P-TDMA-SYS: A TDMA System over Commodity 802.11 Hardware for Mobile Ad-Hoc Networks

Zechen Lin, Zhizhong Ding, Qingxin Hu, and Shuai Tao
School of Computer and Information, Hefei University of Technology, Hefei, 230009, China
Email: {falcon_zechen_lin, 9696210, taoshuai029}@163.com; zzding@hfut.edu.cn

Abstract —In some scenarios such as catastrophes and military operations, there is a need for communication without an infrastructure. In urgent cases, in-time and reliable channel access must be ensured. However, the channel access delay in widely used CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) network increases unpredictably when the channel is sensed busy. Therefore, TDMA (Time Division Multiple Access) gets more and more attention in mobile ad-hoc network research. This paper presents a TDMA network system (named as P-TDMA-SYS) that is designed to support real-time applications in mobile ad-hoc networks (MANETs) and successfully works on commodity 802.11 hardware. The key feature of P-TDMA-SYS is its microsecond synchronization and adaptive medium access policy. P-TDMA-SYS's implementation contains two major modules. One is the P-KER-MAC, a TDMA-based MAC protocol, running over commodity 802.11 hardware. The other is P-TCP, a cross-layer TCP congestion control algorithm specifically for P-KER-MAC. The presented TDMA-based MAC over commodity 802.11 hardware has been tested with a detailed accounting of the hardware and operation system overheads. Furthermore, P-TDMA-SYS has been compared with 802.11 CSMA/CA-based network in the single-hop/multi-hop scenarios. The test results show that our implementation achieves reliable throughput and low delay/jitter for supporting real-time applications.

Index Terms—TDMA MAC, ad-hoc networks, IEEE 802.11

I. Introduction

There has long been a demand for jam-resistant and none-infrastructure communications in the scene of disaster rescue or military campaign. The most suitable network type to meet the need is mobile ad-hoc networks (MANETs). Currently, researches on MANETs are mainly based on CSMA/CA media access control (MAC) protocol due to the wide application of IEEE 802.11 and 802.15.4 standards. However, CSMA/CA-based MAC has problems when it used in MANETs, such as packet collisions and unpredictable transmission delay [1]. An effective way to resolve the problems is to use synchronized TDMA-based MAC protocol [2], but there is no off-the-shelf TDMA wireless network card available in the market, to the best of our knowledge, which can be directly used in MANETs.

On the other hand, 802.11 is an attractive air interface for wireless communication in terms of wideband transmission and low cost network and there are a lot of commercial products on the market. For a fast prototype development of our MANETs product, we implemented a TDMA-based MAC (named as P-KER-MAC) over 802.11 hardware and realized further a TDMA system (named as P-TDMA-SYS) for MANETs by integrating the routing protocol AODV [3] in order to support multi-hop communications.

There are two challenges to implement a TDMA-based wideband MANET:

1) What kind of TDMA mechanism should be adopted in order to support ad-hoc network and to get better performance than that of CSMA/CA?

Traditional TDMA is normally used in an infrastructure network in which the assignation of time-slot is controlled centrally. That is, it cannot be used in MANETs. There are some papers in which TDMA-based are proposed [4]-[5], however they are not so mature to be implemented in a concrete development. A good TDMA mechanism which has been standardized by ITU (International Telecommunication Union) is STDMA (Self-organizing TDMA) [6]. STDMA is mainly designed for position reporting of ships and its Physical layer and MAC layer have to be changed or modified in order to support wideband MANETs of our case. Our MAC implementation (i.e. P-KER-MAC) is designed on the top of STDMA and implemented in the Linux kernel as a driver.

2) How to realize a TDMA network card based on the hardware of a commercial 802.11 CSMA/CA network card?

The 802.11 network card is obviously the first choice when high transmission data rate and low cost are taken into account. The key issue is how to get an effective solution to disable its CSMA/CA functions, and then replace it with a TDMA MAC. Some researchers have done a similar work [7]-[10] and these works lay a good foundation for our job. Detailed comparison will be presented in Section II.

To implement an effective TDMA-based wideband MANET system by taking the above approach, there are still some problems which should be solved.

1) Provide a solution of clock synchronization mechanism for TDMA system if there is no GPS or other time service. In P-TDMA-SYS, we proposed a relative clock synchronization algorithm to ensure reliable transmission.

2) Implement a routing protocol which supports multi-hop wireless communication.

Although most Wi-Fi terminals have an ad-hoc mode now, they cannot support multi-hop transmission. Many researchers have proposed routing protocol for MANETs, but it seems that AODV used in ZigBee/802.15.4 is the

Manuscript received May 2, 2016; revised August 23, 2016.
This work is supported by Hefei University of Technology (JZ2015QSJH0536).
Corresponding author email: zzding@hfut.edu.cn.
doi:10.12720/jcm.11.8.710-725

only one with successful application so far. Therefore, we integrate AODV module into P-TDMA-SYS.

3) Modify the existed TCP congestion control algorithm so that it works more efficiently in a TDMA MANET system with support of TCP/IP.

TCP/IP is originally designed for CSMA-based wire and wireless network. When it is used in a TDMA-based network, its performance is not satisfying in some cases. We develop a new TCP congestion control algorithm (named as P-TCP) in P-TDMA-SYS.

The P-TDMA-SYS is tested on our testbed and its performance is evaluated. The test results show that our implementation achieves reliable throughput and low delay/jitter for supporting real-time applications.

The rest of the paper is organized as follows. Section II compares the implementations of TDMA-based MAC over 802.11 hardware. Section III describes P-KER-MAC design. Subsequently, Section IV presents the details of P-TDMA-SYS's Linux implementation over commodity 802.11 hardware. Section V shows the construction of test cases and the test results of our network system. In the end, Section VI concludes the paper with a few points of discussion.

TABLE I: COMPARISON WITH THE OTHER TDMA-BASED MACS

MAC	Clock Synchronization(Sync)		MAC Mode		Implementation	
	Sync Mode	Sync Precision	Schedule	Routing	Open Source	Testbed
Soft-MAC	in-band	order of μs	distributed	2-hop	MadWiFi	802.11a
Lit-MAC	in-band	order of μs	centralized	multi-hop	MadWiFi	802.11g
Free-MAC	out-band	order of ms	centralized	1-hop	none	none
WiLDNet	in-band	order of ms	centralized	multi-hop	MadWiFi	802.11a/g/n
P-TDMA-SYS (our work)	in-band	order of μs	distributed	multi-hop	ath9k-htc firmware	802.11g

II. TDMA MAC OVER 802.11 HARDWARE

The most important part of the TDMA network system is the MAC protocol. TDMA-based MAC protocols are aplenty, and have been previously proposed for commodity 802.11 networks. Table I compares prior TDMA-based MAC [7]-[10] implementations with our work synoptically, the details are as below.

Free-MAC [9] and WiLDNet [10] are not tightly synchronized. Lit-MAC [7] uses out-of-band synchronization, it can schedule links in any pattern in the TDMA frame. Soft-MAC [8] synchronize after each transmission, so synchronization is done, in-band, through data transmissions.

Lit-MAC outlines the design of a TDMA-based MAC protocol based on out-band clock synchronization and takes into account hardware bottlenecks such as clock drift, processing delay, the guard interval and slot duration, etc.

Soft-MAC proposes a pairwise synchronization algorithm, and takes into account propagation delay to calculate clock difference between two nodes. It makes use of a guard band to account for processing delays encountered while packet transmission, and carries out measurements to empirically tune the value of the guard band.

Our methodology of modifying the open source driver to build a TDMA network system on commodity 802.11 hardware is similar to Soft-MAC, Lit MAC. Soft-MAC proposes a distributed mechanism for nodes to arrive at an agreement on slot usage. The design imposes the significant restriction that all nodes should use the same channel in all time slots. It also assumes that interference is limited to a 2-hop neighborhood. Lit-MAC present a light-weight, centralized, multi-channel and connection oriented TDMA-based MAC protocol, for operation in a wireless multi-hop mesh network.

Soft-MAC, Lit-MAC, Free-MAC and WiLDNet all have built TDMA based MAC protocols over commodity 802.11 hardware with the Atheros MadWiFi driver [11]. Soft-MAC implemented entirely in Linux user space, Lit-MAC and WiLDNet implemented in Linux kernel space.

III. P-KER-MAC DESIGN

P-KER-MAC is the kernel TDMA-based MAC for P-TDMA-SYS. It follows three main design principles. First, tight in-band clock synchronization must be built-into the P-KER-MAC. Second, P-KER-MAC must maintain a dynamic, self-starting and self-organizing slot schedule algorithm to support MANETs. Additionally, the algorithm also should support real-time applications. Third, P-KER-MAC should be robust to wireless network's common errors, such as data packet loss and hidden node. Meanwhile, it should be simple and low overhead for Operating System (OS).

In this section, we describe the design of P-KER-MAC, showing its network composition, slot structure and frame structure, network entry procedure and clock synchronization. Furthermore, we set the P-KER-MAC's parameters and present an adaptive report rate algorithm to support real-time applications. And we solve the S-TDMA's slot collision problem that caused by hidden nodes.

A. Network Composition

The P-TDMA-SYS network boots up when the first node comes online, and we call this node the "master". Master will broadcast the beacons to provide a rough clock reference for the rest of the network. Then, other node entry the network, and we call those nodes the "slave". Each master with some slaves establish a network, and we call this network as a "cluster" network (Fig. 1). In each cluster network, all nodes reuse a slot

flow to send and receive data with each other. Each cluster network may cross the range, so they are distinguished from each other by a unique cluster-ID.

Any node periodically transmits beacons in the cluster network. P-TDMA-SYS's beacon is fixed length and transmitted in per frame duration. Since a beacon can only carry limited information, each node broadcasts its beacon to ensure that all neighbors are advertised for keeping an alive list of its neighbors. We note that "neighbor" refers to all of the nodes that the sender is aware of, except the nodes that the sender cannot hear from directly. Before sending the network beacon, the node marks the current timestamp on it. Particularly, the timestamp of the master will be used by the rough clock synchronization algorithm.

