A Receiver-Centric Multi-Channel MAC Protocol for Wireless Networks

Jaeseon Hwang, Taewoon Kim, Jungmin So, and Hyuk Lim

Abstract

We consider the issues of mitigating interference and improving network capacity in wireless networks from the viewpoint of channel diversity. Multi-channel diversity allows multiple pairs to concurrently use the wireless medium, thus increasing the achievable capacity; this multi-channel diversity can then be fully utilized by enabling wireless nodes to dynamically switch their channels. However, the process of switching channels in a wireless transceiver incurs switching overhead, resulting in the degradation of the throughput performance. We propose a receiver-centric multi-channel MAC protocol (RcMAC) that allows nodes to efficiently utilize multiple channels by reducing unnecessary channel switching. Receiver-centric channel switching enables each sender node to asynchronously and independently switch channels to one where its intended receiver resides, without requiring explicit channel negotiation. Thus, the wait time at the control channel is reduced, in addition to the number of channel switchings, thereby improving channel utilization. As this scheme requires prior knowledge of which channel a node is switched to, each node cooperatively shares its channel usage information in order to recognize the channels that its neighboring nodes are using. Through extensive simulations, we show that the proposed MAC protocol significantly improves network throughput and reduces end-to-end delay compared with other multi-channel MAC protocols.

Index Terms

Wireless networks, multiple channels, medium access control, network protocols.

J. Hwang and H. Lim are with the Department of Information and Communications, Gwangju Institute of Science and Technology (GIST), Gwangju, 500-712, Republic of Korea. Email: {jshwang, hlim}@gist.ac.kr

T. Kim is with the Telecommunications Technology Association (TTA), Seongnam, Gyonggi-do, 463-824, Republic of Korea. Email: twkim@tta.or.kr

J. So is with the Department of Computer Engineering, Hallym University, Chuncheon, Gangwon-do 200-702, Republic of Korea. Email: jungmin.so@gmail.com

I. INTRODUCTION

Wireless networking has received considerable attention as a promising technique for establishing ubiquitous service infrastructure and for providing seamless connectivity in both mobile and static environments. Recently, rapid advances in wireless communication technologies (e.g., MIMO, OFDM, SDR, and smart antennas) make it possible to readily establish wireless communication networks that can satisfy diverse quality of service (QoS) requirements pertaining to parameters such as data throughput, delay, and jitter, with simple deployment and management. In these networks, one critical performance metric is the network throughput performance. In carrier sense multiple access (CSMA)-based wireless networking environments, in which a number of nodes share a single wireless medium, the contention and interference among competing nodes can significantly affect the network throughput performance, because only one transmission is possible within a carrier sense range. Thus, other nodes within the carrier sense range need to defer their transmissions in order to avoid interference with the current transmission. To mitigate this interference and maximize the network capacity, a variety of protocols and algorithms for exploiting multi-channel diversity have been reported [1]–[20]. Based on these studies, it has been determined that multi-channel diversity can be realized by enabling wireless nodes to dynamically switch channels. Because each pair of nodes uses non-overlapping channels, multiple pairs can concurrently use the medium within a carrier sense range. Therefore, the number of transmissions can be increased, and the achievable capacity in a multi-channel network becomes larger than that in a single-channel network.

To exploit the potential of multi-channel diversity, competing nodes need to use non-overlapping channels in order to guarantee that communication on one channel does not interfere with that on any of the other channels. The number of non-overlapping and orthogonal channels is 3 and 12 in IEEE 802.11b and IEEE 802.11a [21], respectively. However, given that the number of sender-receiver pairs is typically larger than the number of orthogonal channels, a dedicated channel cannot be allocated to each pair. Hence, the sender-receiver pair needs to perform frequent channel switching in order to find a non-occupied channel or a less congested channel. Unfortunately, such channel switching in a wireless transceiver may incur a considerable amount of delay (we discuss the issue of channel switching overhead in Section II). Therefore, there is a need to develop a multi-channel MAC protocol that efficiently performs channel coordination

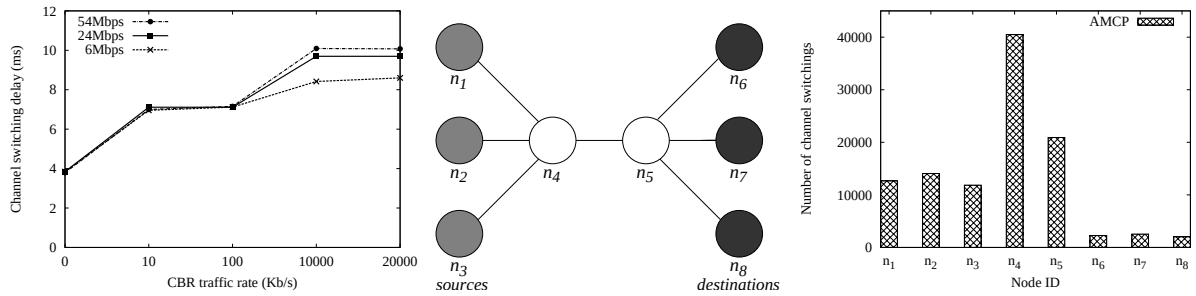
in order to reduce the overhead incurred by channel switching.

In this paper, we propose a receiver-centric multi-channel MAC protocol (RcMAC) that allows nodes to efficiently utilize multiple channels without requiring unnecessary channel switching. To this end, we introduce the following: 1) a fully asynchronous multi-channel MAC protocol that requires only a single transceiver, 2) the notion of receiver-centric channel switching, 3) channel negotiation via the cooperative sharing of channel usage information from neighboring nodes, and 4) multiple data transmissions based on a look-ahead technique in the outgoing queue. It is expected that receiver-centric channel switching will enable nodes to reduce their wait time on the control channel as well as the number of channel switchings that are required. In addition, to further reduce the overhead incurred by switching channels, each node is allowed to transmit multiple data frames for bursty traffic on its current data channel. Finally, through ns-2 simulations [22], we show that network performance is significantly improved in terms of the throughput and end-to-end packet delay.

The rest of this paper is organized as follows. In Section II, we present the motivation of our proposed scheme by investigating channel switching overhead. Next, in Section III, we summarize existing multi-channel MAC protocols. In Section IV, we describe our proposed multi-channel MAC protocol. In Section V, we analyze the protocol in terms of throughput and overhead. We then discuss several issues pertaining to multi-channel diversity in Section VI. In Section VII, we present our simulation results and then conclude the paper in Section VIII.

II. MOTIVATION

In a multi-channel environment, multi-channel diversity can be utilized to increase network capacity by enabling wireless nodes to dynamically switch to a less congested channel. However, the process of switching channels in a wireless transceiver incurs overhead. To estimate this overhead, we measured channel switching delays in a wireless network testbed, in which the sender-receiver pair was a Lenovo 300 N100 laptop equipped with a 3COM 802.11a/b/g wireless card (with an Atheros chipset), and used the Linux MadWifi driver (version 0.9.4) [23] for WLAN networking. In addition to the default two-way handshake DATA/ACK under the IEEE 802.11 DCF, we implemented a channel-switching mechanism in the MadWifi driver for performing channel switching overhead measurements. The pair alternated between two orthogonal channels of 5.24 and 5.32 GHz in the IEEE 802.11a mode by changing the values of the channel ID



(a) Switching delay (b) Topology (c) Number of channel switchings

Fig. 1. Channel switching in a dumbbell topology with three sender-receiver pairs and two relay nodes.

and the frequency. By using Iperf [24], which is a network performance measurement tool for UDP/TCP data streams, we configured the sender to transmit the constant bit rate (CBR) traffic to the receiver in a duration of one second. In this experiment, we measured the channel switching delay for three types of data rates and CBR traffic rates, as shown in Fig. 1(a). We observed that the channel switching overhead was considerably large, and that it increased as the network traffic became heavier. The reason is that it takes time for the Linux device driver in the wireless interface to clear the buffer associated with the current channel before it performs the channel switching process. This time would therefore be extended if there were a large number of packets waiting in the queue. As a result, in the case that channel switching frequently occurs in a bottleneck node that is congested with a large amount of data traffic, the channel switching overhead would not be negligible and would result in significant degradation of the throughput performance.

To further analyze the channel switching overhead, we performed an *ns*-2 simulation for AMCP [8], which is a well-known multi-channel MAC protocol, on a dumbbell topology, where data packets are forwarded by two intermediate relay nodes, n_4 and n_5, as shown in Fig. 1(b). Each source node was configured to transmit CBR traffic at 5 Mb/s, and the simulation time was 50 s. This protocol uses a control channel to negotiate channel allocation among competing nodes and multiple data channels for data transmission. Next, let us suppose that the source-destination pairs initially reside on the control channel. In this case, one of the source nodes n_1, n_2, or n_3 initiates negotiation with the relay node n_4. After this negotiation, both the nodes switch to the data channel, exchange a data packet, and return to the control channel; n_4 then

repeats this same process for each source node. Given that two channel switchings are required for each data transmission, the channel switching overhead is significantly large, as shown in Fig. 1(c). Because n_4 and n_5 relay packets from the sources to the destinations, they perform a large number of channel switchings. The channel switching overhead can cause the network throughput to be significantly degraded.

However, if other source nodes know that the relay node is on the data channel, they can switch to the data channel without any negotiation by allowing the relay node to remain on the same data channel. Consequently, the number of channel switchings performed by the relay node would be significantly reduced. Motivated by this finding, we considered a multi-channel MAC protocol that would allow nodes to efficiently utilize multiple channels by reducing the number of unnecessary and redundant channel switchings.

