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

Pradeep Kyasanur
University of Illinois at Urbana-Champaign
kyasanur@uiuc.edu

Jitendra Padhye
Microsoft Research
padhye@microsoft.com

Paramvir Bahl
Microsoft Research
bahl@microsoft.com

***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. Several small slices of frequency are available in lower frequency bands that are not utilized. We propose utilizing a sliver of unused spectrum in the lower frequency band as a low rate control channel to improve the capacity of infrastructure and multi-hop wireless networks. The proposed Control Channel-based MAC Protocol (C^2M) increases the throughput by moving the contention resolution overheads to the 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.**

I. 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 [1]–[4] 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.

Several small slices[1] of frequency (1 to 2 MHz) are available globally in the lower frequency bands (e.g., below 900 MHz in ISM bands). These orphan slices are not wide enough for deploying large-scale data communication systems and therefore, are not utilized. Radio frequency (RF) communication equipment is becoming inexpensive, and 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. For example, 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. In this paper, we investigate the benefits of using such a small frequency slice available in a lower frequency band to build a split-channel MAC.

Separating control and data traffic on different channels has been studied in the past [1]–[4]. These schemes typically assume that the total available bandwidth can be suitably split among the data and control channels. In the scenario we consider, the bandwidth of the control channel is fixed (equal to the bandwidth of the available frequency slice), and therefore the ratio of bandwidths of data and control channel cannot be appropriately selected. Under this scenario, the control channel may become a bottleneck to performance if the control channel bandwidth is not sufficiently large, and the split-channel approach can have worse performance than a single channel approach [1], [5]. We show that even if the control channel bandwidth is limited, significant performance benefits are possible. To obtain these performance benefits, our solution extends past work on using a control channel by incorporating techniques of *packet aggregation* and *advance reservations*.

Past work has also typically assumed that the transmission range of control and data channels are the same. Since we are considering the scenario where the control and data channels may be in different frequency bands, the ranges of the two channels is expected to be different. Specifically, the control channel is situated in the lower frequency band, and is expected to have longer transmission range. We exploit the longer frequency control channel in reducing the effect of hidden terminals on the data channel.

To quantify the benefits of using a small frequency slice in a lower frequency band for control purposes, 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, long-range channel and the data portion over a high-frequency, high-data rate, short-range 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 evaluate C^2M in infra-structure, static multi-hop, and mobile multi-hop network configurations and find the results to be promising. 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

[1]We use terms slice, band, and channel interchangeably.

the nodes use the lower frequency band for exchanging data packets as well. For example, using a 2 Mbps control channel and a 54 Mbps data channel, we find that:

- In a single-hop Access Point based configuration with 16 nodes running FTP, C^2M gives 40% (12 Mbps) improvement over standard 802.11 protocol.
- In a 4-node multi-hop chain configuration, C^2M gives 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 spectrum, and it is relatively easy to build a 2 Mbps wireless network with as little as 1 MHz of spectrum.

The paper makes the following contributions. We propose a split-channel medium access control protocol that incorporates unused orphan low-frequency spectrum slices into the MAC to increase 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 exploit the longer range of low frequency control channel to reduce the effect of hidden terminals on the data channel. We demonstrate the benefits of our architecture for infra-structure and multi-hop wireless networks.

The rest of the paper is organized as follows: We present an overview of C^2M protocol in Section II and describe the details in Section III. In Section IV we present detailed results from our performance evaluation. We discuss extensions of the split-channel approach in Section V. Related work is covered in Section VI, and we conclude in Section VII.

II. The proposed protocol

In this section, we give an overview of our proposed C^2M protocol. C^2M is similar to IEEE 802.11 [6] in that it uses CSMA/CA for controlling access to the channel, and also uses control packets (RTS-CTS) for channel reservation.

In IEEE 802.11 each data packet exchange has two phases. 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.

A. 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 an 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 1 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]–[4], but the novelty of our approach is to utilize bandwidth available in a low frequency band (that might otherwise be unused) for contention resolution. Furthermore, prior solutions assumed that the control channel bandwidth can be appropriately chosen, while we design solutions that can work with a given control channel bandwidth.

