TMMAC: An Energy Efficient Multi-Channel MAC Protocol for Ad Hoc Networks

Jingbin Zhang †, Gang Zhou †, Chengdu Huang ‡, Sang H. Son †, John A. Stankovic †
† Department of Computer Science
University of Virginia
{jz7q, gzhou, son, stankovic}@cs.virginia.edu
‡ Department of Computer Science
University of Illinois
chuang30@cs.uiuc.edu

***Abstract*— This paper presents a TDMA based multi-channel MAC protocol called TMMAC for Ad Hoc Networks. TMMAC requires only a single half-duplex radio transceiver on each node. In addition to explicit frequency negotiation which is adopted by conventional multi-channel MAC protocols, TMMAC introduces lightweight explicit time negotiation. This two-dimensional negotiation enables TMMAC to exploit the advantage of both multiple channels and TDMA, and achieve aggressive power savings by allowing nodes that are not involved in communication to go into doze mode. Moreover, TMMAC dynamically adjusts its negotiation window size based on different traffic patterns, which further improves communication throughput and energy savings. In this paper, the performance of TMMAC is analyzed and evaluated. The evaluations show that TMMAC achieves up to 113% higher communication throughput while consuming 74% less per packet energy over the state-of-the-art multi-channel MAC protocols for single-transceiver wireless devices.**

I. Introduction

Media access control is an essential part of the wireless communication stack and it has obtained intensive research attention. More recently, to reduce radio congestion and achieve higher communication throughput, multi-channel MAC has been studied.

This paper focuses on how to incorporate both the advantages of multiple channels and TDMA into the MAC design with low overhead, when each node in the network is only equipped with a single half-duplex radio transceiver. Such hardware can not transmit and receive at the same time, but it can switch its frequency dynamically. Many of the previous multi-channel MAC designs [1][2][3] require multiple radio transceivers. Multiple radios not only result in higher product prices, but also consume more power from energy-constrained devices. Plus, most current IEEE 802.11 devices are equipped with a single half-duplex radio transceiver. Therefore, it is important to devise an energy efficient multi-channel MAC protocol based on a single half-duplex transceiver.

In this single transceiver context, conventional multi-channel MAC designs adopt explicit frequency negotiation [4][5][6][7][8], through certain kinds of control messages. This one-dimensional negotiation enables these MAC protocols to take advantage of multiple channels and achieve better performance than IEEE 802.11.

In this paper, we propose an energy efficient multi-channel MAC protocol called TMMAC. In addition to conventional frequency negotiation, TMMAC introduces lightweight explicit time negotiation. In TMMAC, time is divided into fixed periods, which consists of an ATIM (Ad Hoc Traffic Indication Messages) window followed by a communication window. The ATIM window size is dynamically adjusted based on different traffic patterns to achieve higher throughput and lower energy consumption. The communication window is time slotted, each of which is called a time slot. The duration of each time slot is the time needed for a single data packet transmission or reception. During the ATIM window, each node decides not only which channels to use, but also which time slots to use for data communication. Then each node adopts the negotiated frequency for each time slot to transmit or receive data packets. From the TDMA's point of view, TMMAC is a traffic-adaptive and energy-efficient TDMA scheduling algorithm.

This two-dimensional negotiation enables TMMAC to take full advantage of both multiple channels and TDMA. The main contributions of this work can be summarized as follows:

- We present an energy efficient multi-channel MAC protocol called TMMAC. TMMAC avoids contention based communication for data packets in the communication window and allows nodes to use different frequencies within different time slots through two dimensional negotiation. This property allows TMMAC to achieve more efficient bandwidth usage. Further, TMMAC achieves aggressive power savings by putting a node into doze mode in a time slot whenever it is not scheduled to transmit or receive a packet. Finally, TMMAC supports broadcast very efficiently.
- We provide an analytical model for TMMAC. The model accurately characterizes the performance of TMMAC and is validated through simulation. The impact of time synchronization errors is analyzed based on the model.
- We propose a novel scheme to dynamically adjust the ATIM window size efficiently based on different traffic patterns for TMMAC, which improves both the network throughput and energy efficiency of TMMAC.

The rest of this paper is organized as follows. We review related work in Section II. Then we present the details of the TMMAC protocol design in Section III. In Section IV, an analytical model of TMMAC is presented and validated. We describe the design of dynamic ATIM window adjustment scheme in Section V. Section VI contains a complete evaluation of TMMAC. Finally, we give the conclusions in Section VII.

II. State of the Art

A large number of multi-channel MAC protocols and TDMA scheduling algorithms have been proposed in the literature. Many multi-channel protocols are based on special hardware assumptions. [9][10] assume the use of frequency hop-

ping spread spectrum (FHSS) wireless cards, and in [11] the busy-tone ability is required for the radio hardware. In [1][2][3][12][13], either multiple radio transceivers or a single sophisticated transceiver is required to be capable of listening to multiple frequencies at the same time. For example, [1] requires two radio transceivers. One listens to the control channel and the other listens to the data channel simultaneously. In TMMAC, we do not have such special hardware requirements. TMMAC only requires a single half duplex radio transceiver.

Multi-channel MAC protocols that are closely related to TMMAC are the ones that extend IEEE 802.11 Distributed Coordination Function (DCF) protocol [14] and use certain kinds of control messages for frequency negotiation. Typical protocols in this group are [4][5][6][7][8]. Among these protocols, MMAC [4] is the most related one to TMMAC. MMAC assumes time synchronization in the network and time is divided into fixed-length beacon intervals. Each beacon interval consists of a fixed-length ATIM window, followed by a communication window. During the ATIM window, every node listens to the same default channel and negotiates which channel to use for data communication. After the ATIM window, nodes that have successfully negotiated channels with their destinations send out data packets using 802.11 DCF [14] for congestion avoidance. Nodes that do not achieve successful negotiations or do not have packets to send or receive go into doze mode to save power. From simulation studies, MMAC successfully exploits multiple channels to achieve higher throughput than 802.11.

