Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver *

Jungmin So
Dept. of Computer Science, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
jso1@uiuc.edu

Nitin Vaidya
Dept. of Electrical and Computer Eng., and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
nhv@uiuc.edu

ABSTRACT

This paper proposes a medium access control (MAC) protocol for ad hoc wireless networks that utilizes multiple channels dynamically to improve performance. The IEEE 802.11 standard allows for the use of multiple channels available at the physical layer, but its MAC protocol is designed only for a single channel. A single-channel MAC protocol does not work well in a multi-channel environment, because of the *multi-channel hidden terminal problem*. Our proposed protocol enables hosts to utilize multiple channels by switching channels dynamically, thus increasing network throughput. The protocol requires only one transceiver per host, but solves the multi-channel hidden terminal problem using temporal synchronization. Our scheme improves network throughput significantly, especially when the network is highly congested. The simulation results show that our protocol successfully exploits multiple channels to achieve higher throughput than IEEE 802.11. Also, the performance of our protocol is comparable to another multi-channel MAC protocol that requires multiple transceivers per host. Since our protocol requires only one transceiver per host, it can be implemented with a hardware complexity comparable to IEEE 802.11.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms

*This research is supported in part by National Science Foundation and Motorola.

MobiHoc'04, May 24–26, 2004, Roppongi, Japan.

Keywords

Ad hoc network, medium access control, multi-channel

1. INTRODUCTION

IEEE 802.11 standard for wireless LAN [1] provides multiple channels available for use. The IEEE 802.11b physical layer (PHY) has 14 channels, 5MHz apart in frequency [1]. However, to be totally non-overlapping, the frequency spacing must be at least 30MHz. So channels 1, 6 and 11 are typically used for communication in current implementations, and thus we have 3 channels for use. IEEE 802.11a provides 12 channels, 8 in the lower part of the band for indoor use and 4 in the upper part for outdoor use [2].

By exploiting multiple channels, we can achieve a higher network throughput than using one channel, because multiple transmissions can take place without interfering. However, the MAC protocol of IEEE 802.11 Distributed Coordinate Function (DCF) is designed for sharing a single channel between hosts. Designing a MAC protocol that exploits multiple channels is not an easy problem, due to the fact that each of current IEEE 802.11 device is equipped with one half-duplex transceiver. The transceiver is capable of switching channels dynamically, but it can only transmit or listen on one channel at a time. Thus, when a host is listening on a particular channel, it cannot hear communication taking place on a different channel. Due to this, a new type of hidden terminal problem occurs in this multi-channel environment, which we refer to as *multi-channel hidden terminal problem* (we identify this problem in more detail in Section IV). So a single-channel MAC protocol (such as IEEE 802.11 DCF) does not work well in a multi-channel environment where nodes may dynamically switch channels.

In this paper, we propose a MAC protocol which enables hosts to dynamically negotiate channels such that multiple communication can take place in the same region simultanesouly, each in different channel. The network we consider is an ad hoc network that does not rely on infrastructure, so there is no central authority to perform channel management. The main idea is to divide time in to fixed-time intervals using beacons, and have a small window at the start of each interval to indicate traffic and negotiate channels for use during the interval. A similar approach is used in IEEE

802.11's power saving mechanism (PSM) [1], as explained in Section III.

As reviewed in Section II, several MAC protocols are proposed that use multiple channels to improve throughput. But all of them either require multiple transceivers per host or do not solve the multi-channel hidden terminal problem, resulting in degraded performance. To the best of our knowledge, this is the first protocol that requires only one transceiver per host, but still solves the hidden terminal problem in a multi-channel environment. Since the protocol requires one transceiver per host, it can be implemented with a hardware complexity comparable to IEEE 802.11, unlike other multi-channel MAC protocols that require multiple transceivers.

The rest of the paper is organized as follows. Section II reviews the related work. Section III provides some background information. Section IV identifies the multi-channel hidden terminal problem, and explains the problem of using a single-channel MAC protocol in a multi-channel environment. Section V presents our proposed protocol in detail. Section VI describes the simulation model we use, and also discusses the results of our simulations. Section VII discusses some issues in our protocol and possible improvements. Finally, Section VIII concludes the paper.

2. RELATED WORK

There are many related papers that study the benefit of using multiple channels. Dual Busy Tone Multiple Access [3] divides a common channel into two sub-channels, one data channel and one control channel. Busy tones are transmitted on a separate control channel to avoid hidden terminals, while data is transmitted on the data channel. This scheme uses only one data channel and is not intended for increasing throughput using multiple channels.

Hop Reservation Multiple Access [4] is a multi-channel protocol for networks using slow frequency hopping spread spectrum (FHSS). The hosts hop from one channel to another according to a predefined hopping pattern. When two hosts agree to exchange data by an RTS/CTS handshake, they stay in a frequency hop for communication. Other hosts continue hopping, and more than one communication can take place on different frequency hops. Receiver Initiated Channel-Hopping with Dual Polling [5] takes a similar approach, but the receiver initiates the collision avoidance handshake instead of the sender. These schemes can be implemented using only one transceiver for each host, but they only apply to frequency hopping networks, and cannot be used in systems using other mechanisms such as direct sequence spread spectrum (DSSS).

Nasipuri et al. [6] propose a multi-channel CSMA protocol with "soft" channel reservation. If there are N channels, the protocol assumes that each host can listen to all N channels concurrently. A host wanting to transmit a packet searches for an idle channel and transmits on that idle channel. Among the idle channels, the one that was used for the last successful transmission is preferred. In [7] the protocol is extended to select the best channel based on signal power observed at the sender. These protocols require N transceivers for each host, which is very expensive.

Wu et al. [8] propose a protocol that assigns channels dynamically, in an on-demand style. In this protocol, called *Dynamic Channel Assignment* (DCA), they maintain one dedicated channel for control messages and other channels for data. Each host has two transceivers, so that it can listen on the control channel and the data channel simultaneously. RTS/CTS packets are exchanged on the control channel, and data packets are transmitted on the data channel. In RTS packet, the sender includes a list of preferred channels. On receiving the RTS, the receiver decides on a channel and includes the channel information in the CTS packet. Then, DATA and ACK packets are exchanged on the agreed data channel. Since one of the two transceivers is always listening on the control channel, multi-channel hidden terminal problem does not occur. This protocol does not need synchronization and can utilize multiple channels with little control message overhead. But it does not perform well in an environment where all channels have the same bandwidth. When the number of channels is small, one channel dedicated for control messages can be costly. In case of IEEE 802.11b, only 3 channels are available, so having one control channel results in 33% of the total bandwidth as the control overhead. On the other hand, if the number of channels is large, the control channel can become a bottleneck and prevent data channels from being fully utilized.

