Overcoming MAC Overheads Using Packet Size Dependent Channel Widths[†]

Technical Report (November 2010)

Vijay Raman, Fan Wu, and Nitin H. Vaidya
Dept. of ECE & Coordinated Science Lab
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.
Email: {vraman3, fwu24, nhv}@illinois.edu

Abstract—**In this paper, we propose to reduce the MAC overheads in random access protocols by partitioning the channel spectrum used into a narrow channel and a wide channel. The narrow channel is used for transmitting the short packets (approximately 100 bytes long) and the wide channel is used for transmitting the longer packets. We intend to use multiple radios, one each for the different channel partitions. Narrow width channels have a reduced capacity, which lowers the maximum transmission rate achievable on these channels. As a result, the channel wastage due to the rate-independent MAC overhead can be reduced. We propose a protocol called WiSP (channel *Wi*dth *S*election based on *P*acket size) to estimate the appropriate channel widths depending on the relative traffic load involving short and long packets in the network. We evaluate our protocol using extensive simulations and demonstrate its effectiveness in achieving higher throughputs. We propose our algorithm to complement frame aggregation (which is an existing approach for minimizing MAC overheads). We show that there are scenarios during which the frame aggregation can perform poorly, and show that our proposed algorithm can provide a good performance even in those situations.**

I. Introduction

The present day communication networks predominantly involve packets of smaller sizes. For instance, a 2008 study [1] showed that more than 55% of the packets in the internet are of sizes smaller than 100 bytes. This is not surprising given that many of the traffic such as, those generated by VoIP or the ACKs generated by TCP (used commonly by the HTML traffic) are small packets that are smaller than 100 bytes. Even though the transmission time associated with such short packets are small, the channel wastage due to bandwidth-independent overheads of the MAC protocol used is significant for these packets. The bandwidth-independent (or rate-independent) overhead is the channel time consumed independent of the transmission rate used for data packets.

In most of the present day wireless communication techniques that follow a random access scheme, the channel is first assessed to be free before a packet transmission to avoid collisions (e.g., DIFS in IEEE 802.11). If the channel is sensed to be busy, the nodes backoff until the channel becomes free again. The associated overhead due to the time spent in backoff or channel sensing are independent of the packet size or the transmission rate, and are hence bandwidth independent overheads. If, for instance, P_l (in bits) denotes the packet payload size, T (in seconds) denotes the channel time consumed by the rate-independent overhead associated with each transmission, and R (in bits per second) denotes the transmission rate, then $\frac{TR}{P_l+TR}$ fraction of channel capacity is wasted as the rate-independent overhead [2]. Observe that the channel wastage is higher when the packet payload size is small or when we use higher rates of transmission. The wastage in capacity becomes significant when short packets ($\sim$ 100 bytes) are queued in front of longer packets ($\sim$ 1000 bytes), as the long packets have to unnecessarily wait for the short packets whose significant portion of transmission time is spent on the overheads.

Current approaches for reducing the bandwidth independent overhead include frame aggregation [3], [4], where multiple MAC frames are combined into a single large frame and sent using a single transmission opportunity. While frame aggregation is in general effective for reducing the effect of the overheads, there are some

[†]This work was supported in part by NSF grant 06-27074 and in part by US Army Research Office grant W911NF-05-1-0246. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

situations when frame aggregation cannot be adopted. For instance, in the case of voice flows, the packets usually arrive at a low rate and aggregating the voice packets before sending out the combined packets will incur a delay. Moreover, because voice packets are typically only 100 bytes long, multiple voice packets may have to be combined to create a single MAC frame that is large enough to mitigate the effect of MAC overheads. Therefore, the voice packets may end up being delayed further before they are actually transmitted over the network, which may result in a poor voice quality at the receiver. Simply choosing to not aggregate the voice packets may once again result in expensive channel capacity to be wasted on the overheads. With the rapidly growing rates of VoIP calls in the internet, this would imply a significant wastage of capacity.

