Multichannel MAC Protocols for Wireless Networks

Ritesh Maheshwari, Himanshu Gupta and Samir R. Das
Department of Computer Science, Stony Brook University,
Stony Brook, NY 11794-4400, U.S.A.

***Abstract*— In this paper, we develop two new MAC protocols for multichannel operation in wireless ad hoc and mesh networks. The first protocol, *Extended Receiver Directed Transmission protocol* (xRDT) is based on a previously known multichannel solution called *Receiver Directed Transmission* (RDT) that uses a notion of quiescent channel. xRDT solves the problems faced by RDT, such as multichannel hidden terminal and deafness, by using an additional busy tone interface and few additional protocol operations. We also develop a novel single interface solution, called *Local Coordination-based Multichannel* MAC (LCM MAC). LCM MAC performs coordinated channel negotiations and channel switching to provide multichannel support. We demonstrate the effectiveness of these two protocols over two other well-known multichannel protocols – MMAC and DCA – via extensive ns2 simulations.**

I. Introduction

Use of multiple frequency channels offers tremendous potential to improve the capacity of a wireless network. This potential has been recognized in existing standards, such as the IEEE 802.11 [8], that can operate on multiple orthogonal channels. Using multiple frequency channels enables conflict-free transmissions in a physical neighborhood so long as pairs of transmitters and receivers can tune to different non-conflicting channels.

The research community has been addressing the multichannel question using two very different approaches. The first is a *static* approach based on *topology control*. Here, multiple radio interfaces are used on a node and the emphasis is on assigning frequency channels to these radio interfaces such that two nodes that communicate directly in the resulting topology have at least one channel in common. As this approach is necessarily static, the approach is often graph-theoretic and is based on models of interference or protocol behavior, and assumptions on average traffic. The papers in literature using this approach pose the problem as essentially an optimization problem [18], [19], [3], [12], [4], [11].

The other approach is more *dynamic*. It relies on the capability of the radio interface to switch channels on the fly with negligible delay. Here, multiple channels can be utilized even with a single radio interface. Generally speaking, this approach can provide a significant performance benefit over a purely static approach (on a per-interface basis) as it can potentially utilize instantaneous traffic or interference information.

Our goal in this paper is to develop new MAC protocols for ad hoc networks that use such dynamic approaches. We develop two new MAC protocols. The first protocol, called *extended receiver directed transmission* (xRDT), uses one packet interface and one busy tone interface. Note that we differentiate between a *packet interface* and a *tone interface* to contrast our approach with similar approaches that use a separate control channel and thus two packet interfaces (see, for example, the DCA protocol [20]). Tone interfaces are much simpler to implement than packet interfaces. The second protocol, called *local coordination-based multichannel* (LCM) MAC, uses a single packet interface only. We show, via extensive ns-2 simulations, that these two protocols significantly outperform similar protocols that appeared in literature recently, such as DCA [20] and MMAC [22].

The rest of the paper is organized as follows. In the following section similar multichannel approaches in literature are reviewed to provide a context for our work. In Section III the simple receiver directed scheme is described and its problems analyzed. In Section IV, protocol operations are developed to address these problems. This constitutes the xRDT protocol. In Section V, the LCM MAC protocol is developed. In Section VI, ns2 simulation results are presented with realistic traffic scenarios and network models. We finally conclude the paper and outline our future work in Section VII.

II. Background and Related Works

There have been several works on developing new MAC protocols that use multiple channels, and on developing techniques to use the legacy 802.11 MAC with multiple channels efficiently. We review them in this section to provide a context for our work.

A. Dynamic Approaches

In [5] the authors proposed Slotted Seeded Channel Hopping (SSCH), a link-layer protocol that uses unmod-

ified 802.11 MAC layer. Each node in SSCH switches channels at slot-boundaries in a pseudo-random sequence such that channels for neighboring nodes overlap in time periodically. SSCH, requires time synchronization to implement slotting. Also, to be effective, SSCH must adapt its schedule continuously so that frequently communicating nodes overlap in channels frequently.

In [22] the authors proposed the Multichannel MAC (MMAC) protocol which is loosely based on the 802.11 power-saving mechanism [8]. MMAC considers time slotted into *beacon periods* of $100ms$ which are again sub-divided into *ATIM window* of $20ms$ and data window of $80ms$.[1] Nodes tune to a default channel during the ATIM window. A sender picks a receiver to talk to during this window based on the number of packets for each receiver in its interface queue. It then negotiates a channel with the receiver for use during the data window. As all nodes are present in the default channel during the ATIM window, they know about possible future transmissions and the corresponding channels used. Thus, they can make a more informed decision for their own channel negotiations. Nodes switch to their respective selected channels when data window starts. In some sense, MMAC partitions the network into N set of nodes during the data phase, where N is the number of channels. The ATIM window serves as a phase for a node to make the best possible decision on which set it should belong to in the data phase.

Both MMAC and SSCH require network-wide time synchronization to work. They also constraint the nodes to switch channels only at slot boundaries. Thus, they both cannot utilize channel diversity to the maximum extent as they need to stay in a specific channel for fixed periods of time. Much of it is due to network-wide selection of fixed parameters such as slot size in SSCH and beacon periods in MMAC.