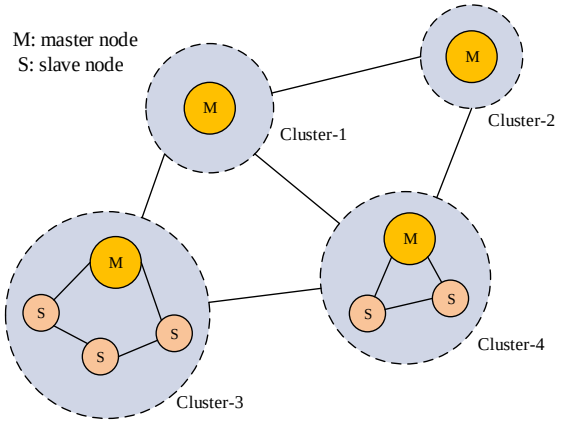

Fig. 1. The network composition of P-TDMA-SYS

B. Slot Structure & Frame Structure

In P-KER-MAC, the time dimension of the channel is divided into fixed N_f slots, whose duration T_{slot} is suited to host one data packet duration (T_{data}) plus a necessary guard interval (T_{guard}). The slot duration (T_{slot}) can be expressed as

$$T_{slot} = T_{data} + T_{guard} \tag{1}$$

Hence, P-KER-MAC views the time in terms of slots, T_{frame} long, and groups TDMA slots into fixed size frames (

Fig. 2). Each frame consists of N_f slots. The number N_f of available slots per frame duration T_{frame} depends on the frame duration and slot duration, and can be expressed as

$$N_f = \lfloor T_{frame} / T_{slot} \rfloor \tag{2}$$

where $\lfloor \cdot \rfloor$ is the floor function.

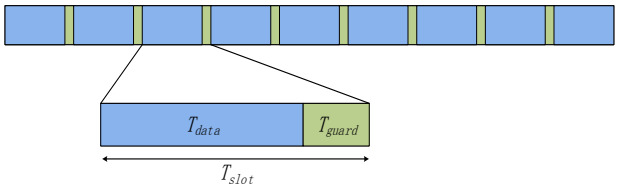

Fig. 2. P-TDMA-SYS's frame and slot structure.

C. P-KER-MAC's Parameters

In the P-KER-MAC, the unit of all time parameters are set as microsecond. Table II summarizes the various parameters in our system.

TABLE II: P-KER-MAC'S PARAMETERS

Notion	Description
t_{sm}	The node's start moment.
$t_{slot-sm}$	The node's slot start moment, which align the cluster network slot boundary.
$t_{current}$	The node's current clock.
T_{frame}	One frame duration.
T_{slot}	One slot duration.
T_{data}	The data duration in a slot duration.
T_{guard}	The guard duration in a slot duration.
N_{slot}	The number of available slots per frame.
L_{slot}	The data length of per slot.
R_{min}	The lower limit of report rate.
R_{max}	The upper limit of report rate.
$R_{current}$	The current value of report rate.
Q_{max}	The maximum length of data queue.
$Q_{current}$	The current length of data queue.
S_{cur}	The current slot index.
$T_{cur-exc}$	The exceed portion duration of current slot.
S_{trig}	Next index of slot trigger.

D. Network Entry Procedure

The P-KER-MAC's medium access policy is based on a slot reservation process: the transmitting node autonomously reserve the slots they are going to use for their transmissions based on other nodes' slots reservation information, which has to be presented in the header of all data packets. The network entry process is organized into 4 phases, which are sequentially performed when a new node enters the network: Initialization, Network Entry & Rough Synchronization, First Frame & Precise Synchronization and Continuous Operations.

TABLE III: SLOT STATES

Slot State	Description
Free	The slot is not being reserved by any node.
Busy	The packet has been detected in the same slot of the previous frame but the packet failed to be decoded successfully.
Externally allocated	The slot is allocated for transmission by another user.
Internally allocated	The slot will be used by the current node self.

Initialization In the initialization phase, a node listens to the channel for 2 frames duration ($2 \times T_{frame}$) at first and marks the slot state of all its N_f slots in its slots table. The possible states that a slot can assume in Table III.

Network Entry & Rough Synchronization The purpose of the network entry phase is for a node to

advertise its presence to the network. To do so, a node transmits a one-time network entry packet in one of the first slots sensed free in the previous phase, randomly chosen according to a p-persistent mechanism [12]. In this phase, the slave selects a latest master's beacon, and use the beacon's timestamp as the rough relative clock boundary.

Once any node is aware of the presence of the new node in a cluster network, it is time for it to allocate a sufficient number of transmission slots to satisfy its default report rate of R_{min} packets per frame duration.

First Frame & Precise Synchronization Once the node has announced its presence, the node enters the first frame phase. The task of this phase is to announce and reserve additional slots in order to fulfill the default report rate. The first frame phase develops as follows (Fig. 3):

1) Set the nominal increment value (NI) to $\lceil N_f / R_{report} \rceil$.
2) Randomly select a nominal start slot (NSS) out of the first NI slots.
3) Derive additional ($R_{report} - 1$) nominal slots (NS) by subsequently adding NI slots to NSS .And NS can be calculated as

$$NS = NSS + n \times NI (0 < n < R_{report}) \quad (3)$$

4) For the NSS and each NS :
   a) Construct a selection interval (SI) by adding $\lceil (N_f / 2R_{report}) \times p_{SI} \rceil$ number of slots to the left and to the right, with p_{SI} being the SI ratio, e.g. 20%.

$$SI = \{NS - (p_{SI} \times NI), NS + (p_{SI} \times NI)\} \quad (4)$$

   b) Compile a set of candidate slots within this SI .
   c) Randomly choose one of these candidate slots as nominal transmission slot (NTS) .

When announcing the allocation of a selected slot, a random timeout value $p_{time-out}$ is drawn from statically defined minimum and maximum timeout limits (e.g. $1 \le p_{time-out} \le 8$). Hence, each allocated slot gets its own timeout value. During this phase, slaves achieve precise synchronization by requesting clock offsets with the master of cluster. And the round trip of synchronization is similar to the Simple Network Time Protocol (SNTP) [13].

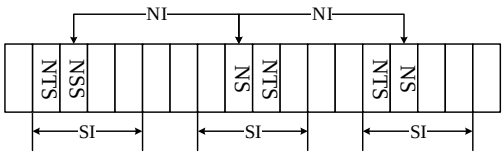

Fig. 3. Slot reservation process.

Continuous Operations After the first allocation within the first frame phase is performed, the nodes enter continuous operation. During this phase the node performs re-allocations whenever the internal timeout of a slot expires. The rules for re-allocation are different with the first frame phase. In this phase, the node uses an adaptive report rate along with the rate of the application layer. Surely, a node is allowed to stick to the current slot if no candidate slot is available. Furthermore, when re-reserving a slot, the semantics of the offset value changes. While it normally indicates the offset to the subsequent transmission, it then indicates the offset to the newly selected slot in the next frame.

E. Clock Synchronization

Highly precise clock synchronization (clock-sync) is essential for efficient TDMA system. For most TDMA systems, they process the clock-sync for aligning the absolute frame boundary. However, in P-KER-MAC, our clock-sync purpose is to align the slot boundary. In a cluster network, optionally, every cluster network's master is specified manually or voluntarily. Each slave node synchronizes to its master node, defining a master/slave synchronization relationship for all nodes in a same cluster network. For the whole network, the master of each cluster network has been specified the parallel level. Each slave determines its synchronization source by using the latest synchronization beacon of the master node.

Rough Clock Synchronization In the *network entry* phase, a slave node selects the latest master's beacon, and use the receiving timestamp of the beacon as the rough relative slot boundary. As for the timestamp, it's the MAC's timestamp.

Precise Clock Synchronization In the *First Frame* phase, the slave node achieve precise synchronization by sending the clock-sync requests to get clock offset with the master of cluster network (Fig. 4).

In one exchange, the slave node sends a clock-sync request with a timestamp $T_1 = CLK_{slave}(t_1)$ which is its clock at time t_1 to the master node. After receiving the request at time t_2 , the master node uses its timestamp, $T_2 = CLK_{master}(t_2)$, to find the clock difference between the clocks relative to itself $T_{offset-ms} = T_2 - T_1$. After some time, the master node sends a clock-sync response at time t_3 , with the timestamp $T_3 = CLK_{master}(t_3)$ and the previously found time difference $T_{offset-ms}$. After receiving the clock-sync response, at time t_4 , the slave node calculates its relative time difference $T_{offset-sm} = T_4 - T_3$, where $T_4 = CLK_{slave}(t_4)$.

At the end of the exchange, the slave node calculates the clock offset:

$$T_{offset} = \frac{1}{2}(T_{offset-sm} - T_{offset-ms}) \quad (5)$$

and round-trip delay with the master node:

$$T_{delay} = \frac{1}{2}(T_{offset-sm} + T_{offset-ms}) \quad (6)$$

The slave node repeats this process several times, and collect the clock offsets to the master node. Then, the clock-sync algorithm first filters the raw clock offsets. The filter removes the values of clock offsets that are out of the valid range. In the final step, the slave node is to use the averaged clock offset of the master node to re-synchronize its local clock.

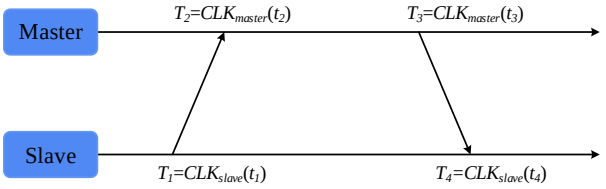

Fig. 4. Precise clock synchronization.

After the clock-sync, there must be a relative relationship between the slave and master node,

$$(t_{sm-master} - t_{sm-slave}) \% T_{frame} = N \times T_{slot} \quad (7)$$

where $t_{sm-master}$ is the start moment of the master node, $t_{sm-slave}$ is the synchronized start moment of the slave node, $N \in \{0,1,2,...\}$. The slot schedule algorithm has no requirement that $t_{sm-master}$ is strictly equal to $t_{sm-slave}$. We just need they can maintain a relative relationship just (7).

F. Slot Collision Avoidance

There are two kinds of situations which can cause slot collision. One is the direct slot collision, the other is the hidden node slot collision.

