On the efficacy of separating control and data into different frequency bands

Pradeep Kyasanur*
Dept. of Computer Science,
University of Illinois at Urbana-Champaign
kyasanur@uiuc.edu

Jitendra Padhye
Microsoft Research
padhye@microsoft.com

Victor Bahl
Microsoft Research
bahl@microsoft.com

July 28, 2005

Abstract

Radio spectrum allocated for use in unlicensed wireless networks is distributed across non-contiguous frequency bands. Existing MAC protocols, like IEEE 802.11, operate only in contiguous bands. We propose a split-channel protocol that improves the capacity of infrastructure and multi-hop wireless networks by utilizing a sliver of unused spectrum in the lower frequency band for control purposes. The proposed Control Channel-based MAC Protocol (C^2M) increases the throughput by moving the contention resolution overheads to a separate low rate channel. We allow simultaneous channel contention and data transmission by incorporating *advance reservation* on the control channel, and *data aggregation* on the data channel. Simulation results show that compared to IEEE 802.11, C^2M significantly improves network performance. Finally, we discuss the potentially large benefits of extending C^2M to networks with directional antennas.

*This research was done when the author was an intern at Microsoft Research

1 Introduction

Over the years, researchers have proposed many different medium access control (MAC) protocols that improve link utilization in wireless networks. Of these, one promising approach is to split the control and data portions of the MAC protocol and move each to a separate time, space, or frequency slice (channel). Nodes arbitrate for medium access on a channel that is separate from the one they use to exchange data. Consequently, the contention overhead is localized to the arbitration channel only.

Evaluations of these schemes have produced mixed results and we cover various approaches in detail in Section 5. Here, we call out one recent result published by Yang et. al [1] in which the authors conclude that splitting a single shared-channel MAC protocol is not a good idea as it reduces the throughput of the wireless network.

In this paper, we revisit and re-evaluate the idea of splitting control from data for the purpose of maximizing the throughput of a wireless network. We analyze a IEEE 802.11-like network and show that the throughput of the network *can* be improved significantly, when the channel arbitration and data transmission are performed simultaneously, and when *packet aggregation* and *advanced reservations* are incorporated into the split-channel protocol.

To quantify these benefits, we have designed a MAC protocol, called control channel-based MAC (or C^2M). C^2M splits the IEEE 802.11 protocol by operating its control portion over a low-frequency, low-data rate channel and the data portion over a high-frequency, high-data rate channel. Contention resolution occurs on the lower-rate channel, which we call the *control channel* while the higher-rate channel, called the *data channel*, is used exclusively for exchanging data packets. In addition to splitting the MAC over two separate frequency bands, we perform advance packet reservation, and data aggregation to ensure that the control channel does not become a bottleneck to the data channel. We show that C^2M avoids the reordering problems of striping packets across two channels.

To understand our motivation for investigating this approach consider the following three observations:

1. Ideally, in wireless networks we would like to remove all control overhead and saturate the license-exempt spectrum with high-rate data transmissions. The IEEE 802.11 MAC protocol does not do this. It is inefficient because it performs both channel contention and data trans-

mission in the same channel.

2. Several small slices[1] of frequency (1 to 2 MHz) are available in the lower frequency bands (i.e. below 900 MHz). These orphan slices are not wide enough for deploying large-scale data communication systems and therefore, are not utilized.

3. In the split-channel approach, the bandwidth (in Mbps) required for control is small. Consequently, the amount of spectrum needed for the control channel is minimal.

Taking these observations together with the fact that radio frequency (RF) communication equipment is becoming inexpensive, we believe that it is possible to build a split-channel MAC protocol that makes use of the available lower frequency spectrum slices to optimize the utilization of the high-frequency license-exempt bands. Such a system could be realized with multiple radios. The radio operating in the lower-frequency band would be a narrow-band radio that would implement the 802.11 PHY with a modified 802.11 MAC. The second radio would operate in the high-frequency band with the 802.11 PHY but with no MAC.

We evaluate C^2M in infra-structure, static multi-hop, and mobile multi-hop network configurations and find the results to be promising. For example, using a 2 Mbps control channel, we find that:

- In a single-hop Access Point based configuration with 16 nodes running FTP, C^2M gives around 40% improvement over 802.11.

- In a 4-node multi-hop chain configuration, C^2M gives around 25% improvement over 802.11

- With 10 random topology configurations with 50 nodes spread over a region of 500m by 500m, and average mobility of 10 m/s C^2M gives an average of 38% improvement over 802.11.

We study the impact of higher control-channel data rate on C^2M's performance, but here highlight results for a 2 Mbps control channel only. This is because our objective is to use a very small slice of the lower-frequency

[1]note, we use terms slice, band, and channel interchangeably

spectrum, and it is relatively easy to build a 2 Mbps wireless network with as little as 1 MHz of spectrum.

Additional dimensions of our evaluation include: protocol fairness, we use Jain's Fairness Index [2] for this, and C^2M's sensitivity to its parameters. We show that by using advance reservation and packet aggregation, C^2M improves performance in excess to the data rate of the control channel. Stated differently, C^2M performs better than a packet striping approach where the nodes use the additional frequency band for exchanging data packets.

Summarizing our contribution, we propose a novel architecture for a split-channel medium access control protocol. This architecture incorporates unused orphan low-frequency spectrum slices into the MAC to increases the throughput of the wireless LAN. In this context, we introduce: (a) simultaneous channel contention and data transmissions, (b) data or packet aggregation, and (c) advance reservations. We demonstrate the benefits of our architecture for infra-structure and multi-hop wireless networks with omni-directional antennas and we discuss the potentially large benefits for networks equipped with directional antennas.

The rest of the paper is organized as follows: In Section 2 we provide a brief overview of the IEEE 802.11 MAC. We describe the C^2M protocol in detail in Section 3. In Section 4 we present detailed results from our performance evaluation. Related work is covered in Section 5, we discuss extensions of the split-channel approach to a network with directional antennas in Section 6 and conclude in Section 7. Details of equations we developed for our analytical framework are provided in the appendix.

2 Overview of IEEE 802.11 MAC

IEEE 802.11 standard has a distributed mode of operation, called Distributed Coordination Function (DCF), which is based on CSMA/CA. A node with a data packet to transmit first selects a random backoff value b from the range $[0, CW]$, where CW (Contention Window) is a variable maintained by each node. The node is required to wait for b idle slots, where slot time (*Slot-Time*) is a constant specified in the standard. After completing the backoff and waiting for DIFS (DCF Interframe Space), if the channel continues to be idle, Request to Send (RTS) and Clear to Send (CTS) packets are exchanged to reserve time for the impending data transmission. The use of RTS-CTS is especially critical in a multi-node scenario that may give rise

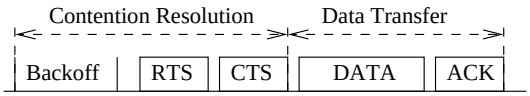

Figure 1: Time line of a typical data transfer

to hidden terminals. In addition, the RTS-CTS exchange ensures that the receiver is ready to receive the data transmission. DATA packet is sent after a successful RTS-CTS exchange, and is acknowledged with an ACK packet. Successive packets in the RTS-CTS-DATA-ACK exchange are separated by a SIFS (Short Interframe Space) interval.

When the sender fails to receive either the CTS or the ACK, CW is doubled, subject to a maximum value $CWmax$. This is followed by a retransmission, unless a specified maximum number of retransmission attempts have failed. On successful reception of an ACK, or if the number of retransmission attempts has reached a maximum threshold, the sender resets CW to $CWmin$.

From the above description, we note that each data packet exchange has two phases (as shown in Figure 1). In the first phase, a node *resolves contention* and reserves the channel for data transfer by first backing off, and then exchanging RTS-CTS packets. This is followed immediately by the second phase that involves the *actual data transfer* by exchanging DATA-ACK packets. We propose to perform the contention resolution on the control channel, and perform the DATA-ACK exchange on the data channel.