One of the earliest works to utilize dynamic channel switching was the Receiver Directed Transmission (RDT) protocol [21]. In RDT, each node has a quiescent channel on which it always listens to when idle. Any transmitter must switch to the receiver's quiescent channel to transmit. We describe RDT in detail in the next section as one of our approaches (xRDT) is based on the RDT paradigm.

The Dynamic Channel Assignment (DCA) protocol [20], unlike the solutions described above, utilizes two packet interfaces. It employs one of the interfaces as a control interface, which is always tuned to a control channel (common to all nodes). This interface allows senders to do a three-way negotiation with the receivers to decide on a channel to be used for data transmission. The selected channel is then used by the other interface to transmit/receive data packet. As one interface is dedicated to the control channel – every node in DCA is informed of channel usage in its neighborhood and thus can make better decisions while negotiating.

However, DCA uses an extra resource – the control interface. Additionally, the right bandwidth for the control channel is traffic dependent. Wide control channel may result in wastage of precious bandwidth, while narrow control channel may become a bottleneck, resulting in wastage of data channel bandwidth.

We also note here that similar protocols were developed in the past, that splits one single channel into multiple subchannels and uses only a subchannel for communication [15]. However, the issue, there, was to reduce the overhead of contention.

B. Static, Multi-radio Approaches

There has been a body of work recently that look at the multi-channel protocols from a different angle. Here, the interest is in using legacy 802.11 protocol with COTS radios – that cannot perform fast channel switching – in multichannel environments. The basic idea is to use multiple radio interfaces assigned (statically, or dynamically – but at a slow time scale) to different channels on each node so that many channels can be used concurrently. However, the channel assignment must be done in a way that interference is minimized. There are several papers in literature that take this broad approach ([18], [19], [3], [12], [4], [11]). While such solutions are amenable to implementation with legacy hardware, the static nature of the solutions limit their effectiveness.

C. Other Related Works

Several other works are also worthy of mention here. The Hop Reservation Multiple Access (HRMA) [23] and Receiver-initiated Channel-hopping with Dual Polling [25] have been proposed for use with frequency hopping spread spectrum (FHSS) wireless cards.

In [17] the authors propose use of multiple radios such that some of them have static channel assignment and the rest do dynamic channel switching. They also propose a new routing strategy based on channel switching and route diversity cost. In [16], the authors develop asymptotic capacity models for multichannel networks with multiple interfaces per node.

III. Receiver Directed Transmission and Performance Issues

In this section, we will develop an understanding of the issues involved in multichannel operations by revisiting the receiver directed transmission (RDT) approach

[1] These possibly could be adapted; but no such protocol exists.

[21]. Our first protocol is based on RDT. However, a straightforward use of RDT with 802.11 MAC results in certain serious performance issues. Our goal in this section is to describe these issues. We start with describing the RDT approach in detail first.

Any dynamic multichannel protocol must ensure that the transmitter and receiver are on the same channel before communicating. To achieve this, it either ensures that they switch to a pre-determined channel at a pre-determined time, or it uses a separate control channel and interface to perform a channel negotiation. This either requires time synchronization or an additional packet interface and channel. RDT uses a clever approach that bypasses both of them.

In RDT, every node is assumed to be have a single interface. Every node also selects (or is assigned) a "well-known" *quiescent* channel for itself. This is the channel the node always listens to when idle. To transmit a packet, a transmitter switches its interface to the quiescent channel of the intended receiver and then transmits using a regular single channel MAC protocol such as 802.11 (with RTS/CTS etc). Following a successful transfer, the sender switches its interface back to its quiescent channel. The protocol assumes that the quiescent channel selection and distribution of this information to the neighboring nodes are done via a separate mechanism. This simplifies the approach greatly in the sense that it is no longer needed that a communicating pair of nodes negotiate a channel beforehand.

A. Multichannel Hidden Terminal Problem

The above scheme presents a new form of the well-known *hidden terminal problem* [24]. Similar problems were also observed in [22] in a slightly different context. When a transmitter A, for data transmission, switches its interface from its quiescent channel p to the receiver B's quiescent channel q, it has no prior information about q's state (i.e., currently ongoing transmissions). For example, there could be another node C in the neighborhood in the same channel q that might be receiving data from a node D that is hidden from A. In case A transmits an RTS for B, it will result in a collision at C.

Note that 802.11 [8] solves the single channel equivalent of this hidden terminal problem by using a virtual carrier sensing mechanism. This uses RTS/CTS exchange and the concept of NAV (*network allocation vector*). We assume that the reader is familiar with the operation of 802.11, and for brevity we do not elaborate on this.

The virtual carrier sensing mechanism, however, is not sufficient to prevent collisions in multichannel environment where only a single interface is used. This is because the control packets such as RTS/CTS could now be sent in different channels and one interface can only work on one channel at a time. Thus, the channel state information in the form of NAV cannot be created. Note that a simple solution to this problem would be for A to wait for the longest packet transmission time before attempting transmission after switching channel. But this is clearly inefficient.

B. Deafness Problem

A second problem, called *deafness*, arises because an intended receiver may currently be in transmit mode, transmitting in the quiescent channel of a third node. This will cause the transmission attempt to fail. In 802.11, this means that the transmission will be retried – after a backoff, that increases exponentially after multiple such failures, suspecting congestion. This wastes network resources and causes unfairness. Also, it is indeed possible that the receiver comes back to its quiescent channel when its current packet transmission is over; however, the transmitter remains in the backoff unaware of this event, waiting for the backoff timer to expire. By the time the latter indeed attempts the next retry, the receiver could have switched to another channel for transmission.