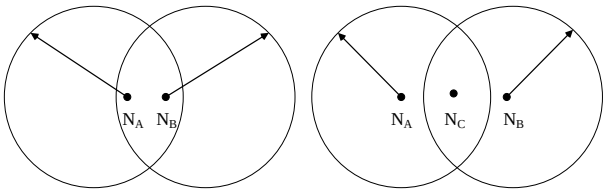

Fig. 5. Slot collisions.

In Fig. 5(a), node A (N_A) and node B (N_B) are within the transmission range of one another. In this case, if N_A and N_B start transmission at the same time, then packets from both nodes will be expected to raise slot collision event. It is the direct slot collision. In Fig. 5(b), the nodes N_A and N_B are not within the transmission range of one another, but there is a third N_C that is within the transmission range of both N_A and N_B. In this case, if N_A and N_B start transmissions at the same moment, then N_C will be expected to receive from both N_A and N_B at the same time. It is the hidden node slot collision.

When any node wants to transmit a packet, it must broadcast the reserved slot index in advance. Hence, we present a passive detection method to solve the slot collision. During the transmission, if any one node finds a slot is reserved more than 2 nodes in its slot table, it will broadcast a slot collision message to all nodes. And the collision message includes the relative slot index and source address. Then, those collision nodes that have chosen the same slot will release the slot. Through this mechanism, the slot collision can be resolved effectively.

G. Real-time Application Support

P-KER-MAC should support the streaming media applications that are real-time voice and video. To adapt the parameter settings to support real-time voice, we consider the G.723.1 codec [14] which needs 24 bytes to be transmitted every 30ms. For the real-time video, we use the MPEG-4 Part 10 (H.264) [15] as video coding standard for the application, and Table IV show the relationship between frame resolution and bit rate.

TABLE IV: H.264'S FRAME RESOLUTION WITH BIT RATE

Typical Frame Resolution	Typical Frames per Second	Maximum Bit Rate (Kbps)
176×144	15	64
320×240	10	192
352×288	15	384
352×288	30	768
720×480	15	4000
1280×720	30	14000
1920×1080	30	20000

In order to support a normal quality video (352×288, 30 frames/s), the P-TDMA-SYS must provide a bandwidth which exceeds 768 Kbps. In addition, the G.723.1 codec require the slot interval must be limited within 30ms. On the basis of application's demand, we set the packet size of each slot (L_{slot}) to 540 bytes with the 54Mbps PHY rate. Then we set $T_{frame} = 1999816\mu s$, $T_{slot} = 310\mu s\ (T_{trans}) + 18\mu s\ (T_{guard}) = 328\mu s$. For matching the bit rate of video application, we set $R_{min} = 66$, $Q_{max} = 475$, $R_{max} = 475$. In these parameters settings, each node can be assigned a bandwidth which is $(R_{\max} / 2)L_{slot} = 1024$ Kbps. The slot interval, just the transmission interval of per packet, can be $T_{frame} / R_{\max} = 4.21ms$.

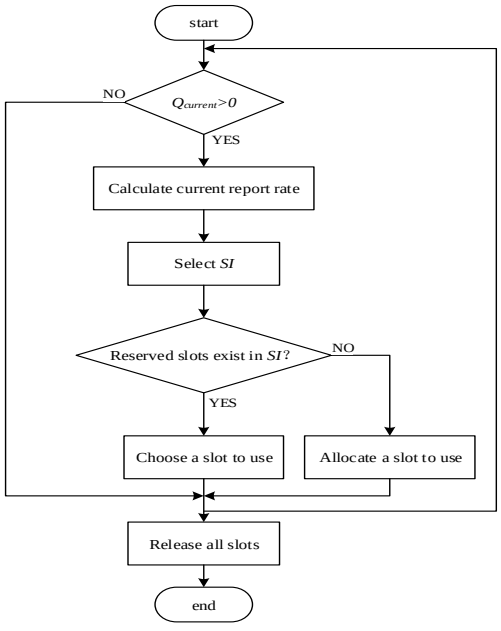

Fig. 6. Adaptive report rate scheduling process.

H. Adaptive Report Rate

Report rate is directly associated with the link bandwidth and latency. P-KER-MAC achieve an adaptive report rate scheduling to ensure real-time application requirements and full use of slot resources. Fig. 6 shows the adaptive report rate scheduling process. When the transport layer raise a transmission request, P-KER-MAC calculate the $R_{current}$ by

$$R_{current} = \min\{R_{max}, \max\{Q_{current}, R_{min}\}\} \quad (8)$$

Then, we can calculate the number of NI (C_{NI}) by

$$C_{NI} = N_f / R_{current} \quad (9)$$

Next, we choose the next free trigger slot index in the range of

$$\{(S_{cur} / C_{NI} + 1) \times C_{NI}, (S_{cur} / C_{NI} + 2) \times C_{NI}\} \quad (10)$$

In the range of (10), we try to choose a reserved slot, if not, we will allocate a new slot to use. This process will continue until the data queue is empty.

IV. P-TDMA-SYS's Linux Implementation

P-TDMA-SYS is fully implemented under the TCP/IP. In the link layer, we have implemented P-KER-MAC which is the kernel MAC of P-TDMA-SYS, a TDMA-based MAC protocol, running over commodity 802.11 hardware. In the internet layer, AODV routing module is added into P-TDMA-SYS to support the multi-hop networks. In the transport layer, we have implemented an improved TCP congestion control algorithm specifically for P-KER-MAC. Fig. 7 shows the various modules and their level.

P-KER-MAC provides precise control over the content and timing of wireless transmission and reception. Its Linux implementation has two major components (Fig. 8). The first component is the kernel module, which resides in the Linux kernel space and connects P-KER-MAC with the Linux kernel network services. The second component is the service module, which provides a slot schedule layer (SSL) and a hardware abstraction layer (HAL). The SSL uses the High-Resolution Timer (HRT) to arrange transmissions. As for the SSL's managements module, it is a collection of modules which concludes the packet management, slot table management, buffer management, etc. The HAL hides the implementation of 802.11 driver to provide a communication interface.

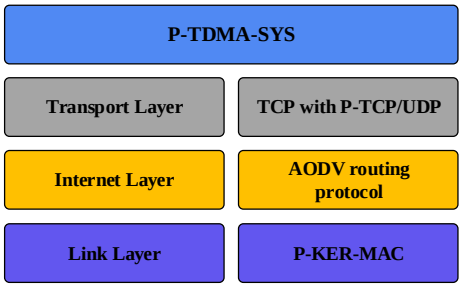

Fig. 7. The composition of P-TDMA-SYS.

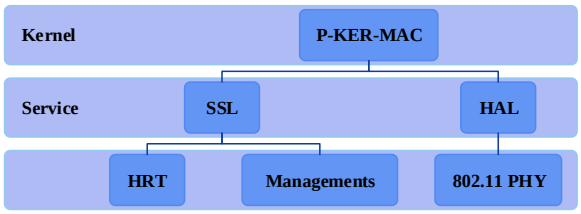

Fig. 8. The composition of P-KER-MAC.

This architecture has two major purposes. First, the kernel MAC is implemented in the Linux kernel IP networking stack, the P-TDMA-SYS can integrate two important modules, one is the AODV module, the other is the TCP/UDP protocol. Second, P-TDMA-SYS is divided into layered architecture, we can build test cases for our system's modules and prove its reliability.

A. Hardware & System Software Setup

We have implemented P-TDMA-SYS on the hardware platform of Lenovo M7150 PC, which use the CORE 2 DUO processors running at 3.06Ghz and install with Atheros wireless cards (TL-WN722N) on the AR9271 chipset. We use this 802.11 platform in the 2.4GHz frequency and 802.11g model. Our software platform is Ubuntu v14.04 (Linux kernel 3.18.24). In particular, we add profiling software to the firmware (open source ath9k-htc firmware) and make it allow us to measure 802.11 transmission duration. Most of important, we add codes to control per-packet transmission.

B. Disabling CSMA/CA

Once the packet is inserted into the hardware queue by the driver, it is essential that the packet is transmitted on-air immediately. However, the majority of existing 802.11 wireless devices are confined to the built-in CSMA/CA protocols. Hence, we had to study the CSMA/CA mechanism, and our work was inspired luckily by [16].

Reconfigurable NONE-CSMA/CA Transmitter The 802.11 MAC service is responsible for exchanging MAC Protocol Data Units (MPDUs) using CSMA/CA. The original standard offered the Distributed Coordination Function (DCF) [17] to manage this process. The 802.11e amendment extends DCF with Quality of Service (QoS) enhancements [18], resulting in enhanced distributed channel access (EDCA). With EDCA four different Access Classes (AC) are defined, allowing the prioritization of traffic.

The time a station must wait before it can transmit depends on several parameters and inter frame spaces (see Fig. 9). The shortest is the Short Interframe Space ($SIFS$) and is the time between successfully receiving a frame and sending an acknowledgement (ACK). Other frames must wait longer than $SIFS$, giving ACKs the highest priority. When a station wants to transmit a data frame of access class AC it first monitors the medium for a period equal to

$$AIFS[AC] = SIFS + AIFS[AC] \times SLOT \quad (11)$$

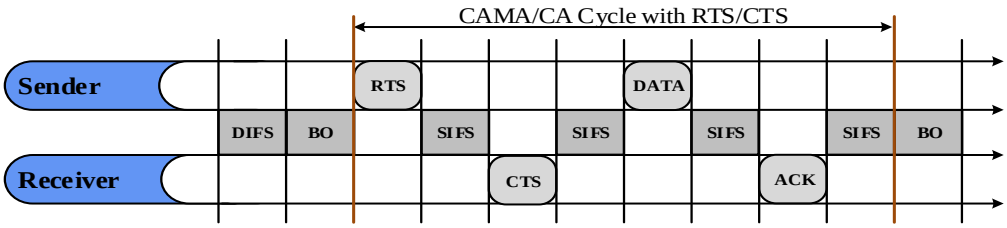

Fig. 9. 802.11's CSMA/CA mechanism.

where *SLOT* is the duration of one slot interval, and with *AIFSN* dependent on the access class. The value of *SIFS* and *SLOT* depend on the physical layer being used. If sender enable the Request To Send/Clear To Send (RTS/CTS) option, it must wait a CTS after send a RTS in each transmission after a *SIFS*. If the medium is idle during this time the station can transmit at once. In case the medium was busy the contention window $CW[AC]$ is set to $CW_{min}[AC]$ and the random back-off procedure is initiated. This procedure initializes the back-off counter to a value uniformly distributed over the interval $[0, CW_{min}[AC]]$. Once the medium has been idle for $AIFS[AC]$, the back-off counter is decremented for each slot where the medium is idle. When the counter is zero the station transmits the frame. Since *AC* is clear from context it will no longer be explicitly included. The values assigned to *AIFSN*, CW_{min}, and CW_{max} determine the priority of an access class.

