC layer creates a fixed-length Tx descriptor that includes the memory address of the packet payload, and sends the descriptor's address to NIC, where the packet can be loaded and sent according to the address. An empty Rx descriptor from the Rx descriptor FIFO will be loaded with the relevant status and the address of the payload when a packet is received or a receive error occurs in NIC, and an IRQ will be created and transmitted to the MAC layer to alert the task. The MAC layer will query NIC for the IRQ cause and handle the associated task when an IRQ is received.

The synchronized time is divided into frames, each of which contains an adjustable number of time slots with a fixed duration. Fig. 6 shows the packets transmitted in each occupied slot: the FI for the TDMA access control; the SIFS is the Short Inter-Frame Space and the DIFS is the DCF Inter-frame Space.

C. TIMER

A key factor in the TDMA system is the time synchronization, which greatly impacts the performance of the protocol. On the basis of the two configurable time pulses of the GPS timing module (u-blox NEO-M8T [63]), denoted by *Tp1* and *Tp2*,

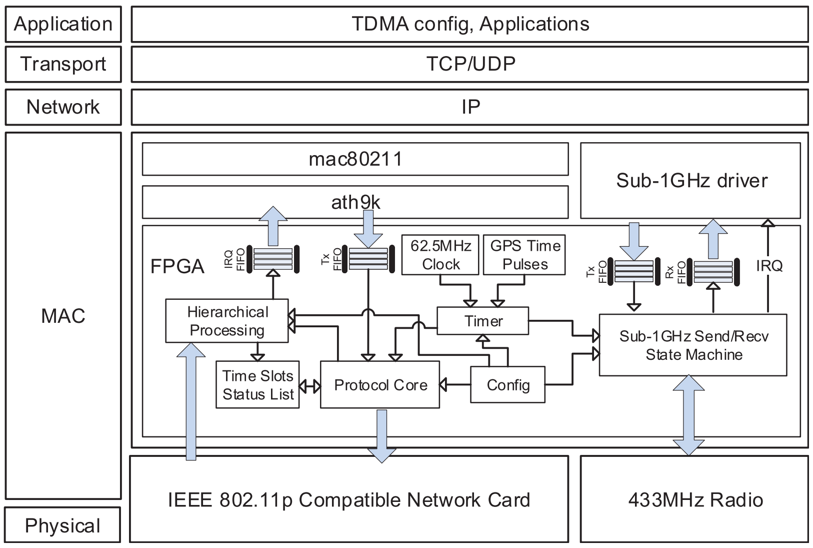


FIGURE 5. System architecture of the proposed system.

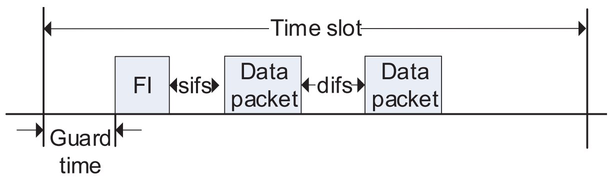


FIGURE 6. Packets transmitted in a time slot.

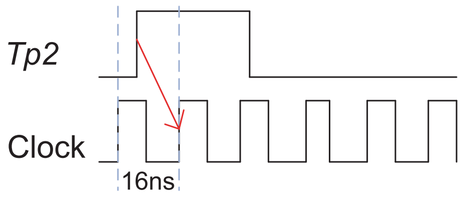


FIGURE 7. Align the time pulse to the clock.

we implement a timer module in the FPGA subsystem. The time pulses *Tp1* is strictly aligned to the Coordinated Universal Time (UTC), and a rising edge represents the start of each second. The period of *Tp2* is set to be equal to the duration of each time slot. In other words, a rising edge of *Tp2* represents the start of each time slot. Time is synchronized and partitioned into frames consisting of i fixed duration time slots, $i \in \{4, 8, 16, 32, \ldots\}$. Each GPS second contains an integer number of frames, and each rising edge of 1PPS marks the start of a new frame. In this case, each GPS second should contain w time slots, $w \in \{4, 8, 16, 32, \ldots\}$. In our work, $w = 1024$ and the duration of a time slot is approximately 976.5 *us* (including the guard time).

In the FPGA subsystem, the system clock is a 62.5 MHz oscillator, where the clock cycle is 16 *ns*. The system aligns the time pluses from GPS to the system clock as shown in Fig. 7, and thus the accuracy of the time synchronization is one system clock cycle (16 *ns*) plus the time pulse accuracy of GPS which is 20 *ns* under clear sky [63]. Such accuracy can meet the requirement of microsecond-level time slot. Note that when the GPS signal is lost [64], e.g., when a node enters tunnels, the local oscillator can maintain sufficient accuracy in a short period for the time synchronization (500 *ns* [63]). If 1PPS is lost for a long duration, a backup time synchronization scheme, e.g., [65] can be used until the GPS signal is recovered. Such backup scheme is not implemented in the prototype.

The frame length and the duration of each time slot are configurable in the testbed prototype, by configuring the frequency of *Tp2* as required. The configuration should meet the alignment of the frame and *Tp1*, e.g., if the frequency of *Tp2* is 1024 HZ (a second contains 1024 slots) and the frame length is 128 slots, then a GPS second contains exactly eight frames.

D. THE HIERARCHICAL PROCESSING MODULE

Fig. 8 shows the structure of the hierarchical processing module. The module is responsible for handling the interaction between the network card and Linux OS, including IRQ response and processing, and the sending of upper-layer data packets. The module intercepts and processes the critical IRQs initiated by the network card, such as the reception of a high-priority packet, and passes the non-critical IRQs to Linux OS for processing.

The critical IRQs include system failure event and the Tx/Rx of the FIs and BSMs. The non-critical IRQs include the Tx/Rx of the application packets and other controlling events such as Rx error. The hierarchical processing module separates different IRQs into three categories: TxOK, RxOK and network card controls. The TxOK indicates that a packet is successfully sent to the air, and the IRQ should be issued by the network card immediately after a Tx process is done. If the TxOK is not received within a certain period, say 1 *ms*, then the watch dog module will reset the whole system.