3 The proposed protocol

As we mentioned earlier, we have designed a new protocol to study the efficacy of splitting control and data. In this section, we give a overview of our proposed C^2M protocol. C^2M is similar to IEEE 802.11 in using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) for controlling access to the channel, and using control packets (RTS-CTS) for channel reservation. We begin with an overview of the design, and then present the details of the protocol.

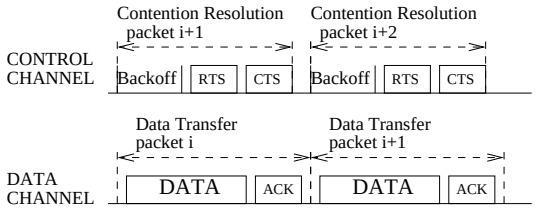

Figure 2: Parallel operation of control and data channels

3.1 Design Overview

Consider the scenario when IEEE 802.11 is being used on the data channel. Contention resolution is required to prevent collisions when transferring data on the channel. Contention resolution is a overhead that uses time on the channel without transferring useful data. Therefore, the throughput over the high rate data channel can be improved if the contention resolution phase is moved to the control channel. Our initial proposal is to perform the contention resolution phase on the control channel in parallel with the data transfer on the data channel. Figure 2 depicts the protocol operation. While the i^{th} packet is being transmitted on the data channel, the contention resolution for the $i + 1^{th}$ packet proceeds on the control channel. Other authors have considered a similar pipelined operation that splits a given channel into two sub-channels [1], but the novelty of our approach is to utilize bandwidth available in a low frequency band (that might otherwise be unused) for contention resolution.

If contention resolution for packet $i + 1$ on the control channel does not complete by the end of transmission of packet i on the data channel, then the data channel has to stay idle until the completion of contention resolution. Contention resolution takes a variable amount of time, as it depends on the backoff values chosen, and RTS or CTS collisions (if any). In addition, data transmission duration depends on the size of the data packet, which may vary from packet to packet. Consequently, with the basic protocol proposed above, the data channel may be frequently idle. The time for contention resolution depends in part on the data rate of the control channel (the impact of which we study in the next section), but it is important that the protocol be insensitive to *variations in the durations of contention resolution and data packet transmission.* It has also been noted by Yang et al. [1] that a two channel approach may be susceptible to inefficiencies on account of such

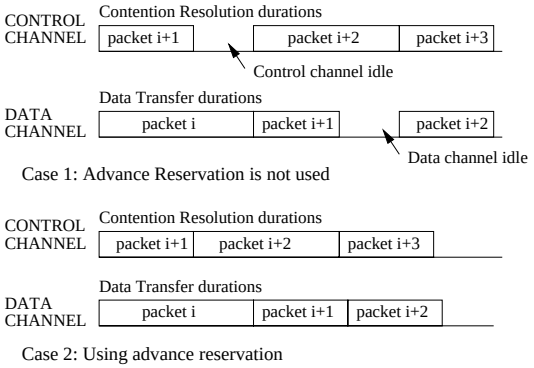

Figure 3: Benefit of using advance reservation on control channel

variability. We propose a technique called "Advance Reservation" to cope with the variable duration of contention resolution.

The *advance reservation* technique allows each node to reserve ahead for k packets, where k is a protocol parameter. A node transmitting packet i on the data channel can reserve (using modified RTS-CTS packets exchanged on the control channel) the data channel for up to k additional packets. Advance reservation is based on the observation that *once the contention resolution and data transfer are separated on two different channels, then they can be decoupled in time as well.* By reserving multiple packets in advance, the protocol can better tolerate the variations in the duration of contention resolution and data transfer. Figure 3 illustrates the benefit of using advance reservation. In the figure, when advance reservation is not used, if the contention resolution for packet $i+2$ takes longer than average, then the data channel will be idle until contention resolution for packet $i+2$ is completed. On the other hand, when the data transfer for packet i is longer than contention resolution for packet $i+1$, the idle time on the control channel cannot be utilized, unless advance reservation is used. By using advance reservation, both the data and the control channel can be better utilized.

Advance reservation with a carefully chosen value for k ensures that the control channel is not a bottleneck to performance as long as the *average duration for contention resolution is smaller than the average duration for data transfer*. In the next section, we will analyze the impact of control channel characteristics on performance, and determine the rate necessary to ensure control channel is not a bottleneck.

3.2 Implication of characteristics of control channel on protocol design

Since we envision that the control channel will operate in small slices of the low-frequency spectrum, it will likely have lower rate than the data channel. The range of the control channel will also be different than that of the data channel. In this section, we will discuss the impact of these two issues on the design of the C^2M protocol.

3.2.1 Control channel data rate

Let us assume that the control channel supports a maximum data rate R_c, and the data channel supports a maximum data rate $R_d \gg R_c$. We define the average time required for contention resolution on the data channel (backoff and RTS-CTS exchange) to be TC_d. Similarly, the average time required for contention resolution on the control channel is defined to be TC_c. We define the time required for data transfer on the data channel with average data packet size s to be $DT_d(s)$.

The data channel is not fully utilized if, $TC_c > DT_d(s)$ i.e., if the average contention resolution time on the control channel is longer than the average data transfer time on the data channel.

If the control channel is not available, the average time to complete one data transmission is $TC_d + DT_d(s)$ seconds, resulting in an average throughput of $s/(TC_d + DT_d(s))$ bytes per second. If the control channel is available, the average time for data transfer is equal to $max(TC_c, DT_d(s))$, since the larger of contention resolution duration and data transfer duration is a bottleneck to performance, and the average throughput is $s/max(TC_c, DT_d(s))$. The time for data transfer depends on the average packet size. Hence, the average throughput of the two schemes is sensitive to the average packet size.

To estimate the average contention and data transfer duration, we use IEEE 802.11a protocol parameters, which are listed in Table 1. Each packet, control or data, is preceded by a physical layer preamble and header that require time $PLCPDuration$. The data rate of control channel decides the actual time required for RTS-CTS exchange when estimating TC_c. The data rate of the data channel decides the actual time required for RTS-CTS exchange when estimating TC_d, and DATA-ACK exchange when estimating $DT_d(s)$. Detailed analysis of C^2M throughput is presented in the Appendix 7. Here, we only present the plots derived from the throughput equations.

Parameter	IEEE 802.11b	IEEE 802.11a
$SlotTime$	$20\mu s$	$9\mu s$
$SIFS$	$10\mu s$	$16\mu s$
$DIFS$	$50\mu s$	$34\mu s$
$PLCPDuration$	$96\mu s$ (short format)	$24\mu s$

Table 1: Parameters of IEEE 802.11a and IEEE 802.11b

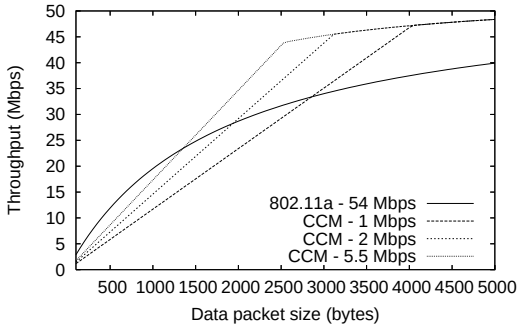

Figure 4: Analytical comparison of C^2M with IEEE 802.11a

In Figure 4, we compare the throughput obtained using the C^2M approach (curves named "CCM"), and when using only the data channel (curve named "802.11a") for different average packet sizes. The data channel transmission rate is assumed to be 54 Mbps (maximum rate of IEEE 802.11a), while the control channel data rate is varied from 1 to 5.5 Mbps (some of the rates available in IEEE 802.11b). The results corresponds to comparing throughput of IEEE 802.11a, with a C^2M implementation that uses IEEE 802.11a as the data channel and IEEE 802.11b as the control channel. As we can see from the figure, C^2M achieves a higher throughput than 802.11a provided sufficiently large packet sizes are used. The C^2M curves initially have a linear slope because for small packet sizes, contention resolution time is larger than the data transfer time. Therefore, the throughput obtained is inversely proportional to contention resolution time, which is of constant length in our analysis, resulting in a linear curve. There is a change in the slope of C^2M curves at a threshold packet size, since for packet sizes larger than the threshold, data transfer time (which increases with packet size) is larger than the contention resolution time.