III. Related Work

Existing multi-channel protocols can be roughly divided into two categories: synchronous and asynchronous protocols. For each class, we enumerate and briefly describe the characteristic features of these protocols.

A. Synchronous Protocols

So *et al.* proposed MMAC [1], which adopts the IEEE 802.11 Power Saving Mechanism (PSM) in order to synchronize the clocks between neighboring nodes. MMAC separates time into two fixed sessions: one for negotiation and the other for data transmission. During a negotiation session, nodes exchange control packets in order that the sender-receiver pairs can communicate with each other on the reserved channel during the following data transmission session. Chen *et al.* devised MAP [2], which allows the data transmission phase to have a variable length depending on the negotiation results. In addition, MAP removes any contention during the data transmission phase by means of a scheduling algorithm. The main advantage of these two protocols is that they use only one interface. However, it should be noted that the negotiation phase in both the protocols needs to be sufficiently long to accommodate all requests, which subsequently limits the maximum achievable throughput performance.

Some protocols in this class require tight synchronization. For example, Tzamaloukas *et al.* devised CHMA [3], in which nodes continuously switch channels according to a common

hopping sequence. When using CHMA, if a sender succeeds in exchanging control messages with its intended receiver, both the nodes stop hopping and start data transmission. When the pair completes its transmission, both the nodes re-synchronize and follow the previous hopping sequence. The authors later improved their work by guaranteeing the collision-free transmission of multicast and broadcast packets [4]. The main advantage of both the protocols is that they require neither an additional interface nor a dedicated control channel. However, the process of frequent channel hopping and the need for tight synchronization both incur overhead, in terms of both implementation and operation.

On the other hand, McMAC [5] is considerably different, in that it can perform a parallel rendezvous on different channels. In McMAC, a node performs periodic channel switching according to its pseudo-random hopping sequence. If there are any pending messages in the queue, a sender temporarily deviates from its default sequence and then transmits them to a receiver on another channel. In SSCH [6], nodes periodically tune into another channel according to a randomized hopping sequence. If a sender wants to transmit a packet to a receiver, it first attempts to rendezvous with its corresponding recipient, and then changes its hopping schedule in order that its schedule can overlap with that of the receiver. Patel *et al.* [7] further divided a network into several sub-networks, and then allocated different channel hopping sequences to each network. In their scheme, the transmission sequence is scheduled such that each sub-network can rendezvous with another sub-network on every channel hop.

B. Asynchronous Protocols

One dominant trend in asynchronous protocols is to allocate one channel for the exchange of control packets and the other channels for data transmission. A node on a data channel cannot hear control messages, and hence, protocols in this class allocate another dedicated interface for control messages, or return to the control channel within a short time with no additional interface.

Shi *et al.* proposed AMCP [8] as a method of alleviating starvation problems. In this approach, each sender-receiver pair decides upon a data channel according to its own internal channel table. When an agreement is reached, both the nodes switch to the data channel and transfer one DATA/ACK packet, after which they immediately return to the control channel. Luo *et al.* devised CAM-MAC [9], in which channel and node information is exchanged cooperatively. In

CAM-MAC, the basic operation is similar to that of AMCP, except that when the requested channel or receiver is temporarily unavailable, the neighboring nodes can notify the sender of this fact. Notably, both the AMCP and CAM-MAC protocols use only one transceiver. Compared with these protocols, a unique feature of the proposed RcMAC is its *receiver-centric channel switching*, which allows multiple senders to asynchronously and independently switch channels to where intended receivers reside, without requiring explicit channel negotiation.

Contrastively, DCA [10] and DPC [25] use two transceivers: one is fixed to the control channel, and the other dynamically tunes in to a data channel to exchange DATA/ACK packets. Both DCA and DPC follow the same basic operations as single-transceiver protocols. The major drawback of DCA and DPC, however, is their low channel utilization, because nodes transfer only one data packet for every two channel switchings. In addition, when the network is congested, the control channel may become bottlenecked. In this case, packet collisions can further reduce the number of successful negotiations, and thereby degrade the overall throughput.

DB-MCMAC [11] is somewhat different from other asynchronous protocols in that it uses the same number of transceivers and channels. For this reason, it does not require a dedicated channel for the exchange of control messages, because every channel can be monitored. In DB-MCMAC, a node maintains its per-neighbor queues, and any idle transceiver can dynamically detach a packet and transmit it. The advantage of this protocol is that the best channel for each receiver is used for transmission, with a high probability of reception, because each channel state is tracked. A notable drawback of DB-MCMAC, however, is its high hardware cost.

IV. Receiver-Centric Multi-Channel MAC

We now consider the problem of channel switching in a multi-channel network. In brief, whenever a node has a packet to be transmitted, it needs to negotiate with its intended receiver on the control channel and then switch to the data channel to transmit data packets. Motivated by the example in Section II, we propose a receiver-centric multi-channel MAC protocol (RcMAC) that allows each node to asynchronously and independently switch channels to one on which its intended receiver resides. The nodes on the control channel are enabled to cooperatively share the channel usage information, and hence, they are able to switch to that channel directly without explicit channel negotiation with an intended receiver, instead of waiting for the receiver to return to the control channel. This receiver-centric channel switching process can thus reduce

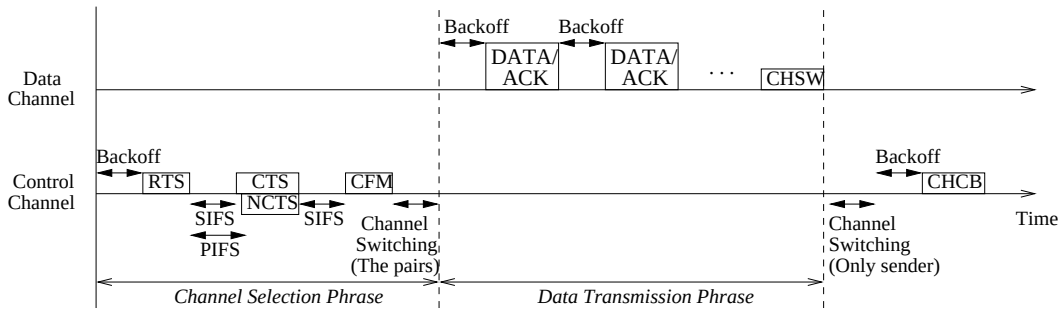


Fig. 2. Basic operation of RcMAC.

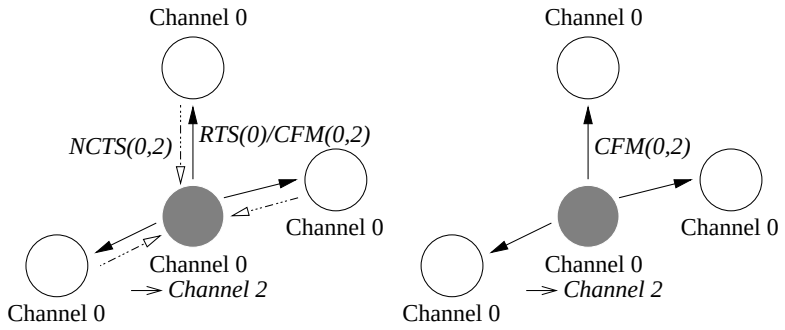
both the wait time at the control channel owing to negotiation, as well as the number of channel switchings.

A. Basic Operation

The data transmission of a node under RcMAC consists of a *channel selection phase* on a control channel and a *data transmission phase* on data channels, as shown in Fig. 2.

In the channel selection phase, a pair of nodes that wish to communicate with each other first select a data channel. They then switch to that channel and exchange data packets. In the case that the sender and its intended receiver are both on the control channel, negotiation is performed using an RTS/CTS/CFM handshake, where RTS and CTS are the request-to-send and clear-to-send control packets, respectively, and CFM is a confirmation control packet that completes the negotiation by confirming which data channel the pair has agreed to switch to. On the other hand, when an intended receiver is not on the control channel, the sender performs *receiver-centric channel switching*, which is done either by an RTS/NCTS/CFM handshake, where NCTS is the neighbor-supported-CTS, or by a CFM broadcast. In brief, the sender gathers the channel information for its intended receiver and switches to the data channel the receiver is currently using without explicit negotiation on the control channel (this receiver-centric channel switching is explained further in Section IV-B).

In the data transmission phase, a sender is allowed to transmit multiple data packets in order to reduce the number of channel switchings. For each data packet, CSMA with the binary exponential back-off (BEB) algorithm adopted in IEEE 802.11 DCF is performed, as shown in Fig. 2. In the BEB algorithm, when it is sensed that a channel has been idle for a specific time



(a) RTS/NCTS/CFM handshake (b) CFM broadcast

Fig. 3. Receiver-centric channel switching when an intended receiver is not on the control channel. Channels 0 and 2 are the control and data channels, respectively.

interval, called the *distributed inter-frame space (DIFS)*, the sender selects a random back-off timer, which is uniformly distributed in $[0, \mathrm{CW} - 1]$, where CW is the size of the contention window. CW is initially set to its minimum value $\mathrm{CW}_{\min}$, and is doubled up to its maximum value $\mathrm{CW}_{\max}$ after each transmission collision. The back-off timer is decreased by one if it is sensed that the channel has been idle for one physical time slot, and it is suspended if it is sensed that the channel is busy. The node transmits its frame when the back-off timer reaches zero. By using BEB, we can reduce the possibility of collisions among senders on the same data channel, and compel them to use the channel in a fair manner. Hence, the data transmissions continue as long as the receiver resides on the same channel and the senders have data packets to transmit. When a sender decides to switch back to the control channel, it performs CSMA with the BEB algorithm and broadcasts a CHSW (channel switching) control packet on the data channel to inform that it is going to leave the data channel, whereas the receiver continues to use that channel. It should be noted that by remaining on the data channel the receiver may receive more data packets from other nodes without incurring a channel switching overhead. After switching back, the sender performs CSMA with the BEB algorithm and broadcasts a CHCB (channel comeback) control packet on the control channel to inform the neighboring nodes that it has returned to the control channel.