Besides the explicit frequency negotiation, pseudo random number generators are also used in [15] to help frequency allocations and switches. Nodes have different random numbers at different times, and communication is allowed when neighboring nodes share the same random numbers.

In recent days, multi-channel MAC protocols in Wireless Sensor Networks (WSNs) are also studied [16]. Due to the limited size of the data packets used in WSNs, [16] uses the static frequency assignment to avoid the overhead of control packets for frequency negotiation.

There are also many TDMA scheduling algorithms [17][18][19][20] proposed for ad hoc networks in the literature. These TDMA scheduling algorithms are mainly designed for sharing a single channel in the network and providing collision-free channel access scheduling in that single channel. In these protocols, frequency diversity is not exploited to improve communication throughput.

III. TMMAC Design

In this section, we present the TMMAC protocol. Figure 1 shows the overall architecture of TMMAC. Like the IEEE 802.11 Power Saving Mechanism (PSM) [14] and MMAC [4], TMMAC requires time synchronization [14][21]. In TMMAC, time is divided into fixed-length beacon intervals and each beacon interval is comprised of an ATIM window and a communication window. Different from 802.11 PSM and MMAC, in TMMAC, the ATIM window size is dynamically adjusted and the communication window is further divided into time slots. For ease of description, in this section, we assume that the ATIM window size is fixed. The dynamic ATIM window scheme will be discussed in Section V.

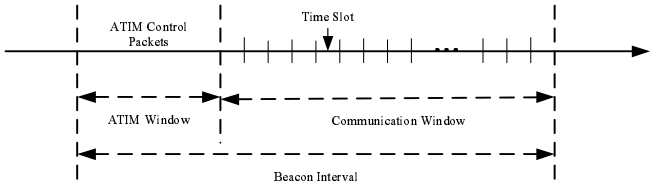


Fig. 1. Overall architecture of TMMAC.

During the ATIM window, all the nodes listen to the same default channel for negotiation. Four types of messages are used for negotiation: ATIM, ATIM-ACK (ATIM-Acknowledgement), ATIM-RES (ATIM-Reservation) and ATIM-BRD (ATIM-Broadcast). They are called ATIM control packets. In TMMAC, the communication during the ATIM window is contention based and uses the same scheme as the one used in 802.11 DCF [14]. During the negotiation, the sender and receiver decide not only which channels to use, but also which time slots to use for a set of data packets, the number of which is specified by the sender. Then in each time slot, each node adopts the negotiated frequency to transmit or receive data packets. The duration of each time slot is long enough to accommodate a data packet transmission, including the time needed to switch the channel, transmit the data packet and the acknowledgement.

A. Data Structures

We first describe two data structures used in TMMAC. They are the Channel Usage Bitmap (CUB), and the Channel Allocation Bitmap (CAB).

The CUB is the main data structure that needs to be maintained at each node. Each CUB represents the current usage information of one channel. So if the radio transceiver has M available channels, there are M CUBs in each node. These CUBs are used to keep track of the allocations of all the $previous$ negotiations in the current ATIM window.

	Slot 1	Slot 2	Slot 3	Slot 4	...	Slot N
CUB_i	1	1	0	0	...	0

Fig. 2. An example of a CUB.

Figure 2 shows an example of a CUB. CUB_i represents the usage information of channel i in different time slots for the current beacon interval. A single bit is used to represent whether a time slot is occupied or not. Therefore, if the beacon interval has N time slots, each CUB contains N bits in it. We use $CUB_{i,j}$ to denote the j^{th} bit in CUB_i. If $CUB_{i,j}$ equals 1, it means that channel i in time slot j is already allocated by its neighbors or itself in previous negotiations. If it is 0, it means that channel i in time slot j is available for allocation.

The CAB has the same data structure as CUB. Different from the CUB, a CAB describes which time slots in that channel are allocated by the $current$ negotiation. We use CAB_i to represent the allocation information of channel i. $CAB_{i,j}$ is used to denote the j^{th} bit in CAB_i. The CABs are not maintained at the node and are only transmitted along with ATIM-ACK, ATIM-RES or ATIM-BRD packets, telling the

neighboring nodes which channels and which time slots are allocated by the current negotiation. If $CAB_{i,j} = 1$, it means that channel i in time slot j is allocated by this negotiation. Otherwise, it is not allocated.

The rules to change the values of the CUBs are described as follows: 1) The value of each bit in all the CUBs is reset to 0 when the node is powered up or it is at the start of each beacon interval. 2) For both unicast and broadcast, if the sender and the receiver(s) are negotiated to use channel i in time slot j for data transmission, the bits in time slot j of all the CUBs in both the sender and the receiver(s) are set to 1. 3) If a node overhears an ATIM-ACK, or ATIM-RES packet, $CUB_{i,j}$ is set to 1 if $CAB_{i,j}$ contained in the packet is 1.

B. Unicast Negotiation

For unicast packets, the ATIM, ATIM-ACK and ATIM-RES packets are used for negotiation. If node S wants to send a set of packets to node R, node S first sends an ATIM packet to node R containing all its CUBs and the number of packets it wants to send. After receiving the ATIM packet, node R decides which channels and which time slots to use based on the information of its own CUBs and the CUBs from the sender S. The selection procedures used by node R are described as follows:

Node R performs an OR operation on node S's CUBs and its own CUBs to generate M combined CUBs. If $CUB_{i,j}$ in the combined CUB_i equals 1, channel i in time slot j can not be allocated for this transmission. Otherwise, channel i in time slot j can be allocated for this transmission. If there are multiple available channels at one time slot, at most one channel can be chosen for data transmission, because a node has a single half-duplex radio transceiver in TMMAC. Then, following the rule above, node R randomly selects channels and time slots, among available ones from the M combined CUBs for this transmission. If node R can not allocate enough time slots as the number of packets specified in the ATIM packet, node R allocates as many time slots as possible.