When an MPDU is transmitted, a Physical Layer Convergence Protocol (PLCP) preamble and header is added. The preamble allows the receiver to synchronize to the transmission, and the header defines the modulation used for the MPDU. We focus on 802.11g long preamble mode, where (slightly simplified) the overhead of the PLCP is 23μs, *SIFS* is 10μs and *SLOT* is 9μs.

To turn a commodity 802.11 hardware into a reconfigurable NONE-CSMA/CA transmitter, we need to be able to carry out the following tasks:

1) Disable carrier sense.
2) Reset all 802.11 packet interframe spaces and disable back-off.
3) Prevent the chip from waiting for an ACK.
4) Disable RTS/CTS option.
5) Control the TX/RX process.

Ath9k-htc Firmware Modifications The open source ath9k-htc firmware runs on AR7010 and AR9271 chips, and is sent to the device after power up. In our platform, we choose the wireless card based on the AR9271 chip. The AR9271 chip is controlled using memory mapped registers. We implemented the following strategies to disable CSMA/CA by setting the register value (Table V). Especially, wireless radios are half-duplex and cannot listen while transmitting. On the contrary, wireless radios cannot transmit while receiving. For avoiding the unnecessary delays while transmitting, we abort the receiving the current frame by setting the register of (WLAN_BASE_ADDRESS+AR_DLAG_SW) to the value of ARDIAG_RX_ABORT when a transmission request arrives PHY. We make the data packet into 802.11's broadcast packet, hence, the traffic will not wait for an ACK.

TABLE V: REGISTER SETTINGS FOR DISABLING CSMA/CA

Register	Description	Value
AR_DIAG_SW	Disable carrier sense, abort ongoing reception, append bad FCS	CLEAR | IGNORE | IDLE
AR_D_GBL_IFS_SIFS	*SIFS*	0
AR_D_GBL_IFS_SLOT	*SLOT*	0
AR_QTXDP	CW_{min}, CW_{max} and *AIFSN*	0
AR_Q_TXE	Contains a pointer to the transmit queue.	-

C. Transmission Control

P-KER-MAC ensures each node's transmission embed to a relative conflict-free slot schedule. Before running the transmission in the network, P-KER-MAC should determine a default and fixed slot duration. For guaranteeing the robustness of slot duration, we calculate the slot duration and ensure that its transmission is over during the node's each slot.

The slot duration (T_{slot}) for an fixed l bytes packet, transmitted at PHY rate (m), is bounded into the message $p_m(l)$, which is measured in each slot. P-KER-MAC views wireless transmissions as consisting of the active part of the transmission, which is directly related to the packet size and the guard duration. The message on transmission duration is calculated with

$$p_m(l) = T_{trans} + T_{guard} \tag{12}$$

where T_{trans} is reserved for the transmission, T_{guard} is the duration of guard contained in a slot, ensures that the message covers various hardware and software delays.

Packet Transmission Duration In our hardware platform, every packet needs to go through the USB, for transfer from radio chip to the CPU, or from the CPU to the radio chip. So, we should take into account the timeout in the OS. Packet transmission process duration is

$$T_{trans} = T_{tx} + T_{rx} \tag{13}$$

In (13), the sending duration (T_{tx}) and the receiving duration (T_{rx}) both depend on the OS processing capacity. The whole process can be seen in Fig. 10.

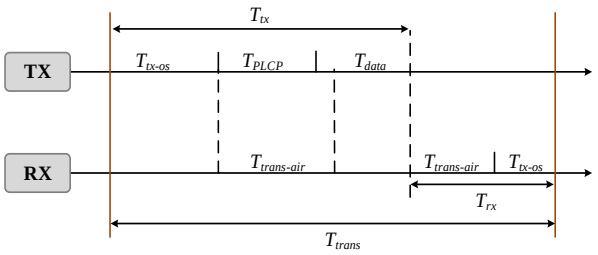

Fig. 10. Transmission process.

T_{tx} is sending process delay, which occurs from the time of the requested timeout, to the time a packet is generated and handed-off to the wireless card. T_{tx} is equal to

$$T_{tx} = T_{tx-os} + T_{PLCP} + T_{data} \tag{14}$$

$$T_{data} = \left\lceil \frac{l + h_{802.11}}{b_m} \right\rceil \times T_{ofdm} \tag{15}$$

where l is the packet size in bytes, $h_{802.11}$ is the number of bytes of 802.11 headers, $b_m \in \{3, 4.5, 6, 9, 12, 18, 24, 27\}$ is the number of bytes carried in each OFDM symbol. $m \in \{6, 9, 12, 18, 24, 36, 48, 54\}$ is the modulation rate (Table VI). The preamble allows the receiver to synchronize to the transmission, and the header defines the modulation used for the MPDU. We focus on 802.11g long preamble mode, where the overhead of the PLCP (T_{PLCP}) is 23μs. T_{ofdm} equals to 4μs, corresponding to single 802.11g Orthogonal Frequency Division Multiplexing (OFDM) symbol. According to test results, the T_{tx-os} is equal to $112 \pm 5\mu s$, representing the sending process duration in the OS.

T_{rx} is the duration from when the wireless card receives the packet, to the time the card finishes transmitting the packet, which includes transmission delay ($T_{trans-air}$) in the air.

$$T_{rx} = T_{trans-air} + T_{rx-os} \tag{16}$$

Due to our testbed is in-door, each test node is in the range of 20 square meters, hence, the value of $T_{trans-air}$ is close to 0μs. In our testbed, we measure T_{rx-os} which represents the receiving process duration in the OS, is equal to $80 \pm 5\mu s$.

TABLE VI: MODULATION RATE WITH PER OFDM SYMBOL

Rate (Mbps)	Modulation	Data bits per OFDM symbol (NDBPS)
6	BPSK	24
9	BPSK	36
12	QPSK	48
18	QPSK	72
24	16-QAM	96
36	16-QAM	144
48	64-QAM	192
54	64-QAM	216

Packet Guard Duration The guard duration should be longer than the delay in the air and the OS's error in the network. For conflict-free slot, it is sufficient that the guard time T_{guard} satisfies

$$T_{guard} = T_{re-trans-air} + T_{clk-drift} + T_{sch-drift} \tag{17}$$

$T_{re-trans-air}$ is reserved for the possible delay in the air. For a single cluster network, we assume that the max diameter of communication region is less than 1000m. Therefore, the maximum transmission delay in the air is approximately equal to 3.3μs. So we set $T_{re-trans-air}$ to 5μs to ensure transmission robust. The $T_{clk-drift}$ is aiming at the node's clock drift after clock synchronization process. We set the maximum clock drift as the clock guard duration. In our platform, the $T_{clk-drift}$ is equal to 8μs. $T_{sch-drift}$ is the maximum timeout error of HRT, and can support a high precision value within 5μs in our platform. Through above analysis, T_{guard} can be expected as $5\mu s\ (T_{re-trans-air}) + 8\mu s\ (T_{clk-drift}) + 5\mu s\ (T_{sch-drift}) = 18\mu s$.

D. Slot Schedule

Schedule Header In each data packet, P-KER-MAC add a schedule header for broadcasting slot reservation information. Table VII show its structure. The h_{type} represent the type of source node, it can be master or slave type. When a node receives a data packet, it can get the schedule header, use the h_{offset} & $h_{timeout}$ to update the slot state in its slot table, and check the packet owner by means of h_{src} & h_{dst}. As for h_{id}, it represents the message type to distinguish data message and management messages (e.g. beacon, clock-sync, etc.).

TABLE VII: SCHEDULE HEADER STRUCTURE

Header	Description
h_{type}	The node type: master or slave.
h_{offset}	The distance between current receiving slot index and next trigger slot index.
$h_{timeout}$	The specific number of receiving slot index's usage.
h_{src}	The source address.
h_{dst}	The destination address.
h_{id}	The message id.
h_{stamp}	The source timestamp.

Transmission Schedule In order to accurately control packet transmission moments, so that nodes transmit only in their reserved slots. P-KER-MAC uses a HRT to arrange the transmission slot event. In Sec. III, we have introduced the method of slot selection. In the following description, we pay close attention to the schedule question that how to arrange a transmission. The transmission schedule process can be showed in Fig. 11.

At time t_0, P-KER-MAC get a data request from the transport layer. First, we get the raw packet from the data queue, then, we search the slot table to get the next arranged slot index (S_{trig}) for calculating the h_{offset} by

$$h_{offset} = S_{trig} - S_{cur} \tag{18}$$

$$S_{cur} = \frac{(t_{current} - t_{slot-sm})\%T_{frame}}{T_{slot}} \quad (19)$$

Second, after adding the schedule header for the packet, at time t_1, we can calculate the exceed portion time of current slot $(T_{cur-exc})$ by

$$T_{cur-exc} = S_{cur}\%T_{slot} \quad (20)$$

Third, we arrange the HRT to register a delay event to send the packet. The delay value is

$$T_{delay} = (S_{trig} - S_{cur}) \times T_{slot} - T_{cur-exc} \quad (21)$$

At time t_2, the HRT raise the elapsed event. At the same time, the packet is put into PHY and transmitted immediately without waiting for the medium to be free.

We use this way to schedule the arranged slot. Because we take into account the cost of the system situation that it takes time for any packet to be generated.

