Joint Decentralized Slot Allocation and Time Synchronization in a TDMA based MAC

Muhammad Hafeez Chaudhary, and Bart Scheers
Royal Military Academy Belgium
{mh.chaudhary, bart.scheers}@rma.ac.be

***Abstract*—Application of wireless networking technology in tactical and mission-critical environments requires wireless connectivity meets certain quality of service requirements; for example, in such environments timely and reliable delivery of data plays a critical rule in the mission success. For such applications TDMA based MAC scheduling is considered an attractive option. However, to have a functional TDMA based MAC for ad hoc multi-hop networks, we need to have efficient and scalable mechanisms for time synchronization and conflict-free slot allocation among nodes. In this paper, we outline progressive decentralized mechanism by which each node can do time synchronization and slot assignment. Any new node wishing to join the network first monitors ongoing transmission activity and ascertains its pattern. Knowing the pattern of the existing transmissions, the new node can synchronize its clock and assign slots. The proposed slot assignment and time synchronization scheme is implemented on a USRP test-bench. We show that we could achieve time-synchronization accuracy on the order of a sampling period. Besides, we can maintain accuracy at this level in a perpetual way without any explicit exchange of synchronization messages.**

I. Introduction

Recently, the phenomenal increase in the wirelessly connected devices and the applications running on them have put an extremely high premium on the communications spectrum, and thus placing great demand on designing spectrum efficient communication and networking protocols to meet the requirements of the current and emerging applications in wireless networking. The conventional wireless networks require infrastructure support connecting the wireless devices with an access point or a control center. However, in many scenarios such a support infrastructure may be non-existent, scarce, unsuitable, or even not desired. Examples in this area are: disaster zones where large-scale emergencies have brought infrastructure-based networks down, and military tactical operations which often take place in areas which are bereft of any support infrastructure. For such environments, wireless ad hoc networking technology holds promise for potential solution as it would allow disparate set of devices to create a network on demand as the need arises.

In a wireless network, simultaneous transmissions of two or more nodes, at the same time and in the same channel, may not be successful if their intended receivers are in the radio interference range of more than one transmissions. So a mechanism to control access to the shared wireless channel is crucial to ensure efficient channel utilization and to meet quality of service (QoS) requirements. The application of ad hoc networking technology in tactical and mission-critical environments requires that the wireless connectivity be provided which meets certain QoS; for example, in such environments timely and reliable delivery of data plays a critical rule in the success of the mission.

MAC schemes can be broadly classified into two categories: *random access* and *deterministic access*. In the random access, a node access to the shared wireless medium is controlled by some random function. The archetypes of the random access scheme are ALOHA and CSMA [1], from which several collision avoidance schemes are derived of which IEEE 802.11 standard [2] for wireless LAN is the most popular example to date. The main drawback associated with the random access schemes is lack of determinism in performance guarantees (e.g., on the delay, reliability, and throughput) which renders these schemes unsuitable for applications in which deterministic performance is required, for example in real time applications. Fixed medium access scheduling schemes such as TDMA, on the other hand, can provide deterministic network performance. Due to the performance determinism, distributed TDMA based MAC scheduling is considered to be an attractive solution in ad hoc networks for mission critical applications [3], [4].

In a TDMA based solution, for successful transmissions, the nodes need to agree on the time-slot boundaries and the ownership of the slots. Over the years, a number of time synchronization and slot assignment schemes appeared in the literature. Recent examples of slot assignment protocols can be found in [4]–[9]. Moreover, some recent examples of time synchronization schemes are outlined in [10]–[12] and references therein.

A majority of the existing slot assignment schemes assumes that the nodes have already achieved time synchronization and have agreed on the slot boundaries. Besides, for slot assignment the problem is broken into two sub-problems: first nodes find conflict free slot assignment minimizing the number of slots used, subsequently nodes define frame lengths in which to use the assigned slots. In this paper first we show that this disjoint and sequential approach, although ensures conflict free slot assignment, is suboptimal from the slot reuse point of view. Subsequently, we outline a TDMA based MAC scheme by which time synchronization and slot allocation are done simultaneously at each node. Besides, for slot allocation, we outline a scheme in which we aim to maximize the slot reuse factor. In maximizing the slot reuse factor, each node decides on its slot assignment and frame length jointly. The

proposed TDMA based MAC protocol works in a progressive decentralized way and is executed at each node independently. In the proposed MAC protocol, each node utilizes information from its one-hop neighbors only.

In this work we investigate the feasibility of implementing the proposed TDMA based solution on the USRP (Universal Software Radio Peripheral) based software defined radio platform. To this end, we implemented the components of the MAC and PHY layers necessary to enable sharing of a common wireless channel among multiple nodes. The application layer is designed to transmit alphanumeric messages. In the implementation, nodes can find conflict-free slot assignment. Regarding time-synchronization, we can achieve accuracy on the order of a sampling period, which in our implementation is set at one micro-second. Interestingly we can maintain accuracy at this level in a perpetual way without any explicit exchange of synchronization messages. Besides, we show that the accumulated delay between two USRP nodes remains at about 23 micro-seconds and shows little variability around this value.

The remainder of the paper is organized as follows. Section II presents preliminaries. Time synchronization is subject of Section III. Section IV deals with the slot assignment mechanism. Section V outlines details of the implementation setup. Section VI presents the results and finally Section VII gives some concluding remarks.