Jain et al. [9] propose a protocol that uses a scheme similar to [8] that has one control channel and N data channels, but selects the best channel according to the channel condition at the receiver side. The protocol achieves throughput improvements by intelligently selecting the data channel, but still has the same disadvantages as DCA.

Compared to the above works, our protocol operates with one transceiver per host. Also, it does not require a dedicated control channel. Instead, our scheme requires clock synchronization among all the hosts. At the start of each interval, we require all hosts to listen to a common channel in order to exchange traffic indication messages. During this interval hosts do not exchange data packets. So this duration of time is an overhead in our scheme. However, as we will see in later sections, it achieves better throughput than maintaining a separate control channel.

3. PRELIMINARIES

In this section, we present some background information on IEEE 802.11's DCF and power saving mechanism.

3.1 IEEE 802.11 Distributed Coordination Function (DCF)

In IEEE 802.11 DCF, a node reserves the channel for data transmission by exchanging RTS/CTS messages with the target node. When a node wants to send packets to another node, it first sends an RTS (Ready to Send) packet to the destination. The receiver, on processing the RTS, replies by sending a CTS (Clear to Send) packet to the sender. RTS and CTS packets include the expected duration of time for which the channel will be in use. Other hosts that overhear these packets must defer their transmission for the duration specified in the packets. For this reason, each host maintains a variable called the *Network Allocation Vector* (NAV)

that records the duration of time it must defer its transmission. This whole process is called *Virtual Carrier Sensing*, which allows the area around the sender and receiver to be reserved for communication, thus avoiding the *hidden terminal problem* [10].

Figure 1 illustrates the operation of IEEE 802.11 DCF. When node B is transmitting a packet to node C, node A overhears the RTS packet and sets its NAV until the end of ACK, and node D overhears the CTS packet and sets its NAV until the end of ACK. After the transmission is completed, the stations wait for DIFS and then contend for the channel. In this figure, node B is a *hidden terminal* to node D. Without virtual carrier sensing, D would not know of B's transmission. So D may start transmitting a packet to C while B is transmitting, which results in a collision at C.

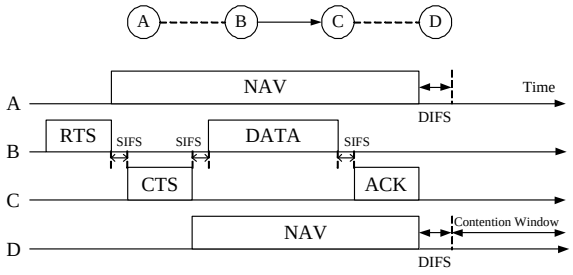


Figure 1: Operation of IEEE 802.11 DCF.

If a node has a packet to send but observes the channel to be busy, it performs a *random backoff* by choosing a backoff counter no greater than an interval called the *contention window*. Each host maintains a variable cw, the contention window size, which is reset to a value CW_{min} when the node is initiated. Also, after each successful transmission, cw is reset to CW_{min}. After choosing a counter value, the node will wait until the channel becomes idle, and start decrementing the counter. The counter is decremented by one after each "time slot", as long as the channel is idle. If the channel becomes busy, the node will freeze the counter until the channel is free again. When the backoff counter reaches zero, the node will try to reserve the channel by sending an RTS to the target node. Since two nodes can pick the same backoff counter, the RTS packet may be lost because of collision. Since the probability of collision gets higher as the number of nodes increases, a sender will interpret the absence of a CTS as a sign of congestion. In this case, the node will double its contention window to lower the probability of another collision.

Before transmitting a packet, a node has to wait for a small duration of time even if the channel is idle. This is called *interframe spacing*. Four different intervals enable each packet to have different priority when contending for the channel. SIFS, PIFS, DIFS, and EIFS are the four interframe spacings, in order of increasing length. A node waits for a DIFS before transmitting an RTS, but waits for a SIFS before sending a CTS or an ACK. Thus, an ACK packet will win the channel when contending with RTS or DATA packets because the SIFS duration is smaller than a DIFS.

3.2 IEEE 802.11 Power Saving Mechanism

In this section, we describe IEEE 802.11 PSM to explain how ATIM windows are used. A node can save energy by going into *doze* mode. In doze mode, a node consumes much less energy compared to normal mode, but cannot send or receive packets. It is desirable for a node to enter the doze mode only when there is no need for exchanging data. In IEEE 802.11 PSM, this power management is done based on *Ad hoc Traffic Indication Messages* (ATIM). Time is divided into beacon intervals, and every node in the network is synchronized by periodic beacon transmissions. So every node will start and finish each beacon interval at about the same time.

Figure 2 illustrates the process of IEEE 802.11 PSM. At the start of each beacon interval, there exists an interval called the ATIM window, where every node should be in the awake state. If node A has buffered packets for B, it sends an ATIM packet to B during this interval. If B receives this message, it replies back by sending an ATIM-ACK to A. Both A and B will then stay awake for that entire beacon interval. If a node has not sent or received any ATIM packets during the ATIM window (e.g., node C in Figure 2), it enters doze mode until the next beacon time.

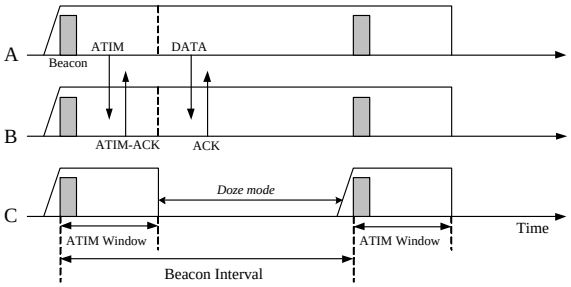


Figure 2: Operation of IEEE 802.11 PSM.

4. ISSUES IN MULTI-CHANNEL ENVIRONMENT

In this section, we describe a new type of hidden terminal problem [10] pertaining to multi-channel environment, which we call the *multi-channel hidden terminal problem*. For the sake of illustration, we start with a simple multi-channel MAC protocol that does not address this problem.

The protocol is similar to [8], except it assumes each node has one transceiver. Suppose there are N channels available. One channel is dedicated for exchanging control messages (control channel), and all the other channels are for data. When a node is neither transmitting or receiving, it listens to the control channel. When node A wants to transmit a packet to node B, A and B exchange RTS and CTS messages to reserve the channel as in IEEE 802.11 DCF. RTS and CTS messages are sent on the control channel. When sending an RTS, node A includes a list of channels it is willing to use. Upon receiving the RTS, B selects a channel and includes the selected channel in the CTS. After that, node A and B switch their channels to the agreed data channel and exchange the DATA and ACK packets. When

this handshake is done, node A and B immediately switch to the control channel.