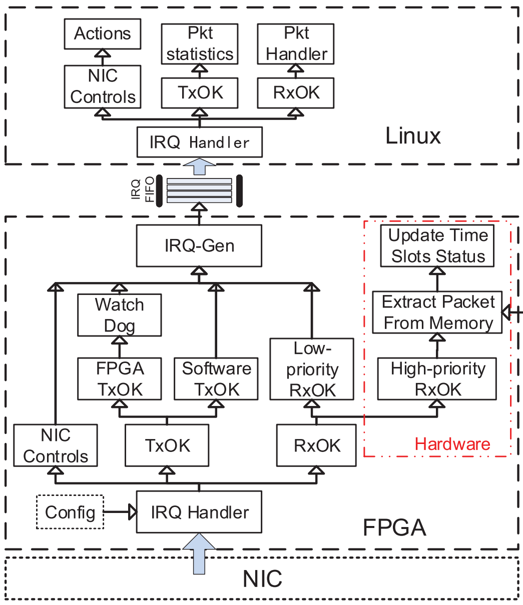


FIGURE 8. **The hierarchical processing.**

The network card has ten Tx queues for transmitting different priority packets, and two Rx queues for receiving different priority packets: low priority Rx queue, denoted by *LP*, and high priority Rx queue, denoted by *HP*. The hierarchical processing module separates the high-priority packets (FI and BSM) and the low-priority packets by utilizing those queues. When a packet is received in the network card, the network card reads the MAC header and parses the priority flag. Then it pushes the packet into *LP* or *HP* accordingly. Then, an IRQ will be issued indicating the Rx event (RxOK). By handling the Rx event in the hierarchical processing module, it can receive the FI packets from the *HP*, and resolve the slot status from the FI packet. Then, it merges the status information into the locally maintained container according to the protocol descriptions in [34] and [35].

Besides the Rx, the hierarchical processing module also handles the Tx packets from the upper layer. When the Tx enable signal is activated, the hierarchical processing module calculates the transmission time of the first packet in the Tx FIFO, and then send the packets one by one to the network card iff the transmission time is less than the remaining time of the slot, which is calculated as follows:

$$T_{Remain} = T_{Slot} - T_{Guard} - T_{FI} - T_{BSM} - T_{Used} \qquad (1)$$

where T_{Slot} is the duration of the time slot, T_{Guard} is the guard time, T_{FI} and T_{BSM} are the transmission time of the FI and BSM (SIFS is included), respectively, and T_{Used} is the time spent on the transmitted upper layer packets in the current slot (DIFS is included).

The processing delay and the memory-access delay make up the hierarchical processing module's delay. The memory-access delay is divided into two parts: the first is the access delay of the NIC registers via PCIe, which is constant (about 100 clock cycles, which take 1.6 μs) because communications between the network card and the FPGA/CPU consume a small portion of the PCIe bus throughput; the second is the access delay of the main memory (DRAM) via AXI, which is about 50 clock cycles ($0.8mus$). Furthermore, the workloads on the DRAM may impact the access latency. Heavy loads on the CPU and DRAM have little effect on the performance of the FPGA subsystem in our testbed, as shown in Section V. The system must get the IRQ code for each IRQ by reading the IRQ register in the network card, which takes around 100 clock cycles, and then process the IRQ by clearing the IRQ register in the NIC and the AXI-PCIe soft core, which takes approximately 200 clock cycles. According to the preceding specifications, processing an IRQ from NIC takes at least 300 clock cycles, that is, $300*16ns = 4800ns = 4.8mus$.

In the hierarchical processing module and the core modules, the processing delay is the time necessary for each processing flow. The process of the FI, which is related to the frame length, is the most time-consuming path. For example, when the frame length is 100, the module needs about 400 clock cycles to traverse all the slot status contained in the FI and update the locally maintained status, resulting in a delay of $400*16ns = 6400ns = 6.4mus$. The processing delay of the FI may be ignored because it is always received at the start of a slot and the $6.4mus$ delay is considerably smaller than the SIFS. Aside from that, all of the other process routes have nanosecond-level delays, which may be ignored as well.

E. IMPLEMENTATION OF THE CORE MODULES

As described in Section IV-B, The core modules control the TDMA access and the transmission of the FI, including the. a new FI will be created based on the status list every frame and sent to the network card when the node enters its own time slot. After that, the Tx enable signal will be activated for the rest of the slot. The Tx descriptor created in the ath9k driver will be buffered in the Tx FIFO of the FPGA subsystem, which will wait for the Tx enable signal to send the descriptors to the NIC. More details will be described in the following subsections.

As shown in Fig. 9, the implementation of the protocol core modules includes six parts: the Protocol State Machine (PSM), the Candidate Slot Finder (CSF), the Tx/Rx Descriptor Manager (TDM/RDM), the FI Constructor, the transmitter and the test/BSM module. In general, we implement the state machine of a revised ADHOC MAC protocol and the SATMAC protocol in the PSM, which controls the transmitting of the FIs, the time slot acquisition and the process of the packet collision (merging collision and access collision [60]). The FI constructor controls the construction of the FI according to the packet format defined in [34] and [35]. Except for the protocol state machine and the data structure of the FI packet, the rest parts are identical in the implementation of the two protocols.