In this work, we propose to partition the channel into a narrow channel and a wide channel. The narrow channel is used for transmitting the short packets and the wide channel is used for transmitting the longer packets. We intend to use multiple radios, one each for the different channel partitions. Narrow width channels have a reduced capacity, which as a result lowers the maximum transmission rate achievable on these channels. As a result, the channel wastage in rate-independent overhead can be reduced. However, it is not straightforward as to how much bandwidth (we interchangeably use the term bandwidth to imply the width of the channel) to allocate for short packet transmissions. This is because, if a node predominantly transmits long packets with very little short packets, then the capacity lost for the long packets while partitioning the channel may cause a negative effect on their throughput. On the other hand, if the node generates more short packets than long packets, then the bandwidth allocated for the short packets, if not sufficient, may result in an eventual packet loss due to buffer overflow at the sender side. It is therefore important to determine the appropriate bandwidth required for each of the packet sizes. Furthermore, it is also important to understand when to partition the channel, depending on the amount of short and long packets generated in the network.

We propose a channel partition scheme called WiSP (channel **Wi**dth **S**election based on **P**acket size), where we use a simple heuristic to determine the channel partition widths. WiSP estimates the relative load of short and long packets in the network and calculates the channel partition widths accordingly. We are not proposing our approach as a replacement for frame aggregation, but to rather use it to complement and provide an alternate means for reducing the MAC overheads during situations when frame aggregation cannot be used.

We show that our proposed protocol achieves a better performance in terms of achieving higher network throughput when compared to a situation where we do not partition. We also compare the performance of our protocol with that of frame aggregation for scenarios where frame aggregation does not provide effective improvements, and show that our approach provides a significant performance in those scenarios.

II. Problem Motivation

In this section, we demonstrate the benefit of choosing variable-width channels based on packet sizes. First, we wish to understand the amount of capacity loss when higher rates are used for short packets. For this, we generate packets of various sizes ranging from 100 bytes to 1500 bytes and plot the capacity loss calculated at various fractions of bandwidths. If α is the fraction of bandwidth allocated for the packet transmission and $DIFS, SIFS$ represent the inter-frame spacing in IEEE 802.11 (chosen to be 34 μs and 16 μs respectively, considering a slot duration of $9\mu s$), the capacity loss, C_{loss} is calculated using the following formula,

$$C_{loss} = \frac{(DIFS + SIFS) * \alpha R}{P_l + (DIFS + SIFS) * \alpha R}$$

In this equation, R is the maximum rate of transmission, which at a bandwidth of 20 MHz (802.11 channel width) is 54 Mbps. We assume that the rate of a packet transmitted at α fraction of the bandwidth is also scaled by α. The C_{loss} values for the different packet sizes are shown in Figure 1. We observe from the plot that shorter packets experience higher capacity loss when they are transmitted at higher fractions of bandwidth than longer packets. In particular, we observe that for a 100 byte packet transmitted at the full bandwidth ($\alpha = 1$), the capacity loss is above 80%, whereas it is lower than 20% for a 1500 byte packet. We also observe that shorter packets experience fewer capacity loss when they are sent at narrower bandwidths. This suggests that choosing bandwidth based on packet sizes can lower capacity loss.

Next, we show that the percentage of channel partitioned for the short packets be proportional to the amount of short packets in the network. For this, we used ns-2 to simulated a 802.11a wireless link between two nodes and generated two constant bit rate UDP flows from one of the nodes to the other. One of the UDP flow generates 1000 byte packets at the rate 24 Mbps. The other UDP flow generates 100 byte packets. The packet

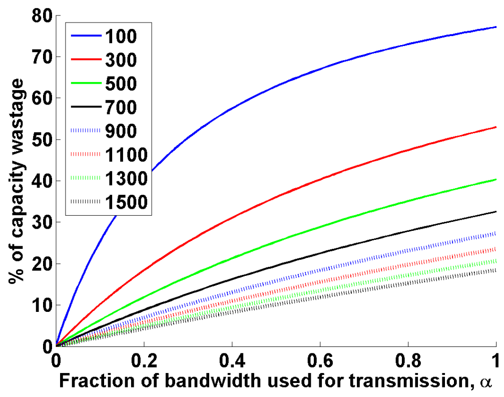

Fig. 1. Percentage of capacity loss as a function of fraction of bandwidth used.

