Load-Aware Channel Hopping Protocol Design for Mobile Ad Hoc Networks

Chih-Min Chao[1], Hsien-Chen Tsai, and Chao-Ying Huang
Department of Computer Science and Engineering
National Taiwan Ocean University, Taiwan

[1] E-mail: cmchao@ntou.edu.tw, Tel: +886-2-24622192 ext 6651, Fax: +886-2-24623249

Abstract—**Using multiple channels in wireless networks improves spatial reuse and reduces collision probability and thus enhances network throughput. Designing a multi-channel MAC protocol is challenging because multi-channel-specific issues such as channel assignment, the multi-channel hidden terminal problem, and the missing receiver problem, must be solved. Most existing multi-channel MAC protocols suffer from either higher hardware cost or poor throughput. Some channel hopping multi-channel protocols achieve pretty good performance in certain situations but fail to adjust their channel hopping mechanisms according to varied traffic loads. In this paper, we propose a Load-Aware Channel Hopping MAC protocol (LACH) that solves all the multi-channel-specific problems mentioned above. LACH enables nodes to dynamically adjust their schedules based on their traffic loads. In addition to load awareness, LACH has several other attractive features: (1) Each node is equipped with a single half-duplex transceiver. (2) Each node's initial hopping sequence is generated by its ID. Knowing the neighbor nodes' IDs, a node can calculate its neighbors' initial channel hopping sequences without control packet exchanges. (3) Nodes can be evenly distributed among available channels. Through performance analysis, simulations, and real system implementation, we verify that LACH is a promising protocol suitable for a network with time-varied traffic loads.**

Keywords: Ad Hoc Networks, Multichannel MAC Protocols, Quorum Systems.

I. Introduction

A Mobile Ad Hoc Network (MANET) consists of a collection of mobile nodes communicating with each other without the support of base stations. The IEEE 802.11 uses only a single channel in MAC layer although multiple channels are supported in physical layer. In a heavy-loaded network, using multiple channels can increase spatial reuse and network throughput. Therefore, a single-channel MAC mechanism such as 802.11 DCF is inefficient in a multi-channel environment wherein nodes may dynamically switch among available channels.

Designing a multi-channel MAC protocol for MANETs is challenging. A good solution should solve the multi-channel hidden terminal problem, the missing receiver problem, and the control channel bottleneck problem. The multi-channel hidden terminal problem exists because the control packets sent on a particular channel, say channel 1, are unable to notify neighbors resided on other channels. These neighbors become potential interference sources if they switch to channel 1 afterwards. The missing receiver problem happens when a sender fails to deliver packets to its intended receiver because they do not switch to the same channel. The control channel bottleneck problem means that a particular control channel (or time interval) becomes a bottleneck because it is dedicated to control packet exchanges.

Chih-Min Chao, Hsien-Chen Tsai, and Chao-Ying Huang are with the National Taiwan Ocean University, Taiwan, 20224. E-mail: cmchao, m97570007, 10057032@ntou.edu.tw

TABLE I
Classification of multi-channel MAC protocols.

	Single-Rendezvous	Multi-Rendezvous
Multi-Transceiver	[1]–[5]	[6]–[12]
Single-Transceiver	[13]–[21]	[22]–[26], ours

There exist several multi-channel MAC protocols for wireless ad hoc networks. We can classify them according to the following two factors:

- *single-/multi-transceiver*: Whether a node is equipped with multiple transceivers or not.
- *single-/multi-rendezvous*: Whether multiple transmission pairs can always achieve handshaking simultaneously or not.

Based on this classification, we categorize existing multi-channel protocols and the protocol proposed in this paper in Table I. A more detailed review of these multi-channel solutions can be found in Section II.

The multi-rendezvous protocols perform better than the single-rendezvous ones [22]–[25]. However, existing multi-rendezvous solutions may be suboptimal because they do not consider each node's traffic load. In this paper, we proposed a load-aware channel hopping MAC protocol for MANETs, denoted as LACH. LACH belongs to the single-transceiver, multi-rendezvous class. It utilizes the same concept of default/switching slots where each node waits for receiving during its default slots [25]. This enables LACH to use a single transceiver to emulate the solutions in the multi-transceiver, multi-rendezvous class. LACH utilizes latin squares to evenly distribute network loads to different channels. A sender running LACH is guaranteed to meet its intended receiver within a finite time span. Each node running LACH can also dynamically adjust the number of default slots based on its own traffic load.

We have extended our previous work [27] in several aspects: (1) The LACH default slot adjustment mechanism is enhanced such that a node's default slots are determined by all of its senders and intended receivers. (2) An analysis section is added to investigate the transmission time of LACH. (3) More simulations are conducted to verify the performance of LACH. (4) A real system implementation is applied to verify if LACH works well in real world.

The rest of the paper is organized as follows. We review some multi-channel MAC protocols in Section II. Our protocol is presented in Section III. Section IV analyzes the protocols. Simulation results are given in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

A number of multi-channel MAC protocols use at least two transceivers to deal with the channel assignment problem [1]–[12]. In the multi-transceiver, single-rendezvous protocols [1]–[5], one transceiver in each node is tuned to a common control channel to negotiate the channel for later data transmissions. The other transceivers are used for data transmissions. With a dedicated control transceiver and a dedicated control channel, these solutions avoid the multi-channel hidden terminal problem and the missing receiver problem. However, they suffer from control channel bottleneck. To avoid such a bottleneck, some channel allocation solutions avoid using a common control channel [7]–[12]. In these protocols, each node uses one of its transceivers to its fixed channel to receive transmission requests. The other transceivers can be switched to any channel to initiate a transmission. Another scheme [6] uses two transceivers; one performs a fast channel hopping and the other performs a slow channel hopping. The fast and slow hopping transceiver is used for transmission and reception, respectively. These protocols belongs to the multi-rendezvous solutions because handshaking of different transmission pairs can accomplish simultaneously at the receivers' fixed channels. A drawback of these multi-transceiver protocols is the increased hardware cost.

To reduce hardware cost, many single-transceiver, single-rendezvous protocols [13]–[21] have also been proposed. Some of them [14], [16]–[18] employ a dedicated control channel to exchange control messages. To avoid the multi-channel hidden terminal problem, a sender usually conservatively waits longer before a transmission. These solutions also suffer from the control channel bottleneck problem. Similar to the ATIM window concept in IEEE 802.11 power saving mode, some other methods [15], [19], [20] utilize a common control period. During this period, all of the nodes switch to a common control channel and contend for channel negotiation. These protocols avoid using a dedicated control channel but often encounter control channel bottleneck during the common control period. To enhance channel utilization, some protocols belonging to the single-transceiver, multi-rendezvous class use one transceiver to enable multi-rendezvous [22]–[26]. These protocols employ a channel hopping scheme such that nodes can switch among different channels. Two nodes can communicate with each other if they switch to the same channel simultaneously. The core of these protocols is to design a channel hopping algorithm such that different nodes can switch to the same channel at the same time. Since solutions in the single-transceiver, multi-rendezvous class are desirable, the protocols belonging to this class are reviewed more detailed in the following.

Bian et al. proposed two kinds of QCH systems [23]. In synchronous QCH systems, time is divided into a series of frames and m channels are selected as the rendezvous channels. One distinct rendezvous channel is assigned to m consecutive frames. A frame consists of k slots and each node chooses a quorum under Z_k. During their quorum slots, nodes turn their transceivers to the rendezvous channel associated with the frame; during the other slots, nodes switch to a randomly selected channel. In asynchronous QCH systems, nodes select two cyclic quorum systems to generate their channel hopping sequences. Nodes use one channel for each of the two quorum systems. Both synchronous and asynchronous QCH systems guarantee that any two nodes hop to the same channel at some slot. However, the QCH system is inflexible. In synchronous QCH, nodes are guaranteed to rendezvous only in a single channel (the rendezvous channel). In asynchronous QCH, at most two channels can be utilized simultaneously.