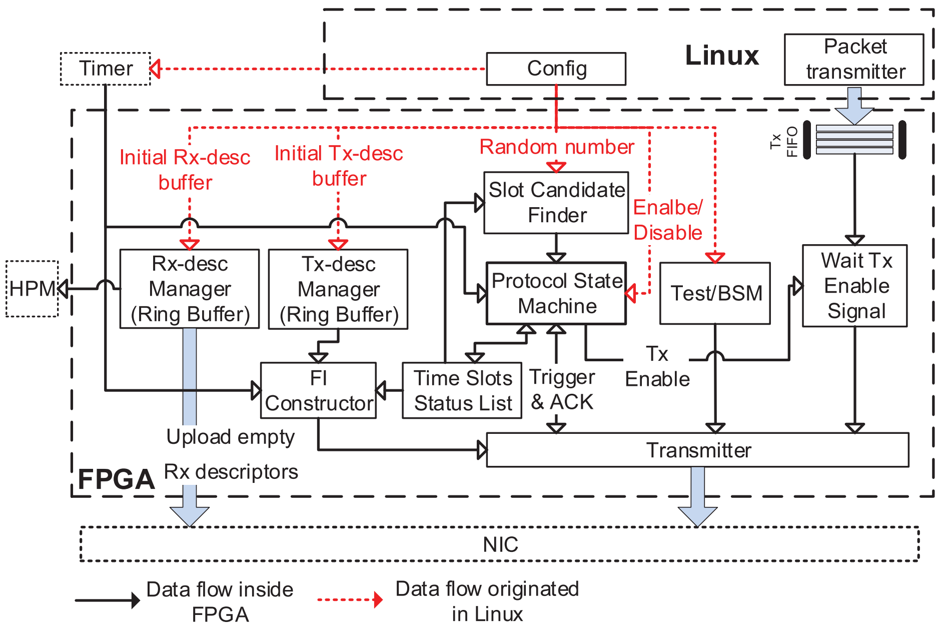


FIGURE 9. Implementation of the core modules.

1) DATA FLOW

When a node needs to acquire a time slot (collisions happen in the occupied time slot, the node is just powered on), The CSF module provides the candidate slots to the PSM module, which should then update the time slot status list and randomly occupy a time slot from the candidates. Furthermore, in each frame, the CSF module continuously updates the list of candidate slots.

In each frame, the FI constructor constructs the FI packet according to the protocol specification in [34], [35] and the time slots status list. When the occupied time slot starts, the PSM module generates signal to trigger the transmission of the FI packet. Then, the transmitter sends the test packet or the BSM packet to the network card after the FI is sent and generates an acknowledge signal to the PSM module to notify the activation of the Tx enable signal for the rest of the slot.

As we described in Section IV-B, the packet sending and receiving interface between the network card and the system is based on EDMA and the Tx/Rx descriptors. The descriptors are initialed in the Linux OS and maintained in the ring buffers of the TDM and RDM. Once initialed, the TDM and RDM will circularly reuse the descriptors until the subsystem is reset. When a packet is received in the network card, it loads the received packet into the empty Rx descriptor, which is uploaded by the RDM module, and the same descriptor will be pushed to the hierarchical processing module for the extracting of the received packet.

2) PROTOCOL STATE MACHINE

Fig. 10 shows the protocol state machine for the ADHOC MAC protocol and the SATMAC protocol. The ADHOC MAC protocol has four states: “Listen”, “Wait_Candidate”,

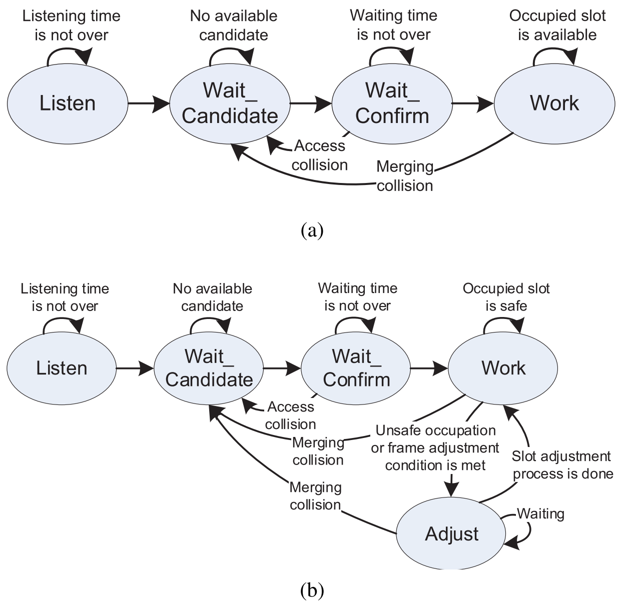


FIGURE 10. The protocol state machines, (a) ADHOC MAC, (b) SATMAC.

“Wait_Confirm”, and “Work”. When a node has just powered on (or reset) and needs to acquire a time slot, it listens for one frame to collect the slot occupancy in the two-hop neighbor set. After that, it will choose a candidate slot and mark it as occupied by the node. If no slots are available, the present state will be maintained; otherwise, the FI packet will be sent at the beginning of the candidate slot. The node will then wait one frame to see if the slot has been successfully occupied without causing an access collision. If true, the node will keep accessing the same slot until a merging collision is detected in subsequent frames. When a collision happens, the protocol will retreat to the ‘‘Wait_Candidate’’ state and reacquires a new time slot.

On the basis of the ADHOC MAC, SATMAC adds an ‘‘Adjust’’ state. When a node is in the ‘‘Work’’ state, it periodically checks whether its time slot is safely occupied [35] or the frame adjustment condition is met. If not, the node will launch a time slot adjustment process to avoid the potential packet collision or adjust the frame length. When the time slot is successfully adjusted to a new position, the node re-enters the ‘‘Work’’ state, otherwise, it enters the ‘‘Wait_Candidate’’ state to reselect a new time slot due to the merging collision.

3) FLEXIBLE CONFIGURATION

In order to evaluate or debug the implemented protocol more flexibly, we first implement the TDMA function switch: the TDMA function can be turned on or off by the upper layer, when off, the core module does not attempt to acquire a time slot and always activates the Tx enable signal (packets from upper layer will be directly pushed to the network card without processing). Second, we can manipulate the time slot acquisition behavior by providing the module with a specified random number, for example, specify the core module to obtain the first available time slot. In addition, we can also monitor the protocol status through the interface, such as Tx/Rx count, packet collision count.

V. EVALUATION

In this section, we evaluate the basic performance of the proposed testbed by conducting two sets of measurements (parameters of the testbed are shown in Table 2):

1) Time slot acquisition: We validate the basic functionality of the TDMA access by investigating how fast the contending nodes can acquire their slot.
2) Round-trip delay (RTD): We measure and compare the RTD of the IEEE 802.11p packets; and the low-priority packets and the high-priority packets. Moreover, In order to investigate the impact of the workloads on the testbed prototype, we add a heavy load on the CPU and DRAM and measure the RTD of the high-priority packet.
3) Channel utilization and multi-hop performance.