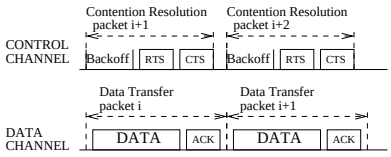

Fig. 1. Parallel operation of control and data channels

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 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 2 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

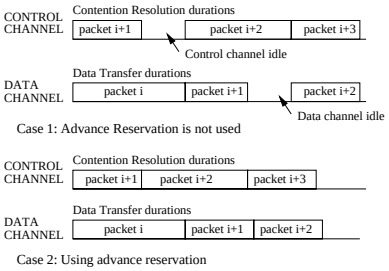

Fig. 2. Benefit of using advance reservation on control channel

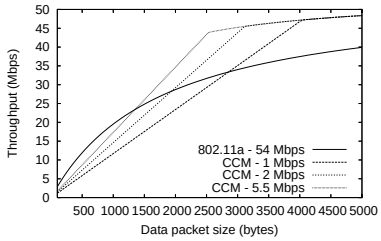

Fig. 3. Analytical comparison of C^2M with IEEE 802.11a

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 data packet size necessary to ensure control channel is not a bottleneck.

B. Implication of characteristics of control channel

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.

1) Control channel data rate: We present a detailed analysis of C^2M throughput assuming IEEE 802.11 parameters and plot the results in [7]. Here, we only present one of the plots derived from the throughput equations.

In Figure 3, 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 [2], 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.

Based on the above plot and from the analysis in [7], 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 larger than a threshold (Figure 3). 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 that control channel is not a bottleneck depends on the data channel rate. For a given control channel 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. 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.
- Collisions during contention resolution and fragmentation overheads is not considered in the analysis. With collisions, average contention resolution time may increase, requiring a larger threshold data packet size. On the other hand, the performance of 802.11a may be worse when fragmentation overheads are accounted for.

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

[2]We ignore fragmentation and collision overheads in the analysis, but our simulation results do account for both fragmentation and collision overheads.

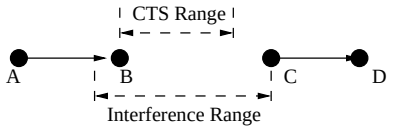

Fig. 4. Scenario where longer control channel range is beneficial

of the aggregation procedure in Section III.

2) Control channel range: Besides having different data rates, the data and the control channel are also likely to have different ranges. The control channel is expected to be located in a lower frequency band. According to the free space propagation model [8], with unit gain antennas, the path loss of a channel is 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.

We exploit the longer control channel range to reduce the effect of hidden terminals on the data channel. Typically, the range up to which a transmission may interfere with another transmission (called "interference range"), is longer than the transmission range. Single channel protocols such as IEEE 802.11, and most split channel approaches as well [1]–[4] use a common transmission range for RTS-CTS, as well as DATA-ACK exchange. Using a common transmission range can result in packet losses on the data channel due to "hidden terminals". For example, in Figure 4, suppose node B is receiving data from a node A (after an earlier RTS-CTS exchange). Node C is outside the communication range of B, and hence would not have received the CTS from B. Therefore, node C can schedule a transmission to node D on the data channel during this time (note that A and B are unaware of RTS sent by C). 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 of the data channel, all nodes in the interference region can be notified of the impending transmission, thereby preventing data packet collisions. In the above example, node C would have received the RTS from node B on the control channel when the control channel range is at least as large as the interference range of the data channel. Although there will be losses on the control channel due to hidden terminals, by using *a longer control channel range hidden terminal losses on data channel can be avoided*. Losses on the control channel can reduce network throughput, only if the average contention resolution time on the control channel becomes larger than the average data transfer time on the data channel. However, the data aggregation technique is used to ensure the average data transfer time is always larger than the average contention resolution time.

If the transmission range of the control channel is too large, 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.

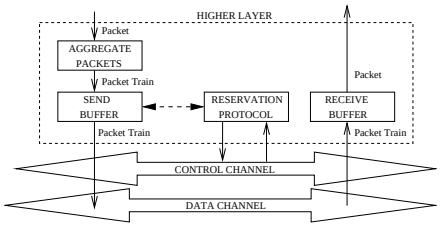

Fig. 5. Architecture of control channel protocol

III. Detailed protocol architecture

C^2M protocol has three main components.