B. Receiver-Centric Channel Switching

We now explain the process of receiver-centric channel switching in a simple network. Fig. 3 shows the control packet exchanges among the nodes during channel negotiation when the intended receiver is not on the data channel. First, when the sender does not have the channel information, the sender transmits the RTS on the control channel (*Channel 0*) and then receives from its neighboring nodes the NCTS in which the channel of the intended receiver is specified (*Channel 2*) as shown in Fig. 3(a). After recognizing the receiver's channel, it broadcasts the CFM and finally switches to *Channel 2.* On the other hand, when the sender already knows the channel of its intended receiver, it does not need to exchange RTS/CTS, as shown in Fig. 3(b). In such a case, it immediately switches to *Channel 2* after broadcasting only the CFM.

Depending upon whether the sender knows the data channel on which its intended receiver resides, the negotiation on the control channel for receiver-centric channel switching is performed based on the following cases.

- Case 1: *The sender does not know the current data channel of its intended receiver.*
  When a sender does not know the channel on which its intended receiver currently resides, it broadcasts an RTS packet on the control channel to the neighboring nodes within its transmission range. Upon receiving the RTS, the neighboring nodes look up the local channel table, and if they recognize that the intended receiver is not on the control channel, one of them replies with an NCTS that includes the current channel of the intended receiver on the control channel. In this case, in order to avoid potential contentions, each node puts a random delay in the range of [0, ($T_{pifs} - T_{sifs}$)] before broadcasting an NCTS packet. The neighboring node with the shortest delay replies first with the NCTS packet, and the other nodes give up the attempt to transmit the NCTS packet when the current NCTS transmission is overheard. It should also be noted that if the intended receiver is on the control channel, it replies to the RTS packet in an SIFS interval and the neighboring nodes do not broadcast the NCTS packet. Upon receiving the NCTS, the sender broadcasts a CFM packet in order to inform its neighboring nodes about the channel to which it is going to switch to.
- Case 2: *The sender knows the current data channel of its intended receiver.*
  In this case, the sender omits the RTS/CTS (or NCTS) handshake and broadcasts only a CFM packet, in order that neighboring nodes can identify which channel the sender is going

to switch to.

It should be noted that receiver-centric channel switching provides flexible channel switching to senders on the control channels; they can switch to the current data channel of their intended receivers without needing to directly negotiate with them.

On the data channel, each node gathers the channel usage information, including that of which nodes are on the data channel, by listening to the DATA and ACK frames.

- Case 3: *The next intended receiver is found on the current data channel.*

  A sender does not need to return to the control channel after completing the current data transmission if it recognizes that its next intended receiver is on the same data channel. This scheme effectively reduces the number of unnecessary channel switchings; i.e., switching the channel to the control channel and then returning to the same data channel.

C. Detailed Procedures

Fig. 4 depicts the overall procedure of RcMAC. Here, we divide the procedure into three phases and provide a detailed explanation of each phase.

1) Channel Selection Phase: For receiver-centric channel switching, each node needs to gather channel usage information about which channels its neighboring nodes are currently using, and then identify the less congested channels that are available both to itself and to its corresponding receiver. Under RcMAC, each node maintains a local channel table consisting of channel usage information that contains a set of node indices for each channel, and cooperatively exchanges channel usage information with its neighboring nodes on the control channel.

As briefly discussed in Section IV-A, an RTS/CTS/CFM handshake during the channel selection phase is performed for channel negotiation, through which the data channel to which a sender-receiver pair is going to switch to for data communication is selected. When a node broadcasts either an RTS or CTS packet on the control channel, the recently updated channel information in the channel table is included in the broadcast RTS or CTS control packet. After a data channel is selected, a CFM packet is broadcast that includes the selected data channel index. If the sender transmits an RTS packet but receives no reply within a timeout interval, it performs a retransmission of the RTS packet. During this channel selection phase, any nodes that overhear the RTS, CTS, or CFM packets can update their channel table with the latest channel information. Finally, whenever a node returns to the control channel after it completes its data

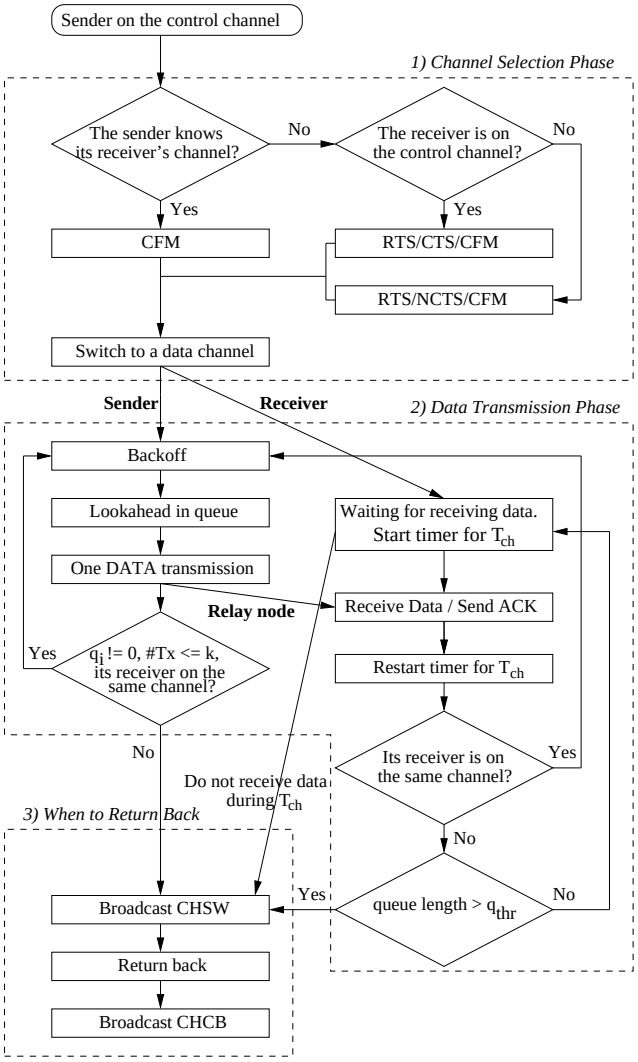


Fig. 4. Schematic of the procedure for RcMAC.

transmission, the channel information in its channel table is considered to be out-of-date and is reset, except for the entry for the most recently used channel.

During the selection of a data channel, a sender-receiver pair attempts to identify a data channel that is recognized as empty by both the nodes. Upon receiving an RTS packet, the receiver compares its channel table with the channel table included in the RTS, and selects from the two channel tables a data channel that is recognized as being empty. If no such channel exists, the receiver selects one of the channels that is designated as empty in its channel table. In a worst case scenario, in which all the channels are occupied by neighboring nodes, the receiver

may randomly select a data channel. Instead of this random channel selection, an advanced scheme can also be applied that provides a workload with a fair distribution for all available channels for efficient channel reuse. More details on channel selection are discussed in Section VI.

2) Data Transmission Phase: We allow a sender on a data channel to transmit multiple data frames for bursty traffic in order to exploit the channel more efficiently without requiring frequent channel switching. Specifically, a sender looks for up to k frames that are destined for its intended receiver in its outgoing queue, and transmits the packets until it returns to the control channel. It should be noted that whenever a DATA packet is ready for transmission, carrier sense multiple access with the BEB algorithm is performed in order to reduce the possibility of collisions among the nodes on the same channel. While transmitting the frames, the sender overhears the DATA and ACK frames on the data channel in order to ascertain the presence of other nodes. After completing the transmissions to its current receiver, it then designates as *the next intended receiver* the next-hop node of the first frame at the head of the outgoing queue. If the next receiver is known to be on the same data channel, it again looks up all the frames destined for the next receiver in its queue and then transmits them instead of immediately returning to the control channel. It should be noted that nodes are not allowed to transmit more than k frames during a data transmission phase.

3) Switching Back to the Control Channel: Under RcMAC, each node on a data channel decides when to switch back to the control channel in an asynchronous manner. If a node returns to the control channel too early, it may lose an opportunity to find other receivers and transmit more frames on the data channel, thereby causing it to eventually switch to that data channel again. However, if a node remains on a specific data channel for a very long time, that channel may become congested, because nodes that have frames destined for that node will decide to join and switch to the same data channel. In this case, the potential of multiple channel diversity cannot be fully exploited. To balance the opportunity for exploiting a channel, RcMAC allows a sender node to return when it has completed at most k data transmissions.