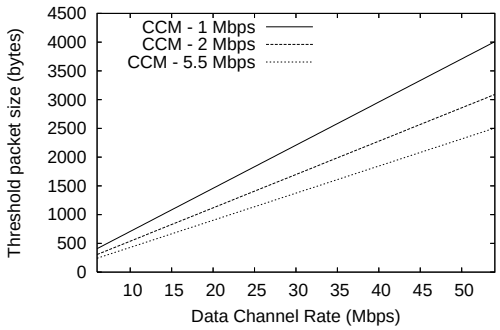

Figure 5: Impact of data channel data rate on C^2M performance

Figure 5 plots the average packet size needed on the control channel to ensure control channel is not a bottleneck to performance, for different data channel data rates (i.e., the packet size s at which $TC_c = DT_d(s)$). Note that IEEE 802.11a supports data rates ranging from 6 Mbps to 54 Mbps. When the data channel rate is small, then the threshold packet size is small as well. For example, even with a 1 Mbps channel, the threshold packet size is only around 2500 bytes for 36 Mbps data channel. Therefore, the threshold packet size required for utilization of the data channel can often be fairly small. In our simulations, we use a 54 Mbps data channel, which may be under-utilized when large packets are not available, to characterize the *worst case* performance of C^2M.

Figure 6 plots the maximum throughput benefit over IEEE 802.11a possible with different control and data channel rates, when the average packet size is chosen to be large enough to ensure control channel is not a bottleneck. As we can see from the figure, the benefit of using a control channel increases with higher data channel rates. When (the fixed duration) contention resolution is moved out of the data channel, the data that can be transmitted in the extra time available on the data channel is greater at higher data rates. Therefore, the total throughput improvement of using a control channel is better at higher data rates. Another observation from Figure 6 is that the throughput improvement achieved by C^2M is *larger* than the control channel bandwidth for higher data channel rates.

Based on the above analysis, we make the following observations:

- Control channel will be a bottleneck to performance, resulting in under-utilization of the data channel, until the average data packet size is

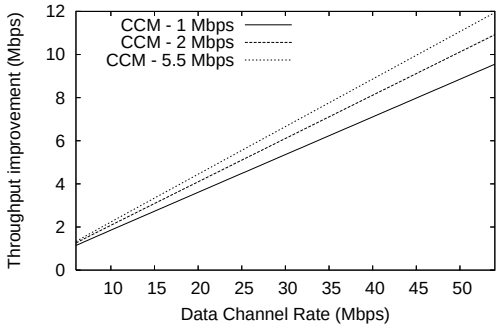

Figure 6: Maximum throughput improvement possible with C^2M

larger than a threshold (Figure 4). For example, the threshold size with 54 Mbps data channel is approx. 4000 bytes for 1 Mbps control channel, and 2500 bytes for 5.5 Mbps control channel.

- The threshold packet size needed to ensure control channel is not a bottleneck depends on the data channel data rate (Figure 5). For a given control channel data rate, a smaller packet size suffices for a lower-rate data channel. Alternatively, for a fixed threshold packet size, a lower-rate control channel suffices with a lower-rate data channel.

- When the average data packet size is larger than a threshold, significant performance benefits are possible (Figure 6). At sufficiently large packet sizes, the improvement in the throughput of the data channel is larger than the data rate of the control channel. This suggests that the C^2M approach has the potential for higher throughputs than other approaches, such as link-layer striping, that transmit data over both the low-rate (control) and high-rate (data) channels.

- Increasing the control channel data rate (say from 1 Mbps to 5.5 Mbps) has a marginal improvement on performance for small packet sizes, and no improvement at all after a threshold size (Figure 4). This suggests that increasing control channel data rate beyond a point is not beneficial.

- Collisions during contention resolution is not considered in the analysis. With collisions, average contention resolution time may increase, requiring a larger threshold data packet size.

The above observations suggest that *the average size of data transmitted for each contention resolution attempt should be sufficiently large* to achieve significant performance improvements. Furthermore, the protocol has to adapt to the different threshold packet sizes required for different control channel data rates that may be available in practice. However, the typical size of packets handed down from higher layers is often smaller than the required threshold size. Since it is difficult to change the packet sizes sent by higher layers, we propose aggregating packets into a train of packets, and reserve the data channel for the whole train with a single contention resolution. Thus, by using an *data aggregation technique*, we can increase the size of "packets" sent on the data channel. The number of packets to aggregate can depend on the desired threshold packet size. We describe the details of the aggregation procedure in Section 3.3.

3.2.2 Control channel range

Besides having different data rates, the data and the control channel are also likely to have different ranges. While the exact difference in the range will depend on several factors such as power levels and environmental factors, it is likely that the range of the control channel will be longer than the range of the data channel for reasons specified below.

The control channel is expected to be located in a lower frequency band. According to the free space propagation model [3], with unit gain antennas, the path loss of a channel is inversely proportional to the square of the frequency. Consequently, for a fixed transmission power, the control channel experiences a smaller path loss, resulting in longer transmission range.

The longer control channel range can be exploited in reducing the effect of hidden terminals. Typically, the range up to which a transmission may interfere with another transmission (called "interference range"), is longer than the transmission range. Protocols such as IEEE 802.11 that use a common transmission range for RTS-CTS, as well as DATA-ACK exchange, cannot completely eliminate packet losses due to "hidden terminals". For example, in Figure 7, node B is receiving data from a node A. Node C is outside the communication range of B, and hence does not receive the CTS from B. Thus, node C may begin a transmission to node D that interferes with the packet reception at B. Such packet losses are expensive as node A has to resend the DATA packet (after backing off for a larger duration).

By using a control channel range that is close to the interference range

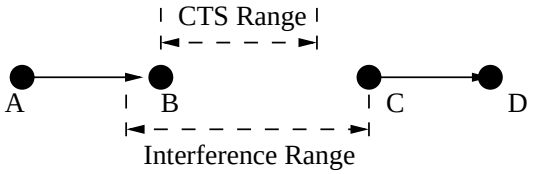

Figure 7: Scenario where longer control channel range is beneficial

of the data channel, all nodes in the interference region can be notified of the impending transmission, thereby preventing data packet collisions. Such collisions may be similarly reduced in IEEE 802.11 by using a higher transmission power (resulting in longer range) for RTS-CTS transmissions. However, since the RTS-CTS transmissions share the channel with DATA-ACK transmissions, increased RTS-CTS transmission power increases the range at which RTS-CTS transmissions may interfere with other ongoing data transmissions. With our proposed two channel approach, RTS-CTS transmissions are on a different channel from the DATA-ACK transmissions. Hence using longer range RTS-CTS transmissions will not interfere with data transmissions on the data channel. A control channel with longer range is also beneficial when using directional antennas, as we will discuss later in the paper.

If the transmission range of the control channel is too large, then then contention resolution process on the control channel will reserve an unnecessarily large area, reducing spatial reuse. Furthermore, contention on the control channel increases (in comparison with using a range equal to the interference range of the data channel). To address this problem, if the transmission range of the control channel is too large, then the transmission power on the control channel can be reduced, thereby setting the control channel range to roughly the interference range of the data channel. Our proposed protocol is not very sensitive to the control channel transmission range, and in our simulations, the control channel range is conservatively set to be sufficiently longer than the interference range of the data channel.

3.3 Detailed protocol architecture

C^2M has three main components.

1. Data aggregation: Aggregate packets to a particular destination into a

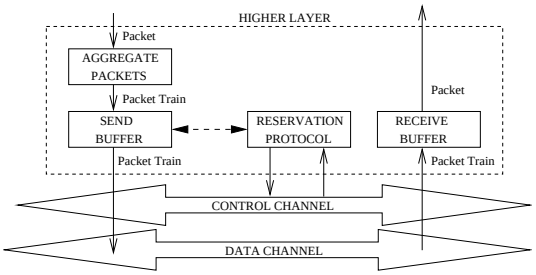

Figure 8: Architecture of control channel protocol

train of packets. Separate queue for each neighbor has to be maintained to support per-destination aggregation.