A. TIME SLOT ACQUISITION

In this experiment, six nodes are fixedly placed together and each node is within in the communication range of the others. All the data are the average of ten repeated measurements. At the beginning of the experiment, each node activates the TDMA function simultaneously (the activation command is sent through the zigbee interface in the testbed prototype). When activated, each node starts to acquire a time slot. When collision happens (more than one node have acquire the same slot), the nodes should detect the collision and reacquire a slot. As shown in Fig. 12, the time slot acquisition in the testbed prototype is functional, and all the nodes can acquire the collision-free slots within five frames. Moreover, less contending nodes require fewer frames to achieve the goal, due to the smaller probability of access collisions.

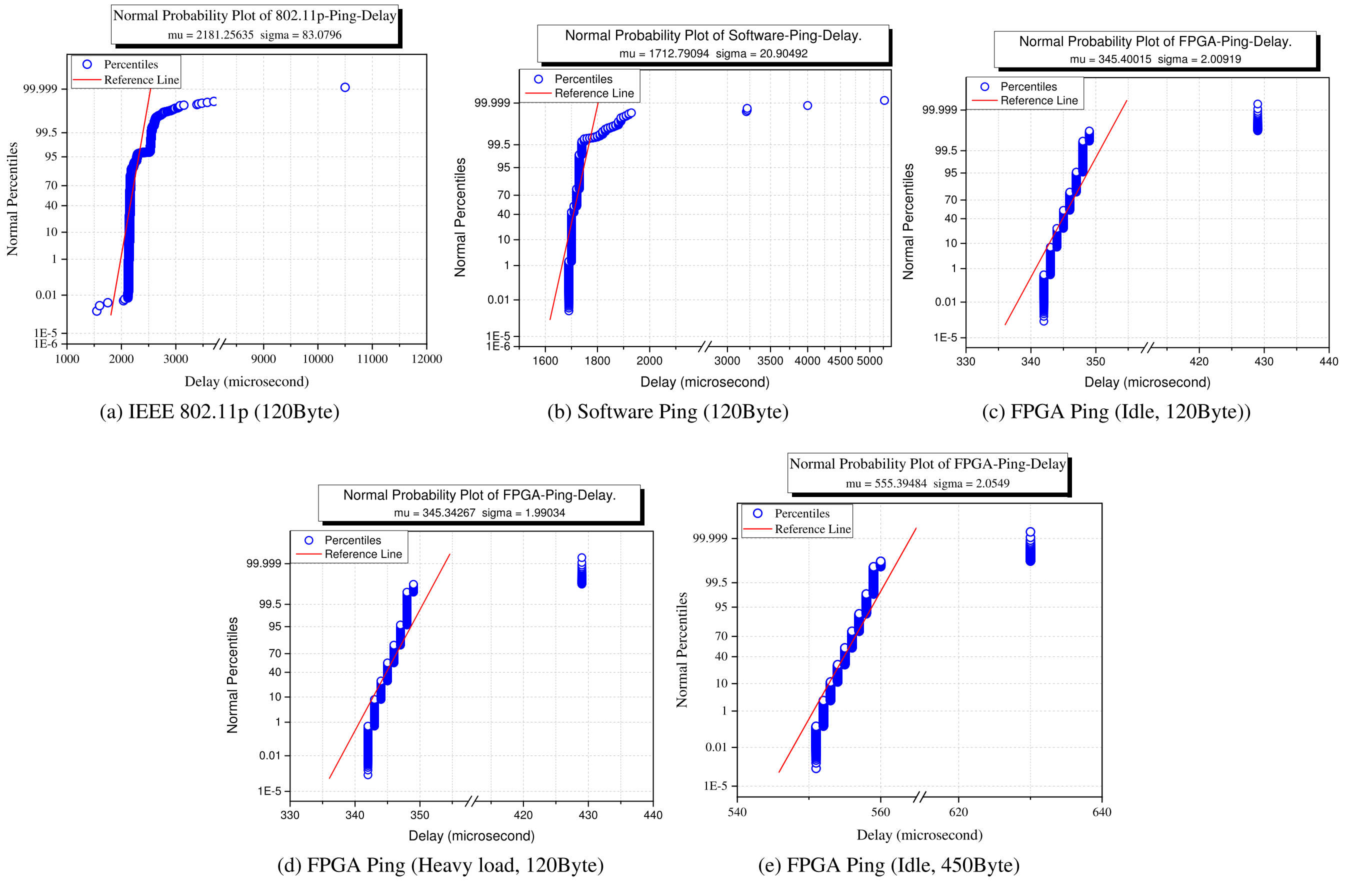
B. ROUND-TRIP DELAY COMPARISON

In this experiment, two nodes are placed closely side by side. The network card is configured in the adhoc mode and the TDMA function is disabled to evaluate the minimum round-trip delay of one hop (when the test packet is received, the node will immediately send the same packet to the source node). Five sets of experiments are conducted using different configurations: (a) IEEE 802.11p, and (b) Software Ping: $100,000$ round trips are measured in the Linux with the ping interval of 10 *ms*. The different between (a) and (b) is that the carrier sense and the back-off functions are disabled in (b). (c) The FPGA Ping (Idle): we use the Test/BSM module in the FPGA subsystem to evaluate the transmission of the high-priority packets. $200,000$ packet round trips are measured with the ping interval of 1 *ms*. (d) The FPGA Ping (Heavy load): the only difference between this set and (c) is that we add a heavy load on the Linux OS: a user-space test program starts 100 threads, and each thread requests three 4K-byte memory blocks, and cyclically copies the value of one block to another block. The requested memory blocks are marked with the ‘‘volatile’’ flag to make sure no cache is involved. (e) The FPGA Ping (Idle, 450 bytes): The size of the test packets are 120 bytes in (a), (b), (c) and (d), and 450 bytes in (e).