1) Data aggregation: Aggregate packets to a particular destination into a 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 5 depicts the interaction between the different components of the control channel MAC. We next describe each component in detail.

A. 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 [9] 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.

The aggregation protocol increases the latency of most packet transmissions due to the delay introduced by 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 belonging to each flow continue to be sent in order.

B. 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.

In a single hop network, since all nodes are in transmission range of each other and can receive all packet transmissions, it may be possible to implement reservations using a simpler mechanism. For example, nodes can only track the number of pending data transmissions. When nodes receive a RTS, the pending transmission count is increased, and when a data transmission completes, the pending transmission count is decreased. However, this approach cannot be used in multihop networks, as all nodes do not see the same channel state.

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.

C. 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.

IV. Evaluation

The control channel MAC protocol has been implemented in Qualnet 3.6 [10]. We have implemented the protocol as a *shim layer* between the MAC and the network layer.

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.

A. 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.

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 6 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* (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 (e.g., packet striping).

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 6 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.

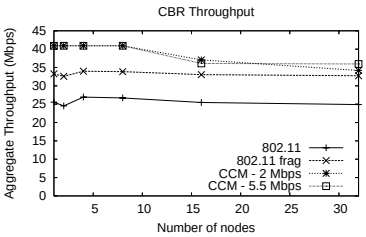

Fig. 6. Access Point: CBR Throughput

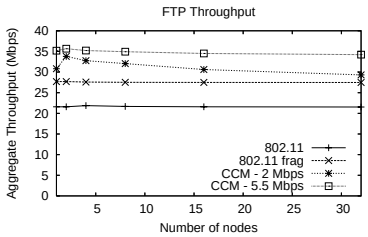

Fig. 7. Access Point: FTP Throughput

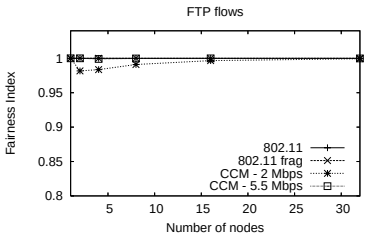

Fig. 8. Access Point: Fairness of FTP flows

Figure 7 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 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. Existing control channel approaches [1]–[4] have mostly assumed UDP traffic, and have therefore not noted the adverse impact added latency of contention resolution over a low rate channel may have on TCP performance.

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 Jain's Fairness Index [11], 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 8 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.

3) Impact of advance reservation on performance: Figure 9 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 9, 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.

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 10 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) Impact of train size on performance: Figure 11 plots the CBR throughput for the access point scenario with different values of *AggregationLimit* parameter, which specifies the maximum 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.

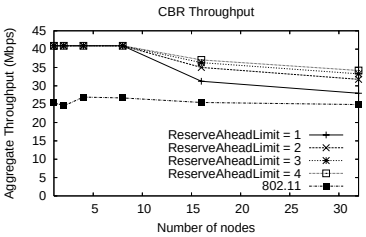

Fig. 9. Throughput with advance reservation

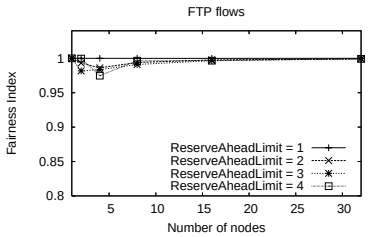

Fig. 10. Fairness with advance reservation

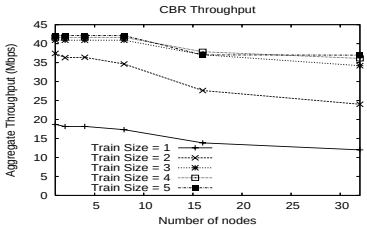

Fig. 11. Impact of packet train size

B. Performance with multi-hop communication

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 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 12 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. In longer chains, there exist pairs of nodes that are in interference range of each other but outside transmission range ("hidden terminals"). The throughput benefit of C^2M increases as the chain length increases as *C^2M can avoid the impact of hidden terminals on the data channel*, while 802.11 can not (recall that the longer range control channel is used to prevent hidden terminal effects on the data channel).