After node R decides which channels and which time slots to use for this transmission, node R updates its CUBs, and generates the corresponding CABs. If channel i in time slot j is selected, $CAB_{i,j}$ is set to 1. Otherwise it is set to 0. Then node R replies back with an ATIM-ACK message, which contains the generated CABs. The nodes in the vicinity of node R update their CUBs by overhearing the ATIM-ACK message based on the rules described in Section III-A and know the current channel usage information. After receiving the ATIM-ACK packet from node R, node S updates its CUBs based on the CABs from node R and sends out an ATIM-RES packet containing the same CABs to node R. By overhearing the ATIM-RES packet, the nodes near node S update their CUBs to obtain the current channel usage information.

C. Broadcast Negotiation

For broadcast packets, the ATIM-BRD packets are used for negotiation. If node S has some packets to broadcast during this beacon interval, node S first selects the channels and the time slots it wants to use based on its own CUBs. The selection procedures are described as follows:

Node S randomly selects the time slots, in which all the channels are not used by any of its neighbors yet. If node S can not allocate enough time slots for the broadcast messages, node S allocates as many time slots as possible. After node S selects the time slots, node S randomly selects a channel for each chosen time slot for the broadcast messages.

After node S decides which channels and which time slots to use for the broadcast packets, node S updates its CUBs, and generates the corresponding CABs. If channel i in time slot j is selected, $CAB_{i,j}$ is set to 1. Otherwise, it is set to 0. Then node S broadcasts the ATIM-BRD packet with the CABs in it. By overhearing the ATIM-BRD packet, the nodes in the vicinity of node S learn in which time slots they will receive the broadcast packets and which channels to use, and update their CUBs.

Since in broadcast negotiation, the time slot to be allocated requires that no channel in that time slot has been used, it is more difficult to allocate time slots for broadcast packets than for unicast packets. To alleviate that problem, we give higher priority to broadcast negotiation. We adopt the following two methods to achieve this: 1) Within a single node, if we have both unicast packets and broadcast packets to send, we initiate broadcast negotiation before unicast negotiation. 2) Among multiple nodes, we use a smaller backoff window for broadcast negotiation, which makes accessing the medium easier for broadcast negotiation.

D. Data Packet Transmission

After the ATIM window, nodes can send packets, receive packets or go into doze mode based on their schedules. If a node has negotiated to send or receive a packet in the i^{th} time slot, the node first switches its channel to the negotiated channel when it enters the i^{th} time slot and then transmits or waits for the data packet. An ACK is sent if a receiver receives a unicast packet. If a sender does not hear an ACK after it sends out a unicast packet, that packet is retransmitted in the next scheduled time slot, which is negotiated with the same receiver. To avoid head of line blocking problem in a single FIFO queue, in which a packet needs to be sent is blocked by the packet at the top of the queue, TMMAC maintains per-neighbor FIFO queues. If a node has not negotiated to send or receive a data packet, including broadcasts, in the i^{th} time slot, the node switches to doze mode to save power.

IV. Analytical Model

In this section, we present an analytical model to compute the throughput of TMMAC in Wireless Local Area Networks (WLAN's). Although multi-hop networks are more complicated than WLAN's, the analysis based on WLAN's helps us to better understand the performance of TMMAC.

The throughput of TMMAC is mainly determined by the number of packets that can be scheduled during the ATIM window and the maximum number of packets that can be accommodated during the communication window. We first compute the number of packets that can be scheduled during the ATIM window, which mainly depends on the number of successful negotiations during the ATIM window. So we compute the number of successful negotiations that can be

accommodated during the ATIM window in the first place. Since in TMMAC, the communication of ATIM control packets follows 802.11 DCF [14], we use the analytical model for 802.11 DCF to characterize the communication performance of ATIM control packets. In [22], Bianchi gives an accurate model for the normalized system throughput S in 802.11 DCF for WLAN's, when the network is saturated. S is "defined as the fraction of time the channel is used to successfully transmit a payload bit". Readers are referred to [22] for details. For simplicity, we use the following equation to express S:

$$S = \frac{P_{succ}E[P]}{P_{idle}t + P_{succ}T_s + P_{coll}T_c} \quad (1)$$

Here, P_{succ} is the probability for a contention slot[1] to have a successful transmission, P_{coll} is the probability for a contention slot to have a collision, P_{idle} is the probability of having an empty contention slot, t is the contention slot size, T_s is the average time the channel is sensed busy due to a successful transmission (i.e., the time needed to complete the RTS-CTS-DATA-ACK handshake in 802.11 DCF, if RTS/CTS is enabled), T_c is the average time the channel is sensed busy during a collision, and $E[P]$ is the average packet payload size. P_{succ}, P_{idle} and P_{coll} depend on the number of contending nodes, given that the sizes of backoff windows and the number of backoff stages are fixed.