Similar deafness problems have been noted before in the context of directional antennas [6]. Similar situation happens in the basic 802.11 protocol as well, but one *additional* hop away. This situation has been referred to as *information asymmetry* in literature [26] and is one cause of fairness problems in 802.11.

Our simulations (not described here due to lack of space) indicate that both these problems occur frequently enough at a high load causing throughput to decrease. The decrease is often large enough that RDT with multiple channels perform poorer than single channel 802.11! The observations in this section motivated us to work on two different approaches to solve the problems associated with multichannel environment. The first protocol, xRDT, tries to *solve* the problems using extra resources but using the same framework as the RDT approach. The second protocol, LCM-MAC, instead *prevents* the above mentioned problems from occurring, by using a novel technique of channel negotiation based on local coordinations. We describe these approaches and their relative merits in the following sections.

IV. xRDT: Extended Receiver Directed Transmission Scheme

xRDT adds two mechanisms to RDT to address the multichannel hidden terminal and deafness problems

that we describe in this section. For the purpose of understanding the protocol assume that the quiescent channel advertisement is done by a separate mechanism. This mechanism is explained separately.

A. Addressing Multichannel Hidden Terminal

One solution of the hidden terminal problem is to implement a "channel memory" that helps propagate the channel state to a potential transmitter *at all times*. An easy and well-known way to implement "channel memory" is by using *busy tones* [24] [7] [27]. Busy tones are single frequency tones used for signaling. The advantage of using busy tones relative to using a separate control channel (as in DCA [20]) is that the issue of determining the right bandwidth to allocate for the control channel does not arise. Also, the channel gain for both the data channel and the busy tone channel (or, the control channel for that matter) must be the same for the techniques to operate correctly. This is relatively easier to achieve for a single frequency tone. In addition, hardware requirement is simpler as only an additional tone interface is needed instead of an additional packet interface.

We assume that there is a different tone channel b_c for each data channel c. However, one single tone interface is sufficient. A receiver, when receiving a data packet on channel c turns on the tone in corresponding busy tone channel b_c. This enables a potential transmitter that has just come to channel c to learn about any receiver in the neighborhood by sensing on the busy tone channel b_c. If the busy tone channel is indeed found busy, a transmitter would defer its transmission on the data channel. This deferment is designed exactly similar to the collision avoidance mechanism in 802.11 [8] that uses a variation of the p-persistent protocol [10]. For brevity, this mechanism is not discussed here.

Note that use of the busy tone prevents any collision of data packets.

B. Addressing Deafness

There are simple solutions to deafness; but they all require additional resources. For example, note that deafness arises because a radio interface is half-duplex. So, in transmit mode it is deaf to any reception. So, if two interfaces are used, one for transmission and the other for reception [17], the problem is solved trivially. In this work, we take an approach that simply softens the impact of deafness instead of completely eliminating it.

Recall from Section III that a receiver might return back to its quiescent channel while the transmitter is still in backoff. A notification that a potential receiver is available to receive data can preempt this backoff and ready the transmitter to transmit immediately following. One way to achieve this would be for the "deaf" nodes to broadcast a *Data Transmission Complete* or DTC notification message in its own quiescent channel. This will ensure that all potential transmitters (who may be in backoff) come to know of the receiver's availability. They now can break out from backoff and start the transmission process. A contention resolution is necessary to resolve between multiple such transmitters. This can be done simply by following the contention resolution scheme in 802.11 [8].

Notice that DTC does not prevent deafness from occurring – transmitters will still send RTS to deaf receivers – but it will alleviate it by capitalizing on the fact that the deaf node will return to its quiescent channel before switching to another channel for transmission. This small window of opportunity is utilized by making the deaf receiver send the DTC to "wake up" the backed off transmitters.

C. Complete Protocol Description

Briefly, the Extended RDT (xRDT) protocol is the same as RDT, except that now (i) there is a receiver busy tone on the appropriate busy tone channel, and (ii) a DTC message after the data transfer is over in the quiescent channel. The details follow.

1) Start of Transfer: In xRDT, every node listens to its quiescent channel when idle. A transmitter A switches channel to the receiver B's quiescent channel q. Then it senses carrier on both q and the busy tone channel b_q using the two interfaces. If any channel is found busy, it uses a contention mechanism similar to 802.11, which we do not describe for brevity. When the channels are found idle (after the appropriate contention resolution, if any), A sends RTS to B on channel q. B after receiving RTS turns on busy tone for b_q. The busy tone works as an implicit acknowledgment for the RTS. On hearing the busy tone, A transmits DATA to B. When DATA transmission is complete, B turns off the busy tone. Then again, after an appropriate interframe spacing, busy tone is turned on briefly as an acknowledgment. This stays for a normal ACK packet duration. The setting of the interframe spacing guarantees that no other transmission in the vicinity can start in the interim. Absence of this busy tone-based acknowledgment signals the transmitter that retransmission is necessary. The retransmission is tried after a backoff. This backoff again is similar to 802.11.