The results are shown in Fig. 11. It is clear that the performance of the FPGA ping (c) is significantly better than the software ping (b), and the average value and standard deviation of RTD are about 5 times and 10 times smaller than that of software ping, respectively. Because the Linux OS cannot handle IRQs (the main component of the packet processing) in real time as we described in Section I, and thus the round time delay is unbounded (the maximum is 5270 μs). Compared to that, because the packet process time in the FPGA subsystem can be ignored, and thus the round time delay of the FPGA ping set is significantly less than that of the software ping set. More importantly, the result of the FPGA ping set has an upper bound of 429 μs, which is only 84 μs higher than the mean value. Furthermore, the difference between Fig. 11c and Fig. 11d is unnoticeable, suggesting that the heavy load on the CPU does not impact the performance of the testbed prototype. In the FPGA ping set, a strange phenomenon is that the RTDs of a small part of the round-trips (68 among $200,000$) are equated to 429 μs and the rest are between 342 μs and 349 μs. Moreover, similar gaps are found in the measurements of the packet having different sizes. We believe this phenomenon is caused by the process within the network card. Nevertheless, the gap does not significantly affect the performance of the system.

C. CHANNEL UTILIZATION AND MULTI-HOP PERFORMANCE COMPARISON

A data rate of 12 Mbps is used in the prototype system, and each time slot is 1 *ms*; therefore, in theory,



(a) IEEE 802.11p (120Byte) (b) Software Ping (120Byte) (c) FPGA Ping (Idle, 120Byte))

(d) FPGA Ping (Heavy load, 120Byte) (e) FPGA Ping (Idle, 450Byte)

FIGURE 11. Round-trip delays.

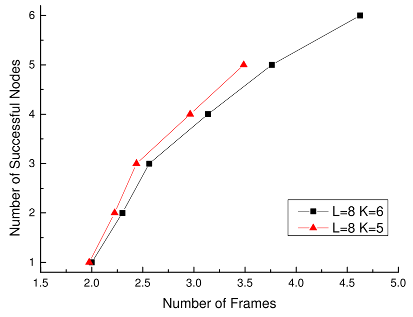


FIGURE 12. Time slot acquisition, *K* denotes the number of the contending nodes.

up to $12 * 10^6 * 0.001/8 = 1500$ bytes of data can be sent in a time slot. The overhead in a time slot includes the transmission of FI messages, the message transmission frame interval (SIFS) and the guard time (the guard time between time slots is set to 100 us, accounting for 10 % of a time slot): According to the analysis in [35], when the frame length is 128, the FI packet length is 194 bytes, which requires at least about 140 us; plus the SIFS of 32 us and the guard interval of 100 us, a total of 272 us. Taking into account the packet processing delay inside the network card, the maximum user data that can be sent in a time slot in the prototype system is about 1000 bytes (including the header).

In this experiment, we use four prototypes to construct a full-connected topology (each node is within the one-hop communication range of other nodes) to test the channel utilization of TDMA and IEEE 802.11p by UDP data stream using the Iperf tool: each node simultaneously sends to any other node a constant UDP data stream. In the TDMA experiment, the frame length is fixed at four, and the packet payload length is limited to 950 bytes. As shown in Fig. 13, it can be clearly seen from the figure that there is almost no jitter in the point-to-point throughput of the TDMA experiment. In contrast, the 802.11p experiment has significantly more jitters. In addition, when the packet length is 950 bytes, the total throughput of the TDMA experiment is about 10 % higher than that of the 802.11p experiment. Moreover, a larger packet length usually brings a higher transmission efficiency, when the packet length is 1470 bytes, the throughput of the TDMA experiment is about 5 % lower than that of the 802.11p experiment.

Next, we use four prototypes to establish a straight-line topology to test the multi-hop delay. It takes three hops to send data from the first node to the last node. In the experiment, we use the ICMP Ping command to compare

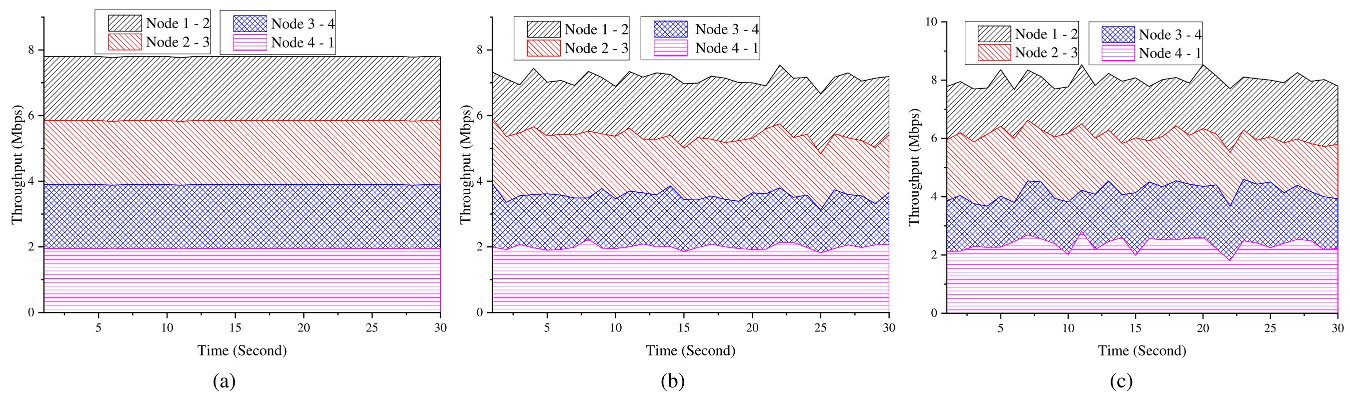


FIGURE 13. Channel utilization using the UDP data stream. (a) TDMA (950 bytes), (b) IEEE 802.11p (950 bytes), (c) IEEE 802.11p (1470 bytes).