RcMAC instructs sender nodes on a data channel to immediately switch back when their intended receivers are not currently on the same channel. On the other hand, the receiver nodes also switch back when they have not received any data frames during a time interval of T_{ch}. The timer for T_{ch} does not pause even when it sensed that the channel is busy because of other

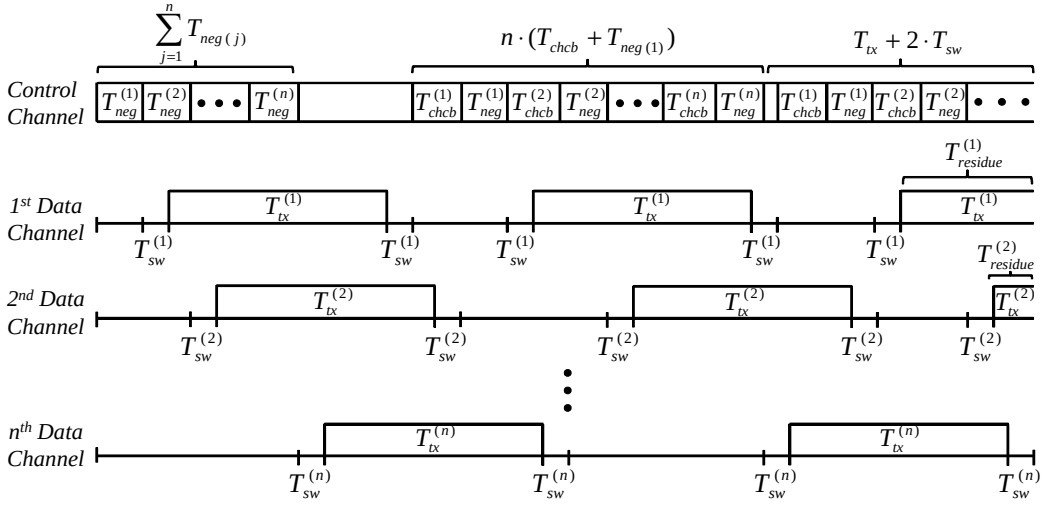


Fig. 5. Timeline diagram for RcMAC operations. The superscript number in parentheses indicates the index of the corresponding sender-receiver pair.

on-going transmissions on the data channel. It should be noted that even when a receiver node is not currently receiving any packets, it does not immediately return to the control channel and remains on the data channel for at least T_{ch}, because some nodes on the same data channel may have data frames that are destined for the node. However, if it is receiving data frames at a high rate, such that its queue length grows rapidly to the point that it reaches the threshold q_{thr}, then the node should immediately return to the control channel to forward the data frames without buffer overflow,— even before the timer set for T_{ch} expires.

It should be noted that the node that are going to switch back to the control channel needs to broadcast a CHSW control frame to notify the other nodes on the same data channel of this switch. After the switch, this node needs to broadcast a CHCB control frame to inform its neighboring nodes that it has just switched back. At this time, the other nodes on the data channel can update their channel tables, to maintain the consistency of the channel information.

V. Protocol Analysis

In this section, we derive the maximum achievable throughput performance of RcMAC in a single-hop network and then validate the obtained analytical results through ns-2 simulations. The assumptions made for this protocol analysis are as follows:

- There are one control channel and c data channels.

TABLE I

CONTROL PACKETS USED IN RCMAC.

Name	Description	MAC payload size (bits)
RTS	Request for transmission with channel table	192
CTS	Response to transmission with channel table	144
NCTS	Notification of the receiver's channel	24
CFM	Notification of the selected data channel	24
CHSW	Notification of leaving a data channel	16
CHCB	Notification of returning to a control channel	16

- Total n sender-receiver pairs are located within each other's transmission range.
- Each of n pairs exploits a data channel exclusively (i.e., $n \leq c$).
- All senders are fully backlogged, and the data payload size is fixed.

Figure 5 depicts a timeline diagram for the first few rounds of n pairs. Each pair performs k back-to-back data packet transmissions on a data channel. After that, the sender of each pair returns to the control channel, whereas the receiver remains on a data channel during a time interval T_{ch} according to the RcMAC operation. Because the sender comes back to the data channel within T_{ch}, it is seen that the receiver does not return to the control channel in Figure 5. Table I lists descriptions and the sizes of the control packets used in RcMAC.

A. Overhead Analysis

We estimate the bandwidth overhead imposed by the control packet exchanges. First, we derive the average length of the RTS/CTS/CFM handshake time for a sender-receiver pair on the control channel. Let $T_s(j)$ and $T_c(j)$ be the transmission time for success and failure of the RTS/CTS/CFM handshake in the jth retransmission attempt caused by collision, respectively, as follows:

$$T_s(j) = T_{difs} + \overline{T_{bo}(j)} + T_{rts} + T_{cts} + T_{cfm} + 2 \cdot T_{sifs}, \tag{1}$$

$$T_c(j) = 2 \cdot T_{difs} + \overline{T_{bo}(j)} + T_{rts}, \tag{2}$$

where T_{difs} and T_{sifs} are the time intervals for the distributed and short inter-frame space, respectively, and T_{rts}, T_{cts}, T_{cfm} are the lengths of the RTS, the CTS, and the CFM control

packet transmissions, respectively. The average backoff time denoted by $\overline{T_{bo}(j)}$, is the time interval for the jth retransmission attempt, and is represented as

$$\overline{T_{bo}(j)} = \frac{\min\left(2^j \cdot \mathrm{CW}_{\min} - 1, \mathrm{CW}_{\max}\right)}{2} \cdot T_{slot}, \tag{3}$$

where $\mathrm{CW}_{\min}$ and $\mathrm{CW}_{\max}$ are the minimum and maximum sizes of the contention window, respectively, and T_{slot} is the basic slot time.

Using Eqs. (1)-(3), we now obtain the average time required for a successful handshake when there are n senders on the control channel, as follows [26]:

$$T_{neg(n)} = (1 - P_{c(n)}) \cdot T_s(0) + \sum_{j=1}^{\infty} \left[(1 - P_{c(n)}) \cdot (P_{c(n)})^j \cdot \left\{ T_s(j) + \sum_{m=0}^{j-1} T_c(m) \right\} \right], \tag{4}$$

where $P_{c(n)}$ is the collision probability in a time slot among n senders. In Bianchi's model [27], $P_{c(n)}$ is given by

$$P_{c(n)} = 1 - (1 - \tau)^{n-1},$$

and

$$\tau = \frac{2(1 - 2P_{c(n)})}{2(1 - 2P_{c(n)})(\mathrm{CW}_{\min} + 1) + P_{c(n)}\mathrm{CW}_{\min}(1 - (2P_{c(n)})^j)},$$

where τ is the probability that a node will transmit in a randomly selected slot time on the jth transmission attempt. Both $P_{c(n)}$ and τ can be solved using numerical techniques, as was done in [27].

Then, when a fully backlogged sender-receiver pair performs the channel negotiation and the data transmission with k frames, the ratio of the control-overhead-to-total-channel-time (P) is given by

$$P = 1 - \frac{k \cdot T_{data}}{T_{neg(n)} + T_{tx} + T_{chcb} + 2T_{sw}},$$

where T_{data} is the length of the transmission time of one data frame, T_{tx} is the transmission time of k data frames and the CHSW control packet, including the average backoff time for the packets, which is given by

$$T_{tx} = k \cdot \left(\frac{\mathrm{CW}_{\min}}{2} \cdot T_{slot} + T_{data}\right) + T_{chsw}.$$

Here, T_{sw} is the channel switching delay, and T_{chcb} and T_{chsw} are the transmission times of the CHCB and CHSW control packet transmissions, including the length of the average backoff time, respectively. It should be noted that this overhead is derived under the presumption that

collisions occur during the RTS/CTS/CFM handshake on the control channel. For example, when the number of sender-receiver pairs (n) and data transmissions (k) are 4 and 10, respectively, it was found that the control overhead occupies a small portion of the total channel time (11.5%).

B. Throughput Analysis

Depending on the time duration of the transmission on a data channel, we consider the following two cases: i) If the time duration for data transmissions on a data channel is considerably long, the exchange of control packets occurs infrequently and the control channel becomes non-saturated. ii) On the other hand, if the time duration of data transmissions is short, in order that one of the pairs returns while the control channel is being used for channel negotiations, the control channel is saturated. In Fig. 5, it can be seen that when the channel negotiations for all the pairs have been completed, the control channel becomes idle until the first pair returns to the control channel. Let T_{idle}^{ctl} denote the duration of the idle time on the control channel when the rth transmission round ends at the nth data channel (e.g., r = 2 in Fig. 5). Then, T_{idle}^{ctl} is calculated by

$$T_{idle}^{ctl} = \begin{cases} 0, & \text{if } T_{tx} + 2T_{sw} \leq (n-1)(T_{chcb} + T_{neg(1)}); \\ T_{tx} + 2T_{sw} - \sum_{j=1}^{n-1} T_{neg(j)} & \text{if } \sum_{j=1}^{n-1} T_{neg(j)} < T_{tx} + 2T_{sw}; \\ \quad +(r-1)\{T_{tx} + 2T_{sw} - (n-1)(T_{chcb} + T_{neg(1)})\}, & \\ (r-1)\{T_{tx} + 2T_{sw} - (n-1)(T_{chcb} + T_{neg(1)})\}, & \text{otherwise.} \end{cases} \quad (5)$$

The first and the second conditions correspond to the cases where the control channel is saturated and unsaturated, respectively. Under the third condition, the control channel is saturated at the first round, and becomes unsaturated at the following rounds. Let T_{total} denote the time duration until the last pair (i.e., the nth pair) completes its rth data transmission. Using Eqs. (4) and (5), T_{total} is given as follows:

$$T_{total} = \sum_{j=1}^{n} T_{neg(j)} + T_{tx} + 2T_{sw} + (r-1) \cdot n \cdot (T_{chcb} + T_{neg(1)}) + \min(r-1, 1) \cdot T_{idle}^{ctl}. \quad (6)$$

Note that T_{idle}^{ctl} includes the total duration of the idle times on the control channel during r rounds.