Now consider the scenario in Figure 3. Node A has a packet for B, so A sends an RTS on Channel 1 which is the control channel. B selects Channel 2 for data communication and sends a CTS back to A. The RTS and CTS messages should reserve Channel 2 in the transmission ranges of A and B, no collision occurs. However, when node B sent the CTS to A, node C was busy receiving on another channel, so it did not hear the CTS. Not knowing that B is receiving on Channel 2, C might initiate a communication with D, and end up selecting Channel 2 for communication. This will result in collision at node B.

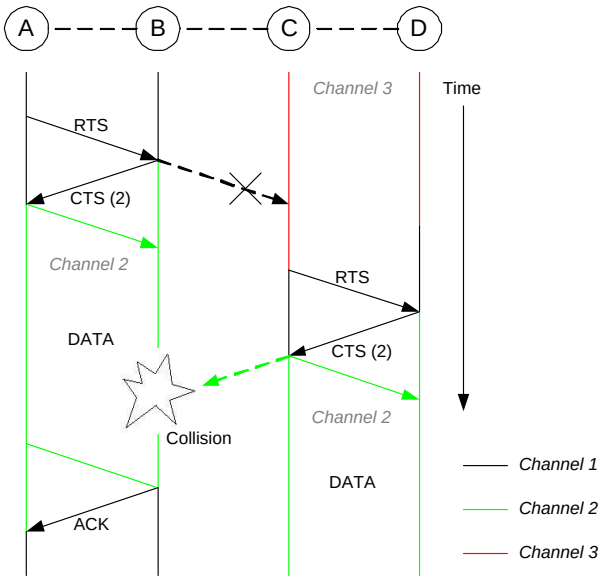


Figure 3: A scenario showing the hidden terminal problem in multi-channel environment. Channel 1 is the control channel. Since C was listening on one of the data channels when B sent a CTS, C does not know about communication between A and B.

The above problem occurs due to the fact that nodes may listen to different channels, which makes it difficult to use virtual carrier sensing to avoid the hidden terminal problem. If there was only one channel that every node listens to, C would have heard the CTS and thus deferred its transmission. Thus, we call the above problem the *multi-channel hidden terminal problem*. As presented in the next section, we solve this problem using synchronization, similar to IEEE 802.11 PSM.

5. PROPOSED MULTI-CHANNEL MAC (MMAC) PROTOCOL

In this section, we present our proposed scheme. Before describing the protocol in detail, we first summarize our assumptions.

- N channels are available for use and all channels have the same bandwidth. None of the channels overlap, so the packets transmitted on different channels do not interfere with each other. Hosts have prior knowledge of how many channels are available.
- Each host is equipped with a single half-duplex transceiver. So a host can either transmit or listen, but cannot do both simultaneously. Also, a host can listen or transmit on only one channel at a time. So when listening to one channel, it cannot carrier sense on other channels. Unlike our scheme, many other multi-channel MAC protocols require each host to have multiple transceivers [11, 9, 8].
- The transceiver is capable of switching its channel dynamically. The time elapsed for switching the channel is $224\mu s$ [1].
- Nodes are synchronized, so that all nodes begin their beacon interval at the same time. Clock synchronization can be achieved using either out-of-band solutions such as GPS, [12], or in-band solutions. If an out-of-band solution can be used, no additional overhead is imposed on the channels used by our protocol. However, if an in-band solution is used, we need to consider the overhead of synchronization. To model this overhead, we implement beaconing mechanism similar to IEEE 802.11 timing synchronization function (TSF) [1] in our simulations (all beacons are sent on a common *default channel* explained later). The issue of clock synchronization is discussed further in Section 7.

Now we describe our proposed scheme in detail. From now on, our protocol will be referred as *Multi-channel MAC* (MMAC).

5.1 Preferable Channel List (PCL)

Each node maintains a data structure called the *Preferable Channel List* (PCL), that indicates which channel is preferable to use for the node. PCL records the usage of channels inside the transmission range of the node. Based on this information, the channels are categorized into three states.

- *High preference* (HIGH): This channel has already been selected by the node for use in the current beacon interval. If a channel is in this state, this channel must be selected. For each beacon interval, at most one channel can be in this state at each node.
- *Medium preference* (MID): This channel has not yet been taken for use in the transmission range of the host. If there is no HIGH state channels, a channel in this state will be preferred.
- *Low Preference* (LOW): This channel is already taken by at least one the node's immediate neighbors. To balance the channel load as much as possible, there is a counter for each channel in the PCL to record how many source-destination pairs plan to use the channel for the current interval. If all channels are in LOW state, a node selects the channel with the smallest count.

The channel states are changed in the following way:

- All the channels in the PCL are reset to MID state when the node is powered up, and at the start of each beacon interval.
- If the source and destination nodes agree upon a channel, they both record the channel to be in HIGH state.
- If a node overhears an ATIM-ACK or ATIM-RES packet (explained in the next section), it changes the state of the channel specified in the packet to be LOW, if it was previously in the MID state. When the state of a channel changes from MID to LOW, the associated counter is set to one. If the channel was previously in HIGH state, it stays in the HIGH state. If the channel was already in the LOW state, the counter for the channel is incremented by one.

5.2 Channel Negotiation during ATIM Window

In MMAC, periodically transmitted beacons divide time into beacon intervals. A small window called the *ATIM window* is placed at the start of each beacon interval (we use the term "ATIM" as in IEEE 802.11 PSM, although it is used for a different purpose in our protocol). The nodes that have packets to transmit negotiate channels with the destination nodes during this window. In the ATIM window, every node must listen to the *default channel*. The default channel is one of the multiple channels, which is predefined so that every node knows which channel is the default channel. During the ATIM window, all nodes listen on the default channel, and beacons and ATIM packets are transmitted on this channel. Note that outside the ATIM window, the default channel is used for sending data, similar to other channels.

If node S has buffered packets destined for D, it will notify D by sending an ATIM packet. S includes its preferable channel list (PCL) in the ATIM packet. D, upon receiving the ATIM packet, selects one channel based on the sender's PCL and its own PCL. As explained in the next section, the receiver's PCL has higher priority in selecting the channel. After D selects a channel, it includes the channel information in the ATIM-ACK packet and sends it to S. When S receives the ATIM-ACK packet, it sees if it can also select the channel specified in the ATIM-ACK. S can select the specified channel only except when S has already selected another channel (according to rules for selecting the channel, explained in the subsequent section). If S selects the channel specified in the ATIM-ACK, S sends an ATIM-RES packet to the D, with S's selected channel specified in the packet. The ATIM-RES (ATIM-Reservation) is a new type of packet used in our scheme, which is not in IEEE 802.11 PSM. The ATIM-RES packet notifies the nodes in the vicinity of S which channel S is going to use, so that the neighboring nodes can use this information to update their PCL. Similarly, the ATIM-ACK packet notifies the nodes in the vicinity of D. After the ATIM window, S and D will switch to the selected channel and start communicating by exchanging RTS/CTS.