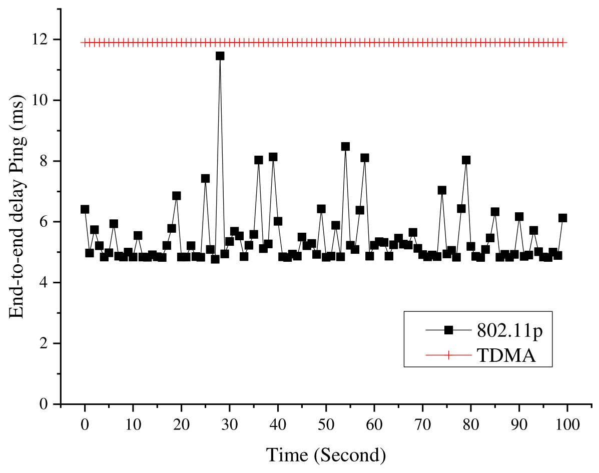


FIGURE 14. End-to-end delay (Ping).

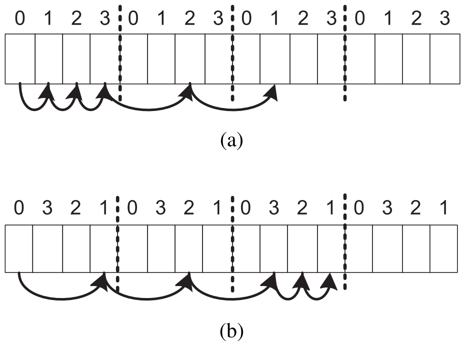


FIGURE 15. Data packet forwarding in the multi-hop scenario.

the multi-hop delay of TDMA and IEEE 802.11p. As shown in Fig. 14, it is clear that although the overall delay of TDMA is higher than that of 802.11p, the stability of TDMA is much better than that of the 802.11p. Moreover, according to the round-trip delay experiment of software Ping in the Section V-B, the round-trip delay of a small number of packets greatly exceeds the average (less than 0.5 %), these outliers are not reflected in the results of this experiment.

In this experiment, the multi-hop Ping delay of TDMA is slightly lower than 12 *ms* and 117 % higher than that of 802.11p, due to the characteristics of the time slots. Fig. 15 shows the multi-hop data forwardings in two different time slot allocation states in TDMA set. In Fig. 15a, time slots 0/1/2/3 are occupied by nodes 0/1/2/3, respectively, node 0 is the data source and node 3 is the data sink, and the theoretical round-trip delay is 9 *ms* (time slot duration is 1 *ms*). In the figure 15b, time slots 0/1/2/3 are occupied by nodes 0/3/2/1, respectively, and the theoretical round-trip delay is 11 *ms*. It can be seen from this example that the time slot occupancy of nodes in a multi-hop link will significantly affect the end-to-end delay.

In summary, the transmission delay of the high-priority packet is bounded, and the time slot acquisition in the testbed is functional. Furthermore, we compare and verify the channel utilization and multi-hop performance of IEEE 802.11p and TDMA implementation. The results show that compared to 802.11p, the TDMA implementation has significantly strong stability.

VI. CONCLUSION AND FUTURE WORK

In this paper, we designed and implemented a real-time testbed for distributed TDMA-based MAC protocols in VANETs. In the proposed system, an FPGA subsystem is inserted between the Linux OS and the network card, to handle the high-priority tasks while passing the low-priority tasks to the Linux OS. Hence, the proposed testbed ensures the program compatibility and provide a full-stack networking capability, and provides real-time data delivery of the high-priority packets. Results of the experiments show that the transmission delay of the high-priority packet is bounded, the time slot acquisition in the testbed is functional, and the TDMA implementation has significantly strong stability compared to IEEE 802.11p.

In the future the ACK and the retransmission functions for the upper layer packets in the FPGA subsystem will be implemented. We will look into more complex scenarios and topologies to further test the performance of the testbed.

ACKNOWLEDGMENT

This article was presented in part at the 2019 IEEE International Conference on Communications (ICC) [DOI: 10.1109/ICC.2019.8761412].

REFERENCES

[1] J. Wu, H. Lu, and Y. Xiang, "A hard real-time testbed for distributed TDMA-based MAC protocols in VANETs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[2] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*, IEEE Standard 802.11p-2010, Jul. 2010, pp. 1–51.

[3] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[4] National Highway Traffic Safety Administration, "Vehicle safety communications project: Task 3 final report: Identify intelligent vehicle safety applications enabled by DSRC," Washington, DC, USA, Tech. Rep. DOT HS 809 859, 2005.

[5] *Study on LTE Support for Vehicle to Everything (V2X) Services (Release 14)*, document TR 22.885 V14.0.0, 3GPP, Dec. 2015.

[6] A. Bazzi, A. Zanella, and B. M. Masini, "Optimizing the resource allocation of periodic messages with different sizes in LTE-V2V," *IEEE Access*, vol. 7, pp. 43820–43830, 2019.

[7] M. Hadded, P. Muhlethaler, A. Laouiti, R. Zagrouba, and L. A. Saidane, "TDMA-based MAC protocols for vehicular ad hoc networks: A survey, qualitative analysis, and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2461–2492, 4th Quart., 2015.

[8] T. V. Nguyen, F. Baccelli, K. Zhu, S. Subramanian, and X. Wu, "A performance analysis of CSMA based broadcast protocol in VANETs," in *Proc. 32rd IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2013, pp. 2805–2813.

[9] *IEEE Draft Standard for Information Technology—Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Standard IEEE P802.11- REVmd/D1.0, Feb. 2018, pp. 1–4226.

[10] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11bd & 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70169–70184, 2019.

[11] H. A. Omar, W. Zhuang, and L. Li, "VeMAC: A TDMA-based MAC protocol for reliable broadcast in VANETs," *IEEE Trans. Mobile Comput.*, vol. 12, no. 9, pp. 1724–1736, Sep. 2013.

[12] P. Chen, J. Zheng, and Y. Wu, "A-VeMAC: An adaptive vehicular MAC protocol for vehicular ad hoc networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[13] F. Lyu, H. Zhu, H. Zhou, L. Qian, W. Xu, M. Li, and X. Shen, "MoMAC: Mobility-aware and collision-avoidance MAC for safety applications in VANETs," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10590–10602, Nov. 2018.