In McMAC [24], a node independently generates its own channel hopping sequence via a common generator with its ID as the seed. A sender follows its receiver's hopping sequence to send the packets. A sender can easily obtain the hopping sequence of its intended receiver because nodes use a common generator. This solution suffers from the missing receiver problem since the receiver may change its channel hopping sequence.

EM-MAC is a receiver-initiated multi-channel asynchronous MAC protocol for wireless sensor networks [26]. A node running EM-MAC independently selects its wake-up time and the wake-up channel according to a pseudo random function which is known to all of the nodes. At each wake-up time slot, a node also sends a wake-up beacon which indicates the unavailable channels (blacklist). A node is able to predict the wake-up channel and wake-up time of the intended receiver and thus can wake up at the right time to initiate a data transmission. EM-MAC also suffers from the missing receiver problem because a node's available channel changes (contained in the wake-up beacon) may not be correctly received by the other nodes.

A node running SSCH [22] independently generates its own channel hopping sequence. The channel hopping scheme guarantees that two nodes hop to the same channel simultaneously at least once in a cycle. Each node's hopping sequence is determined by a set of $(channel, seed)$ pairs. If there are m available channels in the network, $channel$ is an integer between 0 and $m-1$ and $seed$ is an integer between 1 and $m-1$. The next channel to be switched to is obtained by adding $channel$ and $seed$ (mod m). At the end of each cycle, there is a parity slot to guarantee that two nodes hop to the same channel concurrently. The channel being used in the parity is determined by the $seed$ of the first pair. In each cycle,

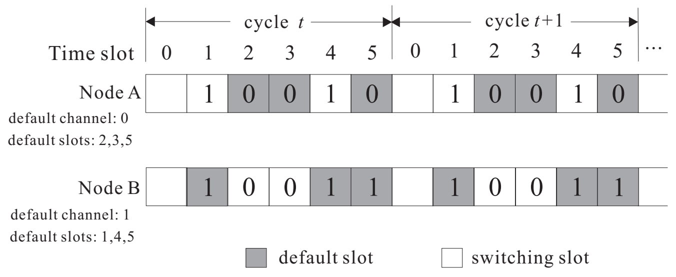

Fig. 1. An example of CQM operation under Z_6 with 3 available channels

nodes running SSCH exchange their hopping sequences with each other. To increase transmission opportunities, a sender may change its hopping schedule to its intended receiver's.

CQM [25] uses the concept of quorum systems and defines default/switching slots to emulate the multi-transceiver, multi-rendezvous solutions. In CQM, time is divided into a series of cycles and each cycle is divided into several time slots. The time slots in each cycle are partitioned into the *default slots* and the *switching slots*. In default slots, a node stays on its *default channel*, waiting for transmission requests. In switching slots, a node may tune to its intended receiver's default channel which is determined by the node ID of the receiver. To solve the missing receiver problem, a cyclic quorum is selected to identify a node's default slots. Fig. 1 is an example of CQM operation for nodes A and B. The number in each slot represents the channel to be switched to. If node A has packets pending to node B, in each cycle, node A can switch to node B's default channel at slots 1 or 4. Similarly, node B can switch to channel 0 at slots 2 and 3 to send packets to node A. CQM uses cyclic quorum systems in a novel way such that a sender is guaranteed to meet its intended receiver. CQM also outperforms the other single-transceiver, multi-rendezvous mechanisms in most situations. However, the performance of CQM can still be improved because it does not adapt to traffic changes.

Several single-transceiver, multi-rendezvous MAC protocols for wireless ad hoc networks are compared in Table II. The QCH protocol is the only mechanism that has channel limitation since at most two channels are guaranteed to be utilized at the same time slot. For the control overhead issue, EM-MAC and SSCH produce heavy burden since nodes running EM-MAC or SSCH have to periodically exchange their blacklists or channel hopping schedules, respectively. In LACH, nodes also have to broadcast their default/switching allocation adjustment information; however, the amount and frequency is less than EM-MAC and SSCH and thus we classify it as having medium overhead. Most solutions solve the three problems mentioned in Section I while McMAC, EM-MAC, and SSCH still have the missing receiver problem. Lastly, only LACH is a load awareness mechanism which provides great flexibility when traffic load changes.

TABLE II
COMPARISON OF SINGLE-TRANSCEIVER, MULTI-RENDEZVOUS PROTOCOLS

	Protocols					
	McMAC	EM-MAC	SSCH	QCH	CQM	LACH
Channel Limitation	N	N	N	Y	N	N
Control Overhead	L	H	H	L	L	M
Multi-Channel Hidden Terminal	N	N	N	N	N	N
Missing Receiver	Y	Y	Y	N	N	N
Control Channel Bottleneck	N	N	N	N	N	N
Load Awareness	N	N	N	N	N	Y

III. The Proposed Load-Aware Channel Hopping Protocol

Similar to CQM, the IEEE 802.11 DCF is adopted as the medium access scheme. This enables nodes running LACH coexist with those running IEEE 802.11. A node running LACH is able to adjust its channel hopping schedule individually according to its own traffic load. LACH uses latin squares to achieve load sharing and multiple rendezvous. In the following, we first introduce latin squares and then describe the proposed LACH protocol.

A. Latin Squares

latin squares have been extensively used for wireless MAC protocol design [28], [29], network coding scheme design [30], and effective interleaver design of network communication devices [31]. A latin square is defined as follows.

Definition 1. A latin square is an $n \times n$ matrix containing n different symbols such that each symbol occurs exactly once in any column and row.

For example, the square A shown below is a 4×4 latin square with symbols 1, 2, 3, and 4:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

B. The LACH protocol

The assumptions we made in this paper are listed below:

- A total of m equal-bandwidth channels are available in the network.
- Each node is equipped with a single half-duplex transceiver which can be dynamically switched to any channel.
- Each node has a unique ID (the ID of node i is i) and knows all of its neighbors' IDs.
- Nodes are time synchronized. Similar to SSCH, McMAC, and CQM, synchronization among nodes is needed in LACH. Synchronization is not an easy task in MANETs.

Fortunately, there exist several schemes that can handle the problem [32], [33].

The LACH protocol consists of two major parts: 1) Initial time/channel allocation and 2) default slot adjustment. A node running LACH first determines its initial channel hopping sequence by the initial time/channel allocation mechanism. Then, using the default slot adjustment mechanism, a node can change its channel hopping schedule to adapt to its load changes. That is, the initial time/channel allocation mechanism is executed once by each node at the network initialization phase while the default slot adjustment mechanism is executed periodically afterwards.

C. Initial time/channel allocation

To enable a communication, a transmission pair must switch to the same channel simultaneously. That is, the most critical task for a multi-channel MAC protocol is to coordinate the time/channel usage for all the nodes. To handle this task, in LACH, time is divided into a series of cycles, each of which is further divided into n time slots. The value of n is determined by the size of the latin square being used. For example, if a 5×5 square is used, the value of n is 5. A node can assign each slot to be either a *default slot* or a *switching slot*. In a default slot, a node must stay on its *default channel*, waiting for transmission requests. In a switching slot, a node can switch to its intended receiver's default channel to initiate a transmission. In a switching slot, no channel switching will be applied if a node has no pending packets. That is, the channel being used in that switching slot is the same one being used in the previous slot. Each node's default slots are determined by its ID and an $n \times n$ latin square. The LACH can be implemented based on any latin square. Since IEEE 802.11 standard provides at most 13 channels, Without loss of generality, we use a 13×13 standardized latin square with symbols 0 to 12 to illustrate the operation of LACH. We use a latin square with the first row labelled from 0 to $n-1$ from left to right. The second row is one position right-rotated and the others can be obtained by analogy.