If S cannot select the same channel as D, because it has already selected another channel, it cannot send packets to D during the beacon interval. It has to wait for the next beacon interval to negotiate channels again. Even though S finishes transmitting all the scheduled packets on the selected channel during the beacon interval, it has to buffer all the packets destined for D until the next beacon interval. Since this can be a waste of bandwidth, we may want to let S send packets to D by switching its channel to the same channel as D in the beacon interval. This issue is discussed in more detail in Section VII.

When multiple nodes start sending ATIM packets at the beginning of a beacon interval, ATIM packets will collide with each other. To avoid such collisions, each node waits for a random backoff interval before transmitting an ATIM packet. The backoff interval is chosen in the range $[0, CW_{min}]$. As RTS and CTS packets, ATIM and ATIM-ACK packets also include NAV information to avoid hidden terminal problems in a multi-hop network.

Note that the receiver can always select a channel for use. Even if all the channels are selected for use in the receiver's transmission range, the receiver can select one of the channels. This is possible because the sender and receiver still exchange RTS/CTS before sending DATA packet, after the ATIM window. If two source-destination pairs that are closely placed choose the same channel, they will have to contend with each other just as in original IEEE 802.11.

Power saving is not the main goal of our protocol, but a node may save power by going into *doze* mode, if it has not transmitted or received ATIM packets during the ATIM window. The possibility of integration with IEEE 802.11 PSM is one advantage of our protocol. However, in our simulations, nodes do not go into doze mode.

5.3 Rules for Selecting the Channel

When a node receives an ATIM packet, it selects a channel and notifies the sender by including the channel information in the ATIM-ACK packet. The receiver tries to select the "best" channel based on information included in the sender's PCL (preferable channel list) and its own PCL. By the best channel we mean the channel with the least scheduled traffic, as elaborated below. This selection algorithm attempts to balance the channel load as much as possible, so that bandwidth waste caused by contention and backoff is reduced. For this reason, we count the number of source-destination pairs that have selected the channel by overhearing ATIM-ACK and ATIM-RES packets and select the one with the lowest count. This scheme assumes that every source-destination pair will deliver the same amount of traffic in a beacon interval, which may not be true. A better approach may be to count the number of packets scheduled to be transmitted on the channel in the beacon interval. To do this, the source needs to include the number of pending packets in the ATIM packet. We take the former approach in this paper and discuss the latter approach in section VI.

Here we describe the channel selection algorithm in detail. Suppose that node A has packets for B and thus sends an ATIM packet to B during the ATIM window, with A's PCL included in the packet. On receiving the ATIM request from A, B decides which channel to use during the beacon inter-

val, based on its PCL and A's PCL. The selection procedure used by B is described as follows.

1. If there is a HIGH state channel in B's PCL, this channel is selected.
2. Else if there is a HIGH state channel in A's PCL, this channel is selected.
3. Else if there is a channel which is in the MID state at both A and B, it is selected. If there are multiple channels in this state, one is selected arbitrarily.
4. Else if there is a channel which is in the MID state at only one side, A or B, it is selected. If there are multiple of them, one is selected arbitrarily.
5. If all of the channels are in the LOW state, add the counters of the sender's PCL and the receiver's PCL. The channel with the least count is selected. Ties are broken arbitrarily.

After selecting the channel, B sends an ATIM-ACK packet to A, specifying the channel it has chosen. When A receives the ATIM-ACK packet, A will see if it can also select the channel specified in the ATIM-ACK packet. If it can, it will send an ATIM-RES packet to B, with A's selected channel specified in the packet. If A cannot select the channel which B has chosen, it does not send an ATIM-RES packet to B.

The process of channel negotiation and data exchange in MMAC is illustrated in Figure 4. During the ATIM window, A sends ATIM to B and B replies with ATIM-ACK indicating to use channel 1. This ATIM-ACK is overheard by C, so channel 1 will be in LOW state in C's PCL. When D sends ATIM to C, C selects channel 2. After the ATIM window, the two communications (between A and B, and C and D) can take place simultaneously.

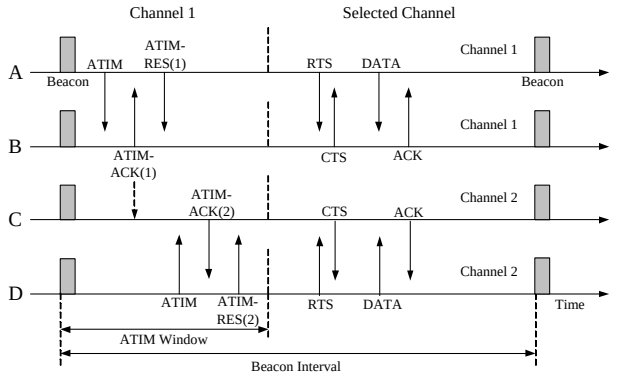


Figure 4: Process of channel negotiation and data exchange in MMAC.

6. PERFORMANCE EVALUATION

In this section we evaluate the performance of our protocol by simulation. We compare our scheme with IEEE 802.11, and the Dynamic Channel Assignment (DCA) protocol [8] (DCA was explained in section II). Recall that the DCA protocol uses a separate channel for exchanging control messages and uses other channels for data. This approach is also taken by [9, 13]. We used two metrics to evaluate the performance of our protocol.

1. *Aggregate throughput over all flows in the network:* Our protocol is expected to increase the total throughput of the network by exploiting multiple channels. Thus, this metric will directly show how our protocol achieves this goal. Ideally, a multi-channel MAC will improve the total throughput by a factor of N over a single-channel MAC given that N data channels are available. This throughput can be achieved if every node has N transceivers. However, with one transceiver per node, the ideal throughput cannot be achieved due to the overhead required for negotiating channels and avoiding the hidden terminal problem. As mentioned earlier, the goal of our protocol is to achieve performance benefit from using multiple channels with one transceiver per node.
2. *Average packet delay over all flows in the network:* Average packet delay is the duration between the time when the Link layer of the sender receives a packet to send, and the time the packet reaches the destination. So the delay is the sum of delays for queueing, backoff, channel negotiation and transmission delay. The queue size at each node is 50 packets. We ignore lost packets in the average delay metric.