Equation (1) shows that the number of successful RTS-CTS-DATA-ACK handshakes per time unit in 802.11 DCF is $\frac{S}{E[P]} = \frac{P_{succ}}{P_{idle}t+P_{succ}T_s+P_{coll}T_c}$, if we assume that RTS/CTS is enabled. This presents a way to compute the number of successful negotiations during the ATIM window. Let N_s represent the number of successful negotiations per time unit. It is easy to see that $N_s = \frac{P_{succ}}{P_{idle}t+P_{succ}T_s^{atim}+P_{coll}T_c^{atim}}$, in which T_s^{atim} and T_c^{atim} denote the average time the channel is sensed busy due to a successful negotiation and a collision of the ATIM packets, respectively. T_s^{atim} and T_c^{atim} are computed as follows when we only consider unicast negotiation:

$$\begin{aligned} T_s^{atim} &= ATIM + SIFS + \delta + ATIMACK + SIFS \\ &\quad +\delta + ATIMRES + DIFS + \delta \\ T_c^{atim} &= ATIM + DIFS + \delta \end{aligned} \quad (2)$$

Here, δ is the propagation delay, and SIFS and DIFS represent short interframe space and distributed interframe space used in 802.11 DCF, respectively. Equation (2) also shows that to achieve higher N_s, we need to keep the sizes of ATIM control packets as small as possible. To reduce the overhead of ATIM control packets, we include only 3 CUBs which have the least utilizations up to now in the ATIM packet when the number of channels is larger than 3. Then the receiver allocates from the 3 channels whose CUBs are contained in the ATIM packet and generates the corresponding 3 CABs for the ATIM-ACK packet.

[1]The contention slot is different from the time slot used in the communication window of TMMAC. The contention slot size is the time unit of the discrete-time backoff algorithm employed by 802.11 DCF, which equals the time needed at any station to detect the transmission of a packet from any other station.

Assume that each successful negotiation schedules η data packets on average and the length of the ATIM window is l_{atim}. The average number of data packets that can be scheduled during a single ATIM window is as follows:

$$n_{schedule} = N_s \times l_{atim} \times \eta \quad (3)$$

Now we compute the maximum number of packets that can be accommodated during the communication window. Let l_{slot} be the length of the time slot. l_{slot} depends on the bandwidth of the network, length of the data packet, and the time synchronization error. Suppose that the bandwidth is B, the length of the data packet header is H, the channel switch delay is t_{cs}, and the maximum time synchronization error is t_{max}. For simplicity, we assume that the maximum data packet payload size equals the average data packet payload size, which is denoted as $E(P_d)$. It is easy to prove that if data packets are transmitted t_{max} after the beginning of each time slot, data transmissions in different time slots do not overlap as long as:

$$l_{slot} = \frac{H + E(P_d) + ACK}{B} + 2\delta + t_{cs} + 2t_{max} \quad (4)$$

Suppose l_{beacon} is the length of the beacon interval. Then we get the maximum number of data packets which can be accommodated during a single communication window as follows:

$$n_{accommodate} = \lfloor \frac{(l_{beacon} - l_{atim})}{l_{slot}} \rfloor \times M \quad (5)$$

Because the actual number of packets scheduled during the ATIM window must be smaller than or equal to the maximum number of packets that can be accommodated during the communication window, the actual number of packets transmitted during the communication window is the smaller value of n_{actual} and $n_{accommodate}$, which is denoted as follows:

$$n_{actual} = min\{n_{schedule}, n_{accommodate}\} \quad (6)$$

So the throughput of TMMAC is $(E(P_d) \times n_{actual})/l_{beacon}$. This presents a convenient way to compute the saturation throughput of TMMAC with certain parameter settings. Based on the above equations, we obtain that when $n_{schedule} = n_{accommodate}$, TMMAC achieves the maximum throughput. We use l_{opt} to denote the optimal ATIM window size, which makes $n_{schedule} = n_{accommodate}$. It can be easily proved that the maximum throughput is achieved when $n_{schedule} = n_{accommodate}$ by showing that either $n_{schedule}$ or $n_{accommodate}$ decreases when the ATIM window size is changed from l_{opt}. When $n_{schedule} = n_{accommodate}$, we obtain l_{opt} as follows:

$$l_{opt} = \frac{l_{beacon}}{1 + \frac{N_s \times \eta \times l_{slot}}{M}} \quad (7)$$

Combining Equation (3) and Equation (7), the maximum throughput of TMMAC is computed as follows:

$$T_{max} = \frac{1}{\frac{1}{N_s \eta E[P_d]} + \frac{H+ACK}{BME[P_d]} + \frac{1}{BM} + \frac{2\delta+2t_{max}+t_{cs}}{ME[P_d]}} \quad (8)$$

Equation (8) shows how to determine the maximum throughput of TMMAC based on different parameter settings. It also presents a theoretical way to analyze the impact of different parameters on the performance of TMMAC, such as the impact of time synchronization errors, the impact of ATIM window size, etc. Due to the page limit, we only show the impact of time synchronization errors in Section IV-B. In brief, Equation (8) presents that the maximum throughput increases as M, N_s, η, or $E[P_d]$ increases, but decreases as t_{max} increases. Therefore, to improve the maximum throughput, the corresponding methods used to change the value of M, N_s, η, $E[P_d]$ and t_{max} can be applied. For example, we can enlarge the data packet size to increase the maximum throughput.

A. Model Validation

We have implemented TMMAC in GloMoSim [23], a scalable discrete event simulator developed by UCLA. We compare the results from the simulation and model to validate the analytical model.

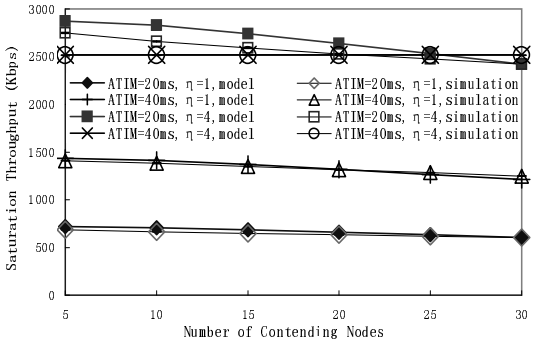


Fig. 3. Saturation throughput: analysis $vs.$ simulation.