To allocate the default slots and the default channel, each node i individually selects a row, R_i, and a symbol, SB_i, of the latin square as follows.

$$R_i = i \text{ (mod } n\text{)},$$
$$SB_i = (i + \lfloor i/n \rfloor) \text{ (mod } n\text{)}.$$

Based on these two parameters, each node's *initial default slot*, IDS_i, and *initial default channel*, IDC_i, are determined by

$$IDS_i = (SB_i + R_i) \text{ (mod } n\text{)},$$
$$IDC_i = SB_i \text{ (mod } m\text{)}.$$

An example of LACH initial default slot/channel allocation with $m = 13$ is shown in Fig. 2. Slot 0 and channel 0 are assigned to node 0 as its initial default slot and initial default channel, respectively. For node 1, we have $IDC_1 = 1$ and $IDS_1 = 1 + 1 = 2$. The allocations of some other nodes are also showed in Fig. 2. If only 13 nodes with consecutive IDs are in the network, each of them will have a unique initial default channel and a unique initial default slot. When the

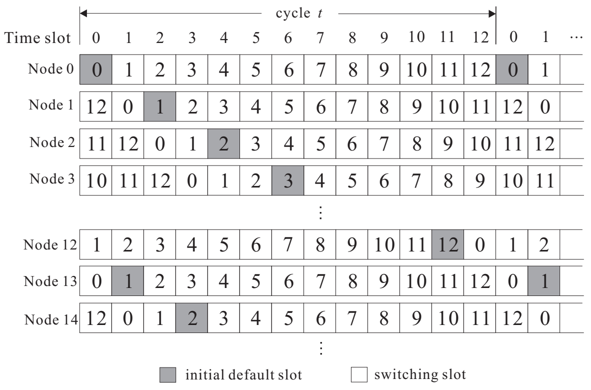
cycle t
Time slot 0 1 2 3 4 5 6 7 8 9 10 11 12 0 1 ···
Node 0: 0 1 2 3 4 5 6 7 8 9 10 11 12 0 1
Node 1: 12 0 1 2 3 4 5 6 7 8 9 10 11 12 0
Node 2: 11 12 0 1 2 3 4 5 6 7 8 9 10 11 12
Node 3: 10 11 12 0 1 2 3 4 5 6 7 8 9 10 11
⋮
Node 12: 1 2 3 4 5 6 7 8 9 10 11 12 0 1 2
Node 13: 0 1 2 3 4 5 6 7 8 9 10 11 12 0 1
Node 14: 12 0 1 2 3 4 5 6 7 8 9 10 11 12 0
⋮
initial default slot
switching slot

Fig. 2. An example for initial default slot/channel assignment of LACH

number of nodes increases, it is inevitable that some nodes have the same default slot/channel. In such situations, LACH will distribute nodes' default channels and default slots as even as possible. The time/channel allocation mechanism needs only each node's ID and the $n \times n$ latin square. Once neighbors are discovered, each node can individually calculate any of its neighbors' initial default slot/channel.

The effectiveness of the LACH protocol depends on the overlapping of the sender's switching slots and the receiver's default slots. If the sender and the receiver do not share the same initial default slot, LACH guarantees such overlapping. We justify the correctness of this mechanism by the following theorem. Let ISS_i represents the set of node i's initial switching slots. The initial default slot/channel allocation scheme implies $IDS_i \cup ISS_i = Z_n$ and $IDS_i \cap ISS_i = \varnothing$.

Theorem 1. *Given two nodes i and j running LACH with an $n \times n$ latin square. If $IDS_i \neq IDS_j$ then $IDS_i \in ISS_j$ and $IDS_j \in ISS_i$.*

Proof: Assume that $IDS_i \notin ISS_j$, which means $IDS_i \in IDS_j$ and $IDS_i \cap IDS_j \neq \varnothing$. We have $IDS_i = IDS_j$ since both IDS_i and IDS_j have only one element. This contradicts to the premise $IDS_i \neq IDS_j$ and we have $IDS_i \in ISS_j$.
Using the same method, $IDS_j \in ISS_i$ can also be proved. ■

The channel/slot allocation scheme of LACH has several advantages. First, if nodes' IDs are randomly distributed, all the nodes' initial channels are distributed among all available channels and thus, traffic loads can be evenly shared. Second, the overlapping default/switching slots for different pairs are scattered, which reduces the packet collision possibility.

It should be noted that two nodes having the same initial default slot may not be able to communicate with each other. This can be solved in the network layer: a common neighbor node with a different initial default slot can help to relay their traffic. In such a case, route discovery between the common node and the two nodes must be applied. If such route discovery fails, one of the nodes, say the one with smaller ID, can temporarily change its initial default slot. Without loss of generality, we assume that any sender and its intended receiver have different initial default slots in this paper to simplify our description.

D. *Default slot adjustment*

In LACH, each node has only one default slot initially. To achieve load awareness, nodes must dynamically adjust the number of default slots based on their traffic loads. That is, some *extended default slots* should be assigned to a node when its traffic load increases. When handling the extended default slot allocation, two issues must be addressed: *How many* and *where* should these extended default slots be allocated. To handle these two issues, our idea is to adjust the number of default slots according to the utilization of the default and switching slots. Specifically, the number of default slots is increased/decreased if the average utilization of the default slots is higher/lower than that of the switching slots. To avoid ping-pong effect, we define a threshold T for the utilization comparison. To facilitate the operation of LACH, each node i broadcasts its default slot allocation of cycle $t+1$ during cycle t at slot IDS_i through channel IDC_i, in the form of an n-bit bitmap. A bit in the bitmap is set if the corresponding slot is a default slot. For example, in Fig. 2, if no extended default slot is needed for cycle $t+1$, node 0 will broadcast its schedule (1000000000000) at slot 0 of cycle t through channel 0. The default slot allocation of cycle $t+1$ is obtained at the end of cycle $t-1$ based on the utilization of that cycle. The number of default slots at cycle $t+1$, denoted as $N_{ds}(t+1)$, is adjusted as follows.

$$N_{ds}(t+1) = \begin{cases} \min(n-1, N_{ds}(t-1) + \lfloor \frac{\overline{U_d} - \overline{U_s}}{T} \rfloor) \\ \qquad\qquad\qquad , \text{if } (\overline{U_d} - \overline{U_s}) > T \\ \max(1, N_{ds}(t-1) - \lfloor \frac{\overline{U_s} - \overline{U_d}}{T} \rfloor) \\ \qquad\qquad\qquad , \text{if } (\overline{U_d} - \overline{U_s}) < T \\ N_{ds}(t-1) \qquad , \text{otherwise.} \end{cases} \tag{1}$$

where $\overline{U_d}$ and $\overline{U_s}$ is the average utilization of the default slots and the switching slots, respectively. The number of default slots is increased/decreased if the utilization of the default slots is larger/smaller than that of the switching slots by T. Otherwise, the number of default slots remains unchanged. LACH adjusts the number of default slots proportional to the utilization difference such that a node can adopt a proper schedule faster. A constraint of the default slot adjustment is that there must be at least one default slot and one switching slot in every cycle to enable nodes to send and receive in each cycle. The threshold T is a system parameter wherein a smaller value implies a sensitive slot adjustment. The best setting of T may vary for different scenarios.