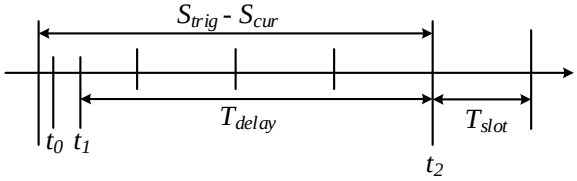

Fig. 11. Single slot schedule.

E. TCP/IP Network Stack Support

The P-KER-MAC has been implemented in the Linux kernel space as a net device driver. There is a huge difference between P-KER-MAC and the other MACs which are designed for CSMA/CA. Hence, we have changed some parameters for the driver.

MTU In computer networking, the maximum transmission unit (MTU) of a communications protocol of a layer is the size (in bytes or octets) of the largest protocol data unit that the layer can pass onwards. The P-KER-MAC has no support for long packets fragmentation, so the default MTU (1500 bytes) is obviously not suitable for P-KER-MAC's fixed data length of per slot. By analyzing the features of P-KER-MAC, we can calculate an appropriate MTU, which is

$$l_{MTU} = l_{slot} - l_{802.11} - l_{schedule} - l_{FCS} \quad (22)$$

where l_{slot} is fixed data length, $l_{802.11}$ is 802.11 header length, $l_{schedule}$ is schedule header length, l_{FCS} is FCS checksum length.

MAC Address A media access control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment. The original IEEE 802 MAC address comes from the original Xerox Ethernet addressing scheme. This 48-bit address space contains potentially 2^{48} possible MAC addresses. In P-KER-MAC's driver, it enable the last 2 octets as MAC-16 (16-bit address), and there is a converter for MAC-16 and MAC-48. There are two reason for choosing the shorter address length, one is for saving data space, the other is P-TDMA-SYS's network capacity would not be great. MAC-16 that contains $2^{16} = 65536$ addresses is enough.

TX Queue The TX queue length is represent the driver's cache, and is equal to Q_{max}. When TX queue is not full, we start the TX queue to receive the packet from the upper layer and deliver it to P-KER-MAC for transmission. Similarly, we stop the TX queue until it has free space.

F. Multi-hop Networks Support

P-KER-MAC is designed for 1-hop network. If two nodes are not within their wireless transmission range, they cannot exchange messages with each other. In order to support the multi-hop networks, we add the aodv-uu-0.9.6 [19] into P-TDMA-SYS. The aodv-uu-0.9.6 snoops all incoming and outgoing packets by utilizing a Netfilter [20] hook to maintain the routing table. Moreover, the AODV's routing table can be mapped into kernel IP routing table. By completing these tasks, each node has the ability to forward IP packets. For a more detailed implementation about AODV, we invite the reader to refer to [3]. In short, P-TMDA-SYS incorporates AODV routing protocol, so it can support the multi-hop networks.

G. TCP Improvement

Traditional TCP is developed for terrestrial wire-line networks, and it can work well over a variety of Internet paths, but there is still a fundamental bottleneck of traditional TCP performance in P-KER-MAC.

With traditional TCP, the network throughput is closely related to the round-trip delay time (*RTT*). However, when the P-KER-MAC initialize transmission, the default report rate (R_{min}) is small, and it lead to a big *RTT* value. Hence, the TCP always cannot keep the pipeline full, because TCP do not make full use of the report rate. In our testbed, we found that the utilization of report rate which in per frame is low by using the traditional TCP.

In order to improve the TCP throughput in our system, we develop P-TCP, that is a new TCP congestion control algorithm to adapt the P-KER-MAC's transmission features.

The TCP congestion-avoidance algorithm is the primary basis for congestion control in the Internet. Slow-start is the default algorithms that TCP uses to control congestion inside the network in the Linux. It is also known as the exponential growth phase. Slow-start begins initially with a congestion window size (*cwnd*) of 1, 2 or 10. The value of the congestion window will be increased with each acknowledgement (ACK) received, effectively doubling the window size each *RTT*. The transmission rate will be increased with slow-start algorithm until either a loss is detected, or the receiver's advertised window (*rwnd*) is the limiting factor, or the slow start threshold (*ssthresh*) is reached. If a loss event occurs, TCP assumes that it is due to network congestion and takes steps to reduce the offered load on the network. These measurements depend on the used TCP congestion avoidance algorithm. Once *ssthresh* is reached, TCP

changes from slow-start algorithm to the linear growth (congestion avoidance) algorithm. At this point, the window is increased by 1 segment for each RTT.

In P-TCP, the network congestion is only determined depending on the R_{max} & $Q_{current}$, rather than the value of *ssthresh* & *RTT*. In the beginning of a slow start, *cwnd* is set to L_{slot}. In each *RTT*, the current *cwnd* will be set as below.

$$cwnd = (R_{max} - Q_{current}) \times L_{slot} \quad (23)$$

The transmission rate will be increased with slow-start algorithm until the $R_{current}$ reach R_{max}, or the data queue is empty, or the receiver's advertised window is the limiting factor. At this point, we reduce the *cwnd* by 1 segment for each RTT until $R_{current}$ is stable. We don't worry that P-TCP sets too large *cwnd* to cause the link conflict due to P-KER-MAC is a TDMA-based MAC to support a none-conflict access control.

V. Testbed Results

In this section, we summarize hundreds of hours of evaluating P-TDMA-SYS on our in-door 10 node testbed (Fig. 12). These test nodes are numbered, from N_0 to N_{10}, and they all in a 1-Hop range. Sec. H has introduced the detail configuration of hardware & system software for each test node. We configure the Linux firewall to build multi-hop environment by using *iptables* [21]. We set up the firewall on the node, make it only receive IP packets from some specific nodes. In all experiments, we set system clock (or system time) [22] as a statistical base of nanosecond clock. We measure the throughput & jitter for UDP and TCP by using *iperf* v2.0.4 tool [23].

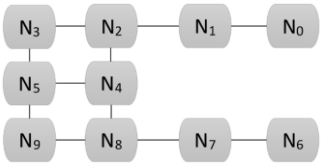

Fig. 12. Test nodes topology.

A. High Resolution Timer Accuracy

High Resolution Timer (HRT) provides a high-precision framework for timer's management. In P-TDMA-SYS, the HRT provide a precise delay to arrange transmissions.

We setup an experiment on the testbed to measure the HRT accuracy. The experiment design was inspired by a stack overflow question [24]. At time $t_{sys-start}$, we arrange different delay ($T_{t-delay}$) to HRT, then we calculate the clock offset ($T_{t-offset}$) between the estimated and actual timeout when the timer expires ($t_{sys-expire}$).

$$T_{t-offset} = (t_{sys-expire} - t_{sys-start}) - T_{t-delay} \quad (24)$$

We repeat this task several times (≥100,000) in our platforms. In the end, we count the average of these clock offsets as the HRT's accuracy.

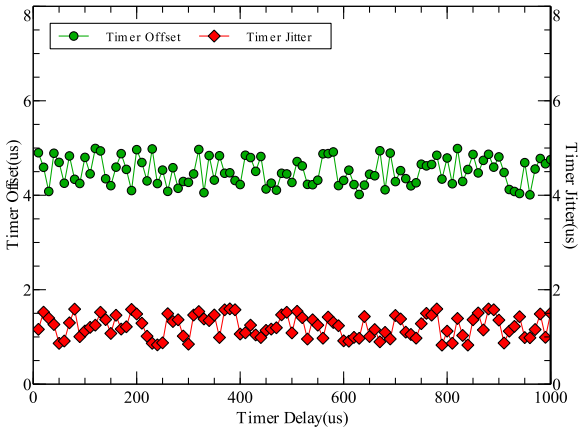

Fig. 13. Timer offset and jitter in different delay.

In Fig. 13, it showed the average offset and jitter of the timer. The result indicates that, with the change of delay, there was no significant change in offset and jitter. The $T_{t-offset}$ can be controlled within 5μs. The jitter of timer offset was always less than 2μs. The offset is referenced to a parameter of T_{guard} in T_{slot}. According to the test results, we set the value of $T_{sch-drift}$ is equal to 5μs in (17).

B. Packet Transmission Evaluation

In order to measure the elapsed time of a packet transmission, we perform series of experiments where a node transmits 10,000 packets at 5ms intervals. We repeat this experiment for different packet length and modulation rates. We use our modifications of the ath9k-htc firmware to record the time between the start time and the hardware interrupt indicating that the card finished sending the packet.

Fig. 14 shows the 802.11 hardware transmission duration (T_{data}) and the sending & receiving process duration (T_{tx-os} & T_{rx-os}) for 540 bytes packets transmitted at the different modulation rate. This experiment corresponds to the fixed slot duration transmissions (T_{trans}). We note that T_{trans} is almost constant with a negligible amount of variability (less than a few microseconds). Disabling the CSMA/CA features of the ath9k-htc firmware also ensures that T_{trans} is almost constant.

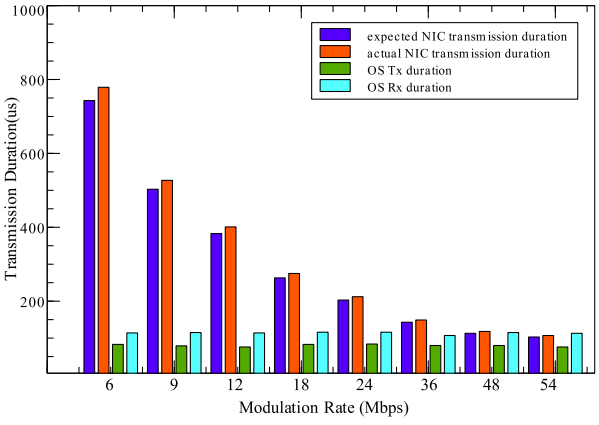

Fig. 14. Transmission duration in different PHY rate.

Table VIII shows the details about the T_{tx-os} & T_{rx-os} in different packet length. Observing the data, we find the T_{tx-os} & T_{rx-os} is relatively stable. Unfortunately, we are unable to establish an accurate relationship between the packet length (l_{packet}) and T_{tx-os} & T_{rx-os}. We *only* found the fluctuation range of T_{tx-os} & T_{rx-os}. In our testbed, $T_{tx-os} = 112 \pm 5\mu s$, $T_{rx-os} = 80 \pm 5\mu s$.