II. Preliminaries

We consider a network in which network connectivity is given by an un-directed graph $G(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ and $\mathcal{E}$ be the set of nodes and the set of edges among the nodes, respectively. The cardinality of $\mathcal{N}$ (i.e., $|\mathcal{N}|$) denotes the number of nodes in the network and let it be N. An edge exists between nodes i and j if and only if they are reachable from each other, i.e., $(i,j) \in \mathcal{E}$ and $(j,i) \in \mathcal{E}$. The set of one-hop neighbors of a node i is defined as $\mathcal{O}(i) = \{j : (i,j) \in \mathcal{E}, \forall j \in \mathcal{N}\}$ and the set of two-hop neighbors of the node i is defined as $\mathcal{T}(i) = \left\{\bigcup_{j \in \mathcal{O}(i)} \mathcal{O}(j) \setminus \{i\} \cup \mathcal{O}(i)\right\}$. The set of all nodes in the contention area of the node i including itself is given by $\mathcal{C}(i) = \{i\} \cup \mathcal{O}(i) \cup \mathcal{T}(i)$.

III. Time Synchronization

In the TDMA scheme, time is slotted in fixed duration epochs, called slots. The nodes need to agree on and respect the slot boundaries in transmitting and receiving data. This is where time synchronization comes into play. For time synchronization, we consider two approaches: master/slave based and decentralized which are discussed in Section III-A and Section III-B, respectively.

A. Master/Slave Based Time Synchronization

The first approach operates on the master/slave principle and use the network time protocol (NTP) type message exchange [13]. Under this scheme each node (called slave node) has its designated master node. The slave node sends out time-stamped synchronization request packet. After receiving this packet, the master node sends out time-stamped reply message. In the reply, the master node also appends the received time-stamp of the request packet, the time of its reception, and the time when the reply is sent out. The slave node on receiving the reply, can estimate the offset of its clock with respect to the master node clock. Besides, the slave node can also estimate the aggregate delay to the master node. Let us assume δ and d be the clock offset and the delay, respectively. As shown in Fig. 1, we can figure out the following relationship between the time-stamps:

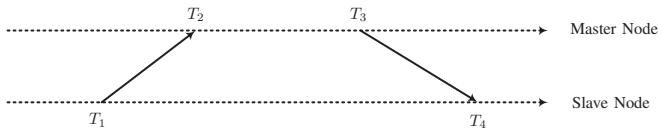


Fig. 1: Message exchange sequence between the master and slave nodes for time synchronization.

$$T_2 = T_1 + \delta + d, \tag{1}$$

$$T_4 = T_3 - \delta + d. \tag{2}$$

Given this information, it is straightforward to get

$$\delta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2}, \tag{3}$$

$$d = \frac{(T_2 - T_1) + (T_4 - T_3)}{2}. \tag{4}$$

Knowing the clock offset and the delay, the slave node adjusts its time to synchronize with the master node time.

The aggregate delay, its magnitude and variability, depends on the level at which the time-stamping of the synchronization request and reply messages is done. For instance, if the synchronization frames are time stamped at the MAC layer, then the delay would include the time it takes for the packet to leave the MAC layer from the sender all the way to the MAC layer of the receiver. The only problem in this approach is that the delay introduced by the processing chain from the MAC layer to the RF front is often non-deterministic. This variability of delay reflects in the high synchronization error. To minimize the synchronization error, ideally, the time stamping of packets shall be done as close to the PHY layer (i.e., RF front-end) as possible. In our implementation. we time stamp the packets at the PHY layer. On the transmit side, the time-stamp specifies the time when first sample of the packet leaves the digital up converter stage of the USRP. While on the receiver side, the time stamp specifies the time when the first sample of the frame leaves the digital down converter stage. With this approach, as shall be shown in the results section, we achieve the accumulated delay on the order of 23 micro seconds and the delay shows very limited variability around this value.

B. Decentralized Time Synchronization

In the preceding scheme, for time synchronization, explicit messages are exchanged between the nodes. Such an explicit

exchange of messages are essential if we would like to measure the delay between the two nodes. Once we know the delay we may choose to compensate for it during the transmission. However, if we do not compensate for the delay then it would be part of the synchronization error. In our implementation, we observe a delay that remains very much constant. So in the second setup, the nodes compensate for this delay. Under this setup, to decide on the slot boundaries and subsequently constrain the transmissions/receptions within the slot boundaries, the new node which joins the network uses the boundary of the ongoing transmissions as a reference and synchronizes with it. In this setup, the node simply listens the channel and once detects a valid transmission, it can estimate the start time of the slot[1] and with that it can synchronize with the ongoing TDMA transmissions. This is in contrast with the master/slave based approach in which for synchronization explicit time stamped messages are exchanged.

The procedure for time synchronization is executed independently at each node and it proceeds as follows. The measured offset from the slot boundary by a node at time instance t can be expressed as follows:

$$y_t = x_t + n_t, \quad t = 1, 2, \ldots \tag{5}$$

where x_t is the actual offset and is a parameter of interest, and n_t is the measurement noise. The n_t is assumed to be zero-mean Gaussian distributed (with variance σ_t^2) independent of x_t. To estimate x_t we use Kalman filter. Assuming a slowly varying parameter, under the Kalman filter model, the state of a system at time t evolves from the previous state at time $t-1$ as follows:

$$x_t = x_{t-1} + v_t, \tag{6}$$

where v_t is the additive noise called process noise. The noise is assumed zero-mean Gaussian distributed with variance α_t^2.