When the number of default slots is determined, the next task is to decide where the extended default slots should be located. In LACH, a priority scheme is used to select these additional default slots. The senders of node i for the last few cycles and the nodes having pending packets in i should be considered in this priority scheme. Let nodes s and r be one of such senders and one of the intended receivers of node i, respectively. To select node i's extended default slots of cycle $t+1$, the following principles should be followed:

1) The time slot where node r's initial default slot is located should be avoided.
2) The time slot where node s's initial default slot is located is preferred to be avoided.
3) The time slots where node r's extended default slots are located at cycle t is preferred to be avoided.
4) The time slots where node i's extended default slots are located at cycle t are preferred to be retained.

These principles are feasible for node i because the initial default slots for nodes s and r can be obtained through their IDs. The extended defaults slots of r is also available because each node broadcasts its schedule of default slots every cycle. Having precise channel hopping schedules of node i's neighbors allows node i to make the best decision. However, when the precise information is unavailable, these principles still help for node i to determine its extended default slots. To realize these principles, the priority of a slot other than node i's initial default slot is

- set to $-\infty$ if it is node r's initial default slot.
- subtracted by two if it is node s's initial default slot.
- subtracted by two if it is one of the node r's extended default slots at cycle t.
- added by one if it is one of node i's extended default slots at cycle t.

These priority setting rules are applied for all i's senders and intended receivers. If multiple principles are matched for a slot, all the matched principles will be executed. Slots with the highest priority values will be selected as extended default slots while random selection is used to break ties. The channel being used at the extended default slots is the one indicated by the corresponding latin square symbol (mod m). For example, in Fig. 2, if node 1 selects slots 5 and 8 as its extended default slots, the channels being used will be 4 and 7, respectively.

Fig. 3 is an example of the LACH operation using seven channels and a 7×7 latin square. The number in each slot stands for the channel a node should be switched to. Assume that nodes 0 and 1 have pending packets to node 2 which

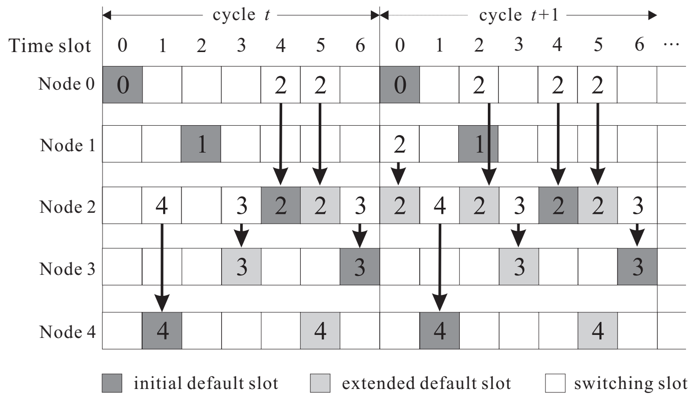

Fig. 3. An example for LACH extended default slot allocation

in turn want to transmit to nodes 3 and 4. The schedule of node 2 for cycle $t-1$ is (0000110). Also assume that node 2 wants to increase two default slots after the utilization comparison at the end of cycle $t-1$. That is, a total of 3 extended default slots will be allocated. After applying the priority setting rules described above, the priorities of the seven slots are $(-2, -\infty, -2, -2, -, -1, -\infty)$ at the end of cycle $t-1$. This means slots 0, 2, and 5 will be chosen as default slots. To clarify the benefit of LACH, we use an arrow in the figure to represent a possible communication. We can see that a proper allocation of extended default slots produces more communication chances at cycle $t+1$ when compared with cycle t.

Based on the selection principles, a node's extended default slots will not overlap with the initial default slots of its intended receivers and thus, the effectiveness of LACH is maintained.

Providing broadcast is not easy for channel hopping protocols. An intuitive solution is to send the packet on each channel separately [22]. This produces longer delay and higher traffic contention. A possible way for LACH to support fast broadcast and to allow new nodes joining the network is to arrange a broadcast slot periodically. For example, a broadcast slot can be allocated once per second (which is equivalent to allocate a broadcast slot every 8 cycles when using a 13 $\times$ 13 latin square). The frequency of broadcast slot depends on the ratio of broadcast traffic and the amount of new nodes. This frequency should be identical for all nodes in the same network.

IV. Performance Analysis

In this section, the time needed for CQM and LACH to deliver a burst of M packets is analyzed. Since we investigate the effect of different channel allocation and scheduling protocols, factors such as interference, transmission collisions, and packet loss are excluded in the analysis. We consider an environment consisting of two nodes A and B while node A has a burst of traffic to B.

In this analysis, a cyclic quorum system $Q = \{G_0, G_1, G_2, G_3, G_4, G_5\}$ generated by the different set $D = \{0, 1, 3\}$ under Z_6 with $G_0 = D$ is used for CQM. Without loss of generality, when running CQM, nodes A and B determine their channel hopping sequences with G_0 and G_1, respectively. The corresponding schedule for nodes A and B is 110100 and 011010, respectively. When running LACH, a 6×6 latin square is employed. Node A selects row 0 and symbol 0 as its channel hopping sequence and initial default slot, respectively. Similarly, node B selects row 1 and symbol 1 as its channel hopping sequence and initial default slot, respectively. The corresponding schedule for nodes A and B is 100000 and 001000, respectively.

Let N be the maximum number of packets a node can send for each rendezvous and R be the number of rendezvous for two nodes in a cycle. The number of cycles needed for node A to deliver a burst of M packets when running CQM, denoted by $P_{CQM}(M)$, is given by

$$P_{CQM}(M) = \lceil \frac{M/N}{R} \rceil$$

Let D be the maximum number of default slots in LACH. The least number of cycles needed for node A to deliver a burst of M packets when running LACH, denoted by $P_{LACH}(M)$, is

$$P_{LACH}(M) = \begin{cases} \lceil M/N \rceil & \text{, if } 0 \le M \le 2N \\ \lceil \frac{(M-2N)/N}{D} \rceil + 2 & \text{, if } M > 2N \end{cases}$$

Nodes running LACH are load awareness but they take two cycles to update their schedules. In the first two cycles, nodes A and B meet once per cycle and therefore a total of $2N$ packets can be transmitted. When M is larger than $2N$, at least three cycles are needed to transmit the burst. After the first two cycles, the number of packets to be delivered is M-$2N$ and the maximum number of rendezvous per cycle between nodes A and B is D.

Here we use an example to illustrate the calculation of $P_{CQM}(M)$ and $P_{LACH}(M)$. Suppose that node A has a burst of 200 packets to B and one packet can be sent in each rendezvous (N = 1). Also, nodes A and B meet twice per cycle (R = 2). In such an environment, $P_{CQM}(200)$ can be calculated as

$$P_{CQM}(200) = \lceil \frac{200/1}{2} \rceil = 100$$

For LACH, since M is bigger than $2N$, nodes A and B will adjust their schedule at the end of cycle 0. The value of $N_{ds}(2)$ for nodes A and B is 1 and 5, respectively. This means that, for cycle 2, the schedule of A remains unchanged while the schedule of B will be changed to 011111. Therefore, $P_{LACH}(200)$ can be calculated as

$$P_{LACH}(200) = \lceil \frac{(200-2)/1}{5} \rceil + 2 = 42$$

This example shows the advantage of using dynamic scheduling: $P_{LACH}(M)$ is much smaller than $P_{CQM}(M)$.