Unless otherwise specified, in all our following simulations, including the figures shown in Section VI, the simulation settings are as follows: 3 channels are used to conform to the IEEE 802.11b which has 3 non-overlapping channels; the bit rate is 2Mbps for each channel; the data packet size is 512 bytes; the channel switch delay is set to $80us$ [15]; the maximum time synchronization error is $0.1ms$; the beacon interval is set to $100ms$; all the simulation results in our performance figures are computed from 20 trials, each of which lasts for 50 seconds, and 90% confidence intervals of the results are also shown in each figure.

Figure 3 shows that our analytical model characterizes the throughput of TMMAC very accurately. When the ATIM window size is $20ms$, the throughput from the analysis has up to 6% error compared to that from the simulation. For example, the throughput from the simulation when $ATIM = 20ms$ and $\eta = 1$ is 0.5% to 6% less than that from the analysis with different number of contending nodes. The main reason for this noticeable error is that when a node is near the end of the ATIM window, it gives up transmitting its ATIM packet if it can not be finished within the ATIM window. This changes the number of nodes which are competing for the bandwidth when it is approaching the end of the ATIM window. The impact is reduced when the ATIM window is enlarged. When the ATIM window size is $40ms$, the maximum throughput error between the analysis and simulation is reduced to 2%. Also note that when $ATIM = 40ms$ and $\eta = 4$, the throughput from the simulation is almost the same as that from the analysis and remains constant as the number of contending nodes changes. This is because when $n_{schedule}$ becomes larger than $n_{accommodate}$, the throughput is determined by the maximum bandwidth supported by the communication window.

B. Impact of Time Synchronization Errors

Equation (8) shows that the maximum throughput of TMMAC is affected by the time synchronization error. In this section, we quantify the impact of time synchronization errors on TMMAC based on the analytical model. We set the number of contending nodes to 20 in this analysis.

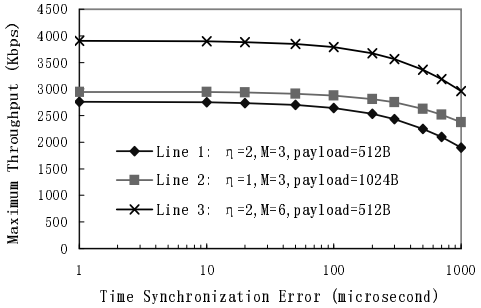


Fig. 4. Maximum throughput $vs.$ time synchronization error.

Figure 4 shows that TMMAC is robust to the time synchronization error. With reasonable time synchronization accuracy, such as $t_{max} < 50us$, the maximum throughput is barely affected by the time synchronization error. For example, the maximum throughput shown in Line 1 is only reduced by 2% when t_{max} is $50us$. Even when the time synchronization error is extremely high, such as $1ms$, the performance of TMMAC remains acceptable. For example, the maximum throughput when $t_{max} = 1ms$ shown in Line 1 still achieves 69% of its optimal maximum throughput when there is no clock skew. Further, we can make TMMAC more robust to large time synchronization errors by either increasing the payload size or the number of channels as shown in Line 2 and Line 3. For example, line 2 shows that if we increase the payload size from 512B to 1024B, we achieve 82% of its optimal maximum throughput even when $t_{max} = 1ms$.

V. Dynamic ATIM Window Adjustment

The maximum throughput in TMMAC is achieved only when the optimal ATIM window size is used. If l_{atim} is different from l_{opt}, it results in bandwidth waste either in the ATIM window or in the communication window. However, there is no fixed l_{opt} in TMMAC which is able to achieve the maximum throughput under all situations. As shown in Equation (7), l_{opt} depends on N_s and η, given that M, l_{beacon} and l_{slot} are fixed. Since the number of contending nodes and η vary over time, the ATIM window size should be changed dynamically to achieve the maximum throughput. Further, it is desirable to use a small ATIM window, when the network is not saturated to save more energy.

Jung et al. [24] propose a scheme to change the ATIM window size dynamically in 802.11 PSM for wireless LANs. However, the same approach can not be applied to TMMAC

because TMMAC introduces channel diversity and TDMA in the communication window. In this section, we present the design for dynamically adjusting the ATIM window size for TMMAC, which helps to improve both the network throughput and energy efficiency.

In TMMAC, each node adjusts its ATIM window size dynamically, allowing different nodes to have different ATIM window sizes. We use a finite set of ATIM window sizes $\{ATIM_1, ..., ATIM_i, ATIM_{i+1}, ..., ATIM_m\}$, in which $ATIM_1$ is the minimal ATIM window size, $ATIM_m$ is the maximal ATIM window size, and $ATIM_{i+1} - ATIM_i = l_{slot}$. To avoid collisions between ATIM control packets and data packets in the default channel, the default channel is never used for data communication in the time slots before $ATIM_m$. However, other channels can be used for data communication in these time slots as long as they are not within a node's current ATIM window. When a node is sending an ATIM control packet, it piggybacks its ATIM window size for the next beacon interval. Thus, the neighboring nodes know its ATIM window size. There are two possibilities when node A wants to send a packet to node B. If node A knows node $B's$ ATIM window size, node A decides whether the negotiation can be finished within $min\{A'sATIMwindow, B'sATIMwindow\}$. If yes, node A sends the ATIM packet to node B. Else, node A waits for the next beacon interval. If node A does not know node $B's$ ATIM window size, node A decides whether the negotiation can be finished within $ATIM_1$. If yes, node A sends the ATIM packet. Else, node A waits for the next beacon interval.