The standard Kalman filter comprises two stages: prediction update and measurement update. The prediction update produces an estimate of x_t given an estimate at $t-1$ using the model in (6). Then the measurement update combines this predicted value with the measurement model in (5) and produces final estimate of x_t. Putting everything together we can write

$$\hat{x}_{t|t} = \hat{x}_{t-1|t-1} + \beta_t^2(\beta_t^2 + \sigma_t^2)^{-1}(y_t - \hat{x}_{t-1|t-1}), \tag{7}$$

$$\gamma_{t|t}^2 = \frac{\beta_t^2 \sigma_t^2}{\beta_t^2 + \sigma_t^2}, \tag{8}$$

where $\hat{x}_{t|t}$ is estimate of x_t given observation and predicted value at time t, $\gamma_{t|t}^2$ is the mean square estimation error, and $\beta_t^2 = \gamma_{t-1|t-1}^2 + \alpha_t^2$.

During each reception, each node estimates its offset from the slot boundary as described above. The node compensate for any timing error in the subsequent slot and transmits/receives accordingly. By this way, the nodes can remain synchronized. Before applying the correction, the node may do a sanity check. If the timing offset is larger than a given threshold (a function of the slot duration and the preamble length), the node ignores the error and assumes that the transmitter of the given transmission has violated the slot boundary. Else the node applies the correction. By doing so the node can remain synchronized in a perpetual way as long as the timing offset remains within bounds and the node receives transmissions. Such kind of synchronization algorithm can be termed as *perpetual time synchronization* and can be applied to any packet-based transmission scheme operating on the time-slotted principle. With this approach, as shall be seen in the results section, we can achieve synchronization error on the order of sampling rate of the signal.

In the master/slave based approach, the slave node synchronize its clock with the clock of the master node—the *slave synchronized to master*. While in the latter approach, a node synchronizes with the ongoing transmissions—the *node synchronized to transmissions*.

[1]Details on how to find the slot boundary are given in Appendix A.

IV. Slot Assignment Mechanism

How the nodes are going to assign slots? In the master/slave setup, the solution is straightforward. When the master node serves the synchronization request message, in the reply the master node also appends the information on the TDMA frame-size and the slots in which the node can transmit. Upon receiving these information, the node knows everything to start transmission in the allocated slots, and receive packets from other nodes.

This master-slave based approach, although simple, does not suit for ad hoc networks. With this in focus, we implemented a second scheme called progressive decentralized MAC (PD-MAC) which we proposed in [14]. According to the PD-MAC protocol, each node allocates a time slot and selects its frame length within which to use the slot. The slot allocation and the frame size selection works in a joint way at each node. The objective of the considered MAC scheduling protocol is to maximize the reuse of the slots with the constraint that no other node in $\mathcal{C}(i)$, for all $i \in \mathcal{N}$, assigns the same slot as node i.

We wish to maximize the communication channel utilization among the nodes. In most of the existing studies, the TDMA based MAC design problem is broken into the following two subproblems:

1) Minimize the number of slots used in the network, and
2) Select frame lengths in which to use the assigned slot to each node.

The two problems are sequentially solved. Where, often the former problem is casted as a graph coloring problem. Given the graph $G(\mathcal{N}, \mathcal{E})$, coloring of nodes can be viewed as a mapping $f : \mathcal{N} \mapsto \mathcal{S}$, where $\mathcal{S}$ is the set of colors (which corresponds to slots). The colors are usually represented by a small set of positive integers, i.e., $\mathcal{S} \subseteq \mathbb{N}_+$, where $\mathbb{N}_+$ denotes the set of all positive integers. In this setting, the slot allocation problem is equivalent to the following problem:

$$\begin{aligned} &\text{minimize } |\mathcal{S}|, \\ &\text{subject to } s_i \neq s_j, \quad s_i, s_j \in \mathcal{S},\ \forall j \in \mathcal{C}(i),\ \forall i \in \mathcal{N}. \end{aligned} \tag{9}$$

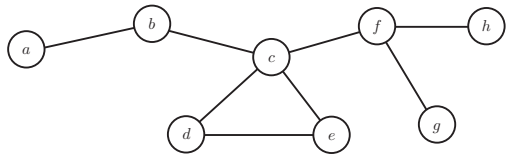


Fig. 2: Example network.

However, finding an optimal solution of this problem is NP-hard [15], [16]. To this end, in the literature, heuristic-based suboptimal solutions are proposed, for instance, in [17], three *greedy* heuristic-based centralized slot assignment procedures are proposed, namely: RAND, MNF, and PMNF. For ad hoc networks, distributed or decentralized schemes are sought because such networks are devoid of coordinating and controlling infrastructure and the global topology knowledge is hard to come by at individual nodes. For ad hoc networks, distributed versions of RAND (named DRAND) and MNF (named HUDSAP) are proposed in [8] and [5], respectively.

Once all nodes in $\mathcal{C}(i)$ have assigned a slot, the node i selects frame length L_i in which to use its assigned slot as a function of $s_{\max}^{(i)} = \max\{s_j : \forall j \in \mathcal{C}(i)\}$, the *maximum slot number* used within the contention area. Concretely, $L_i = 2^{a_i}$, where a_i is an integer number that satisfies the following relation

$$2^{a_i-1} \leq s_{\max}^{(i)} \leq 2^{a_i}, \quad a_i > 0. \tag{10}$$

This splitting of the scheduling problem into two disjoint problems ensures conflict-free transmission schedule. However, from the objective of maximizing the channel reuse, the approach could lead to suboptimal schedule, as shall be seen in the ensuing discussion.