6.1 Simulation Model

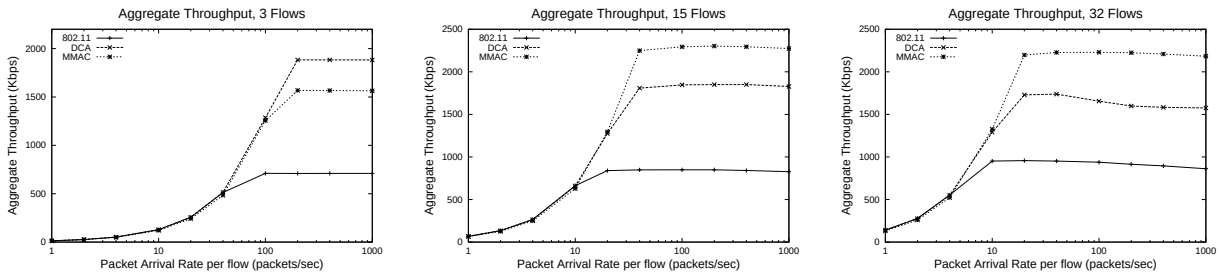
For simulations, we have used ns-2 [14] with CMU wireless extensions [15]. Simulations are performed in two network scenarios, wireless LAN and multi-hop networks. The bit rate for each channel is 2Mbps. The transmission range of each node is approximately $250m$ and the beacon interval is set to $100ms$. Each source node generates and transmits constant-bit rate (CBR) traffic. Each simulation was performed for a duration of 40 seconds. Each data point in the result graphs is an average of 30 runs.

Unless otherwise specified, we assume 3 channels. Also we assume packet size is 512 bytes, and ATIM windows are $20ms$ unless specified otherwise. The parameters we vary are: number of nodes in the network, the packet arrival rate of CBR traffic, ATIM window size, and number of channels.

6.1.1 Wireless LAN

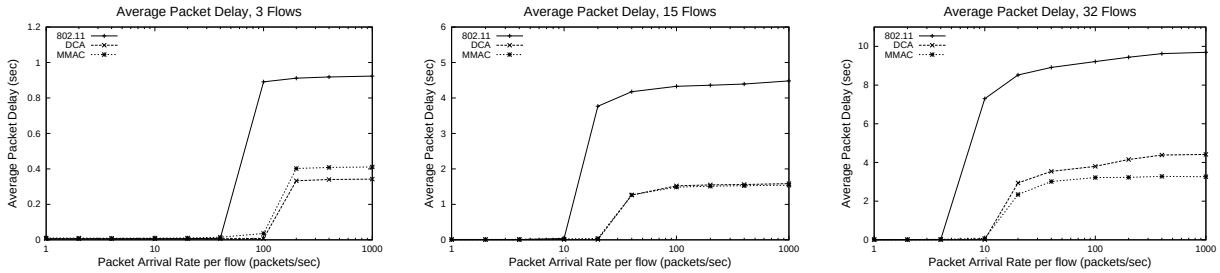
In the simulated wireless LAN, all nodes are within each other's transmission range. So every source node can reach its destination in a single hop. The number of nodes we used are 6, 30, and 64. For each scenario, half of the nodes are sources and the other half are destinations. So a source has at most one destination. The impact of a source having multiple destinations or a destination having multiple sources is not studied in this scenario, but it is studied in the multi-hop network scenario.

First, we examine the throughput and packet delay varying the network load. We use the packet arrival rate of CBR



(a) 6 nodes (b) 30 nodes (c) 64 nodes

Figure 5: Aggregate Throughput vs. Packet Arrival Rate in a wireless LAN.



(a) 6 nodes (b) 30 nodes (c) 64 nodes

Figure 6: Average Packet Delay vs. Packet Arrival Rate in a wireless LAN.

flows to vary the network load. After that, we study the impact of ATIM window size and number of available channels on the throughput.

6.1.2 *Multi-hop network*

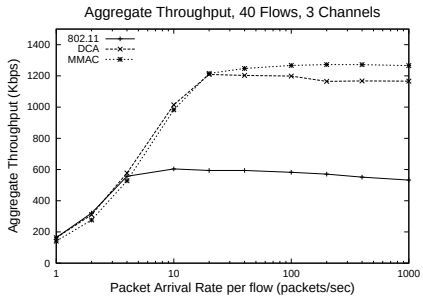
For a multi-hop network, 100 nodes are randomly placed in a $500m \times 500m$ area. 40 nodes are randomly chosen to be sources, and 40 nodes are chosen to be destinations. A node may be the source for multiple destinations and a node may be the destination for multiple sources. In a multi-hop network, we study the situation where different traffic loads are present in different regions, which is not done in the wireless LAN scenario.

6.2 Simulation Results

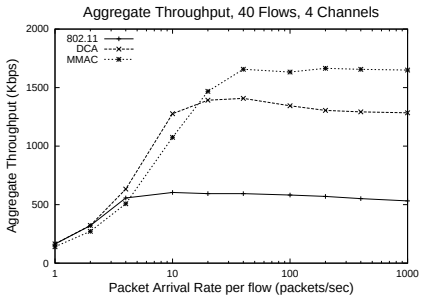
Simulation results are presented in this section. In the graphs, the curves labeled as "802.11" refer to original IEEE 802.11 single channel MAC, the curves labeled as "DCA" indicate the DCA protocol from [8], and the curves labeled as "MMAC" indicate our proposed scheme.

First we present results from simulations performed for a wireless LAN. Figure 5 shows the aggregate throughput of different protocols as the network load increases. The network sizes are 6, 30, and 64 nodes in Figure 5(a), (b), and (c) respectively. When network load is low, all protocols perform similarly. As network load draws near saturation, MMAC performs significantly better than IEEE 802.11, and also does better than DCA. Since there are 3 channels, DCA uses 1 channel for control packets and other 2 channels for data. By using this separate control channel, DCA achieves almost twice the throughput of IEEE 802.11. But as the number of channel increases, the throughput improvement of DCA for the added channel becomes less, because of bottleneck on control channel, as we will see later. MMAC uses all 3 channels for data exchange, but cannot achieve 3 times as much throughput compared to IEEE 802.11 because of its overhead for channel negotiation. The overheads in MMAC are periodic beacon transmissions and ATIM packets. As the graphs show, MMAC performs 20%-30% better than DCA. The throughput improvement of MMAC over DCA may not be dramatic, but it is important that MMAC achieves this throughput using only a single transceiver per node.