Next, we compute the length of the data transmissions on data channels during the time interval of T_{total}. It should be noted that while the nth pair performs exactly r data transmissions, the

other pairs perform more than r data transmissions for T_{total}, as shown in Fig. 5. Each data transmission of the 1st, $\cdots$, $(n-1)$th pairs must be carefully computed. Let $T_{idle}^{data(i)}$ denote the time period during which the ith data channel is idle in T_{total} when $r > 1$. Then, $T_{idle}^{data(i)}$ is expressed as

$$T_{idle}^{data(i)} = \begin{cases} \sum_{j=n-i+1}^{n} T_{neg(j)} + T_{sw} + T_{last}^{unsat(i)} + (r-1)(T_{chcb} + T_{neg(1)} + 2T_{sw}), & \text{if } T_{idle}^{ctl} > 0; \\ \sum_{j=1}^{n} T_{neg(j)} + T_{sw} + T_{last}^{sat(i)} + (nr - 2n + i)(T_{chcb} + T_{neg(1)}) + (1-r)T_{tx}, & \text{otherwise.} \end{cases} \tag{7}$$

where $T_{last}^{unsat(i)}$ and $T_{last}^{sat(i)}$ are the idle time period after the rth data transmissions on the ith data channel in an unsaturated and a saturated case, respectively, and are given by

$$T_{last}^{unsat(i)} = \begin{cases} T_{chcb} + T_{neg(1)} + 2T_{sw}, & \text{if } (\sum_{j=2}^{n-i} T_{neg(j)} - T_{chcb} - T_{sw}) \geq 0; \\ \sum_{j=1}^{n-i} T_{neg(j)} + T_{sw}, & \text{otherwise.} \end{cases}$$

and

$$T_{last}^{sat(i)} = \begin{cases} n(T_{chcb} + T_{neg(1)}) - T_{tx}, & \text{if } (\sum_{j=1}^{n-i} T_{neg(j)} - n(T_{chcb} + T_{neg(1)}) + T_{tx} + T_{sw}) \geq 0; \\ \sum_{j=1}^{n-i} T_{neg(j)} + T_{sw}, & \text{otherwise.} \end{cases}$$

By subtracting the channel usage time from the total time in Eqs. (6) and (7), the time for the data transmission after the rth data transmission on the ith data channel is obtained as follows:

$$T_{residue}^{(i)} = \max(T_{total} - (r \cdot T_{tx} + T_{idle}^{data(i)}), 0). \tag{8}$$

Finally, we compute the aggregate throughput over all the data channels. The number of data packets sent over all the data channels during r rounds is given by

$$K_{total} = rkn + \sum_{i=1}^{n-1} \left\lfloor \frac{T_{residue}^{(i)}}{T_{tx}/k} \right\rfloor. \tag{9}$$

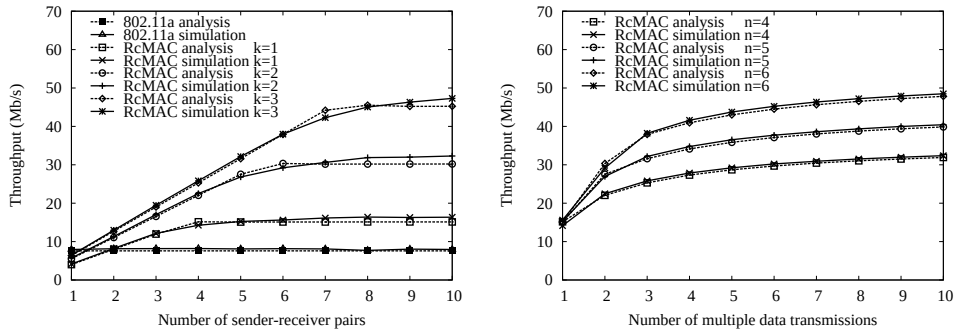
The achievable throughput $S(r)$ during the r rounds is computed by

$$S(r) = \frac{K_{total} \cdot E[P]}{T_{total}}, \tag{10}$$

where $E[P]$ is the packet payload size in bits. Under the assumption that $T_{neg} \approx T_{neg(j)}$ for $j = 1, \cdots, n$, the long-term average throughput S is computed by

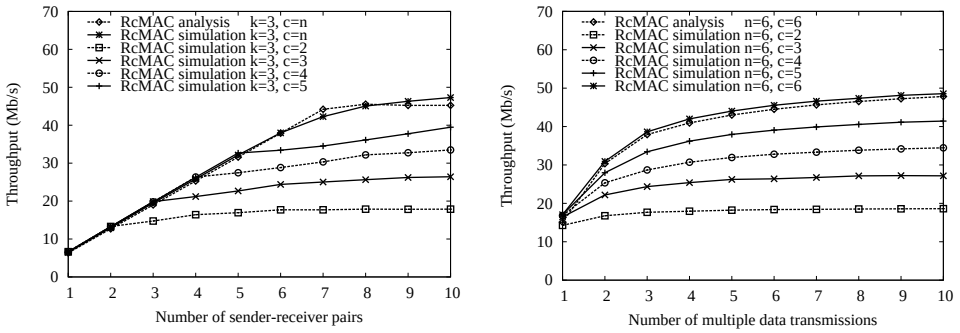
$$S = \lim_{r \to \infty} S(r) = \begin{cases} \frac{k}{T_{neg} + T_{chcb}}, & \text{if } \left\lceil \frac{T_{tx} + 2T_{sw}}{T_{neg} + T_{chcb}} + 1 \right\rceil \leq n; \\ \frac{kn}{k \cdot (\frac{\mathrm{CW}_{\min}}{2} \cdot T_{slot} + T_{data}) + T_{chsw} + 2T_{sw} + T_{neg} + T_{chcb}}, & \text{otherwise.} \end{cases} \tag{11}$$

In Eq. (11), the condition that determines the saturation of the control channel comes from Eq. (5). It is seen that the second and the third condition for the first round in Eq. (11) do not make



(a) The number of sender-receiver pairs (n) (b) The number of multiple data transmissions (k)

Fig. 6. Analysis and simulation results of the RcMAC operation in a single-hop topology (r = 100).



(a) The number of sender-receiver pairs (n) (b) The number of multiple data transmissions (k)

Fig. 7. Simulation results in a wireless network with a small number of data channels.

difference in S, because S is the long-term average throughput obtained by taking $r \to \infty$. Note that the residual time of the channel usage at the last r round in Eq. (8) is also negligible at S.

Remark 1: The control channel is saturated with channel negotiations when $\left\lceil \frac{T_{tx}+2T_{sw}}{T_{neg}+T_{chcb}} + 1 \right\rceil \leq n$. Once the control channel is saturated, the throughput performance of multiple channel networks levels off due to the bottleneck on the control channel.

Remark 2: For $k = \left\lfloor \frac{(n-1)(T_{neg}+T_{chcb})-2T_{sw}-T_{chsw}}{\frac{\mathrm{CW}_{\min}}{2} \cdot T_{slot}+T_{data}} \right\rfloor$, the control channel begins to become saturated. If k is set to a larger value, the control channel is unsaturated. Otherwise, the control channel is congested, and the throughput performance degrades because T_{neg} becomes larger.

C. Throughput Validation

In this section, we numerically obtain the throughput of RcMAC in Eq. (11) and compare it with that obtained by the ns-2 simulations. The simulation parameters are configured to be the

same as those in IEEE 802.11a, and the data packet size is 1000 bytes.

Figure 6 shows the throughput performance for IEEE 802.11a DCF and RcMAC when the number of data channels is the same as the number of sender-receiver pairs. In Figure 6(a), the throughput achieved by IEEE 802.11a DCF is almost constant, regardless of the number of the pairs, because it uses only a single channel. For the multi-channel network under RcMAC, as the number of pairs increases, the throughput performance increases and eventually levels off. The value for which the throughput begins to be saturated matches well with the value of n obtained by Remark 1. It means that it is possible to decide an appropriate value of k by using Remark 1 when the number of pairs is estimated in a multi-channel network. Figure 6(b) shows the throughput performance with respect to the number of multiple data transmissions (k). In both cases, it is seen that the analytical results match well with the simulation results for a variety of n and k.

We next consider the throughput performance in a realistic network with a smaller number of channels. Note that the number of channels is typically smaller than the number of sender-receiver pairs. Figure 7(a) shows the throughput performance when the number of sender-receiver pairs increases. The highest throughput performance is achieved when each pair exclusively exploits a data channel (i.e., $c = n$). When the number of channels is smaller than n, the aggregate throughput performance becomes saturated at a smaller value of throughput. More precisely, in Figure 7(a), it is seen that the throughput linearly increases in accordance with the exclusive channel use case when n is small, and levels off when $n > c$. Figure 7(b) shows the throughput performance on a network with six sender-receiver pairs. The derived analysis results serve as the maximum bound of the throughput performance. As the number of available data channels is smaller, the aggregate throughput performance is worse because the channel contention on data channels becomes severer. These simulation results show that it is possible to decide an appropriate value of k and n for a given configuration of multi-channel networks by using the result for exclusive channel use case derived in this section. An adaptive mechanism for k and an admission control for n would be considered as future work for extending this work.