2) End of Transfer: After the transfer is complete, A switches back to its own quiescent channel (say, p). After proper interframe spacing, it broadcasts a DTC message in this channel if the channel is idle (both p and b_p). Channel sensing eliminates the possibility of DTC collision. If the channel is sensed busy, DTC transmission is deferred till the channel becomes idle. If DTC is indeed sent, any other node C waiting for A in channel p cancels its current backoff timer, erases all backoff and contention window related states, and acts as if it is attempting to transmit a fresh packet.

3) Next Transfer: If A also has another packet to transmit, it prepares to transmit that packet right after transmitting the DTC for the previous transmission. This means again – as in 802.11 – setting the contention window to the minimum value and picking a random backoff time. A transmits – after switching to the receiver's quiescent channel – if its backoff expires earlier than C's. Else, C transmits to receiver A. If this is the case, when C to A transfer is complete, A again attempts to transmit its packet after completing its *remaining* backoff time.

D. Selection of Quiescent Channel

A good quiescent channel selection for all nodes is required to maximize concurrency in the network. When the relative traffic on each link of the network is known a priori, optimized assignment of quiescent channels is akin to the problem of coloring vertices of a graph with available channels. More specifically, when k channels are available, the problem of assigning quiescent channels for maximum concurrency in the network can be shown to be equivalent to the problem of finding a maximum weighted k-cut [1] in the G^2 graph, where G is the original communication graph and G^2 has edges between node pairs (i, j) such that distance between i and j in G is at most 2. We can use the existing approximation algorithm [9] for the Max k-cut problem to do channel assignment in this case.

In the more realistic case where traffic is varies dynamically, we propose a periodic quiescent channel selection mechanism using channel load as a criterion. Each node measures the load on all channels by snooping during its idle time. Traffic directed towards itself is discounted when calculating load on its current quiescent channel, q. If load on the least loaded channel, l, is lower than the load on q, l is chosen as the new quiescent channel. To avoid oscillations, the difference is mandated to be above a threshold for any change to take place. This approach has been used in our evaluations.

V. Local Coordination-based Multichannel (LCM) MAC

The receiver directed approach described before requires an additional busy tone interface. In this section, we develop an alternative approach called LCM MAC where busy tones are not used and *each node has only one interface*. In LCM MAC, the neighboring nodes go through local coordinations to generate *transmission schedules*. A transmission schedule consists of a period when only control packets are transmitted (also called *control window*) followed by a period when only data packets are transmitted (a *data window*). Two basic rules are followed:

- All control packets are transmitted in the same channel during the control window. All nodes in a neighborhood are tuned to this same channel at this time.
- All data packets are transmitted concurrently in different channels during the data window.

The first rule helps ensure that nodes become aware of transmissions in the neighborhood (this avoids the Multichannel Hidden Terminal problem as well as the Deafness problem). Data packets are transmitted concurrently at different channels to exploit parallelism.

The common channel used in the control window is called the *default* channel. Unlike the quiescent channel in xRDT, the default channel in this case is common to all nodes. The default channel is used as a control channel during the control window and as a data channel during the data window.

The key idea in LCM protocol is to setup transmission schedules without the use of any time synchronization. Senders use a contention resolution mechanism similar to 802.11 to gain access to the default channel during the control window. A sender then negotiates a channel to be used during the data window with the intended receiver. Once the negotiation is over, it releases the channel to let other senders contend for its access.

When control window gets over, the communicating nodes switch to their respective selected channels and exchange DATA and ACK. This constitutes the data window. After data window is complete, all these nodes switch back to the default channel for another round of negotiations. The time line showed in Figure 1 illustrates a simple working scenario of LCM MAC.

The protocol is similar in some details to the MACA-P [2] protocol and the POWMAC [14] protocol for transmit power control. LCM also has some similarities with the MMAC [22] protocol in channel negotiations. However, MMAC follows a rigid schedule and the negotiations are for long term. Thus, its benefit is limited by traffic

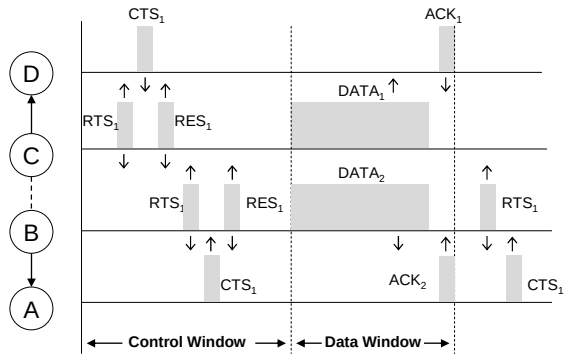


Fig. 1. A example time line showing the working of LCM MAC in a simple three hop network with 2 channels. The subscript for each packet indicates the channel in which the packet is sent.

conditions. MMAC also requires tight time synchronization for the protocol to work whereas LCM has no such requirements.

A. Detailed Protocol Operation

In this section, we describe the methodology used for setting up a schedule without global synchronization. We also talk about how negotiations are performed during the control window and how channels are selected to avoid conflicts during the data window. Later, we talk about the limitations of the protocol and how to overcome them.