A. Rules for Dynamic ATIM Window Adjustment

In this section, we present the rules used for dynamic ATIM window adjustment in TMMAC. Our rules utilize the properties of TDMA and channel diversity in TMMAC.

We apply different rules based on whether the network is saturated. When the network is saturated, we adjust the ATIM window size to achieve the maximum throughput. When the network is not saturated, our objective is to achieve more power savings. In our scheme, each node maintains a variable φ, which means the number of successful negotiations per time unit. The moving average is applied to compute φ to eliminate stochastic anomalies, which is shown as follows:

$$\varphi_{i+1} = \alpha \times \varphi_i + (1 - \alpha) \times \frac{\upsilon_{i+1}}{l_{atim(i+1)}} \tag{9}$$

Here, φ_{i+1} is the computed average number of successful negotiations per time unit in this beacon interval, φ_i is the value in last beacon interval, υ_{i+1} is the number of successful negotiations in this beacon interval, and $l_{atim(i+1)}$ is the current ATIM window size. To decide whether the network is saturated, we compare φ to N_s. If φ is not smaller, the network is considered to be saturated. Otherwise, it is not saturated. However, N_s changes as the number of contending nodes changes and we do not assume that TMMAC has the knowledge of the number of contending nodes. As a result, we use a conservative value for N_s, which serves as the lower bound for a saturated network.

After deciding whether the network is saturated, the corresponding rules are applied. If the network is saturated, we use the information whether all the available bandwidth in the communication window is scheduled for data communication. If yes, i.e., $P_{schedule} >= P_{accommodate}$, we decrease the ATIM window size by one level to leave more bandwidth for data communication. If not, i.e., $P_{schedule} < P_{accommodate}$, we increase the ATIM window size by one level to leave more bandwidth for negotiation. If the network is not saturated, we decrease the ATIM window size by one level to save more power. There is a special case in which a node does not adopt the ATIM window size computed based on the above rules. If a node does not get the opportunity to broadcast its current ATIM window size in the last beacon interval, i.e., no node knows its current ATIM window size, and it does not have any packets to send in this beacon interval, this node resets its current ATIM window size to $ATIM_1$.

VI. Performance Evaluation

In this section, we compare our scheme with 802.11 DCF which uses a single channel, and MMAC [4] which is a typical multi-channel MAC protocol using a single radio transceiver. MMAC is also implemented in GloMoSim [23]. Unless otherwise specified, beside the simulation settings specified in Section IV-A, we use the following settings: the network area size is $1000m \times 1000m$; constant-bit rate (CBR) traffic is used in the application layer; the source and destination of each flow are randomly chosen; Geographic Forwarding Routing protocol [25] is used in the routing layer; 200 nodes are randomly deployed into this area; the communication range is $250m$; the carrier sense range is $500m$; the minimal ATIM window size in TMMAC is $8.57ms$ and the maximum is $31.43ms$; the length of the ATIM window in MMAC is $20ms$.

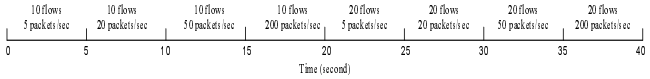
We use the following two metrics to evaluate the performance of TMMAC: 1) Aggregate throughput. Aggregate throughput is the total throughput of all the nodes in the network. 2) Per packet energy. Per packet energy is the value of total energy consumed by the whole network divided by the total number of data packets successfully transmitted.

A. Evaluation of Dynamic ATIM Window Adjustment

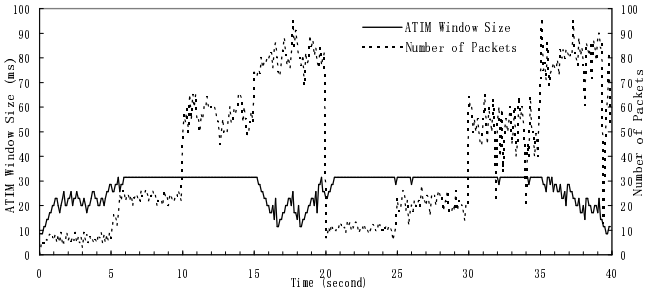
We first evaluate the efficacy of dynamic ATIM window adjustment. We vary both the number of flows and Packet Arrival Rate (PAR) per flow to observe the ATIM window changes. The traffic pattern is shown in Figure 5(a).

We first choose the results of a typical source node in a single run to show the performance, which is plotted in Figure 5(b). The number of data packets transmitted during a beacon interval, which happen in the vicinity of that node is also plotted in that figure. Note that these two curves shown in Figure 5(b) have different units. Because the performance of TMMAC can be significantly improved by using a larger η, which can be achieved by accumulating data packets before negotiation at the expense of transmission delay, we forbid the nodes to accumulate data packets in this set of simulations.

Figure 5(b) shows that while the number of packets transmitted increases as the PAR increases, the ATIM window size changes in a different way based on different traffic patterns. We first explain the ATIM window changes when the number of flows is 10. As we can see from the figure,



(a) The system loads over time



(b) Number of packets transmitted $vs.$ ATIM window size

Fig. 5. Performance evaluation of dynamic ATIM window adjustment.

the ATIM window size increases when the PAR is boosted from 5 packet/sec to 20 packet/sec. The reason why the ATIM window is enlarged is that the increase of the PAR causes each node to compete for the channel more frequently and the ATIM window needs to accommodate more negotiations. After the PAR reaches 20 packet/sec, the ATIM window size remains constant since it has reached the maximum ATIM window size. However, when the PAR increases to 200 packet/sec, the ATIM window size decreases. This is because a high PAR results in a high η, in which case a small number of negotiations utilize all the available bandwidth in the communication window. Therefore, by using a smaller ATIM window, we can reserve more bandwidth for data communication. The similar trend also happens when the number of flows is increased to 20. The only difference is that when the PAR is 5 packet/sec, the ATIM window size reaches the maximum value when the number of flows is 20. With the same PAR per flow, the increase of the number of flows causes more nodes to compete for the channel, which results in a larger ATIM window.