For a network comprising N nodes, let ξ be the average number of conflict-free transmissions over a duration of N slots. We define *slot reuse factor* denoted by η as a ratio of ξ and N, i.e., $\eta = \xi/N$.

Remark 1: We can show that $1 \leq \eta \leq N/3$. The minimum value of the slot reuse factor could be trivially achieved by assigning a slot to each node in a frame length of N without slot reuse. On the other hand the upper bound of the slot reuse factor could be achieved in linear topology, which is relatively easy to prove.

The slot reuse factor basically tells us how many times a slot is used in the network on average. For example, $\eta = 2.5$ would imply 2.5 transmissions per time-slot, on average in the network. Table I gives a comparison of the slot reuse factors for three schemes: namely, DRAND, HUDSAP, and PD-MAC. The frame lengths in DRAND and HUDSAP are selected according to (10). From the tables, we can observe that the PD-MAC could give substantially higher slot reuse factor. As shall be explained in the ensuing discussion, the PD-MAC achieves this higher slot reuse factor by jointly optimizing the slot allocation and frame size selection.

Our objective is to maximize the slot reuse factor η, which is a function of assigned slots and frame lengths, such that no other node in the contention area of a node i transmits in the same slot as node i. Concretely, we consider the following

TABLE I: Comparison of the slot reuse factor of different protocols for network in Fig. 2.

	Node ID	a	b	c	d	e	f	g	h
DRAND	Slot No (s)	1	2	3	1	4	5	1	2
	L	4	8	8	8	8	8	8	8
	η	1.1250							
HUDSAP	Slot No (s)	2	3	1	4	5	2	3	4
	L	4	8	8	8	8	8	4	4
	η	1.3750							
PD-MAC	Slot No (s)	1	4	2	1	6	14	1	4
	L	2	4	8	2	16	32	2	4
	η	2.2188							

optimization problem :

$$\begin{aligned} \underset{s_i, L_i, \forall i \in \mathcal{N}}{\text{maximize}} \quad & \eta(s_i, L_i) \\ \text{subject to} \quad & s_i \neq s_j \quad \forall j \in \mathcal{C}(i),\ \forall i \in \mathcal{N}, \\ & s_i, \log(L_i) \in \mathbb{N}_+,\ \forall i \in \mathcal{N}, \end{aligned} \tag{11}$$

where log denotes the logarithm to the base 2, and the set $\mathbb{N}_+$ contains all positive integers. Finding the optimal solution to this scheduling problem is NP-hard. That means we have to rely on the heuristics based approaches where we have to balance trade-off between optimality and computational complexity. Besides, it is desirable that the solution could be implemented in a distributed or decentralized way. To this end in [14] we proposed a greedy heuristic based solution (called PD-MAC) to the given optimization problem.

The PD-MAC works in a progressive decentralized way, runs on each each node independently, and uses information from one-hop neighbors. All nodes that have at least one-node in their one-hop neighborhood that has completed slot assignment are called *frontier nodes*. At any stage only the frontier nodes attempt slot assignment. For a network in which no node has yet assigned a slot, this last presumption implies that we have to seed the network to start the process of slot assignment. Concretely, that means we have to select a seed node, which could be any node in the network, that completes slot assignment first, and then its one-hop neighbors follows, and so on. According to the PD-MAC, each node decides its slot allocation and frame length jointly using information from its active one-hop neighbors. A node when becomes a frontier node listens to the on going transmission. In doing so it can estimate the start time of slots and hence can synchronize with the transmission, as explained in Section III. Once the node detects the slot boundary, it continues listing the channel in every slot. Based on this process, the node builds a *slot activity vector* (SAV). Each frame of the on going transmissions contains in it the TDMA frame length used by the respective nodes. This information is used by the new node to find out the maximum frame length currently used by its neighbors. The node builds the SAV within this frame-length. If during building the SAV, the node detects increase in the maximum frame length, the node accordingly adapts its SAV. The node does this training for a predetermined time duration. At the end of which, the node decides its frame length and allocates slots as explained in detail in [14]. In assigning slot(s) and

frame length, every node leaves out at least one slot free in its frame to allow another node to join the network at this location. After completing the assignment process, the node starts transmissions in the allocated slots.

V. Implementation on USRP Platform

A. Hardware and Software Platforms

In the implementation we use one USRP N210 and two USRP B100. The RF daughter boards of the USRPs are WBX Rev 2.0 and WBX Rev 3.0, respectively. The USRP N210 is connected with the host computer operating Linux kernel 3.3.1-5.fc16.x86_64, the processor is Intel(R) Core(TM)2 Quad CPU Q6600@2.4GHz, and the RAM is 4 GiB. The USRPs B100 are connected with host computers running Linux kernel 3.3.7-1.fc16.x86_64, processor Intel(R) Core(TM)i7 CPU 920@2.67GHz, and RAM 8 GiB.

The algorithms and protocols related to the PHY, MAC, and application layers are programmed in C++ in Qt development environment. In the programming we use C++ standard library and IT++ library. The IT++ is a C++ library of mathematical, signal processing and communication classes and functions which support simulation of communication systems. From the user perspective, the IT++ provides functionality similar to the MATLAB environment. The Qt is a C++ based cross-platform development framework which is widely used for developing softwares with or without a graphical user interface (GUI).