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 with FTP traffic. Figure 13 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 approach is not appropriate for networks where data traffic is predominantly TCP-based and the average

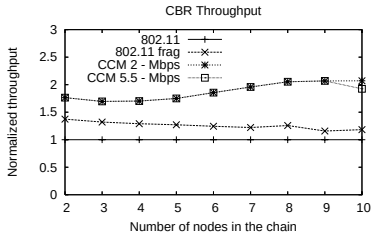

Fig. 12. Chain Topology: CBR Throughput

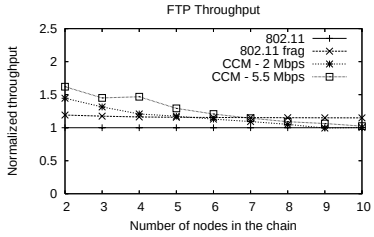

Fig. 13. Chain Topology: FTP Throughput

path length is large (greater than 5). However, as we see from Figure 12, control channel significantly improves CBR performance. Therefore, control channel may still be appropriate for networks that use UDP-based traffic (e.g., multimedia traffic).

C. 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 14 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 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 for all flows. When the path RTT increases because of network contention, the

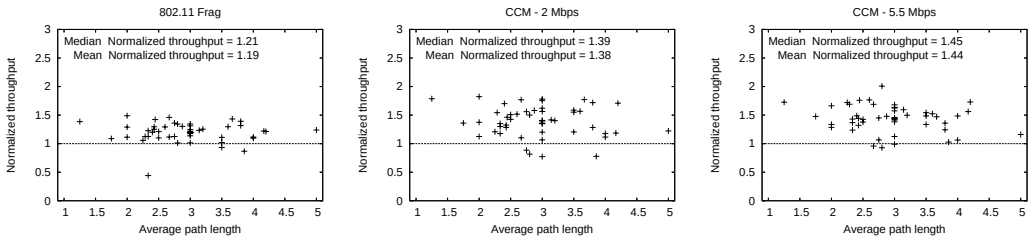

Fig. 14. Impact of mobility: Normalized flow throughputs for varying path lengths

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.

D. Summary of results

The summary of our results is as follows:

1) In access point scenarios, C^2M protocol outperforms both 802.11 and 802.11 with fragmentation enabled, for both FTP and CBR traffic (Figures 6, 7). The magnitude of throughput improvement with C^2M over 802.11 is *larger than the control channel data rate* used (for both 2 Mbps or 5.5 Mbps channels). 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 (e.g., packet striping). Although, C^2M uses data aggregation and advance reservation (Figures 8, 10) fairness is not affected. Therefore, the results suggest that C^2M is a promising solution for use in infrastructure-based networks.

2) In case of multi-hop networks, C^2M offers significant performance benefits over 802.11 for CBR (UDP) traffic (Figure 12) by moving contention to lower rate channel and by reducing hidden terminals on the data channel. The improvement in throughput of a solitary FTP (TCP) flow over long (i.e. several hops) paths (Figure 13) is less significant due to latencies introduced by advance reservation and data aggregation.

3) C^2M continues to offer significant performance improvements over 802.11, in random topologies with mobility (Figure 14). 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.

V. Discussions

The use of a low frequency control channel may be beneficial with directional antennas as well. The control channel can continue to use omni-directional communication to ensure all nodes receive RTS-CTS (channel reservation) packets. However, the data channel may use a directional antenna to improve received signal strength, leading to higher data rates. One requirement with using omni-directional RTS-CTS is that 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. If additional range is necessary, the transmission power on the control channel can be suitably increased. Some of our preliminary evaluations suggest that C^2M offers significant performance benefits with directional antennas as well.

We have restricted our evaluation of C^2M to simulations as it was not feasible to implement the protocol with currently available hardware. As a first step toward deploying control-channel based protocols, we have designed and built 900 MHz radio hardware that supports a CSMA MAC. Initial tests of the hardware have been completed, and we are conducting more detailed performance evaluation studies. In addition, we believe C^2M implementation can be done using some of the features supported by 802.11e. For example, the proposed data train mechanism can be implemented using TXOP and block ACK features in 802.11e. Reservation packets (RTS/CTS) can be implemented as broadcast packets, with the priority feature in 802.11e enabling CTS to be sent immediately after the reception of a RTS.