VI. Discussion

A. Multi-channel Hidden Terminal and Missing Receiver Problems

In a multi-channel environment, the exploitation of multi-channel diversity is impeded by two problems: the multi-channel hidden terminal problem [1] and the missing receiver problem [8]. The multi-channel hidden terminal problem occurs when a node attempts to use a data channel that is currently occupied by other nodes, due to incomplete channel usage information. To solve this problem, RcMAC allows a node to gather channel information from other nodes by overhearing control packets that are exchanged on the control channel when it switches back to the control channel. RcMAC also supports multiple data exchanges on a single data channel. When collisions happen due to channel information incoherence, the carrier sense multiple access with the BEB algorithm is performed, as is done in IEEE 802.11 DCF.

The missing receiver problem occurs when a sender fails to identify the channel on which its intended receiver currently resides, which results in a number of unsuccessful transmission attempts. In RcMAC, when a sender does not know the channel of its receiver, its neighboring nodes cooperatively send an NCTS control frame to let the sender know the current channel of its intended receiver. In addition, when a node leaves a data channel, CHSW and CHCB control frames are sent to prevent this problem by maintaining up-to-date channel information. In the case that any neighboring nodes do not have the channel information for the intended receiver, the sender cannot skip the channel selection phase and is required to retransmit RTS in order to find the intended receiver.

B. Channel Selection Scheme

When all the data channels are already occupied by one or more sender-receiver pairs, a new sender-receiver pair on the control channel has to select any one data channel even when there are other concurrent transmissions. As discussed in Section IV-A, RcMAC simply lets such a node randomly select a data channel during the channel selection phase. In this case, it would be better for the node to select the data channel that has the smallest number of sender-receiver pairs. This approach can help to achieve fair channel utilization among the data channels and eventually improve the aggregate throughput performance.

C. Inconsistent Channel Information

Because a pair of nodes performs channel switching independently, based on the channel information in its local channel tables, inconsistent channel information may cause the nodes to select an improper channel to switch to. For example, a sender may switch to the data channel on which its intended receiver does not reside, and then fails to receive an ACK packet in response to its data transmission. Such channel switching based on inconsistent channel information wastes network resources. To mitigate this undesirable effect, the channel information at each node is reset if it is not updated for $T_{tx} + 2T_{sw}$. In addition, the sender returns immediately to the control channel if it fails to receive an ACK packet in its first attempt on the data channel.

VII. Simulation

To evaluate the performance of RcMAC and compare it with that of IEEE 802.11 DCF, MMAC, AMCP, and AMCP$_k$, we conducted a series of simulations in multi-channel and single-interface environments. In this simulation study, AMCP was configured to transmit one data frame for each switching to a data channel (i.e., k=1), as was originally explored in [8], whereas AMCP$_k$ is a modified version of AMCP, in order to provide a fair comparison with RcMAC, which allows a node to consecutively transmit k data frames (i.e., k>1). It should be noted that IEEE 802.11 DCF uses only a single channel, whereas the other protocols are multi-channel protocols. We implemented the protocols in *ns*-2 (version 2.33), and for more accurate simulation, we modified *ns*-2 such that: 1) the interference perceived at the receiver is the collective aggregate interference from all the concurrent transmissions on its channel, and 2) each node uses a physical carrier sense to determine if the medium is free on each channel.

The default parameters used in the *ns*-2 simulations are presented in Table II. Here, the number of channels is set at 5, and it should be noted that several experiments using Atheros-based 802.11a cards indicate that the number of orthogonal channels is 5 or 6 [28]. One specific channel is assumed to be a designated control channel—except in the case of MMAC, which uses one channel for data exchange as well as channel negotiation. In addition, we used static routing [29] to avoid the effects of routing. In IEEE 802.11 DCF, the RTS/CTS mechanism is enabled; any unspecified factors used in MMAC, AMCP, and AMCP$_k$ follow the guidelines and procedures specified in their original papers. Each source node generates CBR traffic. The simulation time was 50 s, and each data point was obtained by averaging the values of five runs.

TABLE II

DEFAULT PARAMETERS USED IN NS-2 SIMULATIONS.

Propagation	Two-ray	Antenna height	1.5 m	Tx power	24.49 dBm
Channel switching time	2 ms	SNR thresh	10 dB	Queue size	50
Slot time	9 μs	RX thresh	-64.37 dBm	Number of channels	5
SIFS	16 μs	CS thresh	-76.76 dBm	k	3
Basic rate	6 Mbps	Packet size	1,000 bytes	q_{thr}	10
Data rate	12 Mbps	Traffic	CBR	Routing	Static

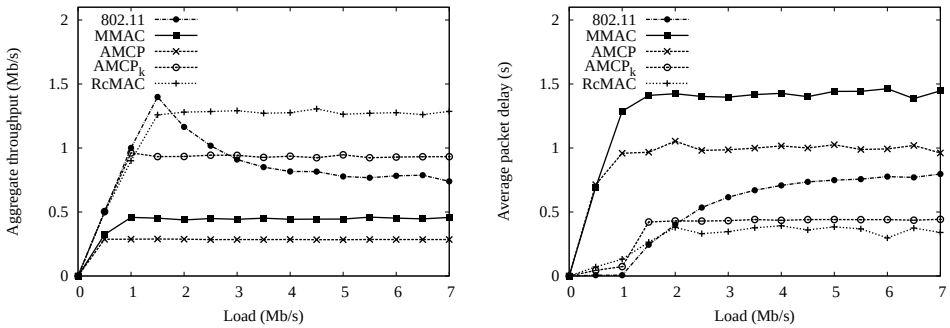
A. Network Topologies

In our simulations, we considered several network topologies, where the transmission range was 250 m, and the carrier sense range was 510 m. The distance between adjacent nodes was set at 200 m. It should be noted that each node could gather channel information only from its direct neighboring nodes within its transmission range. We simulated MAC protocols under various network topologies as follows:

- Chain topology: Seven nodes are placed in a row; the first and the last node on the chain are the source node and the destination node, respectively.
- Cross topology: Two six-hop chain topologies intersect at right angles. Every packet is required to pass the intersection node to reach its destination at the end of each chain topology.
- Grid topology: 100 nodes are placed in a 10 x 10 grid topology, where 20 flows are established from the left-most nodes to right-most nodes, and from the upper nodes to the bottom nodes.
- Single-bell topology: Packets generated from three source nodes are delivered to a destination node via relay by a intermediate node.

B. Throughput and Delay Performance

We measured the aggregate throughput and the average packet delay as a performance metric. The aggregate throughput represents how many bits are delivered from source nodes to the destination nodes during a unit of time. The average packet delay is the average duration for packets to be successfully delivered from the source to the destination node.



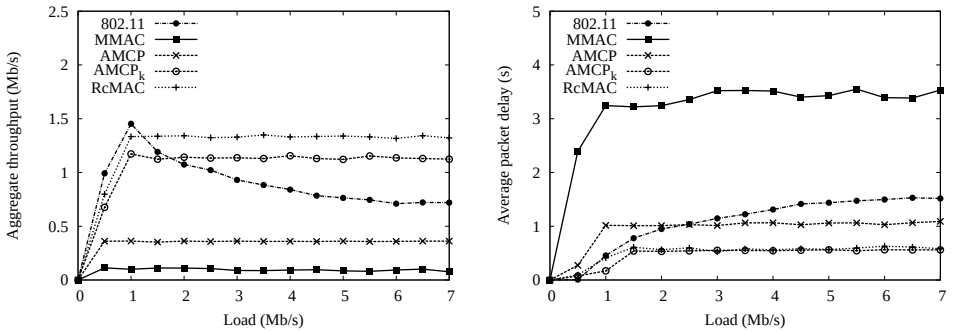
(a) Throughput (b) Delay

Fig. 8. Throughput and delay performance with respect to the offered load in a six-hop chain topology.

1) Chain Topology: Figure 8 shows the throughput and delay performance in a six-hop chain topology. In multi-hop topologies, data packets are delivered to the destination node via intermediate relay nodes along the path. Therefore, the throughput performance greatly depends on the degree to which the interference between adjacent links on the multi-hop path is mitigated by multi-channel diversity. In these simulations, however, the throughputs of MMAC and AMCP were slightly lower than that of IEEE 802.11 DCF, even though they were able to exploit multiple channels for data transmission. One reason for these lower throughputs is that because MMAC is a synchronous protocol and uses a fixed value for the channel reservation time, channel utilization is highly affected by the traffic load of each node, which gradually decreases along the path towards the destination. This uneven traffic load on the path causes MMAC to operate inefficiently on multi-hop networks.

In AMCP, each node defers its channel negotiation for a certain interval in order to prevent the multi-channel hidden terminal problem after it has switched to the control channel. Unfortunately, the interval is extremely short to fully prevent this problem in multi-hop networks, where the intra-flow interference is intense. As a result, RcMAC achieves the highest throughput and the shortest delay in all ranges of the loads offered.

2) Cross and Grid Topologies: Figure 9 shows the throughput and delay performance in a cross topology. The figure shows that the throughput of MMAC is quite low in comparison with the other schemes, mainly due to inefficient channel switching at the intersection node of the cross topology. It should be noted that a node under MMAC needs to remain on the data channel during the data exchange phase even when it does not have any packets destined for



(a) Throughput (b) Delay

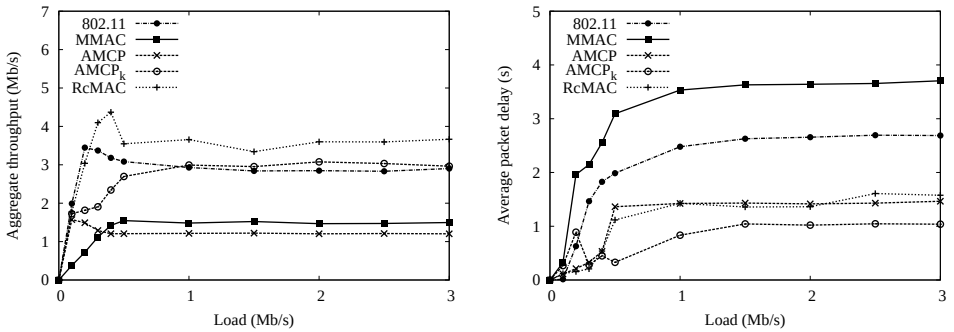
Fig. 9. Throughput and delay performance with respect to the load offered in a cross topology.

the current next-hop node. This inefficiency then becomes significantly more severe when data packets belonging to different traffic flows interleave the queue of nodes.