Figure 6 shows the average packet delay of the protocols as the network load increases. The difference between IEEE 802.11 and other protocols in delay is due to the fact that with only one channel, a packet has to wait longer to use the channel when the network load is high. When comparing DCA and MMAC, MMAC shows higher delay in the network scenario with 6 nodes. Then the delay of the two protocols becomes similar with 30 nodes, and MMAC outperforms

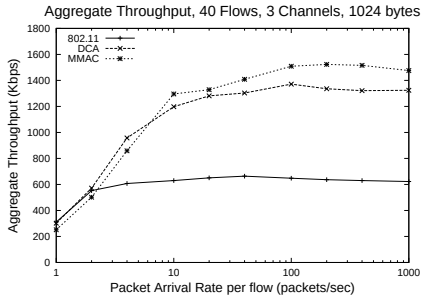


(a) 3 channels

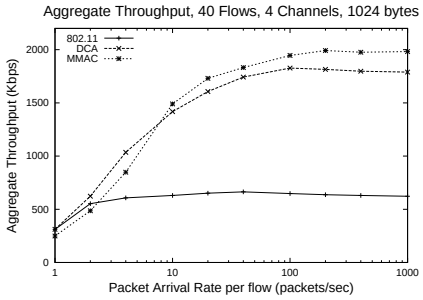


(b) 4 channels

Figure 7: Aggregate Throughput vs. Packet Arrival Rate in a multi-hop network. Packet size is 512 bytes.



(a) 3 channels



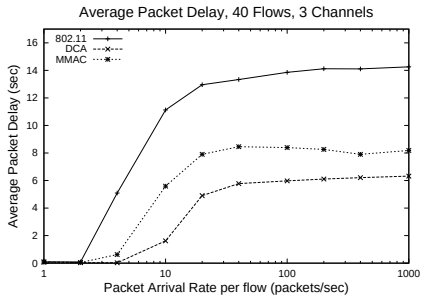
(b) 4 channels

Figure 8: Aggregate Throughput vs. Packet Arrival Rate in a multi-hop network. Packet size is 1024 bytes.

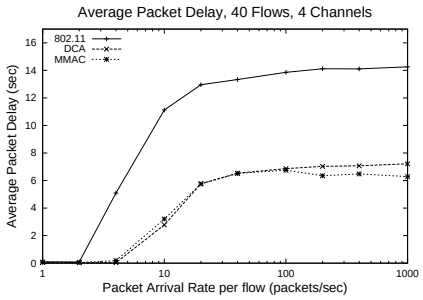
DCA in 64-node scenario. When the number of nodes are small, MMAC shows higher delay because packets have to wait during the channel negotiation phase (ATIM window). But when the number of nodes becomes large, DCA suffers from high contention at the control channel which results in high packet delay. MMAC does not have this problem, because it does not maintain a separate channel for control messages.

Now we look at results from a multi-hop network. As stated in the previous section, in our multi-hop network simulations, a node can be a source for multiple destinations, or it can be a destination for multiple sources. Figure 7 shows the aggregate throughput of different protocols as the network load increases. Three and four channels are used in Figure 7(a) and 7(b) respectively. In 7(a), MMAC performs better than DCA, but the difference is smaller than in the wireless LAN case. This is due to the following reasons. First, in a multi-hop network, all 3 channels may not be fully utilized throughout the entire area. In the region where the network can benefit from having the third channel, MMAC does better than DCA. But in the region where only two channels are needed, DCA does better than MMAC because it can utilize 2 data channels without ATIM window overhead. Second, in MMAC, if a node has flows to two different destinations, each destination may choose a different channel and one flow may have to wait for an entire beacon interval to negotiate the channel again. Also, if a node is a destination for two flows from other sources, these two flows must be transmitted on the same channel, reducing the benefit of having multiple channels. As the network load becomes very high, throughput of DCA drops faster than MMAC. This is because a single control channel is shared by every node in DCA. When the network load is very high, the collision rate of control packets increases, degrading the throughput. We call this *control channel saturation*.

The impact of control channel saturation is also shown in Figure 7(b). MMAC gains significant benefit from not having a dedicated control channel. In [8], it is shown that the maximum number of channels that can be fully utilized using DCA is $L_d/3L_c$, given that L_d is the data packet size and L_c is the control packet size. But even when the number of channels is less than $L_d/3L_c$, the throughput suffers from high contention among the control packets, especially when the network load is high. The impact of control channel saturation can be reduced if larger packets are used, because less control message are needed to transmit the same amount of data.The results using larger packets are shown in Figure 8(a) and 8(b). As the results show, MMAC only does slightly better than DCA, both with three and four



(a) 3 channels



(b) 4 channels

Figure 9: Average Packet Delay vs. Packet Arrival Rate in a multi-hop network.

channels. However, MMAC achieves this improvement with simpler hardware than DCA.

Figure 9(a) and 9(b) shows the average packet delay of the protocols as the network load increases. Packet size is again 512 bytes in these graphs. With three channels, MMAC shows higher delay than DCA, even though MMAC achieves higher throughput. This is due to the same reasons explained in the wireless LAN scenario. However, when four channels are available, MMAC shows lower delay than DCA. We can see that the average packet delay of DCA is almost the same with three and four channels. This is because DCA does not benefit from having one more channel because of control channel saturation, as previously mentioned. But MMAC benefits from having the fourth channel and the average delay becomes lower. ATIM window overhead in MMAC does not increase with the number of channels, as long as the ATIM window is long enough to exchange all the ATIM messages necessary.

We have fixed the ATIM window size to $20ms$ so far in this paper. But this may be undesirable due to the following reason. When there are small number of flows in the network, using 20% of each beacon interval for exchanging ATIM messages is wasteful. Much of the time the channel will be left as idle, because data packets are not allowed to be transmitted in this interval. On the other hand, if there are very large number of flows in the network, a longer ATIM window would be needed to exchange all the ATIM messages between nodes to negotiate channels. Thus the ATIM window size affects the throughput of MMAC protocol[1]. To study this impact, aggregate throughput is measured using different ATIM window sizes and shown in Figure 10. The three curves show results for 256, 512 and 1024 bytes of packet size. For this network scenario, an ATIM window size of around 15-20ms is shown to be the best for throughput. When the ATIM window size is too small, not all nodes can exchange ATIM messages and negotiate channels during this interval. Nodes that have not successfully exchanged ATIM messages stay on the default channel. So the multiple channels cannot be fully utilized, resulting in degraded throughput. If the ATIM window size is too large, the throughput decreases because no benefit is obtained with increased overhead from having a longer ATIM window. The optimal ATIM window size depends mainly on the number of flows in the beacon interval, because an ATIM packet exchange is required for each flow. Since the number of flows is often dynamically changed, it is desirable to also make the ATIM window size adapt to the network situation. Changing ATIM window size dynamically to achieve maximum throughput is left as a future work [2].