We have also observe the effect of different values of M and N. First we observe the impact of varied M. The value of N is set to four, which is also the value used in our simulations. As shown in Fig. 4, LACH consistently uses shorter transmission times in all values of M when compared to CQM. The average increased transmission time per adding packet for LACH and CQM is 0.05 and 0.125 cycles, respectively.

We have also investigated the impact of values of N when the value of M is set to 200. The results can be found in Fig. 5. A larger N means more packets can be transmitted in a rendezvous and thus, a shorter transmission time can be found. Again, nodes running LACH can deliver a burst of traffic much faster than nodes running CQM. This verifies the necessity of using a dynamic scheduling scheme.

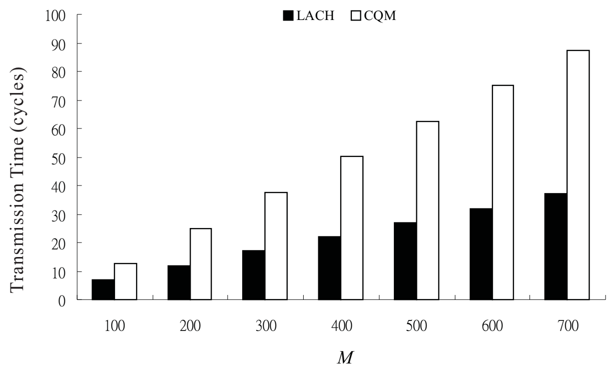

Fig. 4. Transmission time needed in different values of M

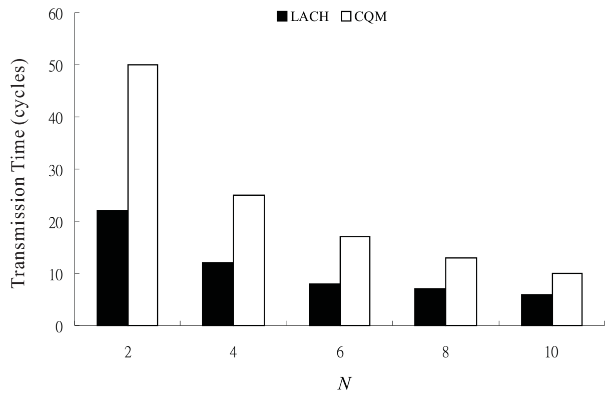

Fig. 5. Transmission time needed in different values of N

V. Performance Evaluation

A. Simulation Setup

We have implemented a simulator to evaluate the performance of the proposed LACH protocol. Three representative multi-channel MAC protocols: CQM, SSCH, and McMAC, were also implemented for comparison purposes. For SSCH, mechanisms such as the dynamically schedule switching, per-neighbor FIFO queues, receiving slots, once-per-slot schedule broadcasting for each node, etc. have been implemented. Whenever a sender wants to change its own schedule, one non-receiving slot is changed to the receiver's schedule. If all slots are receiving slots, any slot can be changed. In our implementation, if multiple slots are allowed to be changed, we always select the next available slot. For McMAC, mechanisms including the linear congruential generator, the dynamically schedule switching (with probability $P_{deviate}$), per-neighbor FIFO queues, etc. have been implemented. Each point in the figures is an average of 10 simulation runs with each simulating 50 seconds. The confidence level shown in the figures was at 95% with the confidence interval of ($\bar{X} - 1.96\sigma/3.16$, $\bar{X} + 1.96\sigma/3.16$), where $\bar{X}$ is the mean

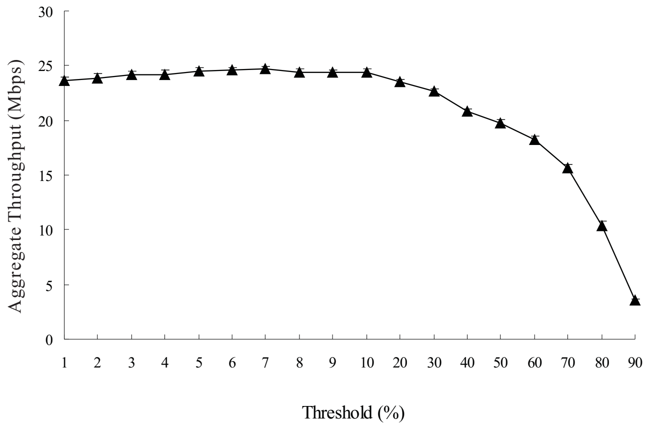

Fig. 6. The impact of threshold

and σ is the standard deviation of the samples.

In the simulations, a total of 100 nodes were uniformly deployed in an area of 800 m $\times$ 800 m. The transmission range is 250 m if not otherwise specified. Each node may act as a sender which randomly selects a one-hop neighbor as its destination. We simulated the bursty traffic model where each node has a traffic arrival probability uniformly distributed between 0 and 1 per second. Each burst of traffic consists of a number of 512-byte packets. The burst length is uniformly distributed between 200 and 300 packets. The number of available channels is 3, 5, 7, 11, or 13; each channel has a bandwidth of 2 Mbps. A time slot is set to 10 ms which means that the maximum number of packets a node can transmit for each rendezvous is four. For LACH, a 13 $\times$ 13 latin square was employed. For CQM, a cyclic quorum system under Z_6 is implemented. The number of (channel, seed) pairs in SSCH is set to 4, and the initial values of all pairs are randomly chosen. For McMAC, the default discovery channel is set to channel 0. The probability of temporary deviation, $P_{deviate}$, is varied from 0.2 to 0.8 (identical to the setting in McMAC) and only the one that has the best performance is shown in the following figures. The notation $SSCH(n)$ means n (channel, seed) pairs are utilized in the SSCH protocol while $McMAC(p)$ means $P_{deviate}$ is set to p in McMAC.

B. Simulation Results

We first determine the best value of the threshold T. As shown in Fig. 6, when T is less than 10%, the aggregate throughput is high and does not differ much. In fact, the throughput increases when T is between 1% and 7% and decreases afterwards. Similar trends can be found when we varied the number of nodes and the number of channels. Thus, we set the value of T to be 7% where the highest throughput was achieved.

In the following, observations are made from five aspects.

1) Impact of Number of Channels:

In this experiment, we varied the number of channels and the results of aggregate throughput can be found in Fig. 7. As