Conversely, RcMAC allows the intersection node to look for up to N data frames belonging to the traffic flow in its outgoing queue, and to then sequentially transmit these data packets. After completing these transmissions, it immediately searches for another set of frames in its queue in order to initiate the next transmission. Thus, the intersection node can relay the interleaved frames that are destined for different next-hop nodes without unnecessary channel switchings and negotiations. Using this operation, RcMAC achieves the best throughput and delay performance among the schemes being compared.

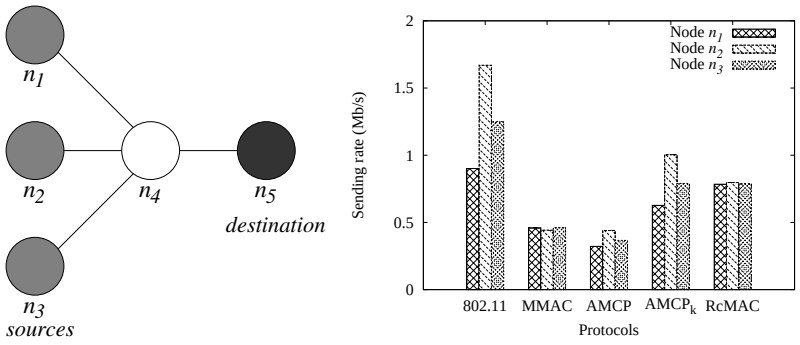
Figure 10 shows the throughput and delay performance for 20 flows in a 10 x 10 grid topology. The figure shows that MMAC, due to its inefficient operation with different traffic flows, as in the cross topology, has the lowest throughput and the longest delay performance. AMCP achieves almost the same throughput performance as IEEE 802.11 DCF; however, it has a longer delay due to the delays incurred after switching back to the control channel. Again, it can be seen that RcMAC significantly outperforms the other schemes in all ranges of the offered loads.

3) Single-bell Topology: To further investigate the issues of starvation and fairness among competing nodes, we considered a single-bell topology in which a relay node is a bottleneck node, as shown in Fig. 11(a). In this topology, each of the source nodes n_1, n_2, and n_3 transmits data packets towards node n_5 via the relay node n_4. Because the relay node n_4 forwards data packets from the three flows at the same time, it may frequently switch channels. If the relay node n_4 performs its channel switching, the nodes n_1, n_2, and n_3 must also change to a new



(a) Throughput (b) Delay

Fig. 10. Throughput and delay performance with respect to the load offered in a grid topology.



(a) Topology (b) Sending rate

Fig. 11. Sending rate of sources n_1, n_2, and n_3, in a single-bell topology where a relay node n_4 forwards data packets from sources to a destination.

channel of node n_4.

Figure 11(b) shows the sending rates that were achieved for each source node. For IEEE 802.11 DCF and AMCP$_k$, one of the source nodes used the wireless channel excessively, and thus, their fairness performance among source nodes was worse, even though their throughput performance was relatively better than that of the others. While both MMAC and RcMAC achieved a higher level of fairness performance, RcMAC achieved the highest throughput performance among the multi-channel protocols. In MMAC, the relay node n_4 can synchronously negotiate with the source nodes during the channel negotiation phase without inconsistency of channels. In RcMAC, even though the relay node n_4 is a bottleneck node, all the source nodes can properly switch to the channel that the relay node n_4 has switched to. Whenever the channel currently used by the relay node is changed, all the other nodes are immediately notified by the CHSW and CHCB

control packets under the receiver-centric channel switching. Because the maximum number of transmissions on a data channel is limited, each node has to return to the control channel, and a new negotiation among competing nodes on the control channel begins. Therefore, RcMAC can achieve long-term fairness even when a relay is a bottleneck node in a single-bell topology.

C. Impact of the Simulation Parameters

1) Effect of Multiple Data Transmissions and Handshakes: We next consider an interesting six-hop bi-directional chain topology, as shown in Fig. 12(a). In this bi-directional chain topology, data packets are generated by nodes A and G, and the packets are delivered to nodes G and A, respectively, via the relay nodes B, C, D, and E. To relay the data packets back and forth, each relay node must decide the channel to switch to, depending on the destination and the channel status.

Figure 12(b) shows the throughput performance with respect to the number of multiple data transmissions. Because the multiple data transmissions reduce the number of channel switchings and allow the nodes to transmit more packets without channel switchings, the network throughput increases with respect to k in this bi-directional chain topology. We also measured the number of negotiation handshakes performed on the control channel. As shown in Fig. 12(c), the handshake of RTS/CTS/CFM corresponds to the cases when none of the senders and none of their neighboring nodes have the channel information for the intended receiver. On the other hand, the handshake of CFM corresponds to the cases when a sender knows the current data channel of its intended receiver. The sender broadcasts a CFM packet and switches to the data channel. As shown in Fig. 12(b), the number of CFM cases is about 85 % of the RTS/CTS/CFM cases. The number of RTS/NCTS/CFM cases supported by neighboring nodes is about 15 % of the RTS/CTS/CFM cases. These results show that RcMAC can reduce channel negotiations to a large degree with the use of these CFM and RTS/NCTS/CFM handshakes.

2) Effect of T_{ch} and q_{thr}: Both T_{ch} and q_{thr} parameters are used for the receivers on a data channel in order to decide when to return to the control channel. A receiver is allowed to remain on a data channel without sending or transmitting for T_{ch}, at the longest, and after that it must switch back to the control channel. In other words, for T_{ch}, it waits to receive data packets, instead of channel switching to extend the time-period in which it can communicate on the current data channel. Figure 13(a) and (b) show the throughput and delay performance as we

varied T_{ch} in a six-hop chain topology. We set $T_{ch} = j \cdot (T_{neg(1)} + T_{sw} + T_{data})$ and vary the parameter j from 1 to 4 in this simulation. In Fig. 13(a) and (b), we can observe that the throughput performance increases as T_{ch} becomes large, and the average packet delay is almost constant regardless of T_{ch}. However, if T_{ch} is set to an extremely large value, both the throughput and delay performance may become worse, because the receiver cannot receive any data packets while waiting on the current data channel. In our simulation, the value of j was set at 2 as a default parameter.

Next, we consider the effect of q_{thr} on the throughput and delay performance. Whenever a relay node has more than q_{thr} packets pending in its outgoing queue, it must return to the control channel in order to relay the packets to the next node. Figure 13(c) shows the throughput performance with respect to q_{thr} in a six-hop chain topology. In the figure, we see that the throughput performance increases linearly as q_{thr} becomes large. The reason for this is that the relay node can receive packets up to q_{thr} on the current data channel without channel switching. However, if q_{thr} is set to an extremely large value, the relay node would not be able to relay packets destined for the next-hop while continuing to receive data packets. The accumulated packets in its outgoing queue may then cause a large packet delay, as shown in Fig. 13(d).

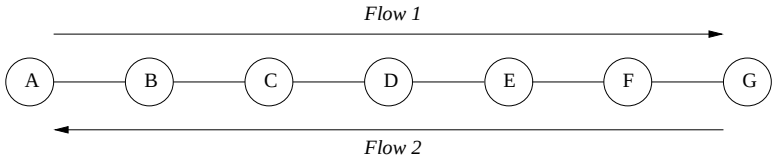
3) Effect of Channels: We then investigated how the number of channels affects the throughput performance in a single-hop network with five sender-receiver pairs. Figure 14(a) shows the aggregate throughput performance with respect to the number of channels. It can be seen that IEEE 802.11 DCF achieves a constant and low throughput regardless of the number of channels, because it uses only a single channel. On the contrary, the throughputs for the other schemes become saturated when the number of channels is six, because each of the five pairs takes one data channel, and they share one control channel. It should be noted that the saturated throughput of RcMAC is higher than that of the other schemes.

Figure 14(b) shows the throughput performance with respect to the channel switching delay in a six-hop chain topology. As the channel switching delay increases, the throughput decreases, because the delay reduces channel utilization. It should be noted that there is a pause in the frame exchanges during a channel switching period. In MMAC, because the nodes perform a small number of channel switchings, the throughput is not greatly affected by the channel switching time. It can be seen that the throughput of AMCP rapidly decreases with respect to the channel switching delay. This decrease is mainly due to the delay overhead caused by the two channel

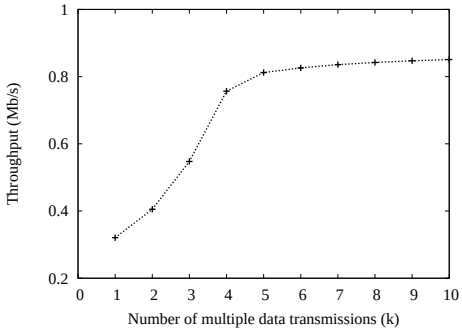
switchings for each data transmission.