1) Control Window Operation: Any node unaware of a control/data window schedule can propose a schedule. Otherwise, it follows a schedule it already knows. When a node has a packet to send and is unaware of any schedule, it transmits an RTS (as in 802.11) in the default channel with a proposed schedule. This node is called a *master* node. The schedule can be defined by two additional fields in the RTS packet: (i) time left for data window to start (control window duration) and (ii) the data window duration. A RTS packet also contains a list of free channels at the sender for transmission during the data window. We use a concept of *Multichannel NAV* for this purpose. This is the same as NAV used in 802.11, except that now NAV is a vector with one element for each channel. If the NAV for a particular channel is set, then that channel is deemed busy otherwise, it is free.

If n is the number of channels, then only n negotiations are possible in a neighborhood in the control window - as it will exhaust the list of channels for data communication. Thus, if T_{neg} is the time needed for a successful negotiation (explained next), then control window duration is set to n times T_{neg}. But, in case the master node has heard only k ($< n$) negotiations in the last control window, it sets the control window size to (k+1) times T_{neg}. The data window size is set to the time needed for DATA-ACK exchange with the proper interframe spacings.

On receiving RTS, the receiver can accept the schedule by replying with a CTS. The CTS also contains a channel id selected from the channel list in RTS. This selected channel is one of the free channels at both sender and receiver's positions. The CTS also contains the schedule information.

When sender receives a CTS, it transmits another packet called RES (for *reserve*) containing the schedule and the selected channel id. RES is needed to allow all neighbors of the transmitter to be aware of the channel to be used for communication. Any node hearing a CTS or RES packet will set its Multichannel NAV for the channel whose id is included in the packet for an appropriate duration of time (end of data window). All such nodes also note the schedule mentioned in the packet and follow that schedule unless they have already been following another schedule. Thus the schedule is propagated to all the nodes in the one-hop neighborhood of the sender and the receiver.

In case, the receiver cannot find a common free channel from the channel list in the RTS, it replies with a channel id value of -1 in the CTS to signal the sender to retry in the next schedule. The sender does not respond with RES to such a CTS message. Any neighboring node hearing such a CTS ignores it.

2) Data Window Operation: After a successful RTS and CTS exchange, the transmitter-receiver pair has now agreed on the channel to be used and all potential interferers have set their NAVs for this channel to allow this transmission to proceed without conflict. Other nodes who overheard the CTS/RES packets are now free to start their own negotiations (using RTS/CTS/RES exchange) as long as they can finish the negotiation within the control window and their data transmission takes time less than or equal to the data window length mentioned in the schedule. At the end of control window the transmitter-receiver pairs switch to their selected channels. This starts the data window. DATA and ACK are transmitted in the selected channel. At the end of the data window, the transmitter-receiver pairs return to the default channel implicitly signaling the start of another control window.

The nodes who heard a schedule in the control window but are not participating in data transmissions, remain in the default channel during the data window.

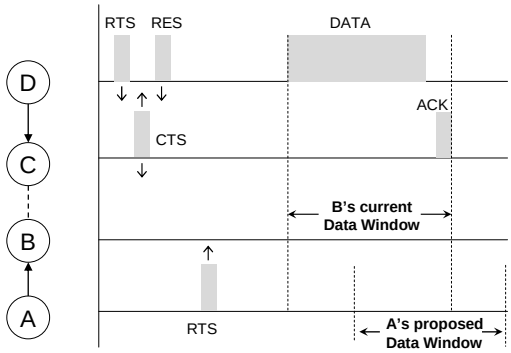


Fig. 2. Example demonstrating how schedules can be adapted in the LCM MAC protocol with the use of inflexible bit.

They are not allowed to communicate during this time.

3) Similarities With 802.11: The other details of the protocol are similar to 802.11. For example, it follows the identical interframe spacings and collision avoidance strategies by using physical carrier sense and backoff before transmitting an RTS. If there is no CTS in response to an RTS, RTS is retransmitted with an increased backoff exactly similar to 802.11. This covers for the case where there is an RTS collision because of a significant load. If the backoff gets over in the same control window, the RTS is retransmitted; but if it gets over during the data window, the node waits for control window to start again.

4) Adapting Schedules: Consider the scenario in Figure 2, node D sends RTS to node C setting up a schedule which node C accepts by sending a CTS. This CTS is heard by node B and it also starts following that schedule. Now, if node A, hidden from C and D, sends an RTS to B, it may propose a different schedule. The protocol dictates that B has to follow C's schedule. Thus, it cannot reply to A. However, in some cases, with some additional protocol operations the schedule can be adapted.

To explain this we come back to the notion of *master node*. A master node is one that proposes a *new* schedule in its RTS. Thus, in Figure 2, D could be the master node, or it could have itself heard a schedule from some other node in its neighborhood and might be following that schedule. We can say the same for node A. If node A is not a master node, then it is impossible for the $A-B$ communication to go on at this time as they both are following different schedules and they cannot violate them. But, in case A is a master node, B can actually reply proposing a change in A's schedule to match its own. Because A is a master node, it can very well accommodate its schedule to facilitate its data transfer to B.

To achieve this, we introduce an *inflexible bit* field in RTS. The inflexible bit is set to zero if the sender of RTS can change its schedule if needed (i.e., it is a master node). On receiving the RTS with inflexible bit set to zero, B can send a CTS with a changed schedule to match its own. On receiving it, A would send a RES with the changed schedule. Note that because neighboring nodes are not meant to follow schedules proposed in RTS, but only in CTS and RES, changing schedules in this manner does not affect any neighborhood node.