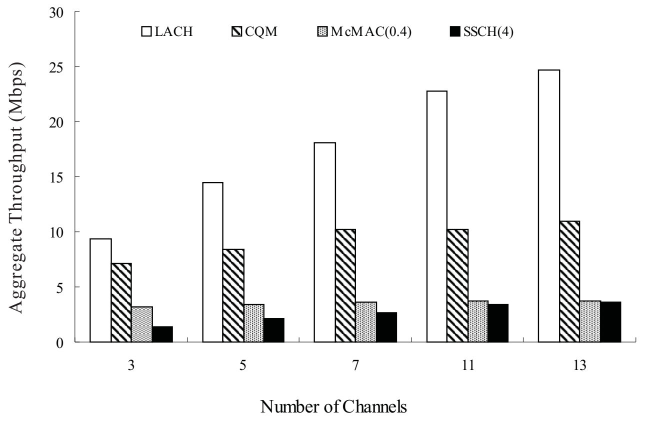

Fig. 7. Aggregate throughput in different channels

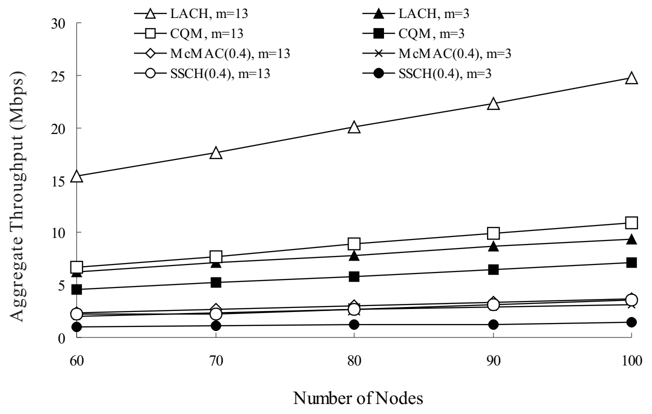

Fig. 8. Aggregate throughput with different number of nodes

expected, LACH outperforms the other three protocols in all situations. It is because LACH enables nodes to dynamically adjust the number of default slots and thus the number of overlapping slots is increased. In CQM, the flexibility is limited since the number of overlapping slots in a cycle for a particular transmission pair is fixed. When a node A running SSCH changes its channel hopping sequence, the other nodes do not aware of such changes until they receive node A's new channel schedule. The missing receiver problem may occur and produces significant performance degradation because the channel schedule information is not guaranteed to be received by all the neighbors. When using 3 channels, the throughput of LACH is 32%, 197% and 564% higher than that of CQM, McMAC, and SSCH, respectively. When using 13 channels, the throughput of LACH becomes 1.26, 5.64 and 5.87 times higher than that of CQM, McMAC, and SSCH, respectively. The average throughput improvement per additional channel for LACH and CQM is 16.38% and 5.46%, respectively. LACH and CQM achieve better performance when the number of channels enlarges because the packet collision probability is reduced. McMAC performs better than SSCH since nodes running McMAC retain some of their original channel hopping sequences if $P_{deviate} \neq 1$. With such a mechanism, although nodes running McMAC suffer from the missing receiver problem, it is not as severe as what is encountered for nodes running SSCH. Note that nodes cannot precisely estimate the channels their neighbors are resided in because the exact slots for nodes running McMAC follow their original schedule are not available for their neighbors. For LACH and CQM, each node's default channel and default slots are deterministic and is known by its neighbors. This means that the rendezvous between two nodes can be correctly specified.

2) Impact of Number of Nodes:

We have also varied the number of nodes from 60 to 100 to observe the influence. The results for aggregate throughput when using 3 or 13 channels are shown in Fig. 8. We can see that network throughput is getting higher as more nodes are deployed in the network. However, transmission contention and packet collision are also increased. In such scenarios, LACH and CQM have better performance because transmission pairs are distributed among all channels, which alleviates the collision problem. When 13 channels are utilized, the average throughput enhancement per adding node is 233, 106, 36 and 34 kbps for LACH, CQM, McMAC, and SSCH, respectively. Again, LACH outperforms CQM because the number of overlapping slots in CQM is fixed. For McMAC and SSCH, we believe the throughput degradation results from the serious missing receivers and packet collisions.

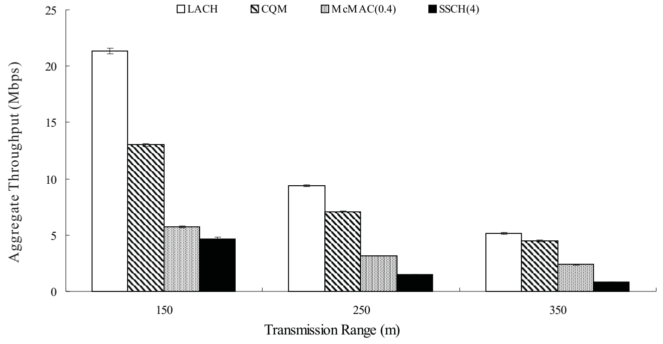

Fig. 9. Aggregate throughput with different transmission ranges

3) Impact of Transmission Range:

We have also changed each node's transmission range to observe the effect on different spatial reuse characteristics. We have tested three different transmission ranges: 150 m, 250 m, and 350 m, which builts a 7-hop, 5-hop, and 3-hop network, respectively. As shown in Fig. 9, the throughput enhancement of LACH over CQM, SSCH, and McMAC is obvious. A larger transmission range results in limited spatial reuse, which lowers the aggregate throughput.

4) Impact of Mobility:

Next, we investigate the effect of nodes' movement. 100 nodes, followed the Random Waypoint mobility model, were randomly distributed in a larger area of 1 km $\times$ 1 km. Nodes

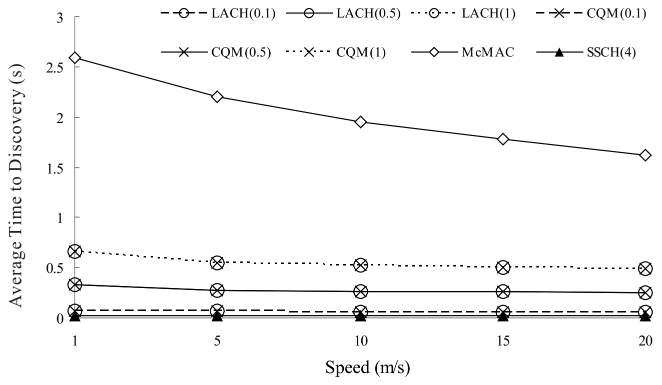

Fig. 10. Average discovery time for different mobility speeds

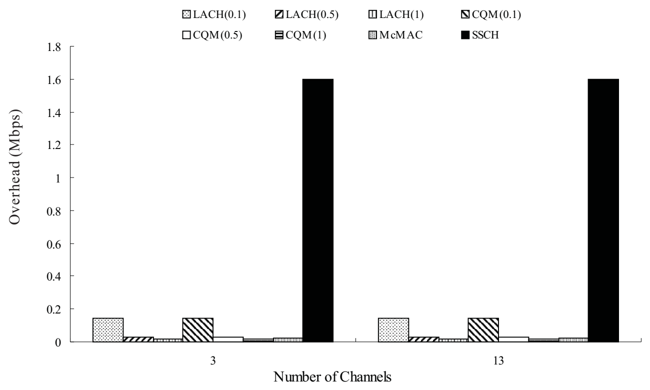

Fig. 11. Average overhead for different mobility speeds

randomly choose a target and move toward it at a speed of 1 to 20 m/s. When the target point is reached, a node stays for 0 to 10 seconds. We observe the time and overhead required for a node to discovery all its one-hop neighbors' channel schedules. In this experiment, a total of 3 channels are available for each node. For SSCH, a node broadcasts its channel schedule once every slot. A node running McMAC broadcasts a beacon once per second and an additional beacon is sent once on its default discovery channel every two seconds. For LACH and CQM, three different broadcast periods are implemented, that is, a broadcast slot is allocated every 0.1, 0.5, and 1 second. We use LACH(p) and CQM(p) to represent a broadcast slot being allocated every p seconds in LACH and CQM, respectively. In this experiment, the results are the average of 20 simulation runs with each of which simulated 600 seconds.