B. Software Architecture

Logically the program is divided into four threads: one main thread (called the GUI thread) and three worker threads as shown in Fig. 3. The worker threads are 'Construct Packet', 'Send/Receive Packet', and 'Process Packet'. Inter-thread communication is realized by the Qt signal-and-slot mechanism. The GUI thread, as the name implies, implements a graphical user interface. Via this interface, a user can pass certain parameters (e.g., transmit and receive frequencies, sampling rate, antenna gain, source and destination node addresses, un/acknowledged transmission mode, and MAC packet filtering) to the program and conversely the program displays its results to the user. Through the GUI, the user can also enter alphanumeric text to send to another node.

The send/receive packet thread is responsible for transmitting a packet from the host computer to the USRP device air interface; the thread is also responsible to receive a burst of samples from the USRP and deliver the burst to the host computer. The send and receive operations are burst-mode operations in which samples are received and transmitted in a timed manner within given slot boundaries.

Once transmission of a packet finishes, the send/receive packet thread requests the construct packet thread to start constructing a new packet. Upon receiving this request, the latter thread constructs a new packet. New burst of samples captured by the send/receive packet thread is delivered to the process packet thread, which processes the burst to find out if it contains any valid packet. If the process packet thread detects a data frame then it sends the received data

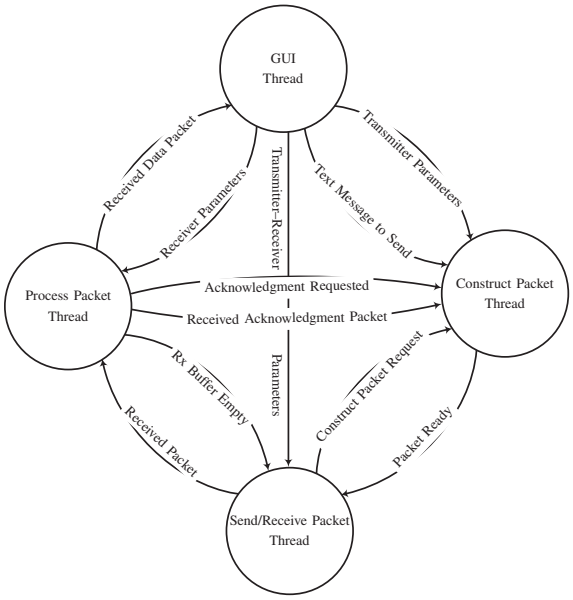


Fig. 3: Architecture of the software program showing the threads and the signals exchanged among them.

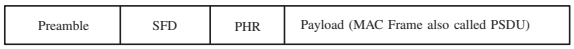


Fig. 4: PHY frame format.

for display to the GUI thread. Besides, if the received frame asks for an acknowledgment, the processing thread signals the packet construction thread to construct an acknowledgment packet. If the received packet is an acknowledgment packet, the construction packet thread is notified about it which takes appropriate action. More concretely, upon receiving such a signal, the packet construction thread removes any packet from its packet queue against which the acknowledgment is received. However, if no such packet exists, the thread simply ignores the signal.

C. PHY and MAC Parameters

The PHY frame format and the coding of information bits are adopted from the 250 kbps chirp spread spectrum (CSS) PHY of IEEE Std 802.15.4-2011. The modulation of the PHY layer is DQPSK. The MAC frame types and formats are also adopted from the IEEE Std 802.15.4-201, albeit we customize the use of some of frame header fields and frame types to serve our purpose.

Fig. 4 shows the format of the PHY frame. The frame comprises the preamble part, SFD (start of frame delimiter), PHR (PHY header), and the payload part. The PHR field represents the length of the payload part in bytes. The lengths of all fields except the PHR field is same as specified in [18]. The PHR field length is limited to one byte. The modulator employed in our implementation is derived from [18]. The modulator outputs DQPSK symbols shaped with raised cosine time-window.

On the receive side, samples are acquired from the USRP in bursts. The receiver processes each burst independently. While

processing a burst, the receiver first checks if a signal is present on the channel. To this end, we calculate signal energy by using running average over samples spanning eight modulation symbols, that is,

$$\mathcal{E}_{\text{avg}}^{(i)} = \frac{1}{8K} \sum_{\kappa=iK}^{(i+8)K-1} |r(\kappa T_s)|^2, \qquad i = 0, 1, 2, \ldots, \quad (12)$$

where $r(\kappa T_s)$ is the sampled base-band received signal, T_s is the sampling period, and $K = T/T_s$ with T denoting the DQPSK symbol duration. Note that (12) is a test statistics for the energy-based sensing scheme. The signal is assumed to be present in the ith averaging window if $\mathcal{E}_{\text{avg}}^{(i)}$ is greater than a threshold. Once signal is detected frequency offset estimation and correction is performed in the next step.