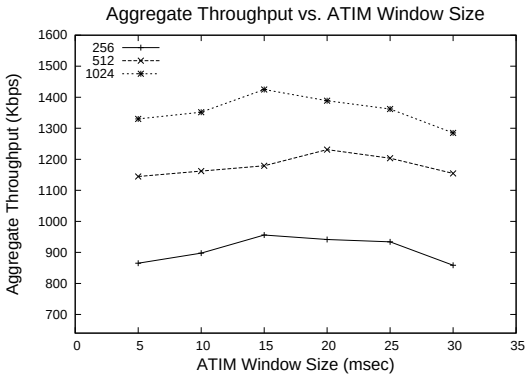
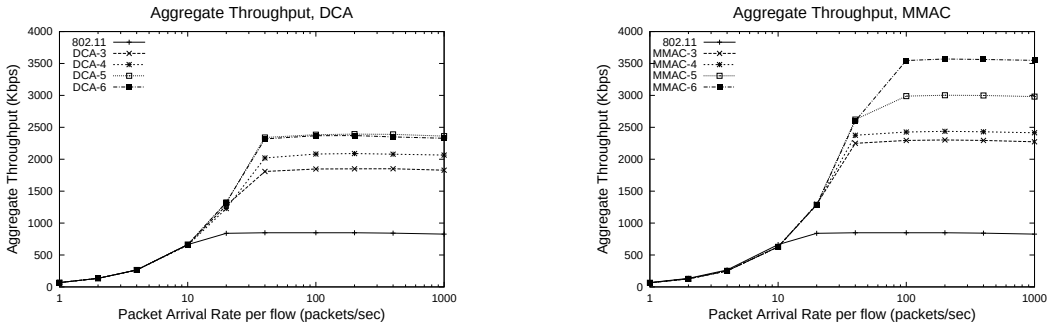


Figure 10: Aggregate Throughput vs. ATIM Window Size in a multi-hop network.

Finally, we measured the throughput of different protocols varying number of channels. This simulation was done for a wireless LAN with 30 nodes. We used 512 bytes for packet size, and the number of channels vary from 3 to 6. The results are shown in Figure 11. In the graphs, "MMAC-3" indicates MMAC protocol with 3 channels, and "DCA-4" refers to DCA protocol with 4 channels. The throughput of IEEE 802.11 is also shown in the graphs. Because of control channel saturation, DCA does not benefit from having additional channels when the number of channels becomes larger. MMAC does better than DCA with the same number of channels, especially when the network load is high.

[1]A similar observation about the impact of ATIM window on IEEE 802.11 PSM is reported in [16, 17].

[2]Changing ATIM window size dynamically to improve throughput in IEEE 802.11 PSM is studied in [17].



(a) DCA (b) MMAC

Figure 11: Aggregate Throughput vs. Packet Arrival Rate in a wireless LAN (30 nodes).

DCA and MMAC have their own ways to avoid the hidden terminal problem and access multiple channels dynamically. DCA uses one separate channel to exchange control packets, whereas MMAC uses ATIM windows to negotiate channels. In DCA, the bandwidth of the control channel has a major impact on the performance. In MMAC, the ATIM window size takes the role. Under our simulation model, MMAC, which uses simpler hardware than DCA, performed better than DCA.

In our simulations, we assume that all channels have the same bandwidth, as in IEEE 802.11 specification. However, if we can control the bandwidth of data and control channels, performance of the DCA protocol can be improved further by tuning the control channel bandwidth properly. The optimal control channel bandwidth in DCA depends on the traffic load, just as the optimal ATIM window size in MMAC. Even though as of now we do not have a scheme that dynamically controls ATIM windows size based on traffic load, it is much easier to adjust ATIM window size dynamically than to adjust bandwidth of the channels. Because of this reason, MMAC has higher flexibility than DCA.

7. DISCUSSION

In this section, we discuss some issues to be considered regarding our scheme and possible ways to improve it.

As stated earlier, the MMAC protocol requires all clocks in the network to be synchronized, so that all nodes start a beacon interval at the same time. Clock synchronization can be achieved using either out-of-band (such as GPS) or in-band solutions. If the nodes are capable of using an out-of-band solution for synchronization, no additional overhead is imposed on channels used by our protocol. However, if an in-band solution is used, it imposes an additional overhead which might affect the performance of the protocol. To model the overhead, we have implemented the beaconing mechanism similar to IEEE 802.11 TSF, which works as follows. At the start of a beacon interval, each node waits for a random delay and transmits a beacon. If a node receives a beacon before transmitting its own beacon, it suppresses and does not transmit its beacon. Since the beacons model the overhead of synchronization, if an out-of-band solution can be used, our protocol will perform better than what our simulations show, since the overhead of beacons will not be necessary.

The beaconing mechanism we use in our simulations is meant to model the potential overhead of synchronization. IEEE 802.11 TSF is designed for wireless LANs, where all nodes are within transmission range of each other. IEEE 802.11 TSF can be applied to multi-hop networks, but in rare cases it may fail to synchronize a multi-hop network, because of the following problem.

Consider the scenario in Figure 12. At the start of a beacon interval, each node chooses a random delay and transmits a beacon. It might happen that node A always transmits a beacon before B, and node D always transmits a beacon before C. Then the clocks of (A, B) and (C, D) may drift away, because they never exchange beacons.

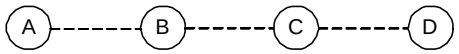


Figure 12: A chain topology of 4 nodes.

One approach to avoid this problem may be to have a node not suppress its beacon transmission even if they receive a beacon. This scheme will solve the problem, but the cost is significant, because all nodes transmit beacons at every beacon interval. Another approach may be to have each node transmits beacon with a probability p at each beacon interval. Developing a scheme for clock synchronization in multi-hop network is outside the scope of this paper.

Also, [18] argues that even if we can achieve clock synchronization, two partitioned network might not be able to discover each other if their schedules are totally out of synchronization such that the ATIM windows do not overlap. We do not address this problem in this paper.

In MMAC, nodes switch to the common channel at the beginning of each beacon interval. However, if a node starts

transmitting a data packet near the end of a beacon interval, the time when the node switches its channel may be pushed back. In this case, the node might miss the ATIM packets sent by other nodes. To prevent this, nodes refrain from transmitting a packet if the time left for the current beacon interval is less than the transmission time of the packet.

When a node is sending packets to two different destinations, these two destination nodes may select a different channel. For example, suppose that we have nodes A, B and C in the network, as in Figure 13. Node A has some packets destined for B and others destined for C. During channel negotiation, node B selects channel 1 and node C selects channel 2. If A selects channel 1, it can only transmit packets destined for B, and all the packets destined for C must wait until next beacon interval to negotiate the channel again. This behavior of MMAC protocol raises several issues. First, to avoid *head of line blocking* problem, the packets that cannot be transmitted because of channel mismatch must be kept in a separate buffer, and restored to the queue at the end of the beacon interval. This complicates the queue management. Also, it is possible that the same channels are selected by each node in the subsequent beacon intervals, starving the flow from A to C. In our scheme, if node A has to send ATIM packets to B and C, A chooses randomly which one to send the packet to first. This randomness should prevent complete starvation, although there can be short-term unfairness among the flows. Instead of randomly choosing among the destinations, node A can send an ATIM packet first to the destination which is the target node of the first packet in its queue. This modification will improve the fairness of the protocol.