As shown in Fig. 10, nodes running SSCH use the least time to find their neighbors' schedules since they broadcast channel schedules most frequently. The impact of mobility is similar for LACH and CQM and both protocols have pretty good performance. It takes much longer time for nodes running McMAC to identify their neighbors' schedules. It is

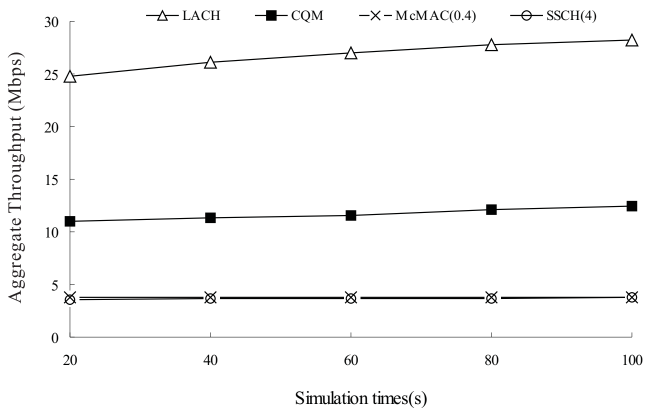

Fig. 12. Aggregate throughput with varied traffic loads

because McMAC has the lowest schedule broadcast frequency. It should be noted that nodes running McMAC and SSCH do not possess broadcast slots. Some nodes may miss some of the schedule advertisements, and hence the effectiveness of channel schedule advertisements is reduced. Fig. 11 shows the overhead for different protocols. As expected, nodes running SSCH have the highest overhead due to the most frequent schedule exchanges. McMAC produces slightly higher overhead when compared to LACH(1) and CQM(1) but has lower overhead when compared to LACH(0.5) and CQM(0.5). Considering Fig. 10 and Fig. 11 together, we comment that all the four protocols can handle node mobility well while LACH and CQM achieve it in an efficient way.

5) Impact of varied traffic loads:

In this experiment, we investigated the impact of varied traffic loads in 100 seconds. The average burst length starts at 250 packets and increases by 100 packets every 20 seconds. A total of 100 nodes use 13 channels in this experiment. The results of aggregate throughput can be found in Fig. 12. LACH obviously outperforms the other three protocols and the aggregate throughput increases as the traffic load increases. This confirms the benefits of the dynamic-adjustable channel hopping mechanism. Without the ability to change channel hopping sequences according to different traffic loads, the aggregate throughput of the other three protocols remains stable as traffic load increases. This experiment verifies that LACH can maintain high throughput in different traffic loads when compared with the other protocols.

C. Real System Implementation Setup

We have made a real system implementation to verify if the simulation results are trustworthy. We implemented LACH, CQM, McMAC, and SSCH in TinyOS 2.x on Octopus II platform. In our implementation, 30 nodes were randomly deployed to form a 5-hop ad hoc network. Any node can be a sender where the destination is chosen from its one-hop neighbors. The Octopus II platform uses an IEEE 802.15.4-compatible CC2420 radio chip, the data packet size is set

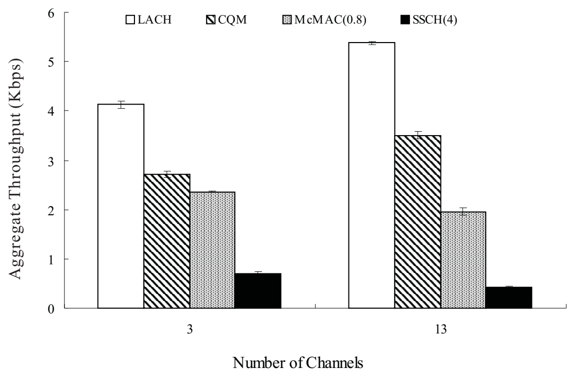

Fig. 13. Impact of number of channels in real system implementation

to the maximum of 127 bytes where the payload size is 114 bytes. Bursty traffic was generated such that the burst arrival probability for each node in every 100 seconds is uniformly distributed between 0 and 1. For each arrival, the burst length is uniformly distributed between 90 and 100 packets. The transmission rate of CC2420 is 250 kbps which is unchangeable. The number of channels was set to 3 and 13. A time slot was set to 1 second. Each experiment in our implementation ran for 300 seconds while a simple synchronization algorithm (all nodes synchronized to a fixed node in each channel) was executed in each slot to keep nodes synchronized. In LACH, a 13×13 latin square was employed. For CQM, we choose the difference set $\{0, 1, 3\}$ under Z_6 as G_0. For McMAC, the parameter $P_{deviate}$ was set to 0.8. The number of (channel, seed) pairs in SSCH was set to 4. Similar to our simulator, we have implemented SSCH and McMAC as faithful as possible. The results are the average of 5 experiments.

D. Real System Implementation Results

The impact of number of channels is shown in Fig. 13. Being able to adjust the number of default slots, nodes running LACH always have better performance than the other protocols. It should be noted that the throughput for McMAC and SSCH decreases when the number of channels increases, which is different from our simulation results. We believe it is the consequence of the reduced node density in the real system implementation. With less number of nodes, the throughput enhancement results from increased channels cannot compensate for the throughput degradation due to the missing receiver problem. On the contrary, without the missing receiver problem, both LACH and CQM benefit from increased number of channels. These results have verified the effectiveness of the proposed LACH in real systems.

The impact of number of nodes can be found in 14. A higher throughput is achieved for all the protocols when the number of nodes increases. As expected, LACH still performs the best while SSCH the worst. Again, due to reduced node density, the throughput for McMAC and SSCH decreases as the number of channels increases.

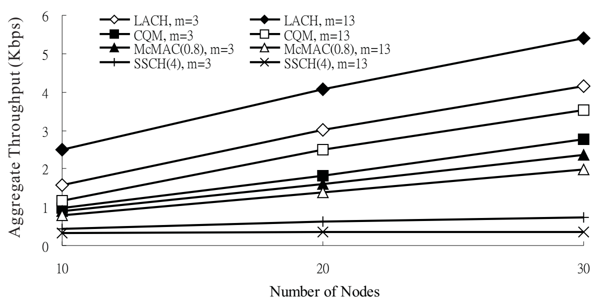

Fig. 14. Impact of number of nodes in real system implementation

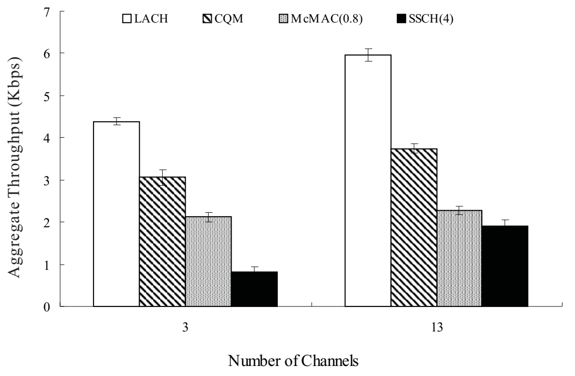

Fig. 15. Simulation results of aggregate throughput with different channels using the same parameter settings as in the real system implementation

To verify if the results of the real system implementation are trustworthy, a simulation with the same parameter settings used in the real implementation was also conducted. The simulation results of aggregate throughput, for a network with 30 nodes, using different channels can be found in Fig. 15. Compared with Fig. 13, we can see the results of simulations and real test beds are very close.

VI. Conclusions

In this paper, we proposed a load-aware channel hopping protocol for mobile ad hoc networks. The proposed LACH protocol requires only one transceiver to achieve multi-rendezvous. Utilizing latin squares, different nodes' initial default slots are distributed evenly in a cycle if their IDs are evenly distributed. The schedule selection scheme of LACH guarantees a sender meets its intended receiver. Nodes can also dynamically adjust their schedules according to their own traffic loads. Simulation and real system implementation results verified that LACH performs better than existing multi-channel MAC protocol, McMAC and SSCH. We believe that the proposed LACH achieves significant throughput improvement, especially in a network with unbalanced traffic loads.

Acknowledgements

This research was sponsored by Ministry of Science and Technology, R. O. C., under grant NSC 102-2221-E-019-017-MY3.

References

[1] K. H. Almotairi and X. Shen. Multichannel medium access control for ad hoc wireless networks. 2011.

[2] L. Hongjiang, R. Zhi, G. Chao, and G. Yongcai. A New Multi-channel MAC Protocol for 802.11-based Wireless Mesh Networks. In *IEEE ICCSEE*, pages 27–31, 2012.