Frequency offset exists in the received signal due to a finite difference between the clock frequency of the transmitter and the receiver. Since the received signal can have frequency offsets, (coarse) carrier offset estimation and compensation needs to be done before frame detection can proceed. In the presence of frequency offset, the received base-band signal with additive noise can be expressed as follows

$$r(\kappa T_s) = e^{j(2\pi v t + \theta)} \sum_i c_i g(\kappa T_s - iT - \tau) + n(t), \quad (13)$$

where v denotes the frequency offset, θ is phase offset, c_i's are modulation symbols, $g(t)$ is the pulse shaping filter, and $n(t)$ is the noise term. Since we are working with the DQPSK modulation, where information are transmitted in the phase difference, the phase offset θ can be ignored as long as it remains constant over the feedback length of the differential encoder used at the transmitter. To estimate the frequency-offset, we use *delay-and-multiply* method from [19]. This is an open-loop offset estimator that does not rely on any specific signal features—blind offset estimation. The acquisition range of this estimator is about $[\frac{-1}{T}, \frac{1}{T}]$ and its mean-square-error performance is comparable with the closed-loop methods [19]. Besides, the acquisition time of this method (being an open-loop method) is shorter than the closed-loop methods. This last feature, makes such a method more suitable for bust-mode transmissions like TDMA system. According to the delay-and-multiply method, an estimate of v can be given as follows:

$$\hat{v} = \frac{2}{\pi T} \arg \left\{ \sum_{k=0}^{4L_0-1} r\left(\frac{kT}{4}\right) r^*\left(\frac{(k-1)T}{4}\right) \right\}, \quad (14)$$

where $L_0 = \frac{T_0}{T}$ with T_0 denoting the length of the signal burst. Knowing the estimate of the frequency offset, we compensate for it as $e^{-j2\pi\hat{v}\kappa T_s} r(\kappa T_s)$.

After correcting the frequency offset, we find beginning point of the PHY frame. For this purpose, we use known preamble part of the frame. Concretely, we correlate the sample sequence of known preamble with the signal burst and find location of the correlation peak. From the peak index, we can estimate starting point of the frame. Let $p(n)$ denotes the preamble part, of length M, of a transmitted frame[2]. Moreover, let N be the length of the received signal burst $r(n)$. For computation of the cross-correlation function (CCF), $p(n)$ is zero-padded to the length of $r(n)$[3]. The CCF can be given by:

$$z(i) = \sum_{n=0}^{N-1} r^*(n) p(n+i), \quad i = -N+1, \ldots, N-1. \quad (15)$$

The test statistics for the frame start position is defined as

$$I = \arg \max_{-N+1 \le i \le N-1} |z(i)|. \quad (16)$$

Now in finding frame start position one of the following cases may arise depending on the position of the preamble part within the received burst:

1) $I = 0$, the preamble is completely contained in the beginning of the burst, that is, the frame starts at $r(0)$.
2) $I < 0$, only later part of the preamble is contained in the beginning of the burst. In this case $|I|$ samples of the preamble are lost and $r(0)$ is $|I+1|$th sample of the preamble.
3) $I > 0$, complete preamble is sandwiched within the burst, and in this case frame starts from $r(I)$.

Thus from the knowledge of index I we can ascertain starting position of a frame within a received burst. Besides, knowing the frame start position we can recover the symbol timing. From the frame starting point, we keep samples of the current signal burst equivalent to the maximum expected frame length and discard rest of the burst. The received burst then enters the demodulator stage which performs converse functions of the corresponding modulator stages, and outputs the MAC layer frame.

VI. Results

In this section performance results are obtained by operating the PHY with DQPSK symbol duration of 16 micro seconds. We use the sampling frequency of one MSPS[4]. In the experiments, transmissions are done in the ISM radio band 433.92 MHz. With the given symbol duration, the maximum length of the transmitted frames could be as much as 43.777 milliseconds. And the minimum frame length could be as low as 4.864 milliseconds[5].

A. Master/Slave Based Scheme

In the master/slave setup, NTP-based message exchange is used to synchronize time between the master and slave nodes. Fig. 5 shows the synchronization request and response messages captured on a real-time spectrum analyzer. After the synchronization reply message is received by the slave node, it can calculate the time-offset and the delay. Having calculated these values, the slave node is synchronized with the master node. In the same figure, after the time synchronization, we can see a sequence of data messages exchanged between the two nodes. Fig. 6 plots the measured delay at two nodes named

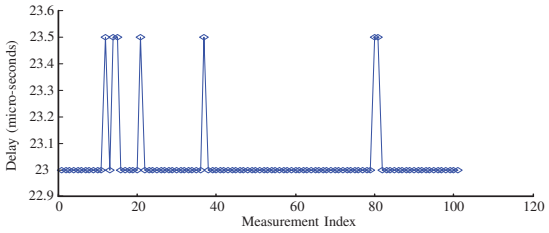
[2]As both preamble and SFD fields are fixed and known, for start of frame detection we use both of them. Thus $p(n)$ comprises the preamble and SFD fields.

[3]We assume length of $r(n)$ greater than that of the $p(n)$, else no valid frame can be contained in the received signal burst.

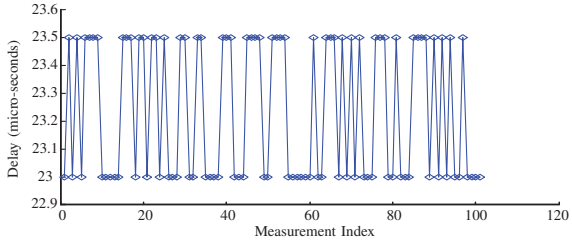
[4]The symbol duration and the sampling frequency are re-configurable.

[5]The shortest frame is the acknowledgment frame which does not have any payload in the MAC part.