2. Channel reservation and contention resolution: Resolve contention and reserve time on the data channel for a packet train by exchanging RTS-CTS packets on the control channel. This component implements *advance reservation*.

3. Data train management: Manage the buffers at the sender and receiver for holding data trains. Selectively ACK the received packets in the train, and support retransmission of lost packets.

Figure 8 depicts the interaction between the different components of the control channel MAC. We next describe each component in detail.

3.3.1 Data Aggregation

Control channel will become a bottleneck to performance unless each reservation on the control channel reserves time for large packets. If a path MTU discovery [4] is in use, the new MAC cannot simply expose a large MTU and expect the higher layers to send larger packets. Thus, there is a need to build a train of packets that has a total size greater than a specified threshold, and transmit the whole train using a single reservation on the control channel.

The RTS-CTS exchange reserves the data channel for communication between a sender and exactly one receiver. Thus, to use the reservation mechanism in conjunction with packet trains, it is necessary that all packets in the train have a common destination. However, successive packets from the higher layer may have multiple destinations. Hence, the aggregation protocol has to separately aggregate packets for each destination.

The aggregation protocol maintains a queue for each known neighbor. The assembly of a new packet train is initiated whenever a packet is received from the higher layer to a destination for which there are no packets pending. Subsequently received packets to the same destination are added to the existing packet train. When the size of the packet train is equal to *AggregationLimit*, a parameter of the aggregation protocol, the packet train is handed off to the reservation protocol for scheduling and subsequent transmission.

Multiple packets to a single destination are not always available, or may require an unbounded period of waiting before *AggregationLimit* number of packets are available. The aggregation protocol uses another parameter, called *AggregationTimeout*, which specifies the maximum time a partially built packet train may wait for a new packet. A timer is associated with each packet train being built, and is reset to *AggregationTimeout* whenever a new packet is added to the train. When the timer expires, the packet train associated with the timer is handed off to the reservation protocol even if the size of the packet train is less than the *AggregationLimit*. The timeout mechanism ensures that there is an upper bound on the maximum delay introduced by the aggregation protocol.

The aggregation protocol can be enhanced to use a threshold based on the total size of packets in the packet train, in addition to using a threshold based on the number of packets in the packet train. This extension may be useful when the size of individual packets vary significantly.

The aggregation protocol adds latency to most packets by delaying them for aggregation. This latency results in a larger RTT for each link, and our simulation results characterize the impact higher latency has on TCP performance. The aggregation protocol also partially modifies the traditional FIFO semantics. We continue to maintain FIFO ordering among packets to any particular destination, but packets to different destinations are no longer guaranteed to follow the FIFO order. The reordering we may introduce does not adversely impact TCP, since packets along a flow (using a common route) continue to be sent in order.

3.3.2 Channel reservation and contention resolution

The reservation protocol exchanges control packets between the sender and the receiver. We label these packets as RTS and CTS respectively, in line with the commonly used terminology, although the packet formats are different from that used in IEEE 802.11. The reservation protocol has a parameter

ReserveAheadLimit that indicates the maximum number of packet trains that may be reserved for transmission at any time (*advance reservation*). When the aggregation protocol hands a packet train to the reservation layer, if the number of already reserved packets awaiting transmission is less than *ReserveAheadLimit*, then a new reservation is initiated. Otherwise, the packet train is buffered till a reservation opportunity arises later.

Each node maintains a reservation table to keep track of the reservations already done on the data channel. A sender initiating the reservation first computes the time T needed for transmitting the associated packet train, and its ACK. The sender then looks up in the reservation table for the earliest time E, starting from the estimated end of the RTS-CTS exchange, when the data channel is continuously free for T duration. The pair (E, T) is sent in the RTS. The receiver looks up in its own table to check if the duration (E, T) is indeed free. If the channel is free during (E, T), then (E, T) is sent back in the CTS. Otherwise, the next possible time after E, say E1, when the channel is free for duration T is chosen, and (E1, T) is sent back in the CTS. The receiver adds the pair sent in the CTS to its reservation table. When the sender receives a CTS with some pair (E, T), it checks if this pair conflicts with any entry in the reservation table. If there is no conflict, then the reservation is successful, and (E, T) is added to the reservation table. Otherwise, a new reservation attempt is initiated.

RTS transmissions are initiated after a random backoff. RTS or CTS packets may be lost due to errors or collisions. The reservation protocol uses a retransmission procedure similar to that used by IEEE 802.11, which doubles the contention window on collision. When a successful reservation is completed by the sender, the packet train is scheduled for transmission on the data channel at time E. The actual protocol for managing the data transmission (and retransmission) is described later.

As discussed earlier, using advance reservation can effectively hide the variations in contention resolution and data transfer durations. The advance reservation technique requires loose synchronization among nodes. The time specified in a control packet is relative to the reception of that packet. The reservation interval is set to be sufficiently larger than the data train transmission duration to account for propagation delay. However, if reservation is done too far ahead of the data transmission, clock drift among nodes may result in two transmissions overlapping with each other (resulting in data packet collisions). We set the *ReserveAheadLimit* to be at most a few packet trains, to reduce the impact of clock drift errors.

3.3.3 Data train transmission and management

The proposed protocol transmits a burst of packets during a single transmission opportunity. Since the underlying channel is not error-free, some of the packets in a packet train may be lost or corrupted, and require retransmission. To reduce overhead, we propose using a single ACK at the end of the packet burst that uses a bit map to indicate which packets were correctly received (selective acknowledgments). Based on the received ACK, the lost and corrupted packets of a packet train are assembled into a new packet train and retransmitted (after obtaining a reservation from the reservation protocol). In case an ACK is not received at the end of a packet train transmission, the whole train is retransmitted.

Each packet train is retransmitted at most a fixed number of times, as specified by a retransmission threshold. Each packet train has a sequence number, and individual packets within the packet train are identified by a number relative to the packet train. The packet train transmission management is similar to the fragmentation mechanism specified by IEEE 802.11. The main difference is that the individual packets are not ACK-ed in the proposed protocol.

The receiver attempts to hand off the received packets in order to the higher layer. When a packet in a packet train is lost, but subsequent packets are received correctly, then the received packets are buffered. After the missing packet is received following a retransmission, all the buffered packets are handed off in order. A timeout is associated with the receive buffer to ensure that if some packet in a train is never received, then subsequently received packets are handed off to the higher layer (out of order) after the timeout expires. By using this mechanism we approximate the behavior of IEEE 802.11 in handing off packets in order whenever possible.

4 Evaluation

The control channel MAC protocol has been implemented in Qualnet 3.6 [5]. We have implemented the protocol as a *shim layer* between the MAC and the network layer. Our simulations account for the overheads (e.g., extra packet headers) introduced by a shim layer implementation.

We compare the performance of C^2M with IEEE 802.11a protocol. The data channel data rate is set to 54 Mbps (the maximum data rate available

with IEEE 802.11a). C^2M is evaluated with the control channel data rate set to 2 Mbps and 5.5 Mbps. The protocol parameters *ReserveAheadLimit* and *AggregationLimit* are set to 2 and 3 respectively, unless explicitly stated otherwise. As we noted in the analysis, the higher the data rate of the data channel, the higher the threshold packet size needed on the control channel. As it is not always possible to meet the threshold packet size requirement, C^2M performance may be degraded with higher data channel rates. Therefore, we have assumed a 54 Mbps data channel to characterize the *worst case* performance of C^2M.

For clearly separating out the benefits of moving overheads to the control channel from the benefits of aggregating data into larger packets, we also evaluate the performance of IEEE 802.11a with large application packets (4500 bytes for FTP and CBR simulations). IEEE 802.11 fragments packets that are larger than a specified fragmentation threshold (2346 bytes in IEEE 802.11a specification). A single RTS-CTS exchange is sufficient to transmit all the fragmented packets. Therefore, using large data packets, with MAC layer fragmentation enabled, approximately quantifies the benefit of data aggregation (recall that a single reservation suffices for multiple packets in a train). We designate this approach as "802.11 Frag" in simulation results. The results with "802.11 Frag" are biased against our proposed C^2M, especially with FTP traffic. The bias arises because TCP increases its congestion window in terms of packets. Therefore, when "802.11 Frag" is used, during every RTT (in the congestion avoidance phase) one additional 4500 byte packet is transmitted. However, with C^2M, only one additional 1500 byte packet is sent (as C^2M evaluations are performed with 1500 byte packets). Despite this bias, we have included "802.11 Frag" results to quantify the performance improvement that would be possible with large data packets.