VIII. Conclusion

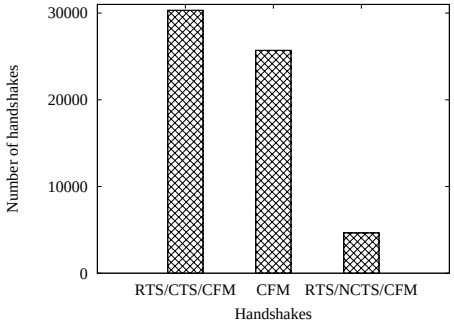
In this paper, we have considered the issues of mitigating interference and improving a network's performance by dynamically exploiting its multi-channel diversity. We proposed a receiver-centric multi-channel MAC protocol (RcMAC) that allows nodes to efficiently utilize multiple channels without requiring unnecessary channel switching. Then, based on the cooperative sharing of channel usage information from neighboring nodes, receiver-centric channel switching enables nodes to reduce both the wait time on the control channel as well as the number of channel switchings. The receiver-centric channel switching operations were performed using either an RTS/NCTS/CFM handshake or a CFM broadcast. To further reduce the overhead of the channel switchings, RcMAC allows a node to transmit multiple data frames for bursty traffic on the current data channel. Through *ns*-2 simulations, we then showed that RcMAC's performance in terms of both network throughput and end-to-end packet delay offered a significant improvement in comparison with other multi-channel MAC protocols. As a future work, we plan to implement RcMAC with a Linux MadWifi driver in order to empirically evaluate its performance in a multi-hop wireless testbed.



(a) Topology

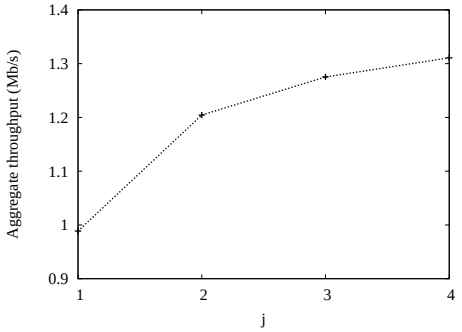


(b) Throughput

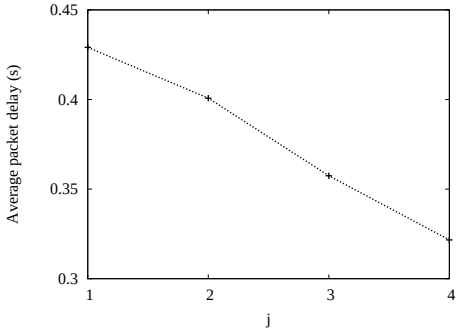


(c) Number of handshakes

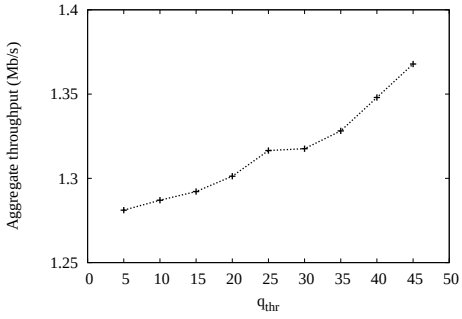
Fig. 12. Throughput performance and number of handshakes in a six-hop bi-directional chain topology.



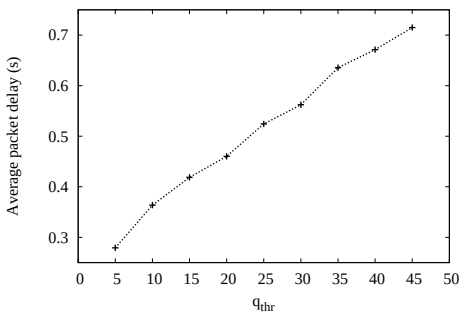
(a) Throughput for T_{ch}



(b) Delay for T_{ch}

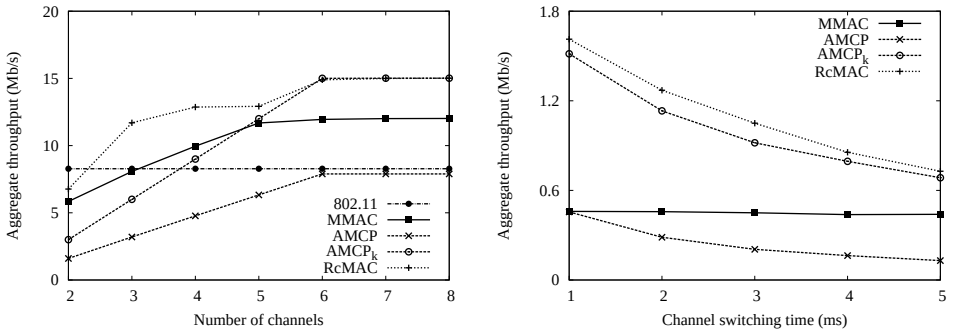


(c) Throughput for q_{thr}



(d) Delay for q_{thr}

Fig. 13. Throughput and delay performance to investigate the point of returning operation in a six-hop chain topology.



(a) Number of channels (b) Channel switching delay

Fig. 14. Throughput performance for the effect of channels in (a) a five-pair single-hop and (b) a six-hop chain topology.

REFERENCES

[1] J. So and N. Vaidya, "Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver," in *Proc. of ACM MobiHoc*, 2004.

[2] J. Chen, S. Sheu, and C. Yang, "A new multichannel access protocol for IEEE 802.11 ad hoc wireless LANs," in *Proc. of IEEE PIMRC*, 2003.

[3] A. Tzamaloukas and J. J. Garcia-Luna-Aceves, "Channel-hopping multiple access," in *Proc. of IEEE ICC*, 2000.

[4] ——, "Channel-hopping multiple access with packet trains for ad hoc networks," in *Proc. of IEEE Mobile Multimedia Communications*, 2000.

[5] H. So, J. Walrand, and J. Mo, "McMAC: A parallel rendezvous multi-Channel (MAC) Protocol," in *Proc. of IEEE WCNC*, 2007.

[6] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks," in *Proc. of ACM MobiCom*, 2004.

[7] J. Patel, H. Luo, and I. Gupta, "A cross-layer architecture to exploit multi-channel diversity with a single transceiver," in *Proc. of IEEE INFOCOM Minisymposium*, 2007.

[8] J. Shi, T. Salonidis, and E. Knightly, "Starvation mitigation through multi-channel coordination in CSMA multi-hop wireless networks," in *Proc. of ACM MobiHoc*, 2006.

[9] T. Luo, M. Motani, and V. Srinivasan, "Cooperative asynchronous multichannel MAC: design, analysis, and implementation," *IEEE Transactions on Mobile Computing*, vol. 8, no. 3, pp. 338–352, 2009.

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

[11] M. Cao, V. Raghunathan, and P. Kumar, "Cross-layer exploitation of (MAC) layer diversity in wireless networks," in *Proc. of IEEE ICNP*, 2006.

[12] T. Al-Meshhadany and W. Ajib, "New CDMA-based MAC protocol for ad hoc networks," in *Proc. of VTC*, 2007 Fall.

[13] W. Chen, T. Huang, Y. Chang, and J. Liu, "An adaptive multi-channel MAC protocol for wireless ad hoc networks," in *Proc. of IEEE ICC*, 2006.

[14] S. Wu, Y. Tseng, C. Lin, and J. Sheu, "A multi-channel MAC protocol with power control for multi-hop mobile ad hoc networks," *The Computer Journal*, vol. 45, no. 1, pp. 101–110, 2002.

[15] N. Jain, S. Das, and A. Nasipuri, "A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks," in *Proc. of IEEE IC3N*, 2001.

[16] R. Maheshwari, H. Gupta, and S. Das, "Multichannel MAC protocols for wireless networks," in *Proc. of IEEE SECON*, 2006.

[17] J. Zhang, G. Zhou, C. Huang, S. Son, and J. Stankovic, "TMMAC: An energy efficient multi-channel MAC protocol for ad hoc networks," in *Proc. of IEEE ICC*, 2007.

[18] R. Huang, H. Zhai, C. Zhang, and Y. Fang, "SAM-MAC: An efficient channel assignment scheme for multi-channel ad hoc networks," *Elsevier Computer Networks*, vol. 52, no. 8, pp. 1634–1646, 2008.

[19] F. Chen, H. Zhai, and Y. Fang, "An opportunistic multiradio MAC protocol in multirate wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 5, pp. 2642–2651, 2009.

[20] Y. Tanigawa, H. Tode, and K. Murakami, "Multi-channel MAC protocols with two transceivers pursuing effective use of vacant resources," in *Proc. of IEEE CCNC*, 2010.

[21] IEEE Computer Society, "Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Standard 802.11*, 2007.

[22] "The network simulator ns-2," http://www.isi.edu/nsnam/ns/.

[23] "The madwifi project," http://madwifi-project.org/.

[24] "Iperf," http://iperf.sourceforge.net/.

[25] W. Hung, K. Law, and A. Leon-Garcia, "A dynamic multi-channel MAC for ad hoc LAN," in *Proc. of 21st Biennial Symposium on Communications*, 2002.

[26] M. Abusubaih, S. Wiethoelter, J. Gross, and A. Wolisz, "A new access point selection policy for multi-rate IEEE 802.11 WLANs," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 23, no. 4, pp. 1–20, 2008.

[27] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[28] P. Kyasanur and N. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 10, no. 1, pp. 31–43, 2006.

[29] "No ad-hoc routing agent (NOAH)," http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/.