[14] V. Dragonas, K. Oikonomou, K. Giannakis, and I. Stavrakakis, "A disjoint frame topology-independent TDMA MAC policy for safety applications in vehicular networks," *Ad Hoc Netw.*, vol. 79, pp. 43–52, Oct. 2018.

[15] B. Li, F. Hou, C. Zhang, S. Li, S. Ji, and S. Chen, "MAC-AC: A novel distributed MAC protocol for accessing channel in vehicular ad hoc networks," in *Proc. IEEE 92nd Veh. Technol. Conf. (VTC-Fall)*, Nov. 2020, pp. 1–5.

[16] X. Jiang and D. H. Du, "PTMAC: A prediction-based TDMA MAC protocol for reducing packet collisions in VANET," *IEEE Trans. Veh. Technol.*, vol. 65, no. 11, pp. 9209–9223, Nov. 2016.

[17] S. Li, Y. Liu, and J. Wang, "An efficient broadcast scheme for safety-related services in distributed TDMA-based VANETs," *IEEE Commun. Lett.*, vol. 23, no. 8, pp. 1432–1436, Aug. 2019.

[18] S. Bharati, H. A. Omar, and W. Zhuang, "Enhancing transmission collision detection for distributed TDMA in vehicular networks," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 13, no. 3s, p. 37, Aug. 2017.

[19] Z. Tianjiao and Z. Qi, "Game-based TDMA MAC protocol for vehicular network," *J. Commun. Netw.*, vol. 19, no. 3, pp. 209–217, 2017.

[20] F. Lyu, N. Cheng, H. Zhu, H. Zhou, W. Xu, M. Li, and X. Shen, "Towards rear-end collision avoidance: Adaptive beaconing for connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 1248–1263, Feb. 2021.

[21] Y. Deng, D. Kim, Z. Li, and Y.-J. Choi, "Implementing distributed TDMA using relative distance in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7295–7305, Jul. 2020.

[22] F. Bai, D. D. Stancil, and H. Krishnan, "Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2010, pp. 329–340.

[23] M. E. Renda, G. Resta, P. Santi, F. Martelli, and A. Franchini, "IEEE 802.11p VANets: Experimental evaluation of packet inter-reception time," *Comput. Commun.*, vol. 75, pp. 26–38, Feb. 2016.

[24] J. Wu, H. Lu, and Y. Xiang, "Measurement and comparison of sub-1 GHz and IEEE 802.11p in vehicular networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 1063–1066.

[25] H. A. Omar, N. Lu, and W. Zhuang, "Wireless access technologies for vehicular network safety applications," *IEEE Netw.*, vol. 30, no. 4, pp. 22–26, Jul./Aug. 2016.

[26] X. Shen, H. A. Omar, N. Lu, S. Bharati, and W. Zhuang, "Method, system and apparatus for enabling vehicular communications," U.S. Patent 15 519 526, Apr. 14, 2017.

[27] P. Djukic and P. Mohapatra, "Soft-TDMAC: A software-based 802.11 overlay TDMA MAC with microsecond synchronization," *IEEE Trans. Mobile Comput.*, vol. 11, no. 3, pp. 478–491, Mar. 2012.

[28] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications," in *Proc. IEEE 34th Real-Time Syst. Symp. (RTSS)*, Dec. 2013, pp. 140–149.

[29] W. Torfs and C. Blondia, "TDMA on commercial of-the-shelf hardware: Fact and fiction revealed," *AEU-Int. J. Electron. Commun.*, vol. 69, no. 5, pp. 800–813, May 2015.

[30] Y. Cheng, D. Yang, and H. Zhou, "Det-WiFi: A multihop TDMA MAC implementation for industrial deterministic applications based on commodity 802.11 hardware," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–10, Apr. 2017.

[31] P. Agostini, R. Knopp, J. Härri, and N. Haziza, "Implementation and test of a DSRC prototype on OpenAirInterface SDR platform," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Jun. 2013, pp. 510–514.

[32] OpenAirInterface. *The OpenAirInterfaceTM Software Alliance*. Accessed: Aug. 3, 2021. [Online]. Available: https://openairinterface.org/

[33] SRS Team. *srsRAN*. Accessed: Aug. 3, 2021. [Online]. Available: https://www.srsran.com/

[34] F. Borgonovo, A. Capone, M. Cesana, and L. Fratta, "ADHOC MAC: New MAC architecture for ad hoc networks providing efficient and reliable point-to-point and broadcast services," *Wireless Netw.*, vol. 10, no. 4, pp. 359–366, 2004.

[35] J. Wu, H. Lu, Y. Xiang, and F. Wang. *SATMAC: Self-Adaptive TDMA-Based MAC Protocol for VANETs*. Accessed: Aug. 2021. [Online]. Available: https://github.com/wujingbang/satmac-codes

[36] S. Johari and M. B. Krishna, "TDMA based contention-free MAC protocols for vehicular ad hoc networks: A survey," *Veh. Commun.*, vol. 28, Apr. 2021, Art. no. 100308.

[37] J. Liu, F. Ren, L. Miao, and C. Lin, "A-ADHOC: An adaptive real-time distributed MAC protocol for vehicular ad hoc networks," *Mobile Netw. Appl.*, vol. 16, no. 5, pp. 576–585, Oct. 2011.

[38] W. Yang, W. Liu, P. Li, and L. Sun, "TDMA-based control channel access for IEEE 802.11p in VANETs," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 8, Aug. 2014, Art. no. 579791.

[39] F. Lyu, H. Zhu, H. Zhou, W. Xu, N. Zhang, M. Li, and X. Shen, "SS-MAC: A novel time slot-sharing MAC for safety messages broadcasting in VANETs," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3586–3597, Apr. 2018.

[40] F. Lyu, J. Ren, P. Yang, N. Cheng, W. Tang, Y. Zhang, and X. S. Shen, "Fine-grained TDMA MAC design toward ultra-reliable broadcast for autonomous driving," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 46–53, Aug. 2019.