4.1 Performance with single hop communication

We first evaluate the performance of C^2M in single-hop topologies. In these simulations, we characterize the performance improvement obtained by C^2M, its fairness properties, and study the impact of protocol parameters.

4.1.1 Access Point scenario

Infrastructure-based mode, where nodes communicate with an access point, is a commonly used architecture for IEEE 802.11-based wireless networks.

We evaluate the performance of C^2M in an infrastructure-based scenario with one access point. We vary the number of nodes within communication range of the access point from 1 to 32.

Figure 9 plots the aggregate network throughput when each node sets up a Constant Bit Rate (CBR) connection to the access point (the CBR transmission rates are chosen to be large enough to saturate the channel). As we can see from the figure, C^2M offers significant capacity improvements over both "802.11" and "802.11 Frag", especially with a small number of nodes. The difference between the "802.11" and "802.11 Frag" curves is indicative of the performance improvement obtained by using larger trains, while the difference between the "802.11 Frag" and "C^2M" curves indicates the performance improvement obtained by moving contention resolution to the control channel and using advance reservation. The magnitude of throughput improvement with C^2M over 802.11 is larger than the control channel data rate used (2 Mbps or 5.5 Mbps). Therefore, using the low rate channel as a control channel can achieve higher throughput than using both the low rate and high rate channel for data transmission.

When the number of nodes around the access point increases, the performance of C^2M approaches that of "802.11 Frag" primarily because of the increased contention resolution overheads. It should be noted that in our simulations, 32 nodes around the access point corresponds to 32 *simultaneously active flows*. In practice, the number of simultaneously active flows is often small, and C^2M can offer significant performance improvement in infrastructure-based networks.

Another interesting observation from Figure 9 is that the performance of C^2M is nearly the same with both 2 Mbps and 5.5 Mbps control channels for a small number of nodes. When the number of contending nodes is small, the average contention resolution time is smaller than the average data transfer time even with a 2 Mbps control channel. Hence, a 5.5 Mbps control channel does not improve performance over a 2 Mbps control channel. When the number of nodes increases, the average time for contention resolution increases because of RTS collisions, and at this point larger control channel data rate is useful.

Figure 10 plots the aggregate network throughput when each node sets up a FTP connection (that runs over TCP) to the access point. In this figure, we see that C^2M still performs better than "802.11", although performance of C^2M with 2 Mbps data rate degrades under higher congestion. The link latency of C^2M is larger than "802.11" as a lower control channel data rate

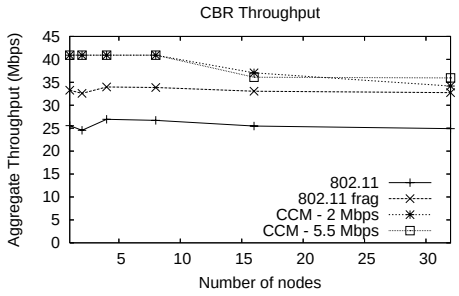

Figure 9: Access Point: CBR Throughput

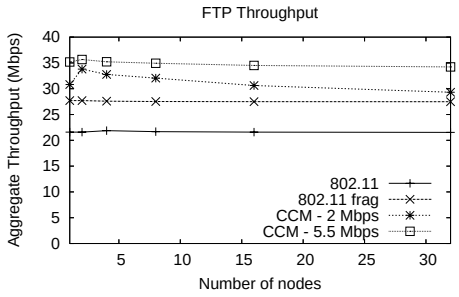

Figure 10: Access Point: FTP Throughput

results in a longer contention resolution duration, and the use of data aggregation adds some latency as well. Higher link latency translates to lower TCP throughput because the throughput of TCP is inversely proportional to the RTT of the path between the source and the destination. In this case, the use of a higher rate (5.5 Mbps) control channel improves performance over using a lower rate (2 Mbps) control channel as the link latency due to contention resolution is reduced. Thus, for performance improvements with TCP traffic, especially with high degrees of contention, larger control channel data rates may be beneficial. Alternatively, data has to be aggregated into bigger trains to reduce the contention resolution overhead per data packet.

4.1.2 Fairness properties

The C^2M protocol aggregates data into a train of packets, and sends the whole train together. Transmitting multiple packets back to back may affect the fairness of C^2M. We compare the fairness of C^2M with 802.11 using

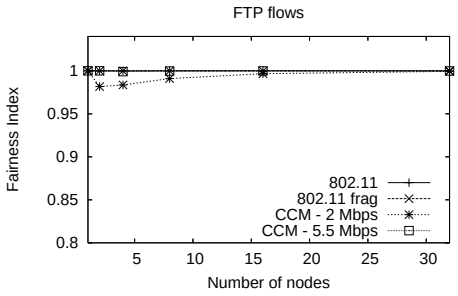

Figure 11: Access Point: Fairness of FTP flows

Jain's Fairness Index [2], defined as,

$$\text{fairness index} = \frac{(\sum_f T_f)^2}{N * \sum_f T_f^2}$$

where T_f represents the throughput of a flow f (between a node and the access point), and N is total the number of flows. Fairness index values closer to 1 indicate better fairness. We evaluate the fairness of C^2M under the access point scenario.

Figure 11 compares the fairness index of C^2M with 802.11. As we can see from the figure, the fairness properties of C^2M are comparable to that of 802.11. This indicates that the proposed C^2M protocol preserves long-term fairness properties, though in the short-term (order of few packet transmission times) there may be some unfairness.

4.1.3 Impact of advance reservation on performance

Figure 12 plots the CBR throughput for the access point scenario with different values of *ReserveAheadLimit* parameter. C^2M is evaluated with a 2 Mbps control channel data rate. The results are for packet trains of size 3. *ReserveAheadLimit* of 1 implies a node can have at most one packet train reservation pending, and corresponds to the simple "pipelining" scheme proposed in the past [1].

As we can see from the Figure 12, higher values of *ReserveAheadLimit* results in significantly better performance under high contention. For example, with 32 nodes, using a parameter value of 4 improves throughput by 7 Mbps over using a parameter value of 1. On the other hand, when the contention is low, even a *ReserveAheadLimit* value of 1 suffices.

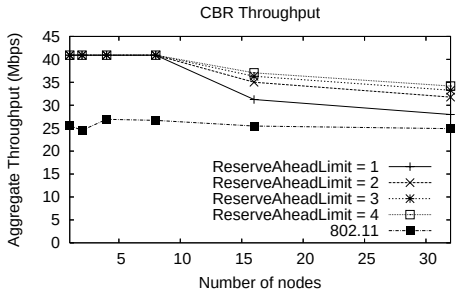

Figure 12: Impact of advance reservation

The advance reservation technique is used to smoothen large variations in the contention resolution duration. When the number of contending nodes is small, variations in contention resolution duration are small as well, and advance reservation is not necessary. On the other hand, when the number of contending nodes increase, variations in contention resolution duration become large. As a result, the benefits of advance reservation are clearly evident at higher contention.

We next evaluate the impact of advance reservation on fairness. Figure 13 compares the fairness index of C^2M with different *ReserveAheadLimit* values. As we can see from the figure, using larger values of advance reservation has minimal impact on fairness. The advance reservation protocol does not bias toward any particular node. The contention resolution is still based on random backoff values, even when advance reservation is being used. As a result, C^2M does not discriminate against any flow, leading to good fairness properties.

Although using large values of advance reservation does not affect fairness, it is still not appropriate to use very large values. When the clock drift among nodes is high, two non-overlapping reservations made far in advance may overlap at the time of packet transmission due to clock drifts, resulting in packet collisions. We have not evaluated the impact of clock drifts in this paper, but we believe that clock drifts will impact the advance reservation only if very large values of advance reservation are used.