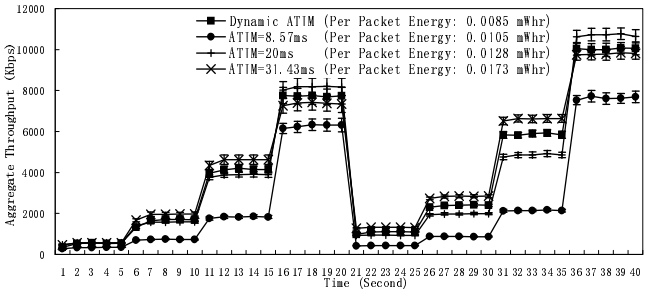


Fig. 6. Aggregated throughput over time.

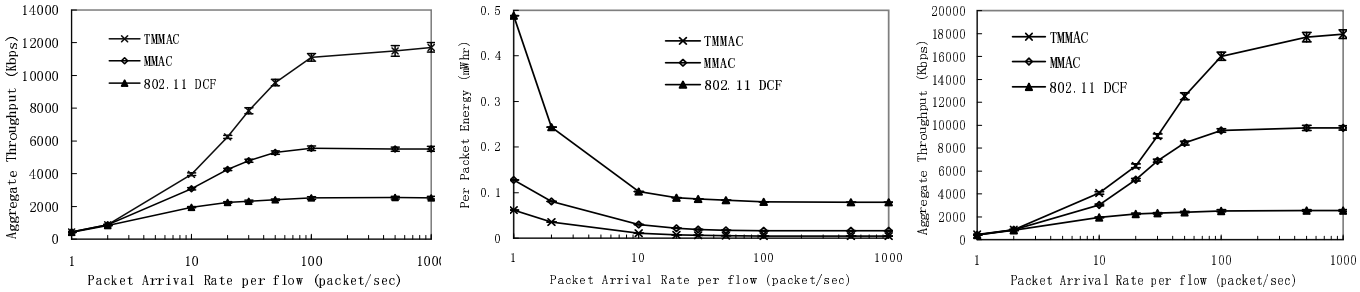
Figure 6 plots the aggregate throughput of TMMAC over time with different ATIM window schemes. The per packet energy consumption of each scheme is also depicted within the parenthesis. We first investigate the performance of TMMAC when the ATIM window size is fixed. TMMAC with ATIM window size $8.57ms$ always achieves the lowest throughput, since its inability to accommodate enough negotiations to fully utilize its communication window. TMMAC with ATIM window size $31.43ms$ achieves the highest throughput when the PAR is below 50 packet/sec, because a larger ATIM window size allows more negotiations, which utilizes the bandwidth in the communication window more efficiently. This is also why the maximum ATIM window size is adopted by the node shown in Figure 5(b), when the PAR is 20 packet/sec and 50 packet/sec. However, its throughput becomes the second worst when the PAR is 200 packet/sec, because it causes bandwidth waste in the ATIM window. TMMAC with ATIM window size $20ms$ achieves the highest throughput when the PAR is 200 packet/sec, because it is close to the optimal ATIM window size with that PAR. This also explains why the ATIM window size shown in Figure 5(b) decreases with the PAR is 200 packet/sec. However, it becomes the second worst when the PAR is below 50 packet/sec, because it does not leave enough bandwidth for negotiation. Different from the fixed ATIM window schemes, whose performance depends on the specific traffic patterns, the performance of the dynamic ATIM window scheme is always comparable to the best performance a fixed ATIM window scheme can achieve, if it does not achieve the best, under different traffic patterns. For example, the dynamic ATIM window scheme always achieves the second best when the PAR is between 20 packet/sec and 200 packet/sec and it even achieves the best when the PAR is 5 packet/sec sometimes. Moreover, the dynamic ATIM window scheme consumes much less per packet energy. Its per packet energy consumption is 66% and 49% of that with ATIM window size $20ms$ and $31.43ms$, respectively. This shows that the dynamic ATIM window scheme adjusts the ATIM window size successfully based on different traffic patterns to achieve higher throughput and lower energy consumption.

B. Performance $vs.$ System Loads

In this section, we vary the PAR of the CBR flows to show the performance of TMMAC, MMAC and 802.11 DCF at different network loads. The number of flows used in this set of simulations is 40. Figure 7(a)-(b) and Figure 7(c) show 3 and 6 channel results, respectively.

Figure 7(a) shows that when the PAR is low, such as 1 or 2 packet/sec, the 3 MAC protocols achieve similar aggregated throughput. However, as the network load increases, TMMAC starts to outperform MMAC and 802.11 DCF. TMMAC achieves 113% more aggregate throughput than MMAC when the network is overloaded, which is 4.5 times that of 802.11 DCF. TMMAC avoids contention based communication during the communication window. Contention based communication adopted by MMAC and 802.11 DCF wastes some bandwidth in backoff, and sending RTS and CTS control packets. Besides, TMMAC is able to dynamically adjust the ATIM window size, which further improves throughput.