TABLE VIII: PACKET LENGTH WITH SYSTEM OVERHEAD DURATION

l_{packet}(bytes)	T_{tx-os}(μs)	T_{rx-os} (μs)
128	107	77
256	109	78
512	110	80
640	113	82
896	114	83
1024	116	87

Table IX shows more details about the measurement of actual hardware transmission duration ($T_{data-actual}$). Meanwhile, we compare the $T_{data-actual}$ and the expected transmission duration ($T_{data-expect}$). We find that $T_{data-actual}$ is always slightly larger than $T_{data-expect}$, hence, we measure the T_{ofdm} by using

$$T_{ofdm} = \frac{T_{data-actual} - T_{PLCP}}{l_{packet} / b_m} \quad (25)$$

where $T_{data-actual}$ is the average transmission duration of the 540 bytes of a packet, $b_m \in \{3, 4.5, 6, 9, 12, 18, 24, 27\}$ is the number of bytes carried in each OFDM symbol. $m \in \{6, 9, 12, 18, 24, 36, 48, 54\}$ is the modulation rate. In the 802.11g's standard, T_{ofdm} should be equal to 4μs. However, in our testbed, $T_{ofdm} = 4.16 \pm 0.04\mu s$.

Using our measurements over various packet sizes, we can precisely calculate the T_{trans}. In particular, we use the upper limit of the measured parameters for the final T_{trans},

$$T_{trans} = T_{PLCP} + \left\lceil \frac{l + h_{802.11}}{b_m} \right\rceil \times T_{ofdm} + T_{tx-os} + T_{rx-os} \quad (26)$$

In (26), $T_{PLCP} = 23\mu s$, $T_{ofdm} = 4.2\mu s$, $T_{tx-os} = 117\mu s$, $T_{rx-os} = 85\mu s$, $b_m \in \{3, 4.5, 6, 9, 12, 18, 24, 27\}$ is the number of bytes carried in each OFDM symbol. $m \in \{6, 9, 12, 18, 24, 36, 48, 54\}$ is the modulation rate.

TABLE IX: PACKET TRANSMISSION DURATION

Modulation Rate (Mbps)	$T_{data-actual}$ (μs)	$T_{data-expect}$ (μs)	T_{ofdm} (μs)
6	779	743	4.17
9	527	503	4.18
12	401	383	4.20
18	275	263	4.12
24	212	203	4.18
36	149	143	4.17
48	118	113	4.15
54	107	103	4.16

C. Disabling CSMA/CA Evaluation

It is essential to quantify how accurately the packet transmission duration can be controlled with disabling CSMA/CA. We complete this experiment by measuring the T_{trans} in a 2-node topology. Similar to packet transmission experiment, we still measure the packet transmission duration (T_{trans}). In different ways, we carry out this experiment for the two cases of CSMA/CA enabled and disabled.

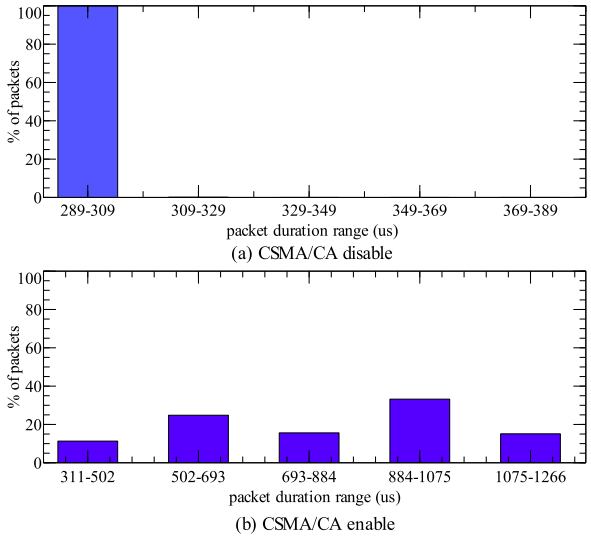

Fig. 15. CSMA/CA disabled vs. CSMA/CA enabled.

Fig. 15 (a) shows the packet transmission duration for the case of CSMA/CA disabled, while Fig. 15 (b) shows the packet transmission duration for the case of CSMA/CA enabled. For the case of CSMA/CA disabled, as can be seen from Fig. 15 (a), 100% packet transmission duration can be stably controlled in $299 \pm 10\mu s$. However, for the case of CSMA/CA enabled, the transmission duration of each packet presents a stochastic state, and its range varies from 299-1258μs.

Thus, above results indicate that with CSMA/CA disabled it is possible to precisely control packet transmission duration over commodity 802.11 hardware.

D. Clock Synchronization Evaluation

We setup an experiment on the testbed to measure the clock synchronization error, between the nodes in the network. To measure the clock synchronization error, we use 10 nodes to set up a cluster network (each node in a 1-hop range), and set one of the nodes to be master node (N_{master}). The system clock of N_{master} is set the global clock (t_{global}). As each slave node is synchronized with the master node, the master node sends packets with an increasing sequence number and timestamp at 200ms intervals. Each slave node receives the packet at the same time and records the sequence number & timestamp by using

$$t_{(seq)} = t_{ori(seq)} - T_{trans} \quad (27)$$

where *seq* is the sequence number, T_{trans} is the transmission duration of the packet, $t_{ori(seq)}$ is the original the sequence number & timestamp which is marked when the packet was received. We run the experiment for an hour and record the average synchronization errors for all nodes.

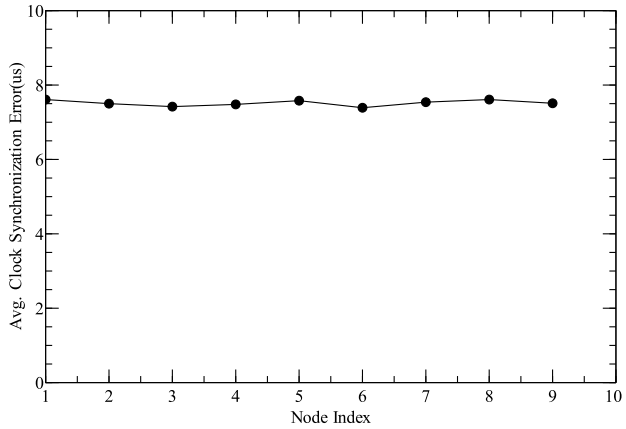

Fig. 16. The avg. clock synchronization error of each node.

In Fig. 16, as can be seen the average synchronization error of each node is basically stable at $7.5 \pm 0.5\mu s$. Table X show more detailed statistics for all nodes. Out of the 90,000 collected errors, the maximum synchronization error of 88μs just occurred only 3 times. We note that almost all of the errors (99%) are within 8μs.

TABLE X: MORE DETAILS OF CLOCK SYNCHRONIZATION

Nodes Count	9 (except for master node)
Avg. Synchronization Error	7.8μs
Max Synchronization Error	88μs
Min Synchronization Error	-7μs
Std. Dev. Synchronization Error	0.5μs

We conclude that the clock synchronization algorithm works well for tight synchronization. With high confidence, it synchronizes all nodes of a cluster network to within 8μs. Meanwhile, we set $T_{clk-drift} = 8\mu s$, where $T_{clk-drift}$ is a part of slot guard duration.

E. TDMA Mechanism Evaluation

Packet Transmission in Fixed Slots In P-TDMA-SYS, the kernel MAC use the fixed slot to transmit packet. Hence, we must prove that each packet transmission can be tucked into the slot correctly.

After the slave node have been synchronous, any node can measure each packet transmission duration in each slot. When a packet was received by a node, it would be marked the receiving timestamp ($t_{current}$). And then, we can use (20) to calculate the $T_{cur-exc}$. It represents the actual transmission time in the receiving slot. If the packet transmission is tucked into the slot correctly, the $T_{cur-exc}$ must satisfy the following relationship,

$$T_{trans} \le T_{cur-exc} \le T_{slot} \tag{28}$$

We setup 2 nodes for this experiment, and they send the maximum length packets ($L_{slot} = 540$) with each other by using the $R_{max} = 475$. The whole process lasts 30 minutes, and in the final we count the numbers of correct transmission which was fit for (28).

TABLE XI: TRANSMISSION DURATION IN SLOT DURATION STATISTICS

Packet Num.	Correct Num.	Error Num.	Avg. Trans duration
40,000	39,998	2	320μs

Table XI show the results of this experiment. As we can see, 99.9% of the transmission can satisfy the (28)(26), and the average transmission duration accounts for $320/328 = 97.5\%$ of slot duration. We conclude that each packet transmission can be tucked into the slot correctly.

Packet Transmission by Slot Schedule On the basis of previous experiment, we also can test packet transmission scheduling process. Likewise, when a packet is received by a node, the h_{offset} which is in the common header of the packet notices the next reserved slot index that would be used. In Sec. H, the h_{offset} has been described in detail. Now, we record the all used and reserved slots into a table for a node. The record format is as follows:<*Node_Id*, S_{cur}, h_{offset}>.

If the slot scheduling is correct, there must be a following relationship for a fixed *Node_Id*,

$$S_{cur}(r) = S_{cur}(r-1) + h_{offset}(r-1) \tag{29}$$

where r is the row number of the table.

We setup 5 nodes for this experiment, they send packets with each other by using the $R_{max} = 475$. The whole process lasts 30 minutes, and then we gather the record tables from each node.

TABLE XII: SLOT SCHEDULING STATISTICS

Node	Records Num.	Correct Num.	Correct Rate(%)
0	1,691,323	1,681,533	99.42
1	1,692,211	1,624,712	96.01
2	1,694,154	1,660,423	98.01
3	1,694,310	1,643,480	96.99
4	1,689,131	1,628,322	96.39

According to the Table XII, we found the 97.42% process could meet the (29). However, some packet transmission process cannot meet the (29). There are two main reasons, one is the decoding failure, the other is the slot collision.

Slot Collision Each slot can only belong to one node in a moment. If a slot is used or reserved more than 2 nodes, we believe it is a slot collision event. We setup 10 nodes for this experiment, and count the numbers of packet transmission (c_{trans}) and slot collision event (c_{coll}) for all nodes.