4.1.4 Impact of train size on performance

Figure 14 plots the CBR throughput for the access point scenario with different values of *AggregationLimit* parameter, which specifies the maximum

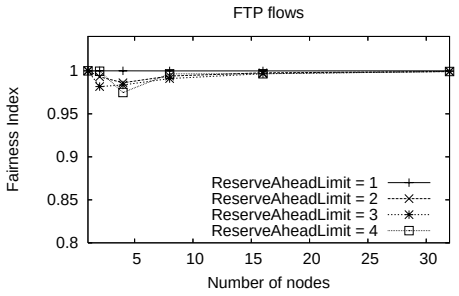

Figure 13: Access Point: Fairness with different values of advance reservation

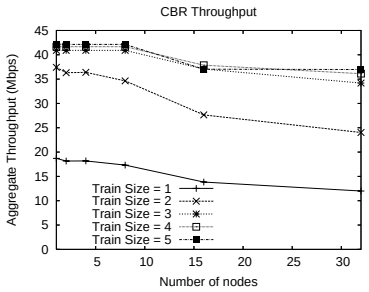

Figure 14: Impact of packet train size

number of packets in a packet train. *ReserveAheadLimit* is set to 2. As we can see from the figure, we need the train size to be at least 3 packets to achieve good performance. If smaller train sizes are used, control channel will become a bottleneck, degrading performance. On the other hand, using too large a train size does not help either, as the data channel will then be a bottleneck to performance, and therefore the throughput obtained stabilizes beyond a threshold train size.

4.2 Performance with multi-hop communication

C^2M increases link latency on account of data aggregation and the use of a lower rate channel for contention resolution. To quantify the impact of the additional latency, we evaluate the performance of C^2M in a multi-hop chain topology.

We setup a chain of nodes to evaluate the multi-hop performance of C^2M. The number of nodes in a chain is varied from 2 to 10. Nodes in a chain

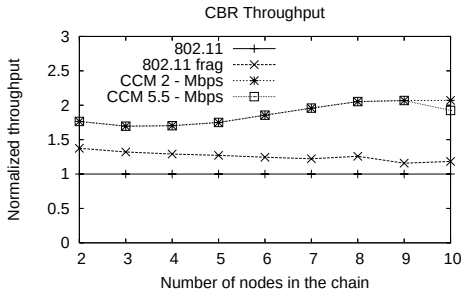

Figure 15: Chain Topology: CBR Throughput

are stationary, and direct communication is possible only between adjacent nodes on the chain (distance between adjacent nodes is 40m). One flow is setup from the first node of the chain to the last node of the chain. Since, the end-to-end throughput varies with the length of the chain, we normalized all throughputs with the throughput of "802.11", for ease of comparison.

Figure 15 plots the throughout of a CBR flow for different length chains. C^2M significantly improves the throughput even with multi-hop traffic. For example, C^2M attains around 75% improvement over "802.11", and 30% improvement over "802.11 Frag", for a single multi-hop CBR flow. Since the contention is low when a single multi-hop flow is active, the throughput obtained with both the 2 Mbps control channel, and the 5.5 Mbps control channel is the same.

Figure 16 plots the throughout of a FTP flow for different length chains. As we can see from the figure, C^2M offers better performance over "802.11" when the length of the chain is small, but the performance of C^2M significantly degrades with longer chains. As we noted earlier, C^2M increases the link latency, when compared to 802.11. This in turn results in higher end-to-end RTT, degrading TCP performance. When the number of hops on a path increases, the cumulative increase in the end-to-end RTT is larger, resulting in higher throughput degradation.

The increased link latency seems to be inherent in any approach that uses a low rate control channel. When contention resolution is performed over a slower channel, it inevitably requires more time, adding to the link latency. Furthermore, if the control channel bandwidth is insufficient, it may be necessary to send data on the data channel after aggregation, which further increases the latency. As a result, we believe that the control channel

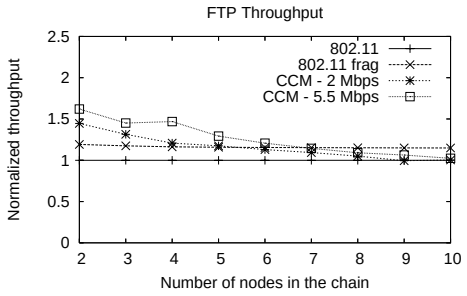

Figure 16: Chain Topology: FTP Throughput

approach is not appropriate for networks where data traffic is predominantly TCP-based and the average path length is large (greater than 5). However, as we see from Figure 15, control channel significantly improves CBR performance. Therefore, control channel may still be appropriate for networks that use UDP-based traffic (e.g., multimedia traffic).

4.3 Impact of mobility

We evaluate the performance of C^2M with mobility in 10 random scenarios. 50 nodes are placed in a 500m square area, and the initial location of the nodes is randomly generated. 5 FTP connections are set up between randomly selected pairs of nodes. Therefore, the total number of flows over the 10 topologies is 50. We normalize the throughput for each flow with respect to 802.11 for ease of comparison.

Figure 17 plots the normalized throughput with respect to the average path length for each flow (each of the 50 flows is represented by a point). With mobility, a single flow will use multiple routes over the course of the simulation. The path length of a flow is averaged over all routes used by the flow. The average path length in the simulation ranges from 1 to 5.

As we can see from the figure, C^2M performs better than 802.11 for most of the flows. Hence, C^2M is suitable for operation in mobile networks as well. Furthermore, the throughput improvement seems to be similar with different path lengths as well. Although the multi-hop performance of C^2M with a single FTP flow is not good (as we saw earlier), when multiple FTP flows are active, the impact of increased latency is less severe. When multiple FTP flows are active, the flows contend with each other, resulting in longer RTTs

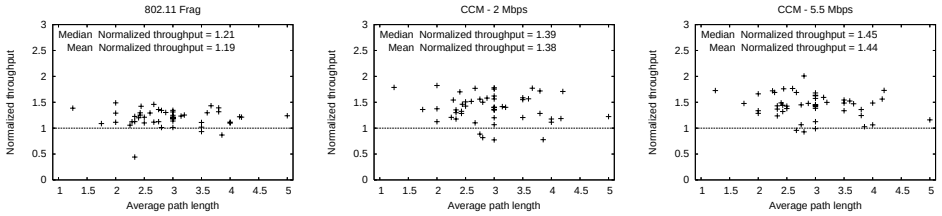

Figure 17: Impact of mobility: Normalized flow throughputs for varying path lengths

for all flows. When the path RTT increases because of network contention, the link latency introduced by C^2M is a less significant fraction of the total RTT. Hence, C^2M does perform well with FTP when there are multiple contending flows.

4.4 Summary of results

The simulation results have validated our findings from the analysis. The summary of our results is as follows:

- In access point scenarios, C^2M protocol outperforms both 802.11 and 802.11 with fragmentation enabled, and for both FTP and CBR traffic (Figures 9, 10). Although, C^2M uses data aggregation and advance reservation (Figures 11, 13) fairness is not affected. Therefore, the results suggest that C^2M is a promising solution for use in infrastructure-based networks.

- With very high contention, performance benefits of C^2M may be smaller (Figure 10). The control channel, due to its lower bandwidth can become a bottleneck in such situations. Impact of higher contention can be addressed by increasing the size of the packet trains sent on the data channel, so that the increased contention cost is amortized over more packets.

- In case of multi-hop networks, C^2M offers significant performance benefits over 802.11 for CBR (UDP) traffic (Figure 15). The improvement in throughput of a solitary FTP (TCP) flow over long (i.e. several hops) paths (Figure 16) is less significant due to latencies introduced by advance reservation and data aggregation.

- C^2M continues to offer significant performance improvements over 802.11, in random topologies with mobility (Figure 17). The results also suggest that the multi-hop performance of FTP is good when there are *multiple* FTP flows sharing the network. Therefore, it appears that C^2M is a promising solution for use in multi-hop networks as well.