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 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.

VI. 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 uses a single channel.

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*. We restrict our comparisons to CSMA/CA MAC-based solutions (other possibilities include using TDMA approaches [12], [13], link layer approaches [14], routing approaches [15] , etc.).

A control channel has been used in some protocols for transmitting "tones" instead of packets (e.g., [4], [16]). Since tones do not encode any information, it is not possible to use tones for reserving the channel in advance. Thus, our

proposed protocol relies on packets, instead of tones, for channel reservation.

Many multi-channel MAC protocols (e.g., [17], [18]) use a dedicated *control channel* for exchanging control packets. However, the control channel is typically used to enable a node to rendezvous with other nodes. Furthermore, 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.

A few proposals have considered the use of a control channel specifically to improve the throughput of the data channel [1]–[4], [19]. Li et al. [2] 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. Tantra et al. [3] propose the use of a low-rate control channel to improve the performance of a high-rate data channel. However, their proposal requires the presence of an access point, and is therefore restricted to infrastructure-based networks, while our proposal is suitable for multi-hop networks as well.

Ravichandran's thesis [20] studies the benefits of a split-channel strategy and demonstrates that the bandwidth required for control channel is dependent on the data packet sizes. In another study, Deng et al [5] 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 split-channel approach is appropriately splitting the available channel bandwidth between the control channel and the data channel. In contrast, our proposal is designed to operate with a given control channel rate, and also exploits the benefits of a longer range control channel.

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 low frequency bands, which does not require splitting an existing channel into multiple sub-channels.

Aggregating packets before transmission has been proposed in the past in single channel networks [21], [22] to reduce the overheads with small data packets. We differ from those approaches by using packet aggregation for ensuring control channel does not become a bottleneck. Another related approach is the block ACK feature in 802.11e which allows using a single ACK for multiple data packets.

VII. Conclusion

We have proposed the use of a sliver of unused spectrum that is available in the lower frequency bands as a low-rate control channel. We have studied the benefits of using such a small frequency slice as a control channel through the design of C^2M, a split-channel wireless MAC protocol. We have shown that moving the control traffic to a separate low-rate channel in conjunction with the use of advance reservation and packet aggregation can significantly improve the performance of wireless networks. C^2M provides performance improvement in excess of the data rate of the control channel, and is therefore superior to a data striping approach.

References

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

[2] 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 *ICDCSW'03*, May 2003.

[3] Juki Wirawan Tantra, Chuan Heng Foh, and Bu Sung Lee, "An efficient scheduling scheme for high speed ieee 802.11 wlans," in *IEEE Vehicular Technology Conference (VTC Fall 2003)*, 2003.

[4] 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 conjunction with IEEE Globecom 2004*, Dallas, Texas, USA, Nov 2004.

[5] 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.

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

[7] Pradeep Kayasnur, Jitendra Padhye, and Paramvir Bahl, "On the efficacy of separating control and data in 802.11," Tech. Rep. MSR-TR-2005-100, Microsoft Research, July 2005.

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

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

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

[11] 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.

[12] 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.

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

[14] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks ," in *Broadnets*, 2004.

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

[16] 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.

[17] 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.

[18] 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 *ISPAN*, 2000.

[19] Xue Yang, Nitin H. Vaidya, and Priya Ravichandran, "Split-channel pipelined packet scheduling for wireless networks," *IEEE Transactions on Mobile Computing*, To appear.

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

[21] Ashish Jain, Marco Gruteser, Mike Neufeld, and Dirk Grunwald, "Benefits of packet aggregation in ad-hoc wireless network," Tech. Rep. CU-CS-960-03, University of Colorado at Boulder, 2003.

[22] "KarlNet's TurboCell^{TM}: Enhancing the Capabilities of Standard 802.11 ," White paper, available at http://www.karlnet.com/Documents/DocumentsWhitePapers.htm.