TABLE XIII: SLOT COLLISION STATISTICS

Nodes Count	c_{trans}	c_{coll}	c_{coll} / c_{trans}(%)
2	10,000	201	2.01
4	20,000	516	2.58
6	30,000	1442	4.80
8	40,000	2083	5.21
10	50,000	2778	5.55

As the number of nodes increases in in the network, the slot collision frequency (c_{coll} / c_{trans}) also increases. However, the (c_{coll} / c_{trans}) is low. Hence, the vast majority of transmission can be guaranteed to perform correctly.

Through the above three experiments, in conclusion, we believe the P-MAC-KER implements the TDMA mechanism.

F. 1-Hop Transmission Performance

1-Hop UDP Performance In order to test each node can reach the design bandwidth (1000 Kbps). We setup 10 nodes in 1-hop range and make them send data to each other in pairs with continuous backlogged UDP traffic. We run the experiment for 1 minutes and repeat it 30 times.

TABLE XIV: PAIRS TRANSMISSION ON HEAVY UDP TRAFFIC

Pair	Avg. Rate (→, Kbps)	Avg. Rate (←, Kbps)
N_0-N_1	1011	1008
N_2-N_3	1011	998
N_4-N_5	1001	1012
N_6-N_7	999	1002
N_8-N_9	1006	997

In Table XIV, "→" & "←" represent the data transmission direction. By analyzing the test results, we found the bandwidth of each link is stable and reach the design bandwidth. Furthermore, we found each node used the maximum report rate (R_{max}) in the heavy traffic.

1-Hop TCP Performance The purpose for testing 1-hop TCP performance is to evaluate the P-TCP which is the specialized TCP congestion control algorithm for P-KER-MAC. In this experiment, we only setup 2 nodes to test the link rate. And, we put P-TCP and other classic TCP congestion control algorithm together for comparison under the same external condition. In each algorithm test, we run the link rate test for 1 minutes and this procedure is repeated 30 times to get the average of rates.

In Table XV, we found the most classic TCP congestion control algorithm cannot enhance the potential for P-KER-MAC, because these algorithms rely mainly on the RTT to estimate the bandwidth, however, the P-KER-MAC's initial report rate is low. These algorithms mistakenly believe that a very narrow bandwidth in the MAC. Hence, they use the short congestion window to send packet and cannot fully occupy the bandwidth. However, in P-TCP, it can know exactly available bandwidth and increase the congestion window until the report rate reach the maximum value. From Table XV, the increase in TCP throughput is lesser than that for UDP throughput. This is since for TCP we transmit both the TCP data packet as well as the TCP ACK at the same time.

TABLE XV: COMPARISON WITH OTHER TCP ALGORITHM

Algorithm	Avg. Rate (Kbps)	Avg. Report Rate
High Speed	131.2	36
Westwood	96.7	42
Cubic	153.6	58
Vegas	142.3	44
Reno	122.3	47
Scalable	46.7	30
P-TCP	676.7	420

G. P-TDMA-SYS vs. CSMA/CA

In order to compare the performance between P-TDMA-SYS and CSMA/CA, we setup fair scenarios to run the test cases.

Experiment Scenarios Design For CSMA/CA, we set the wireless card into 802.11's " ad-hoc model" [25]. The 802.11's ad-hoc model was implemented in Linux kernel under mac80211 [26], so we can set the wireless card model directly. For P-TDMA-SYS, we use the same settings that have been mentioned above. We use UDP & TCP to simulate the real-time application, and put the throughput, jitter and packet loss as the main evaluation criteria. In particular, we set P-TCP as P-TDMA-SYS's TCP congestion control algorithm, and set the Cubic algorithm [27] for CSMA/CA. Meanwhile, we add the AODV module into our OS as routing for both of them.

1-Hop CBR Performance We setup 10 nodes in 1-hop range and make them send data to each other in pairs with constant bit rate (CBR) UDP & TCP flows of 1000Kbps & 100Kbps source rate. We run the experiment for 1 min and repeat it for 30 times. The CBR UDP & TCP flows of 1000Kbps can simulate the real-time video transmission, and the CBR UDP & TCP flows of 100Kbps can simulate the real-time audio transmission.

Table XVI shows the transmission details of all pair nodes. Observing the data, we found that P-TDMA-SYS could provide a more stable transmission environment, but CSMA/CA not.

TABLE XVI: P-TDMA-SYS VS. CSMA/CA IN 1-HOP SCENARIO

		Avg. UDP				Avg. TCP			
	Pair	Rate (←,Kbps)	Rate (→,Kbps)	Jitter (←,ms)	Jitter (→,ms)	Rate (←,Kbps)	Rate (→,Kbps)	Jitter (←,ms)	Jitter (→,ms)
P-TDMA-SYS	N0-N1	977	987	4.4	3.8	651	640	6.4	7.2
	N2-N3	988	975	4.1	3.6	581	633	6.8	6.6
	N4-N5	984	971	3.7	3.2	642	627	6.7	6.8
	N6-N7	942	956	4.6	4.1	631	642	6.9	7.7
	N8-N9	956	985	4.4	3.2	661	637	6.1	7.0
CSMA/CA	N0-N1	998	960	4.1	4.2	851	922	4.3	3.8
	N2-N3	981	968	21.2	14.3	742	688	7.2	6.4
	N4-N5	991	720	15.6	44.1	238	417	51.3	32.4
	N6-N7	936	514	36.8	32.8	172	642	22.3	51.2
	N8-N9	712	654	51.2	61.2	238	172	88.4	67.2

TABLE XVII: P-TDMA-SYS VS. CSMA/CA IN TRANSMISSION JITTER

	UDP （Std. Dev.）		TCP （Std. Dev.）	
	Jitter (←,ms)	Jitter (→,ms)	Jitter (←,ms)	Jitter (→,ms)
P-TDMA-SYS	0.31	0.35	0.29	0.37
CSMA/CA	16.51	20.41	31.60	24.73

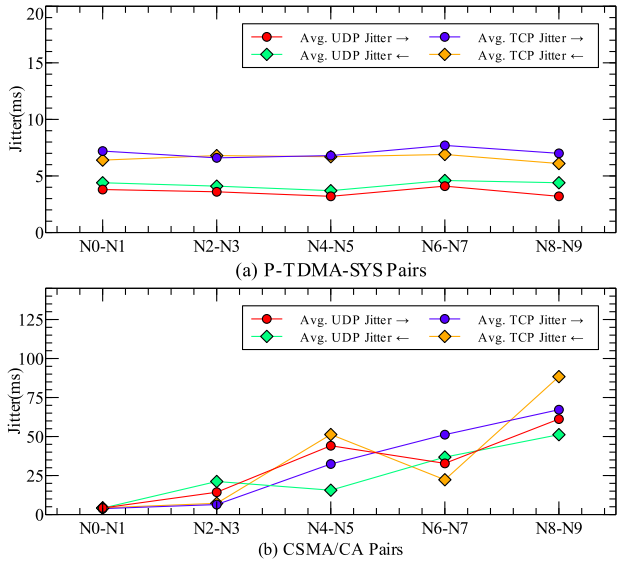

Fig. 17. P-TDMA-SYS vs. CSMA/CA in avg. transmission jitter.

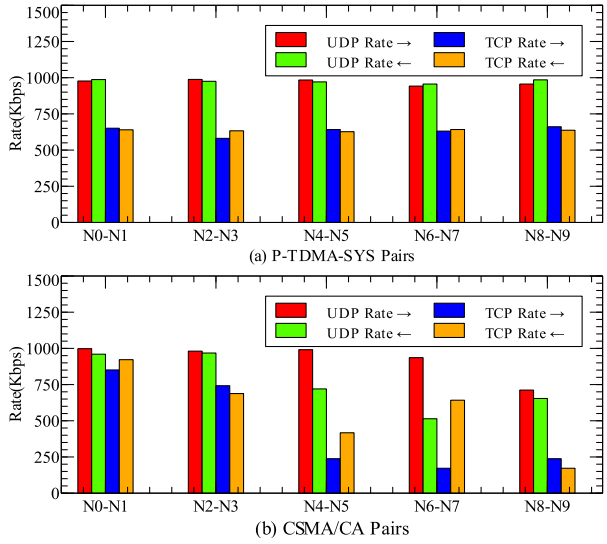

Fig. 18. P-TDMA-SYS vs. CSMA/CA in avg. transmission rate.

Fig. 17 shows the average of UDP & TCP jitter of all nodes with P-TDMA-SYS & CSMA/CA. We found that P-TDMA-SYS performed smoothly in the bidirectional transmission, but CSMA/CA performs unsteadily. Furthermore, in Table XVII, we calculate the standard deviation of transmission jitter of P-TDMA-SYS & CSMA/CA. The P-TDMA-SYS's standard deviation of jitter is much lower than CSMA/CA's standard deviation of jitter.

In Fig. 18, we find that P-TDMA-SYS can provide all nodes a relatively stable bandwidth in the bidirectional transmission. However, the CSMA/CA only can provide some of the nodes with abundant bandwidth (e.g. N_0-N_1, N_2-N_3), but the bandwidth of some nodes is very low (e.g. N_8-N_9). Especially, the N_8-N_9 pair always performed pool, because they were always the last to enter the rate test, and we believed that the congested channel causing them to low rate.

According to the test results, we conclude that P-TDMA-SYS can support better than CSMA/CA for real-time application.

Multi-Hop CBR Performance We also evaluate P-TDMA-SYS & CSMA/CA multi-hop performance in our testbed. In this experiment, we construct a linear multi-hop environment ($N_0 \rightarrow N_1 \rightarrow \ldots \rightarrow N_9$). Fig. 19 show the topology of nodes in this experiment. And we set the transmission rate of N_0 which is the CBR UDP & TCP flows of 1000Kbps.

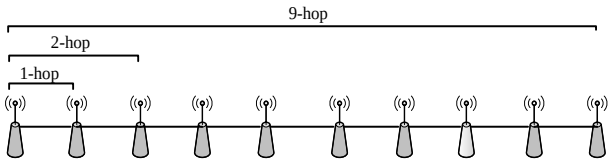

Fig. 19. The multi-hop topology for experiment.