[3] Myunghwan Seo, Yonggyu Kim, and Joongsoo Ma. Multi-Channel MAC Protocol for Multi-Hop Wireless Networks: Handling Multi-Channel Hidden Node Problem Using Snooping. In *Proceedings of IEEE MILCOM*, 2008.

[4] Jianfeng Wang, Yuguang Fang, and Dapeng Wu. A Power-Saving Multi-radio Multi-channel MAC Protocol for Wireless Local Area Networks. In *Proceedings of IEEE INFOCOM*, pages 1-12, April 2006.

[5] Shih-Lin Wu, Yu-Chee Tseng Chih-Yu Lin, and Jang-Ping Sheu. A New Multi-channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks. In *Proceedings of IEEE ISPAN*, pages 232-237, December 2000.

[6] K. H. Almotairi and X. Shen. Fast and Slow Hopping MAC Protocol for Single-Hop Ad Hoc Wireless Networks. In *IEEE ICC*, pages 1–5, 2011.

[7] Jun-Ho Kim and Sang-Jo Yoo. TMCMP: TDMA based Multi-channel MAC Protocol for Improving Channel Efficiency in Wireless Ad Hoc Networks. In *Proceedings of IEEE MICC*, pages 429-434, December 2009.

[8] Chi-Yu Li, An-Kai Jeng, and Rong-Hong Jan. A MAC Protocol for Multi-Channel Multi-Interface Wireless Mesh Network using Hybrid Channel Assignment Scheme. *Journal of Information Science and Engineering*, 23:1041–1055, 2004.

[9] Jaya Shankar Pathmasuntharam, Amitabha Das, and Anil Kumar Gupta. Primary Channel Assignment based MAC (PCAM) - A Multi-Channel MAC Protocol for Multi-Hop Wireless Networks. In *Proceedings of IEEE WCNC*, pages 1110-1115, 2004.

[10] Zhenxia Zhang, Azzedine Boukerche, and Hussam Ramadan. Design and evaluation of a fast MAC layer handoff management scheme for WiFi-based multichannel Vehicular Mesh Networks. *Journal of Network and Computer Applications*, 36(3):992–1000, 2013.

[11] Khaled H. Almotairi and Xuemin (Sherman) Shen. A distributed multi-channel mac protocol for ad hoc wireless networks. *IEEE Transactions on Mobile Computing*, 14(1):1–13, 2015.

[12] Ting-Yu Lin, Kun-Ru Wu, and Guang-Chuen Yin. Channel-hopping scheme and channel-diverse routing in static multi-radio multi-hop wireless networks. *IEEE Transactions on Computers*, 64(1):71–86, 2015.

[13] O. D. Incel, L. van Hoesel, P. Jansen, and P. Havinga. MC-LMAC: A multi-channel MAC protocol for wireless sensor networks. *Ad Hoc Networks*, 9:73–94, Jan. 2011.

[14] Stepan Ivanov, Dmitri Botvich, and Sasitharan Balasubramaniam. Cooperative Wireless Sensor Environments Supporting Body Area Networks. *IEEE Transactions on Consumer Electronics*, 58(2):284–292, May 2012.

[15] Wen-Hwa Liao and Wen-Chin Chung. An Efficient Multi-Channel MAC Protocol for Mobile Ad Hoc Networks. In *Proceedings of IEEE CMC*, pages 162-166, 2009.

[16] Cheng-Shien Lin, Meng-Chun Wueng, Ting-Hung Chiu, and Shyh-In Hwang. Concurrent Multi-Channel Transmission (CMCT) MAC Protocol in Wireless Mobile Ad Hoc Networks. In *Proceedings of ICACT*, pages 445-449, Feb. 12-14, 2007.

[17] Tie Luo, Mehul Motani, and Vikram Srinivasan. Cooperative Asynchronous Multichannel MAC Design, Analysis, and Implementation. *IEEE TRANSACTION ON MOBILE COMPUTING*, MARCH 2009.

[18] Jingpu Shi, Theodoros Salonidis, and Edward W. Knightly. Starvation Mitigation Through Multi-Channel Coordination. In *Proceedings of ACM MobiHoc*, pages 214-225, May 22-25, 2006.

[19] Jungmin So and Nitin Vaidya. Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver. In *Proceedings of ACM MobiHoc*, pages 222-233, May 24-26, 2004.

[20] Duc Ngoc Minh Dang, Huong Tra Le, Hyo Sung Kang, Choong Seon Hong, and Jongwon Choe. Multi-channel mac protocol with directional antennas in wireless ad hoc networks. In *International Conference on Information Networking (ICOIN)*, pages 81–86. IEEE, 2015.

[21] Duc Ngoc Minh Dang, Choong Seon Hong, and Sungwon Lee. A hybrid multi-channel mac protocol for wireless ad hoc networks. *Wireless Networks*, 21(2):387–404, 2015.

[22] Paramvir Bahl, Ranveer Chandra, and John Dunagan. SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks. In *Proceedings of ACM MobiCom*, pages 216-230, September 2004.

[23] Kaigui Bian, Jung-Min Park, and Ruiliang Chen. A Quorum-based Framework for Establishing Control Channels In Dynamic Spectrum Access Networks. In *Proceedings of ACM MobiCom*, September 20-25, 2009.

[24] Hoi-Sheung Wilson So, Jean Walrand, and Jeonghoon Mo. McMAC: A Prarllel Rendezvous Multi-Channel MAC Protocol. In *Proceedings of IEEE WCNC*, pages 334-339, March 2007.

[25] Chih-Min Chao, Hsien-Chen Tsai, and Kuan-Ju Huang. A New Channel Hopping MAC Protocol for Mobile Ad Hoc Networks. *IEEE Transactions on Vehicular Technology*, 63(9):4464–4475, Nov. 2014.

[26] L. Tang, Y. Sun, O. Gurewitz, and David B. Johnson. EM-MAC: A Dynamic Multichannel Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *ACM MobiHoc*, 2011.

[27] Chih-Min Chao, Hsien-Chen Tsai, and Chao-Ying Huang. Load-aware channel hopping protocol design for mobile ad hoc networks. In *IEEE ISWPC*, pages 1–6, July 2012.

[28] Lichun Bao. MALS: Multiple Access Scheduling Based on Latin Squares. In *Proceedings of IEEE MILCOM*, pages 315-321, December 2004.

[29] Ji-Her Ju and Victor O. K. Li. TDMA Scheduling Design of Multihop Packet Radio Networks Based on Latin Squares. *IEEE Journal on Selected Areas in Communications*, 17:1345–1352, August 1999.

[30] Moonseo Park and Seong-Lyun Kim. Minimum Distortion Network Code Design for Source Coding Over Noisy Channels. In *Proceedings of IEEE PIMRC*, pages 1-5, 2008.

[31] Hyun-Young Oh, Dae-Son Kim, Joon-Sung Kim, and Hong-Yeop Song. Collision-free Interleavers using Latin Squares for Parallel Decoding of Turbo Codes. In *Proceedings of IEEE VTC*, pages 1589-1592, 2007.

[32] Jang-Ping Sheu, Chih-Min Chao, Wei-Kai Hu, and Ching-Wen Sun. A Clock Synchronization Algorithm for Multihop Wireless Ad Hoc Networks. *Wireless Personal Communications*, pages 185–200, 2007.

[33] Hoi-Sheung Wilson So, Giang Nguyen, and Jean Walrand. Practical Synchronization Techniques for Multi-Channel MAC. In *Proceedings of ACM MobiCom*, September 23-26, 2006.