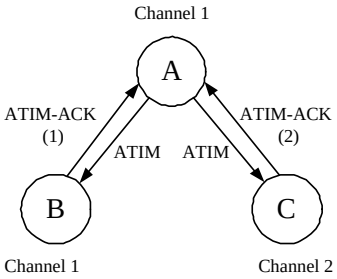


Figure 13: An example network scenario. Assume there are other nodes in the vicinity of these three nodes, that affect the PCL of these three nodes. Node A has packets for B, and also packets for C. A exchanges ATIM messages with B first, and both select channel 1. After that, A sends an ATIM packet to C, and C selects channel 2. Since A will stay in channel 1 for the beacon interval, packets for C must be deferred until the next beacon interval.

In addition to the problems stated above, this situation might also have impact on the throughput. Suppose that A had only a few packets for B. Then after sending all the scheduled packets, A becomes idle for the rest of the beacon interval. But A cannot send packets to C, even though A has received C's ATIM-ACK during ATIM window and knows which channel C will be listening on. To avoid waste of bandwidth, we can extend MMAC to allow nodes to switch channels inside the beacon interval. A node may switch channels according to the following rules.

- If node A finishes sending packets on its selected channel, and does not know of any node that is planning to send packets to A, A may switch its channel. If A has received any ATIM packet during the ATIM window, A must stay on the selected channel for the entire beacon interval.
- After switching to another channel, node A must wait for some time to gather information on the condition of the new channel before transmitting a packet. The amount of time it has to wait is $SIFS + t_{MTU} + p$, where t_{MTU} is the transmission delay for maximum tranfer unit (MTU), and p is the propagation delay for one-hop distance. This delay is required to avoid collision, because A does not have NAV (Network Allocation Vector) information for the new channel at the time it switches channels.

In our example, node A can switch to channel 2 after sending all of its scheduled packets to B, because it has not received any ATIM packets during the ATIM window. Node C has to stay in channel 2 for the entire beacon interval, because it received an ATIM packet from A. So the communication between A and C can take place in channel 2, for the rest of the beacon interval. This extended scheme might increase the throughput, and the performance of this scheme will be studied in the future work.

Another issue in MMAC is channel load balancing. In the original MMAC, a node counts the usage of a channel based on the ATIM-ACK or ATIM-RES packets it overhears, and selects the channel with the least count to balance the channel load. It means that the node is counting the number of *source-destination pairs*. The assumption here is that every flow has the same amount of traffic ready to be transmitted in the beacon interval, which may not be true. Different flows may have different number of packets pending to be sent. So it may be better to count the number of *pending packets* for each channel rather than number of source-destination pairs. To do this, the source node counts the number of pending packets for the destination and include the value in the ATIM packet. This number is echoed in the ATIM-ACK and ATIM-RES packet, so that the nodes in the vicinity of the source or destination can obtain the information. When selecting a channel, the node selects a channel with the least number of packets scheduled on the channel. This selection mechanism will achieve a better load balancing than the original scheme.

8. CONCLUSION AND FUTURE WORK

In this paper, we have presented a multi-channel MAC protocol that utilizes multiple channels to improve throughput in wireless networks. The proposed scheme requires only one transceiver for each host, while other multi-channel MAC protocols require multiple transceivers for each host [11, 9, 8]. In order to avoid the multi-channel hidden terminal problem, we require nodes to be synchronized, so that every node starts each beacon interval at about the same time. At the start of each beacon interval, every node listens on

a common channel to negotiate channels in the ATIM window. After the ATIM window, nodes switch to their agreed channel and exchange messages on that channel for the rest of the beacon interval.

Simulation results show that MMAC successfully exploits multiple channels to improve total network throughput over IEEE 802.11 single-channel. The performance of MMAC and DCA depends on the network situation, but as the simulation results show, MMAC performs better or at least comparable to DCA in most cases. It is important that MMAC achieves this performance using simpler hardware than DCA. Since MMAC only requires one transceiver per host, it can be implemented with hardware complexity comparable to IEEE 802.11. Also, power saving mechanism used in IEEE 802.11 can easily be integrated with MMAC for energy efficiency, without further overhead.

As discussed in the previous section, the ATIM window is a major overhead in MMAC. Nodes cannot exchange data packets during the ATIM window, even if they already finished exchanging the ATIM packets. So it is desirable to change the size of ATIM window dynamically, based on the traffic condition. We are going to investigate this problem as a future work.

9. REFERENCES

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

[2] IEEE 802.11a Working Group, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 1: High-speed Physical Layer in the 5 GHz band," 1999.

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

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

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

[6] A. Nasipuri, J. Zhuang and S. R. Das, "A Multichannel CSMA MAC Protocol for Multihop Wireless Networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, September 1999.

[7] A. Nasipuri and S. R. Das, "Multichannel CSMA with Signal Power-based Channel Selection for Multihop Wireless Networks," in *Proc. of IEEE Vehicular Technology Conference (VTC)*, September 2000.

[8] 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 *Int'l Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN)*, 2000.

[9] N. Jain and S. Das, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks," in *Proc. of the 9th Int. Conf. on Computer Communications and Networks (IC3N)*, October 2001.

[10] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II - the hidden terminal problem in carrier sense multiple-access modes and the busy tone solution," *IEEE Transactions on Communications, COM-23*, 1975.

[11] A. Nasipuri, S. Ye, J. You and R. Hiromoto, "A MAC Protocol for Mobile Ad Hoc Networks using Directional Antennas," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, Chicago, IL, September 2000.

[12] I.A. Getting, "The Global Positioning System," *IEEE Spectrum 30*, December 1993.

[13] 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*, April 2002.

[14] VINT Group, "UCB/LBNL/VINT network simulator ns (version 2)," .

[15] The CMU Monarch Project, "Wireless and Mobility Extension to ns," .

[16] H. Woesner, J. Ebert, M. Schlager and A. Wolisz, "Power-saving mechanisms in Emerging Standards for Wireless LANs: The MAC Level Perspective," *IEEE Personal Communications*, June 1998.

[17] E.-S. Jung and N. H. Vaidya, "An Energy Efficient MAC Protocol for Wireless LANs," in *Proc. of IEEE INFOCOM*, June 2002.

[18] Y.-C. Tseng, C.-S. Hsu and T.-Y. Hsieh, "Power-Saving Protocols for IEEE 802.11-Based Multi-Hop Ad Hoc Networks," in *Proc. of IEEE INFOCOM*, June 2002.