B. Improving Efficiency

The constraint in LCM MAC that all nodes in a neighborhood tune to the default channel during control window ensures that all neighbors are aware of what channels are being used in the neighborhood. But, this also leads to an inefficiency – all the non-default channels remain idle during the control window, resulting in a considerable loss of bandwidth.

We counter this inefficiency by letting senders transmit a maximum of m (>1) DATA packets to the receivers per negotiation; or in effect, increasing the data window duration by a factor of m. This amortizes the loss of bandwidth during the control window over a number of data transmissions. But the value of m should be chosen carefully. For example, if the size of data window is too large, many senders might run out of packets to send to the respective receivers – resulting in more wastage of bandwidth. If the data window duration is too short, the protocol may still remain inefficient.

We propose a way to adapt the data window duration. For every node, if d is the intended receiver for the next packet in the outgoing interface queue, then the node counts the number of packets in the queue that have d as the next hop. This information is included by all nodes in their CTS and RES packets for the neighbors to learn about their future transmission requirements. Thus, at the start of the next control window, all nodes have information about the requirements of other neighboring nodes. A *master* node can use a statistic of these values (e.g., average) to decide on a suitable value m to be used in current schedule. To eliminate outliers, we also constraint m to be no more than the number of channels.

When data window starts, a sender, after sending the current DATA packet can proceed to transmit the next one only if: a) it receives an ACK for the current DATA packet and b) the time left for data window to end is

greater than the time required for another DATA/ACK exchange.

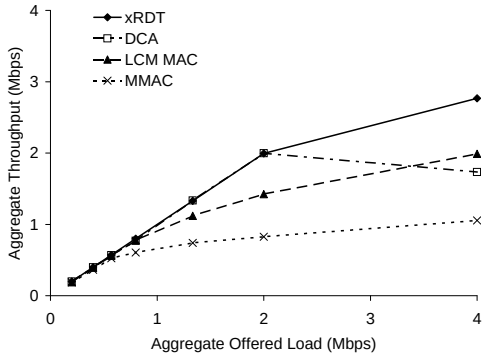
C. Question of Starvation

It is possible that some nodes suffer from periods of starvation. To see this, consider Figure 2 again. It is possible that A is inflexible, and thus A to B transfer cannot proceed with the current control and data window for B. In that case, A simply has to continue trying in its next control window. It is possible that B again follows a schedule set by another node hidden from A. In such cases B can starve – as it is not able to receive packets intended for it. As the average route length increases in the network, the probability of more than one schedule operating in the network increases. This, as a result, could cause more starvation. However, because each transmission starts with a contention period that uses randomization, it is unlikely that a single node will suffer for a long time.

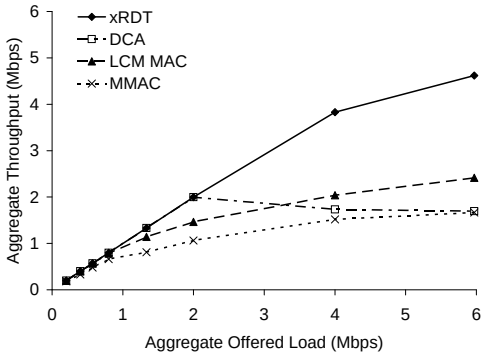
For our implementation, we used an ad hoc approach to alleviate the problem. Whenever a node suffers from long periods of starvation, it disrupts a negotiation which is imposing a second schedule on it by causing a control packet collision. For example, after hearing a CTS from C (intended for D), B can cause a collision at C when C was supposed to receive the RES from D. In case, A attempts to send a RTS to B again, it can safely reply with a CTS now.

VI. Simulation-Based Performance Evaluation

We evaluate xRDT and LCM MAC and compare them against two known multichannel protocols, DCA and MMAC, using the $ns2$ simulator with CMU wireless extensions [13]. The simulations were performed for two scenarios with different density of nodes. The following common parameters were used in each experiment, all the radio parameters being $ns2$ defaults. The radio power and threshold levels are such that transmission range of each node is 250m, the carrier sense range is 500m and the nominal bit rate of each channel is 1 Mbps. The two-ray ground reflection model is used to model radio propagation. The number of nodes in each scenario is 100 and positioned at random locations. There are 50 CBR flows with randomly selected source-destination pairs. Statically chosen shortest path routing is used. The data packet sizes are 1000 bytes. The data packet generation rate for each flow is varied to vary the load in the network and simulations are done for different number of channels. Each simulation is performed long enough for the output statistics to stabilize. Each data point in the plots is an average of five runs where each run used a different randomly generated topology.



(a) 6 Channels.

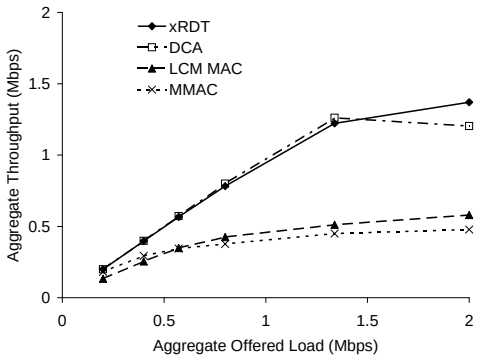


(b) 13 Channels.