5 Related Work

There are a large number of CSMA/CA based proposals for single channel wireless networks. IEEE 802.11 [6] is a widely used standard for wireless networks that extends earlier proposals such as MACA [7], MACAW [8], among others. C^2M is designed to use two channels, and improves the performance of a high rate *data channel* by using a low rate *control channel*.

Several proposals have considered the use of multiple channels by deploying a radio on each channel. In this paper, we have explored the scenario where the available channels are heterogeneous in the data rates supported. One approach for utilizing the heterogeneous channels is to use a link layer solution [9] that stripes data simultaneously across the available channels. However, past studies have shown that striping at the granularity of packets over heterogeneous links may lead to reordering of packets, which may adversely impact the performance of higher layer protocols such as TCP. Another approach is to use a routing layer solution [10] that is designed to operate with heterogeneous channels. However, with a routing layer solution, the lower rate radio may not always be used. On the other hand, C^2M always uses the lower rate channel.

Many multi-channel MAC protocols [11, 12, 13, 14] use a dedicated *control channel* for exchanging control packets. While, we propose to use the control channel for exchanging control packets as well, the motivation and design of our solution is different. The multi-channel MAC protocols use a control channel for enabling a node to rendezvous with other nodes. The control packets contain extra information for negotiating a channel to use for subsequent data transmission. The typical assumption in these multi-channel MAC solutions is that all channels can support the same data rate, and any channel can be used as a control channel. In contrast to these approaches, our proposed C^2M protocol is aimed at improving the efficiency of a high rate data channel by using a low rate control channel.

A few proposals have considered the use of a control channel specifically

to improve the throughput of the data channel. Li et al. [15] propose MAC-SCC, a control channel based protocol to improve network performance by moving the backoff and RTS-CTS exchange to the control channel. Yang et al. [1] present *pipelining* strategies to improve the performance of wireless MAC protocols. The available bandwidth is split into two sub-channels - a data channel and a control channel. Contention resolution on the control channel is *pipelined* with the data transmission on the data channel to reduce the resources spent on resolving contention. Ravichandran's thesis [16] studies the issues with using a *pipelining* strategy. The thesis demonstrates that the bandwidth required for control channel is dependent on the data packet sizes, and hence may be hard to estimate. In another study, Deng et al [17] conclude that splitting the bandwidth between a control and a data channel may not be beneficial, if the contention resolution duration is randomly distributed. Therefore, the key difficulty with using a pipelined approach is appropriately splitting the available channel bandwidth between the control channel and the data channel.

Earlier proposals could not sufficiently exploit the benefits of using a control channel, because the bandwidth needed for the control channel depends on the contention on the channel, and the application dependent data packet size. C^2M differs from past work by using *advance reservation* to overcome randomness in contention resolution duration, and *data aggregation* to overcome variations in the control channel data rate and size of data packets. Furthermore, the primary motivation for using a control channel in C^2M is to utilize bandwidth available in low frequency bands, and does not require splitting an existing channel into multiple sub-channels.

Control channel has been used in some protocols for transmitting "tones" instead of packets. Haas et al. [18] present Dual Busy Tone Multiple Access protocol that uses two tones on a control channel to reduce the impact of hidden terminals. Zhai et al. [19] have proposed a protocol that uses one control channel for tones, and a second control channel for exchanging RTS-CTS packets. Since tones do not encode any information, it is not possible to use tones for reserving the channel in advance. Thus, our proposed protocols rely on packets, instead of tones, for channel reservation. Tones are also susceptible to *aliasing*, as the identity of the sender of a tone is not known with certainty.

Cidon et al. [20] present a TDMA scheduling algorithm that uses a control channel for exchanging reservation information. Data channel uses TDMA scheduling, and the focus of this paper is on achieving *good* TDMA schedules

in a distributed fashion. Zhu et al. [21] present a TDMA-based broadcast scheduling protocol. Nodes reserve time slots by contending using a multi-hop ALOHA scheme, and exchange control packets to complete the reservation. Our proposed protocol uses *advance reservation* to reserve the data channel, which to an extent is similar to a TDMA schedule. We differ from TDMA-based solutions in using CSMA/CA on the control channel for contention resolution (i.e., binary exponential backoff with RTS-CTS exchange).

Directional communication has been advocated as an approach for improving network performance. Most of the directional MAC protocols proposed in literature [22, 23, 24, 25] are designed to use a single antenna. In contrast, we have proposed to use omni-directional antenna on the control channel, and a directional antenna on the data channel.

For obtaining the full benefits of directional communication, existing proposals have advocated the use of directional transmissions for data as well as control packets. However, the use of directional control packets leads to a problem known as "Deafness" [23], wherein the performance of directional communication is poor in certain topologies.

Recent research has proposed two solutions for addressing deafness. Korakis et al. [25] have proposed a "Circular RTS" protocol wherein a directional RTS packet is transmitted in multiple directions to inform all nodes in the neighborhood of impending communication. However, transmitting multiple RTS packets significantly increases overhead, degrading performance in topologies that are not subject to deafness.

Roy Choudhury et al. [26] have proposed "ToneDMAC", a protocol that uses "tones" on a control channel to signal the end of a communication. Tone "aliasing" may occur if two tones overlap in time, or if there is noise or fading on the channel. As a result, the use of tones does not guarantee deafness will be alleviated in all cases. In our discussions, we outline extensions to C^2M that allows the full benefits of directional communication to be exploited, while overcoming the deafness problem.

6 Extensions for directional antennas

Directional communication has been proposed as a means of improving the capacity of wireless networks. However, such antennas are still expensive and are not yet in widespread use. Therefore, we have designed our C^2M

protocol with omni-directional antennas in mind. However, it is possible to extend the protocol to work with directional antennas. We present our initial design here.

We begin by noting that the IEEE 802.11 MAC is also not designed for use with directional antennas. Many schemes have been proposed to extend IEEE 802.11 for use with directional antennas. The directional MAC proposals fall into two key categories based on the antenna mode used for RTS-CTS exchange.

1. Omni-directional RTS-CTS: RTS and CTS packets are sent omni-directionally, and DATA-ACK packets are exchanged directionally. However, using omni-directional RTS-CTS reduces the spatial reuse.

2. Directional RTS-CTS: RTS and CTS packets are sent directionally. The direction to be used for communication is assumed to have been discovered a priori. When directional RTS and CTS is used, spatial reuse increases. However, directional RTS-CTS may result in poor performance in certain topologies due to a problem known as "Deafness" [26]. The problem can be illustrated by the following example.

Assume nodes A and B are communicating after a prior directional RTS-CTS exchange. Node C wishes to communicate with B and initiates a directional RTS. But B is beam-formed toward A and fails to receive the RTS. Node C may misinterpret the absence of CTS as a sign of RTS collision due to congestion, and increase its backoff. It is shown in [26] that deafness can seriously degrade throughput and fairness. Deafness problem arises primarily because the use of directional RTS-CTS results in some nodes in the neighborhood being unaware of the ongoing communication.

We propose to use omni-directional transmissions on the control channel, while using directional transmissions on the data channel. Our proposal is simple to implement, and derives the spatial reuse benefits of directional antennas, while not suffering from deafness. Consider the example we mentioned earlier. When node C initiates a RTS to node B on the control channel, node B will respond with a CTS since the data communication with A is proceeding on the data channel. As a result, the control channel architecture provides a simple solution for exploiting the spatial reuse benefits of directional antennas, while not incurring the performance penalty of deafness. These benefits are in addition to the benefits of using the control channel approach described earlier. The use of an omni-directional control channel also simplifies the problem of neighbor discovery.

For the correct operation with omni-directional RTS-CTS, the range of

omni-directional RTS-CTS must be at least as large as the range of directional transmissions on data channel. But, the control channel inherently supports a longer range as it operates at a lower frequency, and if additional range is necessary, the transmission power on the control channel can be suitably increased. Our initial simulation results show that our protocol outperforms the ToneDMAC protocol proposed in [26]. We are currently carrying our a more detailed comparison study.