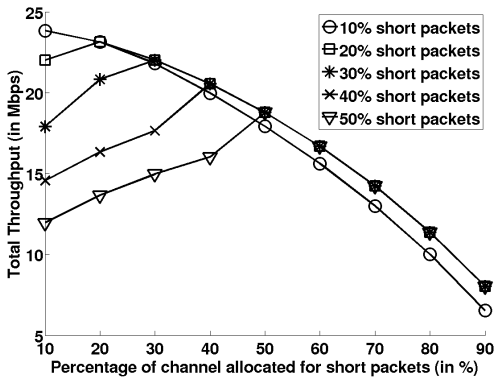

Fig. 2. Illustration on technique used for partitioning the channels.

generation rate of the 100 byte packets is varied so that the percentage of short packets in the network (calculated by dividing the packet generation rate of the 100 byte UDP flow by the total packet generation rate of both the UDP flows) is in the range of 10% up to 50% in steps of 10% (accordingly, the packet generation rates for the 100 byte packets are evaluated to be 2.5 Mbps, 6 Mbps, 10 Mbps, 16 Mbps, and 24 Mbps). In each case we varied the percentage of channel allocated to short packets from 10% to 50% and measured the combined throughput of the both the flows in each case, which is plotted in Figure 2. We first observe that the throughput values peak at the channel percentage value that is same as the percentage of short packets. Furthermore, we observe that the throughput falls if the percentage of channel allocated to short packets goes beyond the actual percentage of short packets in the network, as this will reduce the amount of channel allocated to the long packet flows. We use this motivation for developing our channel partitioning algorithm.

III. Network Model

We assume a single hop infrastructure network consisting of a set of static wireless clients controlled by an access point (AP). We consider a small to medium network consisting of 5 to 25 clients that are typical of a home or an office network. We assume that the available spectrum can be split into multiple sub-channels, each of varying widths. The center frequency of the sub-channel depends on the width of that channel. For all of our evaluations in this paper, we only consider situations where a channel is split into two sub-channels. Furthermore, we consider IEEE 802.11a channels and protocols. However, our algorithm is more generic and can be extended to any wireless technologies and for any number of sub-channels. Figure 3 shows an example where a 20 MHz channel is split in two possible ways, (a) two 10 MHz channels, and (b) a 5 MHz and a 15 MHz channel. Note that the center frequencies of the sub-channels change depending on their widths.

We assume that the clients and the AP are equipped with multiple radios. The wireless radios in a node are capable of transmitting over any one of the sub-channels at any instant of time, and are capable of switching across sub-channels. We assume that the sub-channels have sufficient guard band between them, so that the interference due to transmissions on adjacent channels is reduced.

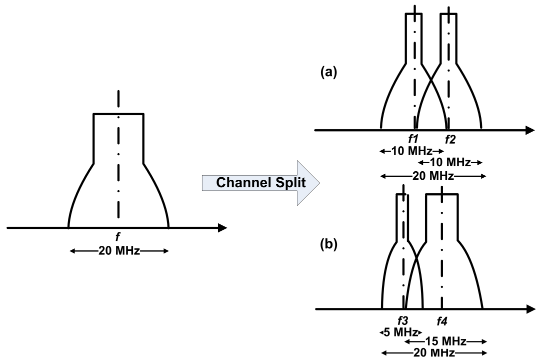

Fig. 3. An example where a 20 MHz channel is split into (a) two 10 MHz channels, and (b) a 5 MHz and a 15 MHz channel.

IV. The WiSP Protocol

The WiSP protocol is a centralized approach, where the bandwidth partition values are decided by the AP. Note that for successful communication between a pair of nodes, they have to be communicating on the same sub-channel (involving the same center frequency and channel width). Thus, the channel partitions chosen by an AP has to be used by all the N clients controlled by this AP. The AP chooses the percentage of channel for the short packets in increments of 10%, and the remaining channel is used for the long packets (after discounting for a guard band of W_{guard} in wither cases). We make the following assumptions in our algorithm:

a) Assumptions: 1) For simplicity of analysis, we restrict ourself to a single data rate. However, in practical scenarios an autorate algorithm can be used to determine the best rates for a given channel width.