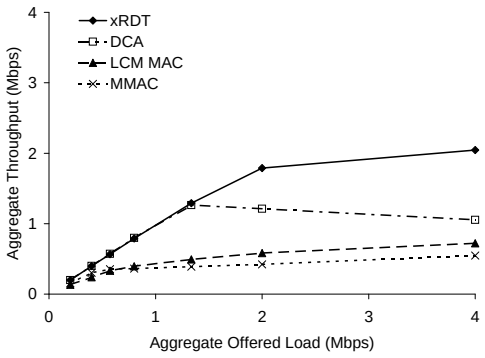
Fig. 3. Throughput vs. load in ns2 simulations with 1Mbps channels, 100 nodes in a 500m $\times$ 500m area.

We have simulated four protocols xRDT, LCM MAC, DCA and MMAC. For DCA, if there are k channels available, then 1 channel is designated as the control channel and the rest $k-1$ are used as data channels. For MMAC, the specified values in [22] of 80ms for data window and 20ms for the ATIM window are used. Two 100 node network scenarios – with different density – are simulated. The first (second) scenario is created by randomly placing 100 nodes in a 500m $\times$ 500m (1000m $\times$ 1000m) area. 6 and 13 channel results are presented in Figures 3 and 4 for each of the two scenarios.

As expected, the two-interface protocols (xRDT and DCA) usually perform better than the single-interface protocols (LCM and MMAC). xRDT provides much superior performance among all protocols. DCA is a close second, except at high loads. Also, DCA's performance suffers for the 13 channel experiments, likely because of the control channel becoming a bottleneck resulting in wastage of data channel bandwidth. Note that both



(a) 6 Channels.



(b) 13 Channels.

Fig. 4. Throughput vs. load in ns2 simulations with 1Mbps channels, 100 nodes in a 1000m $\times$ 1000m area.

xRDT and DCA use two interfaces, although the second interface in xRDT is a much simpler busy tone interface. So, it is fair to compare them together.

Similarly, it is fair to compare LCM MAC and MMAC together as they both use one interface. LCM MAC performs better than (or similar to) MMAC at all times, although LCM is much better in the 500$\times$500 scenario. The degradation in 1000$\times$1000 scenario is probably due to starvation problem as mentioned in section V-C. Also note that, inspite of using time synchronization, MMAC's performance in some scenarios is not good at low loads. This is due to the large data window size. At low loads senders run out of packets to send to the receivers present in their current channel. As they cannot change channel until the end of data window, this results in wastage of bandwidth. LCM MAC also does not give proportional improvement with the increase in channels. We will discuss this aspect momentarily.

One goal of our work is also to showcase the performance benefit of using multiple channels in wireless networks. To demonstrate this aspect, we plotted the

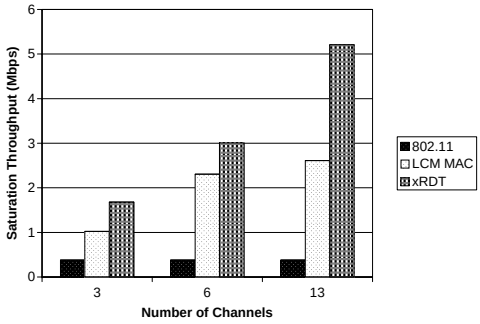


Fig. 5. A chart comparing saturation throughput of xRDT and LCM MAC with varying number of channels in a 500m $\times$ 500m area. The chart also includes single channel 802.11 for baseline comparison.

average saturation throughput of xRDT and LCM MAC in Figure 5 with varying number of channels (m) and compared them against single channel 802.11. Single channel 802.11 is only used for baseline comparison. The earlier mentioned scenario with 100 nodes in 500$\times$500 area is used for this plot.

In Figure 5 note that xRDT's performance increases almost linearly with increase in number of channels. This demonstrates the efficiency of the xRDT scheme. It does not face control channel bottleneck issues as in DCA, nor does it face any control period inefficiencies as in LCM MAC or MMAC. Also, note that xRDT, in fact, provides more than k times throughput relative to 802.11 while using k channels. This is due to lesser control packet overheads in xRDT.

LCM MAC also provides substantial improvement over 802.11, slightly less than k times for the 3 and 6 channel experiments. But, the saturation throughput does not increase proportionately for 13 channels. This is due to the earlier mentioned problem about loss of bandwidth during the control window (Section V-B), where only one channel (the default) is used. In fact, the loss of bandwidth is $O(k^2)$ as it is proportional to the product of number of non-default channels ($k-1$) and control window size, and control window size is itself proportional to k. Thus, we advocate the use of LCM MAC for small number of channels and dense networks only.

VII. Conclusions and Future Work

We have developed two multichannel MAC protocols – xRDT with a packet and a busy tone interface, and LCM MAC, with just one interface. Both protocols show superior performance relative to similar protocols proposed in literature that use similar resources in terms of interfaces. In particular, xRDT provides a far superior performance relative to control channel based protocol

(DCA), but only using busy tones instead of control channels. LCM MAC also provides superior or at par performance relative to a similar protocol (MMAC), but uses an entirely asynchronous operation.