Fig. 5: An example of message-exchange sequence in master/slave setup. Upon receiving sync reply message, slave node is synchronized with the master node. After synchronization, the master and slave nodes exchange data messages according to their slot assignment.



(a) Delay measured at STB.



(b) Delay measured at YNG.

Fig. 6: Delay measured at two nodes, named YNG and STB. The node at which delay is measured acted as a slave node while the other acted as the master node.

YNG and STB, respectively, connected to the USRP B100 and the USRP N210. From the figures, we can observe that the delay switches between 23 and 23.5 micro-seconds over the measurements. We can conclude that the delay remains fairly constant at around 23 micro-seconds for the two USRP platforms. This constant delay is achieved by moving the time-stamp close to the physical layer and thus excluding the variability contributed by the upper layers which is observed in other implementations [11].

B. Distributed Scheme

In this distributed scheme, the delay d is assumed to be known (e.g., from the previous section) and the nodes compensates for this delay during transmissions. Here for time synchronization and slot allocation nodes use the decentralized scheme outlined in this paper. Fig. 7 shows screen shots captured from the real-time spectrum analyzer for a three node network. The first screen-shot shows when there is only one node in the network which occupies the first slot in a frame length of two. Subsequently, when the second node joins the

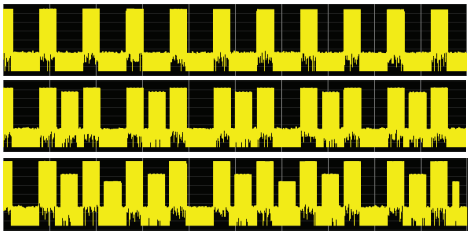


Fig. 7: Screen shots from a real time spectrum analyzer showing working of the PD-MAC protocol for a three-node network implemented on a USRP based test-bench.

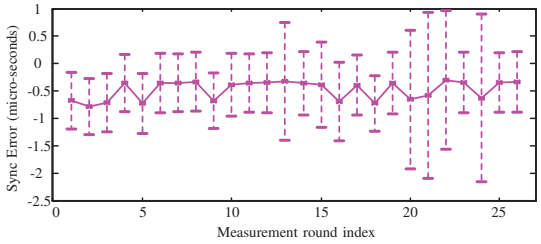


Fig. 8: Time synchronization error measured at a node. Each point on the solid line shows mean error measured over a round of 30 minutes. The dotted vertical bars show variations where length of each bar is equal to two times the standard deviation.

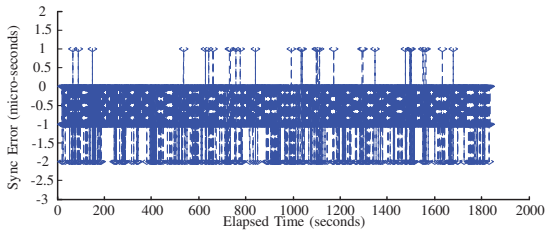


Fig. 9: Time synchronization error measured at the given node during a measurement round.

network, the second screen-shot shows, it occupies the second slot in a frame length of four. When the third node arrives, it occupies the forth slot in a frame length of eight as shown in the third screen-shot.

Once a node completed slot assignment, we measured the time-offset from the slot boundaries over measurement period of 30 minutes. The results of time-offset measured at a node are plotted in Fig. 8 and Fig. 9. We can observe that the synchronization error remains within 2 micro-seconds and mostly the error is within 1 micro-second (the sampling period).

VII. Concluding Remarks

The TDMA based MAC scheduling is an appropriate choice for many applications in which deterministic medium access scheduling is required. There exists plentiful literature on the TDMA based scheduling. However, majority of these cases focus on finding a conflict-free slot allocation among nodes. In those works, the problem of TDMA based scheduling

is broken into two disjoint problems which are sequentially solved: first find conflict-free slot allocation, minimizing the number of slots used, and subsequently select frame sizes in which to use the assigned slots. This disjoint and sequential treatment results in suboptimal spatial reuse of the slots in the network. Time synchronization is an integral part of a TDMA based MAC in which nodes need to agree and respect the slot boundaries in transmitting and receiving data. The existing slot assignment schemes assume that the nodes have already achieved time synchronization by some other method. In this paper, we outlined a fully functional TDMA based MAC protocol by which nodes can simultaneously do time synchronization, assign slots, and select frame lengths in which to use the assigned slots. The MAC protocol works in a progressive decentralized way. Each node runs the protocol independently, and for time synchronization and slot assignment the node use information from its one-hop neighborhood. The proposed slot assignment scheme can give better slot reuse than the existing schemes. This is realized by jointly optimizing the slot allocation and frame length selection.

We implemented the TDMA based MAC protocol on a USRP test-bench. We showed that the nodes can find conflict-free slot assignment. We measured overall delay between two USRPs and found that it remains at around 23 micro-seconds with little variability. This approximately constant delay is achieved by moving the time-stamp close to the physical layer and thus avoiding the variable delays contributed by the upper layers. This constant delay has important implication on the achievable synchronization accuracy insomuch as we can compensate for it and thus reduce the synchronization error. We also showed that time-synchronization accuracy on the order of a sampling period can be achieved. Besides, the accuracy at this level can be maintained at this level without explicit exchange of synchronization messages.