2) An earlier work has shown that narrow channel have longer transmission range for a fixed power of transmission than wider channels [5]. In order to overcome the resulting asymmetry in links between the clients and the AP for the short and long packets, we scale the transmission power depending on the channel widths used. Thus, narrower channels transmit at a lower power than a wider channel. We also scale the carrier sense and received SNR thresholds accordingly.

3) Because all the clients use the same channel widths, they can carrier sense each other independently on each of the partitions.

4) We do not scale the slot size, DIFS, SIFS, and other system parameters. However, we scale the symbol time based on the channel width used. Scaling the symbol time will also affect the transmission rate used for data packets.

b) Algorithm: Our WiSP algorithm is as follows:

```
WISP: Channel Partitioning Algorithm:
Parameters: Total channel width - $W_{total}$
            Small packet Threshold - $P_{th}$
            Guard band - $W_{guard}$

// Algorithm executed at client $i$

1. // Initialize current width for short packets
2. $W_{curr} \leftarrow W_{total}$
1. // During each probing interval $T_{cl}$
2. // size($packet$) gives the size of packet in bits
3. if size($packet_i$) $\leq P_{th}${
4.  Send packets using $(W_{curr} - W_{guard})$ to AP
5.  $numShortBits_i += $ size($packet_i$)
6. }else{
7.  Send packets using $(W_{total} - W_{curr} - W_{guard})$ to AP
8.  $numLongBits_i += $ size($packet_i$)}

9. // At the end of interval $T_{cl}$
10. $\lambda_i = \frac{(numShortBits_i + numLongBits_i)}{T_{cl}}$
11. $\beta_i = \frac{numShortBits_i}{(numShortBits_i + numLongBits_i)}$
12. sendToAP($\lambda_i$, $\beta_i$)
13.  return
            ______________________
14. recvFromAP($w$)
15. if($W_{curr} \neq w$)
16.  $W_{curr} \leftarrow w$
17.  return
            ______________________

// Algorithm executed at the AP

18. //From each client $j$, receive $\lambda_j$ and $\beta_j$
19. recvFromClient($\lambda_j$, $\beta_j$)
20. // Calculate the network wide % of short packets
21. $\beta^* = \frac{\sum_i \beta_i \lambda_i}{\sum_i \lambda_i}$
22. // $\lceil x \rceil^{10}$ rounds $x$ to the nearest multiple of 10.
23. $w_{percent} = \lceil \beta^* \rceil^{10}$
24. sendToClient($w_{percent} * W_{total}$)
```

In the above algorithm, each of the segments (demarcated using a line) are executed at different instants of time. The algorithm starts by sending both the short and long packets on the same channel (using single radio) using the full channel width W_{total}. Every client i then estimates the arrival rate of packets at its side, λ_i and computes the percentage of short packets β_i using the formula, $\beta_i = \frac{\text{No. of packets of size} \leq P_{th}}{\text{Total no. of packets}}$, where P_{th} is a packet size threshold, such that packets smaller than P_{th} are considered short and are otherwise considered long. The clients then send the estimate of arrival rate and the percentage of short packets to the AP periodically every T_{cl} seconds. The AP, after receiving the arrival rate and β values from all the clients, calculates the aggregate percentage of short packets in the network, using $\beta^* = \frac{\sum_i \beta_i \lambda_i}{\sum_i \lambda_i}$. The AP then chooses the percentage of channel for the short packets to the closest multiple of 10 using β^*, and broadcasts the channel width to all the clients (using sendToClient()). Once the new channel widths are received by the clients (in recvFromAP()), they use they use the appropriate percentage of channels for the short and long packets.

Note that the AP estimates the bandwidth partitions based on the packets received from all the clients and all the flows. To enable new clients that may later join the AP's network, the AP sends the beacon packets (or neighbor advertisement packets) on the full bandwidth W_{total} along with the information on the current bandwidth partition. Thus the new client can start using the new partitions right away. The AP can then re-calculate the bandwidth partitions for the whole network based on the packets that the new client generates. If any of the channel queue is full at a client, then the client can choose to send any additional packets arriving at that channel queue through the other channel.