7 Conclusion

We have studied the benefits of separating control from data through the design of C^2M, a control channel-based wireless MAC protocol. We have shown that moving the control traffic to a separate low-rate channel can significantly improve the performance of wireless networks. By using advance reservation and packet aggregation, C^2M provides performance improvement in excess of the data rate of the control channel. A low-rate control channel be implemented over a sliver of unused spectrum that is available in the lower frequency bands.

There are several avenues for future work. *(i)* Several researchers have proposed modifications that improve performance of 802.11 contention resolution algorithm. Many of these schemes can be applied to C^2M as well. *(ii)* We are considering schemes that perform optimistic and anticipatory reservations in order to lower the cost of contention resolution. *(iii)* We are working on more detailed evaluation of C^2M protocol with directional antennas. *(iv)* We plan to explore whether in some situations it might be better to use the control channel for transmitting data. Note that the range of the data channel is generally lower than that of the control channel. If two nodes are far apart, the data channel may either fail to provide connectivity or may operate only at very low rates. In such cases, the protocol should automatically use the control channel to transfer data.

References

[1] Xue Yang and Nitin Vaidya, "Explicit and Implicit Pipelining for Wireless Medium Access Control," in *Vehicular Technology Conference*, 2003.

[2] R. Jain, G. Babic, B. Nagendra, and C. Lam, "Fairness, call establishment latency and other performance metrics," Tech. Rep. ATM_Forum/96-1173, ATM Forum Document, August 1996.

[3] Theodore Rappaport, *Wireless Communications Principles and Practice*, Prentice Hall, 2002.

[4] J. Mogul and S. Deering, "Path mtu discovery," RFC 1191, Apr. 1991.

[5] Scalable Network Technologies, "Qualnet simulator version 3.6," http://www.scalable-networks.com.

[6] *IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification, P802.11*, 1999.

[7] P. Karn, "MACA - A New Channel Access Method for Packet Radio," in *Proceedings of 9th Annual ARRL Networking Conference*, London, Ontario, Cananda, 1990.

[8] V. Bhargavan, A. Demers, S. Shenker, and L.Zhang, "MACAW: A media access protocol for wireless lans," in *ACM SigComm*, London, UK, September 1994.

[9] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks ," in *IEEE International Conference on Broadband Networks (Broadnets)*, 2004.

[10] Richard Draves, Jitendra Padhye, and Brian Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *ACM Mobicom*, 2004.

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

[12] A. Nasipuri and S.R. Das, "Multichannel CSMA with Signal Power-based Channel Selection for Multihop Wireless Networks," in *Vehicular Technology Conference*, September 2000.

[13] Nitin Jain, Samir R. Das, and Asis Nasipuri, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop-Wireless Networks," in *9th Int. Conf. on Computer Communications and Networks (IC3N*, Phoenix, Arizona, Oct. 2001.

[14] Shih-Lin Wu, Chih-Yu Lin, Yu-Chee Tseng, and Jang-Ping Sheu, "A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks," in *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, 2000.

[15] Yijun Li, Hongyi Wu, Dmitri Perkins, Nian-Feng Tzeng, and Magdy Bayoumi, "MAC-SCC: Medium Access Control with a Separate Control Channel for Multihop Wireless Networks ," in *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)*, May 2003, pp. 764–769.

[16] Priya Ravichandran, "Explicitly Pipelining IEEE 802.11 to Enhance Performance," M.S. thesis, University of Illinois at Urbana-Champaign, December 2003.

[17] Jing Deng, Yunghsiang S. Han, and Zygmunt J. Haas, " Analyzing Split Channel Medium Access Control Schemes with ALOHA Reservation," in *Ad-Hoc, Mobile, and Wireless Networks - ADHOC-NOW '03*, S. Pierre, M. Barbeau, and E. Kranakis, Eds. 2003, pp. 128–139, Springer-Verlag.

[18] Zygmunt J. Haas and Jing Deng, "Dual Busy Tone Multiple Access (DBTMA) - A Multiple Access Control for Ad Hoc Networks," *IEEE Transactions on Communications*, vol. 50, no. 6, pp. 975–985, June 2002.

[19] Hongqiang Zhai, Jianfeng Wang, Yuguang Fang, and Dapeng Wu, "A Dual-Channel MAC Protocol for Mobile Ad Hoc Networks," in *IEEE Workshop on Wireless Ad Hoc and Sensor Networks, in conjuction with IEEE Globecom 2004*, Dallas, Texas, USA, Nov 2004.

[20] Israel Cidon and Mishe Sidii, "Distributed assignment algorithms for multihop packet radio networks ," *EEE Transactions on Computers*, vol. 38, no. 10, pp. 1353–1361, Oct. 1989.

[21] Chenxi Zhu and M. S. Corson, "A Five-Phase Reservation Protocol (FPRP) for Mobile Ad Hoc Networks," *Wireless Networks*, vol. 7, no. 4, pp. 371–384, 2001.

[22] Ram Ramanathan, "On the performance of ad hoc networks with beamforming antennas," in *Mobihoc*, 2001.

[23] Romit Roy Choudhury, Xue Yang, Ram Ramanathan, and Niitin H. Vaidya, "Using directional antennas for medium access control in ad hoc networks," in *MOBICOM*, 2002.

[24] Karthikeyan Sundaresan and Raghupathy Sivakumar, "A unified MAC layer framework for ad-hoc networks with smart antennas," in *MobiHoc*, Roppongi Hills, Tokyo, Japan, 2004, pp. 244–255.

[25] Thanasis Korakis, Gentian Jakllari, and Leandros Tassiulas, "A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks," in *MobiHoc*, Annapolis, Maryland, USA, 2003, pp. 98–107.

[26] Romit Roy Choudhury and Nitin H. Vaidya, "Deafness: A MAC Problem in Ad Hoc Networks when using Directional Antennas," in *ICNP*, 2004.

Appendix: Detailed Equations

We derive the detailed equations used in the analysis. We use IEEE 802.11a parameters for the data channel, and IEEE 802.11b parameters for the control channel. The parameter values *SlotTime*, *SIFS*, *DIFS*, and *PLCPDuration* that are used below are listed in Table 1.

The contention resolution duration involves a backoff followed by a RTS-CTS exchange. The average number of slots per backoff attempt, *AverageSlots*, in the absence of collision is equal to $(CWmin+1)/2$, which we assume to be 8 slots for both the data and the control channel. The average backoff duration, T_b, is given by,

$$T_b = SlotTime * AverageSlots + DIFS$$

The RTS and CTS packets are assumed to be transmitted at the maximum rate allowed by the channel. The size of RTS is 160 bits, and the size of CTS is 112 bits. Each packet is prefixed by a PLCP header, which requires *PLCPDuration* for transfer. Between a RTS and a CTS there is a wait of *SIFS* duration. The time required for RTS-CTS exchange, T_e, is given by,

$$T_e = 2 * PLCPDuration + \frac{(160 + 112)}{ChannelRate} + SIFS$$

The total time for contention resolution is $T_e + T_b$. Substituting the appropriate values (IEEE 802.11a values for the data channel, and IEEE 802.11b values for the control channel) we can estimate the contention resolution time for control and data channel.

The time for contention resolution on the control channel, TC_c, with control channel data rate R_c is given by,

$$TC_c = 412 + \frac{272}{R_c}$$

Similarly, the time for contention resolution on the data channel, TC_d, with data channel data rate R_d is given by,

$$TC_d = 170 + \frac{272}{R_d}$$

The data transfer phase consists of DATA-ACK exchange, each of which is also preceded by a PLCP header. The DATA packet is sent *SIFS* duration after CTS, and the ACK packet is sent *SIFS* duration after the data. The DATA packet is assumed to be s bits long, and the size of ACK is 112 bits. The DATA packet is preceded by a header which is 224 bits long. Therefore, the total time for data transfer on the data channel, $DT_d(s)$, is given by,

$$\begin{aligned} DT_d(s) &= 2 * SIFS + 2 * PLCPDuration + \frac{(s + 336)}{R_d} \\ &= 80 + \frac{(s + 336)}{R_d} \end{aligned}$$

The equations for TC_c, TC_d and $DT_d(s)$ are used to generate the analysis graphs.