Appendix A
Finding Time-Offset from Slot Boundary

Let there be a pre-defined slot boundary at which time Tx and Rx are supposed to transmit and receive, respectively. However, due to their local clock drifts and erroneous knowledge of the slot boundaries, Tx or Rx, or both, may go into their respective modes earlier or later than they are supposed to do. The timing-offset errors may be classified into three categories: *Rx directed*, *Tx directed*, and *Tx–Rx directed*. In the Rx directed offset, the Tx transmits at the slot boundary whereas Rx either starts receiving earlier or later than the boundary. In the Tx directed offset converse happens. The Tx–Rx directed type appears as a combination of the earlier two. At the Rx we can estimate timing-offset with respect to its notion of the slot. And to do that, Rx use knowledge acquired from the receiver signal sensing stage and the start of PHY frame detection stage (cf., Section V-C). Concretely, let K_{s} be the number of samples discarded in the signal sensing from the captured signal burst. Also, let τ_{off} be the timing offset. The offset can be given by:

$$\tau_{\mathrm{off}} = (K_{\mathrm{s}} + I)\, T_s, \tag{17}$$

where I is index of the CCF peak defined in (16).

The accuracy of the above method depends on the accuracy of the index I of the CCF peak. The index is determined by correlating the received burst sequence with the known preamble sequence. The detection performance of correlation based methods usually degrades under multi-path fading environments and residual frequency offset.

References

[1] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: Part-I carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. on Commun.*, vol. 23, no. 12, pp. 1400–1416, Dec. 1975.

[2] B. P. Crow, I. Widjaja, L. G. Kim, and P. T. Sakai, "IEEE802.11 wireless local area networks," *IEEE Commun. Mag.*, vol. 35, no. 9, pp. 116–126, Sep. 1997.

[3] P. Suriyachai, J. Brown, and U. Roedig, "Time-critical data delivery in wireless sensor networks," in *Proc. of 6th IEEE Int. Conf. on Distributed Computing in Sensor Systems (DOCSS'10)*, Santa Barbara, CA (USA), Jun. 2010, p. 216229.

[4] P. Suriyachai, U. Roedig, and A. Scott, "A survey of MAC protocols for mission-critical applications in wireless sensor networks," *IEEE Commun. Surveys & Tutorials*, vol. 14, no. 99, pp. 1–24, Mar. 2011.

[5] M. H. Chaudhary and B. Scheers, "High spatial-reuse distributed slot assignment protocol for wireless ad hoc networks," in *Proc. of Military Commun. and Inform. Systems Conf. (MCC'12)*, Gdańsk (Poland), Oct. 2012, pp. 1–8.

[6] S. Perumal and J. S. Baras, "Performance evaluation of single channel virtual-circuit MAC protocols for manets," in *Proc. of IEEE Int. Conf. on Military Commun. (MILCOM'12)*, Baltimore, MD (USA), Nov. 2011, pp. 926–931.

[7] S. Dastangoo, T. G. Macdonald, D. Reinharth, and C. Burns, "Performance analysis of distributed time division multiple access protocols in mobile ad hoc environments," in *Proc. of IEEE Int. Conf. on Military Commun. (MILCOM'09)*, Boston, MA (USA), Oct. 2009, pp. 1–7.

[8] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized TDMA scheduling for wireless ad hoc networks," *IEEE Trans. on Mobile Computing*, vol. 8, no. 10, pp. 1384–1396, Oct. 2009.

[9] I. Rhee, A. Warrier, M. Aia, , J. Min, and M. L. Sichitiu, "Z-MAC: A hybrid MAC for wireless sensor networks," *IEEE/ACM Trans. on Networking*, vol. 16, no. 3, pp. 511–524, Jun. 2008.

[10] S. M. Lasassmeh and J. M. Conrad, "Time synchronization in wireless sensor networks: A survey," in *Proceedings of the IEEE SoutheastCon*, March 2010, pp. 242–245.

[11] S. Ganeriwal, R.Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. of 1st Int. Conf. on Embedded Networked Sensor Systems (SenSys)'03*, Los Angeles, CA (USA), Nov. 2003, pp. 138–149.

[12] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proceedings of ACM Conference SenSys (SenSys04)*, Nov. 2004.

[13] D. L. Mills, "Internet time synchronization: The network time protocol," in *Global States and Time in Distributed Systems*, Z. Yang and T. Marsland, Eds. IEEE Computer Society Press, 1994.

[14] M. H. Chaudhary and B. Scheers, "Progressive decentralized TDMA based MAC: Joint optimization of slot allocation and frame lengths," in *IEEE Int. Conf. on Military Commun. (MILCOM'13)*, San Diego, CA (USA), Nov. 2013.

[15] E. L. Lloyd and S. Ramanathan, "On the complexity of link scheduling in multi-hop radio networks," in *Proc. of 26th Conf. on Inform. Science and System (ISS'92)*, 1992.

[16] S. Even, O. Goldreich, S. Moran, and P. Tong, "On the NP-completeness of certain network testing problems," *Jr. on Networks*, vol. 14, no. 1, pp. 1–24, 1984.

[17] S. Ramanathan, "A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks," in *Proc. of 16th IEEE Int. Conf. on Computer Commun. (INFOCOM'97)*, vol. 2, Kobe (Japan), Apr. 1997, pp. 900–907.

[18] IEEE, "IEEE Std 802.15.4-2011, IEEE standard for local and metropolitan area networkspart 15.4: Low-rate wireless personal area networks (WPANs)," 2011.

[19] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*. New York: Plenum Press, 1997.