V. Performance Results

We divide the performance evaluation section in to two parts. In the fist part, we validate our WiSP algorithm to show that our algorithm correctly estimates the percentage of channel to allocated to short packets. We provide simulations results that cover a variety of scenarios for this purpose. Later, in the second part, we compare the performance of our algorithm to that of frame aggregation for scenarios where frame aggregation performs poorly. As we mentioned in Section I, the main purpose of our algorithm is to not to replace frame aggregation, but to complement it in scenarios where frame aggregation cannot be performed (or performs poorly). All of our simulations are performed using a IEEE 802.11a network and the packet transmission rate is fixed at 54 Mbps, and a guard band of 5% is used between the sub-channels while partitioning the channels. Furthermore, we use a threshold, P_{th} of 128 bytes to determine whether a packet is short or long.

A. Algorithm Validation

To validate our WiSP algorithm, we first repeated the simulation discussed is Section II using two UDP flows between a pair of nodes and used our WiSP algorithm to choose the best channel partitions for each of the percentage of short packets generated. We then plot in Figure 4, the total throughput obtained for the different percentages of the short packets. We also plot the maximum throughput obtained in Figure 2, which are labeled as 'Fixed Partition', as the partition values are fixed manually and are not chosen by WiSP, and the throughput obtained without using channel partitioning (using a single radio and a single channel for both short and long packets), which are labeled as 'No Partition'. The plot also shows the percentage improvement obtained using WiSP and the fixed partition scheme over the no partition scheme, and the values are displayed on top of the corresponding bars. We observe that the WiSP protocol achieves almost the same throughput as the maximum throughput obtained using fixed partition values. Furthermore, we observe that partitioning the channels improves the throughput performance significantly, and the percentage of improvement is higher for higher percentage of short packets in the network. We also found that the percentage of improvement starts to decrease as the percentage of short packets in the network is increased beyond 50% (we have not shown those plots here to avoid cluttering the figure). This is because, the benefit from partitioning the channel can be useful only when there are a significant mix of long and short packets in the network. When a network has predominatly short packets, then there are not much long packets that can benefit from the capacity saved by using channel partitioning.

We observe that the throughput achieved using the WiSP algorithm is slightly lower than the throughput achieved using a fixed partition. This is because, our WiSP algorithm initially does not partition the channels, as it has no estimate of the amount of short and long packets in the network. Later, as the clients start estimating the percentage of short packets and reporting them to AP, they start to use different sub-channels for the two packet sizes. The associated latency involved in estimating the percentage of short packets and getting the amount of channel form AP, therefore creates a throughput difference. There is also latency involved in estimating the throughput values when the percentage of short packets vary within a flow depending on the flow dynamics, such as in the case of TCP or variable bit rate UDP flows. Before proceeding to evaluate the performance of our protocol for these cases, we first show that our algorithm can correctly estimate the percentage of short packets even when they vary.

For this, we generated two UDP flows, as before between a client and the AP. One of the UDP flows generates a constant bit rate traffic of 24 Mbps consisting of 1000 byte packets. The other UDP flow generates 100 byte packets. However, the rate of this flow is varied at intervals of 15 seconds starting from 24 Mbps to 10 Mbps, and then to 3 Mbps before finally increased to 16 Mbps. We plot the actual percentage of short packets as evaluated using these rates, and that estimated by our algorithm in Figure 5. The interval at which the clients send the reports on percentage of packets is set to 5 seconds. We, therefore observe that except for a latency of 5 seconds, our algorithm correctly tracks the percentage of short packets.

Next, we further validate our protocol using TCP flows. For this, we considered a network where the

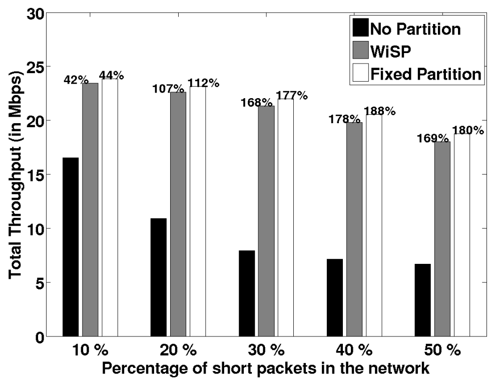

Fig. 4. Validation of WiSP algorithm.