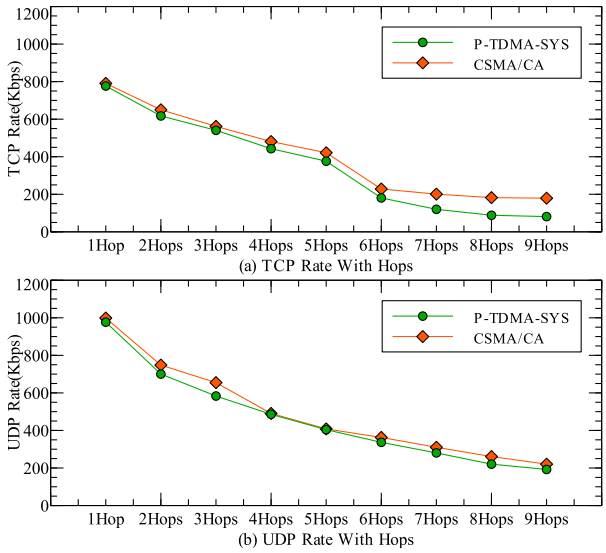

Fig. 20. P-TDMA-SYS vs. CSMA/CA in multi-hop avg. transmission rate.

TABLE XVIII: P-TDMA-SYS VS. CSMA/CA IN PACKET LOSS

Hops	P-TDMA-SYS's packet loss rate (%)	CSMA/CA's packet loss rate (%)
1	2.8	0.1
2	3.7	0.8
3	3.9	1.8
4	4.4	2.4
5	4.4	3.2
6	3.8	2.8
7	3.7	2.3
8	4.4	2.1
9	5.4	2.6

In Fig. 20, we found that P-TDMA-SYS & CSMA/CA both could not guarantee rate in the multi-hop transmission. With the increase of hop, transmission rate is reduced. The main reason is the loss of data forwarding in each hop. However, in Table XVIII, for multi-hop transmission, the test results showed that P-TDMA-

SYS's packet loss rate was higher than CSMA/CA, because P-KER-MAC is unable to guarantee reliable transmission, if P-KER-MAC loss a packet, it would directly feedback to application layer. By contrast, the CSMA/CA try to ensure reliable transmission, if CSMA/CA loss a packet, it would retransmit the packet. Hence, in the application, the P-TDMA-SYS's packet loss rate was higher than CSMA/CA. Even in the 1-hop transmission, there still was a lost packet phenomenon in P-TDMA-SYS transmission.

Due to the limitation of experimental conditions, we cannot guarantee that all wireless card have a "clean" channel. All of the wireless card worked at 2.4GHz with other wireless devices, so a certain amount of interference is inevitable.

Based on P-TDMA-SYS and CSMA/CSMA comparative experiments, P-TDMA-SYS can provide lower delay jitter and rated bandwidth. Hence, we conclude that P-TDMA-SYS can provide a more reliable transmission than CSMA/CA in 1-hop CBR transmission. However, in the multi-hop transmission, because P-KER-MAC do not have a retransmission mechanism, its performance is slightly inferior to the CSMA/CA in packet loss control.

VI. Conclusions

In this paper, we design and implement P-TDMA-SYS over commodity 802.11 hardware for MANETs. P-TDMA-SYS is an integrated TDMA network system based Linux kernel TCP/IP network stack and it can support real-time applications in mobile multi-hop ad-hoc networks.

In P-TDMA-SYS, the kernel TDMA-based MAC is P-KER-MAC. It contains a tight microsecond clock synchronization to enable high TDMA efficiency. What's more, P-KER-MAC inherits the S-TDMA MAC and provides an adaptive report rate algorithm to support real-time applications. On the basis of P-KER-MAC, we present a modified TCP congestion control algorithm to fully utilize the bandwidth resources.

In the testbed, we put forward a complete and feasible solution of disabling the wireless card's CSMA/CA functions. By measuring the delay of hardware and software, we summarize the transmission function of wireless cards. Furthermore, we evaluate the P-KER-MAC's TDMA mechanism in detail to prove its high transmission efficiency. For evaluating the P-TDMA-SYS, we compared the performance between P-TDMA-SYS and CSMA/CA in 1-hop/multi-hop scenarios. In the final, we conclude that P-TDMA-SYS indeed can provide a reliable throughput and low delay/jitter transmission for real-time applications in MANETs.

References

[1] S. Xu and T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks," *Computer Networks*, vol. 38, pp. 531-548, 2002.

[2] Y. R. Kondareddy and P. Agrawal, "Synchronized MAC protocol for multi-hop cognitive radio networks," in *Proc. IEEE ICC*, 2008, pp. 3198-3202.

[3] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," No. RFC 3561. 2070-1721, 2003.

[4] E. Jung and N. H. Vaidya, "A power control MAC protocol for ad hoc networks," in *Proc. 8th ACM International Conference on Mobile Computing and Networking*, 2002, pp. 36-47.

[5] S. Wu, C. Lin, Y. Tseng, and J. Sheu, "A new multi-channel MAC protocol with on-demand channel assignment for multi-hop mobile ad hoc networks," in *Proc. I-SPAN*, 2000, pp. 232-237.

[6] R. I. M. Series, "Technical characteristics for an automatic identification system using time division multiple access in the VHF maritime mobile frequency band," I. T. Union, Ed., 2014.

[7] V. Gabale, B. Raman, K. Chebrolu, and P. Kulkarni, "Lit mac: Addressing the challenges of effective voice communication in a low cost, low power wireless mesh network," in *Proc. 1st ACM Symposium on Computing for Development*, 2010, p. 5.

[8] P. Djukic and P. Mohapatra, "Soft-TDMAC: A software TDMA-based MAC over commodity 802.11 hardware," in *Proc. IEEE INFOCOM*, 2009, pp. 1836-1844.

[9] A. Sharma and E. M. Belding, "FreeMAC: Framework for multi-channel mac development on 802.11 hardware," in *Proc. ACM Workshop on Programmable Routers for Extensible Services of Tomorrow*, 2008, pp. 69-74.

[10] R. K. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. A. Brewer, "WiLDNet: Design and implementation of high performance WiFi based long distance networks," in *Proc. NSDI*, 2007, p. 1.

[11] S. Leffler. (Sep 2015). Madwifi-project.org-Trac. [Online]. Available: http://madwifi-project.org/

[12] P. Persistent, "Environmental medicine, Part 4: Pesticides--biologically persistent and ubiquitous toxins," *Alternative Medicine Review*, vol. 5, pp. 432-447, 2000.

[13] D. Mills, "Simple network time protocol (SNTP) version 4 for IPv4," *Ipv6 and OSI, RFC2030*, 1996.

[14] S. Jung, K. Kini, and H. Kang, "A bit-rate/bandwidth scalable speech coder based on ITU-T G. 723.1 standard," in *Proc. IEEE ICASSP*, 2004, pp. I-285-8.

[15] I. Draft, "recommendation and final draft international standard of joint video specification (ITU-T Rec. H. 264| ISO/IEC 14496-10 AVC)," *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050*, vol. 33, 2003.

[16] M. Vanhoef and F. Piessens, "Advanced Wi-Fi attacks using commodity hardware," in *Proc. 30th Annu. Conf. Computer Security Applications*, 2014, pp. 256-265.

[17] H. Wu, S. Cheng, Y. Peng, K. Long, and J. Ma, "IEEE 802.11 distributed coordination function (DCF): Analysis and enhancement," in *Proc. IEEE ICC*, 2002, pp. 605-609.

[18] S. Mangold, S. Choi, G. R. Hiertz, O. Klein, and B. Walke, "Analysis of IEEE 802.11 e for QoS support in wireless LANs," *IEEE Wireless Communications*, vol. 10, pp. 40-50, 2003.

[19] AODV-UU Linux Implementation. (Mar. 2016). [Online]. Available: https://sourceforge.net/projects/aodvuu/

[20] Netfilter Architecture. (Oct. 2015). [Online]. Available: http://www. netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO-3.html

[21] Iptables Description and Target. (Oct. 2015). [Online]. Available: http://ipset.netfilter.org/iptables.man.html

[22] Computer System Time. (Dec. 2015). [Online]. Available: https:// en.wikipedia.org/wiki/System_time
[23] iPerf. (Nov. 2015). [Online]. Available: https://iperf.fr/
[24] Stack Overflow: Reliability of Linux kernel add_timer at resolution of one jiffy? (Dec. 2015). [Online]. Available: http://stackoverflow.com/questions/16920238/reliability-of-linux-kernel-add-timer-at-resolution-of-one-jiffy/17055867#17055867
[25] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "IEEE 802.11 ad hoc networks: performance measurements," in *Proc. 23rd International Conference on Distributed Computing Systems Workshops*, 2003, pp. 758-763.
[26] Linux Wireless: Mac80211. (Dec. 2015). [Online]. Available: https:// wireless.wiki.kernel.org/en/developers/documentation/mac80211
[27] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, pp. 64-74, 2008.

Zechen Lin was born in Fujian Province, China, on Nov. 14, 1990. He received his B.E degree in Communication Engineering from Hefei University of Technology, Anhui province, China, in 2009 and 2013. He is currently working toward Master's degree at Hefei University of Technology. His major interests are mobile networks, wireless communications and multimedia applications.

Zhizhong Ding received his B.E degree in Radio Communications from Nanjing University of Aeronautics and Astronautics, Nanjing, China, Master's degree in Circuit and System from Hefei University of Technology, Hefei, China, and Ph.D. in Information and Communication Engineering from University of Science and Technology of China. He currently is a Professor with the Department of Communication Engineering and with the Institute of Communications and Information Systems, Hefei University of Technology. His research interests include wireless communications, network communications and information theory.

Qingxin Hu received his B.E degree and Master's degree in Hefei University of Technology, Anhui province, China. He currently is a Professor with the Department of Communication Engineering and with the Institute of Communications and Information Systems, Hefei University of Technology. His research interests include wireless communications, wireless signal processing and railway information security.

Shuai Tao was born in Shanxi Province, China, in 1989. He received his B.E degree in Communication Engineering from Hefei University of Technology, Anhui province, China. He is currently working toward Master's degree at Hefei University of Technology. His research interests include wireless communications and network management.