Figure 7(b) shows that TMMAC consumes much less per packet energy compared to MMAC and 802.11 DCF. When the PAR is 1 packet/sec, per packet energy consumption in TMMAC is 48% of that in MMAC. However, as the PAR increases, the energy savings in TMMAC becomes more significant. Per packet energy consumption in TMMAC is only 26% of that in MMAC when the network is overloaded. We conclude the following 3 reasons for the low per packet energy consumption in TMMAC. First, TMMAC allows a node to switch to doze mode in a time slot whenever it is not scheduled to transmit or receive a packet. In MMAC, due to the lack of time negotiation,



(a) Aggregate throughput (3 channels) (b) Per packet energy (3 channels) (c) Aggregate throughput (6 channels)

Fig. 7. Performance evaluation with different system loads.

a node needs to stay awake during the whole communication window when it has negotiated to transmit or receive packets. Second, TMMAC dynamically adjusts the ATIM window size based on different traffic patterns to save power. Third, TMMAC achieves much higher aggregate throughput, which further reduces its per packet energy consumption.

Figure 7(c) shows that both the throughput of TMMAC and the throughput of MMAC are improved significantly when the number of channels is increased from 3 to 6. The throughput of TMMAC is 7.07 times that of 802.11 DCF when 6 channels are adopted. However, the performance gain of TMMAC over MMAC is reduced. When the network is overloaded, TMMAC achieves 84% more aggregate throughput than MMAC, lower than 113%. The main reason for the reduced performance gain is that in MMAC when the channel number increases, the contention in each channel becomes smaller because the traffic is distributed to different channels. With less contention in each channel, we obtain smaller performance gain of the TDMA scheduling adopted by TMMAC over the contention based communication adopted by MMAC. However, as long as the contention exists, TMMAC always achieves higher throughput than MMAC.

VII. Conclusions

In this paper, we present the TMMAC protocol, which is an energy efficient multi-channel MAC protocol using a single half duplex radio transceiver. TMMAC requires time synchronization in the network and divides time into fixed beacon intervals. Nodes that have packets to transmit negotiate which channels and which time slots to use for data communication with their destinations during the ATIM window. This two-dimensional negotiation enables TMMAC to exploit the advantage of both multiple channels and TDMA in an efficient way. Further, TMMAC is able to support broadcast in an effective way and is highly power-efficient. In this paper, we also propose an accurate analytical model for TMMAC and present a novel scheme to dynamically adjust the ATIM window size based on different traffic patterns. From our performance evaluation, TMMAC achieves up to 113% higher communication throughput while consuming 74% less per packet energy over the state-of-the-art multi-channel MAC protocols using a single half-duplex radio transceiver.

References

[1] 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 *ISPAN*, 2000.

[2] A. Raniwala and T. Chiueh, "Architecture and Algorithm for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *IEEE InfoCom*, 2005.

[3] A. Adya, P. Bahl, J. Padhye, A.Wolman, and L. Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks," in *IEEE Broadnets*, 2004.

[4] J. So and N. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver," in *ACM MobiHoc*, 2004.

[5] F. Fitzek, D. Angelini, G. Mazzini, and M. Zorzi, "Design and Performance of an Enhanced IEEE 802.11 MAC Protocol for Multihop Coverage Extension," *IEEE Wireless Communications*, 2003.

[6] J. Li, Z. J. Haas, M. Sheng, and Y. Chen, "Performance Evaluation of Modified IEEE 802.11 MAC for Multi-Channel Multi-Hop Ad Hoc Network," in *IEEE AINA*, 2003.

[7] N. Jain, S. R. Das, and A. Nasipuri, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks," in *IEEE IC3N*, 2001.

[8] A. Tzamaloukas and J.J. Garcia-Luna-Aceves, "A Receiver-Initiated Collision-Avoidance Protocol for Multi-Channel Networks," in *IEEE InfoCom*, 2001.

[9] Z. Tang and J.J. Garcia-Luna-Aceves, "Hop-Reservation Multiple Access (HRMA) for Ad-Hoc Networks," in *IEEE InfoCom*, 1999.

[10] A. Tyamaloukas and J.J. Garcia-Luna-Aceves, "Channel-Hopping Multiple Access," in *IEEE ICC*, 2000.

[11] J. Deng and Z. Haas, "Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks," in *IEEE ICUPC*, 1998.

[12] A. Nasipuri, J. Zhuang, and S. R. Das, "A Multichannel CSMA MAC Protocol for Multihop Wireless Networks," in *IEEE WCNC*, 1999.

[13] A. Nasipuri and S. R. Das, "Multichannel CSMA with Signal Powerbased Channel Selection for Multihop Wireless Networks," in *IEEE VTC*, 2000.

[14] IEEE 802.11 Working Group, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 1997.

[15] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," in *ACM MobiCom*, 2004.

[16] G. Zhou, C. Huang, T. Yan, T. He, J.A. Stankovic, and T. Abdelzaher, "MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks," in *IEEE InfoCom*, 2006.

[17] I. Chlamtac and S. Kutten, "A Spatial-Reuse TDMA/FDMA for Mobile Multi-hop Radio Networks," in *IEEE InfoCom*, 1985.

[18] I. Chlamtac and A. Farago, "Making Transmission Schedules Immune to Topology Changes in Multi-hop Packet Radio Networks," *IEEE/ACM Transactions on Networking*, 1994.

[19] L. Bao and J.J. Garcia-Luna-Aceves, "A New Approach to Channel Access Scheduling for Ad Hoc Networks," in *ACM MobiCom*, 2001.

[20] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," in *ACM SenSys*, 2003.

[21] I.A. Getting, "The Global Positioning System," *IEEE Spectrum*, 1993.

[22] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE JSAC*, 2000.

[23] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," in *PADS*, 1998.

[24] E.S. Jung and N.H. Vaidya, "An Energy Efficient MAC Protocol for Wireless LANs," in *IEEE InfoCom*, 2002.

[25] B. Karp, *Geographic Routing for Wireless Networks*, Ph.D. thesis, Harvard University, 2000.