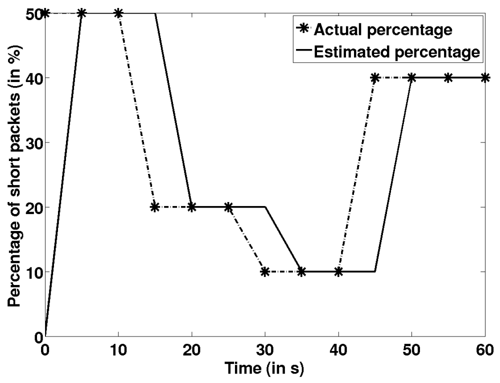

Fig. 5. Effectiveness of WiSP in estimating the percentage of short packets.

number of clients are varied from 5 to 25 in steps of 5. Each client generates a TCP flow towards the AP for 60 seconds; the TCP frame size is fixed at 1000 bytes, so that the TCP ACKs (which are 40 bytes long) are the only short packets in the network. We then plot the aggregate throughput obtained using WiSP algorithm, the throughput obtained using fixed partitions, and that obtained without partitioning the channel. For the case of fixed partitions, we observed that the channel partition at which the maximum throughput was achieved was different for different number of clients. We therefore, plot only the maximum throughput achieved across multiple partitions. The plots are shown in Figure 6. We once again observe that the WiSP algorithm achieves a throughput that is close to the maximum throughput achieved using the fixed partition case. Furthermore, we observe that, except for the 5 clients case, partitioning the channels consistently offers a throughput improvement of around 10 to 12%. This is because, in the case of TCP flows that we generated the performance improvement can be achieved only from the ACK packets, which are relatively fewer than the amount of data packets sent. The throughput improvement in the case of 5 clients may be high because of lower contention due to fewer clients in the newtork, which may have have resulted in more data packets and ACKs.

Finally, we wish to validate our protocol for the case of variable bit rate UDP flows. For this, we once again consider a network consisting of 5 to 25 clients. Each client generates a constant bit rate UDP flow at 24 Mbps rate consisting of 1000 byte packets, and a variable bit rate UDP flow consisting of 100 byte packets. The rates for the variable bit rate traffic is chosen randomly form the set {2.5 Mbps, 6 Mbps, 10 Mbps, 16 Mbps, 24 Mbps}. Furthermore, the rate of packet generation is varied every 15 seconds. The simulation time is set to 60 seconds. Figure 7 plots the throughput values averaged over 10 different runs of our simulation (where the rates for the variable bit rate flows are randomly chosen each time) obtained using WiSP, the maximum throughput obtained using the fixed partition algorithm, and the throughput when the channel is not partitioned. We observed that the percentage of channel at which the maximum throughput was achieved varied for each run of our simulation. However, in each case our WiSP algorithm correctly estimated the percentage of short packets, as we can observe from the plots. Furthermore, we observe that unlike the case of TCP, we achieve throughput improvement of at least 25% and up to 79% using our WiSP algorithm. This shows that a significant percent of channel capacity has been saved using our algorithm.

B. Comparison With Frame Aggregation

In Section I, we discussed an example scenario in the case of VoIP flows where frame aggregation cannot be used. We now provide throughput results for such a scenario both using frame aggregation and WiSP. For this, we considered a single client-AP network. The client is made to send constant bit rate traffic. One of the flows is set to send 1000 byte packets at a rate of 24 Mbps. We then generated a number of flows that send 100 byte packets at a rate of 1 Mbps. The 1 Mbps flows are intended to simulate a voice traffic. The number of 100 byte packet flows is varied from 1 to 5. In each case, we measure the throughput obtained without using frame aggregation or WiSP, the throughput obtained using WiSP, and that obtained using frame aggregation (we used the code shared with us by the authors of [6] for frame aggregation). Because the largest

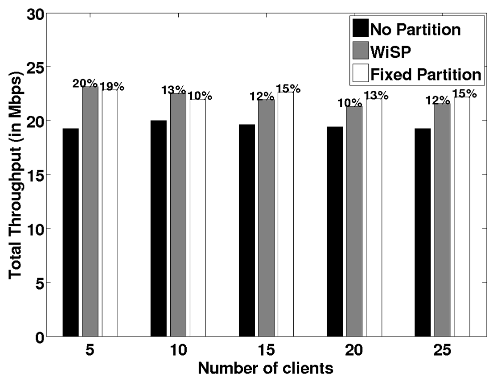

Fig. 6. Performance of WiSP for TCP flows.