Some of the issues not discussed in this paper are the effects of mobility, non-negligible channel switching delay and different data packet sizes as well as mechanisms for broadcasts in our protocols. We intend to study them in our future work. Effects of mobility is expected to be minimal on xRDT as it uses busy tones to avoid collisions, while it may affect LCM MAC somewhat because of its large data window size. There should be no effect of varying data packet sizes on both the protocols. The channel switching delay is expected to affect only xRDT to any significant extent as it follows a per-packet channel switching approach. For LCM the delay is amortized over multiple packets. Also, in LCM, broadcast packets can be sent during the control window. But no intelligent mechanism exists for xRDT, currently the broadcast must be done individually in each channel.

Our future work will also involve implementing and testing the studied protocols in real wireless testbeds using different software-based MAC platforms. Also, several intelligent approaches are possible to solve the starvation problem in LCM MAC that is worth investigating. Our ad hoc solution only serves as a first step.

REFERENCES

[1] C. Moore A. Coja-Oghlan and V. Sanwalani. Max k-cut and approximating the chromatic number of random graphs. In *Proc Thirtieth Int Coll Automata, Languages, Programming, Lecture Notes in Computer Science 2719, Springer*, pages 200–211, 2003.

[2] A. Acharya, A. Misra, and S. Bansal. Maca-p : a mac for concurrent transmissions in multi-hop wireless networks. In *Proc. of IEEE PerCom*, pages 505–508, March 2003.

[3] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proc. of Broadnets*, 2004.

[4] M. Alichery, R. Bhatia, and L. Li. Joint Channel Assignment and Routing for throughput optimization in Multi-Radio wireless mesh networks. In *ACM MobiCom*, 2005.

[5] 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.

[6] R. R. Choudhury and N. Vaidya. Deafness: A mac problem in ad hoc networks when using directional antennas. In *ICNP*, Berlin, October 2004.

[7] Jing Deng and Zygmunt J. Haas. Dual busy tone multiple access (DBTMA): A new medium access control for packet radio networks. In *Proceedings of IEEE ICUPS*, 1998.

[8] IEEE Standards Department. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE standard 802.11–1997, 1997.

[9] Alan M. Frieze and Mark Jerrum. Improved approximation algorithms for max k-cut and max bisection. In *Proceedings of the 4th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 1–13, London, UK, 1995. Springer-Verlag.

[10] L. Kleinrock and F. A. Tobagi. Packet switching in radio channels: Part-I - carrier sense multiple access modes and their throughput-dely characteristics. *IEEE Transactions in Communications*, COM-23(12):1400–1416, 1975.

[11] M. Kodialam and T. Nandagopal. Characterizing the Capacity Region in Multi-Radio, Multi-Channel Wireless Mesh Networks. In *ACM MobiCom*, 2005.

[12] M. K. Marina and S. R. Das. A topology control approach to channel assignment in multi-radio wireless mesh networks. In *Proc. of Broadnets Symposium*, 2005.

[13] S. McCanne and S. Floyd. Network simulator ns-2. In *http://www.isi.edu/nsnam/ns/*, 1997.

[14] Alaa Muqattash and Marwan Krunz. Powmac: A single-channel power-control protocol for throughput enhancement in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 23(5), May 2005.

[15] A. Nasipuri N. Jain and S. R. Das. A multichannel mac protocol with receiver-based channel selection for multihop wireless networks. In *IEEE IC3N*, pages 431–439, 2001.

[16] N. Vaidya P. Kyasanur. Capacity of multi-channel wireless networks: Impact of number of channels and interfaces. In *Mobicom*, 2005.

[17] N. Vaidya P. Kyasanur. Routing and interface assignment in multi-channel multi-interface wireless networks. In *WCNC*, 2005.

[18] A. Raniwala and T. Chiueh. Architechture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *IEE Infocom*, 2005.

[19] A. Raniwala, K. Gopalan, and T. Chiueh. Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(2):50–65, 2004.

[20] S.-L.Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu. A new multi-channel MAC protocol with on-demand channel assignment for multi-hop mobile ad hoc networks. In *In I-SPAN*, 2000.

[21] N. Shacham and P. King. Architectures and performance of multichannel multihop packet radio networks. *IEEE Journal on Selected Areas of Communication*, SAC-5(6):1013–1025, 1987.

[22] J. So and N. Vaidya. Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proc. ACM MobiHoc*, 2004.

[23] Z. Tang and J. J. Garcia-Luna-Aceves. Hop-reservation multiple access (hrma) for multichannel packet radio networks. In *Proc. of IEEE IC3N*, October 1998.

[24] F. A. Tobagi and L. Kleinrock. Packet switching in radio channels: Part-II - the hidden terminal problem in carrier sense multiple-access models and the busy-tone solution. *IEEE Transactions in Communications*, COM-23(12):1417–1433, 1975.

[25] A. Tzamaloukas and J. J. Garcia-Luna-Aceves. A receiver-initiated collision-avoidance protocol for multi-channel networks. In *IEEE Infocom*, 2001.

[26] B. Sadeghi V. Gambiroza and E. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *Proc. of ACM Mobicom*, 2004.

[27] C. Wu and V. Li. Receiver-initiated busy-tone multiple access in packet radio networks. In *SIGCOMM '87: Proceedings of the ACM workshop on Frontiers in computer communications technology*, pages 336–342, New York, NY, USA, 1988. ACM Press.