[41] S. Li, Y. Liu, and J. Wang, "ASTSMAC: Application suitable time-slot sharing MAC protocol for vehicular ad hoc networks," *IEEE Access*, vol. 7, pp. 118077–118087, 2019.

[42] V. Nguyen, T. Z. Oo, P. Chuan, and C. S. Hong, "An efficient time slot acquisition on the hybrid TDMA/CSMA multichannel MAC in VANETs," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 970–973, May 2016.

[43] V. Nguyen, T. A. Khoa, T. Z. Oo, N. H. Tran, C. S. Hong, and E.-N. Huh, "Time slot utilization for efficient multi-channel MAC protocol in VANETs," *Sensors*, vol. 18, no. 9, p. 3028, Sep. 2018.

[44] S. Li, Y. Liu, J. Wang, Y. Ge, L. Deng, and W. Deng, "TCGMAC: A TDMA-based MAC protocol with collision alleviation based on slot declaration and game theory in VANETS," *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 12, p. e3730, Dec. 2019.

[45] X. Zhang, X. Jiang, and M. Zhang, "A black-burst based time slot acquisition scheme for the hybrid TDMA/CSMA multichannel MAC in VANETs," *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 137–140, Feb. 2019.

[46] S. Li, Y. Liu, J. Wang, and Z. Sun, ''SCMAC: A slotted-contention-based media access control protocol for cooperative safety in VANETs,'' *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3812–3821, May 2020.

[47] H. A. Omar, W. Zhuang, and L. Li, ''Evaluation of VeMAC for V2V and V2R communications under unbalanced vehicle traffic,'' in *Proc. IEEE VTC Fall*, Sep. 2012, pp. 1–5.

[48] 3GPP. *Release 12: TR 21.101*. Accessed: Aug. 2021. [Online]. Available: https://www.3gpp.org/specifications/releases/68-release-12

[49] 3GPP. *Release 14: TR 21.914*. Accessed: Aug. 2021. [Online]. Available: https://www.3gpp.org/release-14

[50] R. Molina-Masegosa and J. Gozalvez, ''LTE-V for sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications,'' *IEEE Veh. Technol. Mag.*, vol. 12, no. 4, pp. 30–39, Dec. 2017.

[51] R. Molina-Masegosa and J. Gozalvez, ''System level evaluation of LTE-V2V mode 4 communications and its distributed scheduling,'' in *Proc. IEEE 85th Veh. Technol. Conf. (VTC-Spring)*, Jun. 2017, pp. 1–5.

[52] L. M. Lopez, ''Performance of sensing-based semi-persistent scheduling (SPS) in LTE-V2X release 14 distributed mode,'' M.S. thesis, Univ. Politècnica Catalunya, Barcelona, Spain, 2019.

[53] A. Bazzi, A. Zanella, and B. M. Masini, ''Optimizing the resource allocation of periodic messages with different sizes in LTE-V2V,'' *IEEE Access*, vol. 7, pp. 43820–43830, 2019.

[54] A. Nabil, K. Kaur, C. Dietrich, and V. Marojevic, ''Performance analysis of sensing-based semi-persistent scheduling in C-V2X networks,'' in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–5.

[55] F. Eckermann, M. Kahlert, and C. Wietfeld, ''Performance analysis of C-V2X mode 4 communication introducing an open-source C-V2X simulator,'' in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2019, pp. 1–5.

[56] T. Shimizu, B. Cheng, H. Lu, and J. Kenney, ''Comparative analysis of DSRC and LTE-V2X PC5 mode 4 with SAE congestion control,'' in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2020, pp. 1–8.

[57] R. Molina-Masegosa, J. Gozalvez, and M. Sepulcre, ''Comparison of IEEE 802.11p and LTE-V2X: An evaluation with periodic and aperiodic messages of constant and variable size,'' *IEEE Access*, vol. 8, pp. 121526–121548, 2020.

[58] T. Park and K. G. Shin, ''SNB: Reduction of consecutive message reception failures in C-V2X communications,'' in *Proc. IEEE VTC-Fall*, Nov. 2020, pp. 1–6.

[59] P. Wendland, G. Schaefer, and R. Thomä, ''An application-oriented evaluation of LTE-V's mode 4 for V2V communication,'' in *Proc. ACM/SIGAPP SAC*, New York, NY, USA, Apr. 2019, pp. 165–173.

[60] F. Borgonovo, L. Campelli, M. Cesana, and L. Fratta, ''Impact of user mobility on the broadcast service efficiency of the ADHOC MAC protocol,'' in *Proc. IEEE 61st Veh. Technol. Conf.*, vol. 4, May 2005, pp. 2310–2314.

[61] Ministry of Internal Affairs and Communications of Japan. *Current Situation of 335.4-960 MHz Spectrum Use in Japan*. Accessed: Aug. 2021. [Online]. Available: http://www.tele.soumu.go.jp/resource/e/search/myuse/use0303/335m.pdf

[62] *Ardupilot*. Accessed: Aug. 2021. [Online]. Available: http://ardupilot.org/

[63] U-Blox. *NEO/LEA-M8T Data Sheet*. Accessed: Aug. 2021. [Online]. Available: https://www.u-blox.com/sites/default/files/NEO-LEA-M8T-FW3_DataSheet_

[64] K. F. Hasan, Y. Feng, and Y.-C. Tian, ''GNSS time synchronization in vehicular ad-hoc networks: Benefits and feasibility,'' *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3915–3924, Mar. 2018.

[65] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer, ''Position-aware ad hoc wireless networks for inter-vehicle communications: The Fleetnet project,'' in *Proc. ACM MobiHoc*, 2001, pp. 259–262.

JINGBANG WU received the B.S. degree from Xiangtan University, and the M.S. and Ph.D. degrees from Beijing Institute of Technology. He is currently an Assistant Professor with Beijing Technology and Business University. His research interests include routing protocols and MAC layer algorithms in VANETs.

LUYUAN ZHANG received the B.S. degree from Beijing Technology and Business University, where she is currently pursuing the master's degree. Her research interest includes operating systems.

YICHEN LIU is currently pursuing the bachelor's degree with Beijing Technology and Business University.