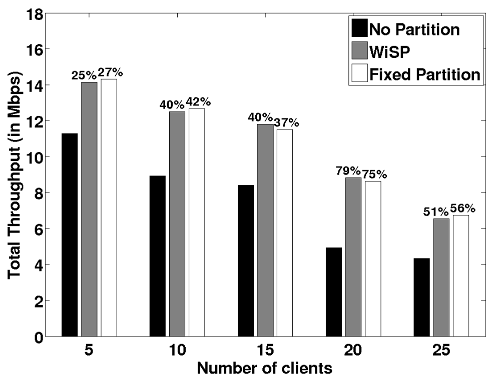

Fig. 7. Performance of WiSP for variable bit rate UDP flows.

packet size used in our simulations is 1000 bytes, we set the maximum frame size for frame aggregation also to be 1000 bytes to get a fair comparison. Figure 8 shows the corresponding results. We observe that frame aggregation does not achieve a throughput improvement that is higher than 11%. Whereas the WiSP algorithm achieves an improvement of up to 43% depending on the number of 100 byte flows in the network. This is because, the frame aggregation algorithm maintains a FIFO ordering on the interface queue from which the packets are combined. This FIFO ordering is maintained per destination, rather than per flow. Therefore, rather than benefiting from combining the 100 byte packets after buffering them for a while, the frame aggregation algorithm aggregates the packets as they arrive. Because most of the packets are 1000 byte long, a significant portion of the packets go un-aggregated.

Next, we wish to simulate a scenario where a user attempts to open multiple web sessions. Web pages are TCP connections where the associated HTTP packets are transferred within a few seconds. To emulate this scenario, we simulated five different TCP flows every 5 seconds, each lasting for just 5 seconds. We varied the number of clients in the network from 5 to 25 in steps of 5. Every client in the network is made to simulate the same number of TCP connections as explained. We then plot in Figure 9 the combined throughput of all the TCP connections across all clients for the case where no WiSP or frame aggregation is used, and for WiSP and frame aggregation scenarios. We observe that, once again, frame aggregation does not provide significant throughput improvements; the maximum percentage improvement is 5%. However, we observe that WiSP provides at least 11% and up to 45% improvement for the cases shown. The reason for the poor performance in the case of frame aggregation is becasue the TCP flows are short lived and therefore not much ACK packets are generated. Furthermore, we found that the AP does not aggregate ACKs belonging to different clients, eventually leading to a significant amount of ACK packets being not aggregated resulting in a bad performance.

The above two scenarios provide evidence that frame aggregation can perform poorly in certain scenarios, during which partitioning the channel using WiSP can be useful.

VI. Extensions to a Distributed Scenario

Our centralized algorithm can be easily modified to a distributed setting suitable for a multihop network. In the distributed case, the bandwidth partitions are estimated by the receive nodes, which for instance, may be the next hop node for a flow. However, the nodes that send the packets to a common next-hop node j, have to send their estimate of the arrival and transmission rates of the packets of only those flows that are sent through j. Thus, different flows in a node can be assigned different bandwidth values depending on the next-hop of a flow. The nodes, therefore, may have to switch across different bandwidth pairs for transmitting the packets belonging to the different flows. Furthermore, every hop of a given flow may be using a different bandwidth values. Note that a single flow targeted at a given next-hop node have to use two different bandwidths, one for the short packets in the flow and the other for the long packets in the flow. Thus, two radios are required for simultaneously transmitting the packets of a flow on the two bandwidths. Furthermore, two more radios are required per node for receiving packets sent by other nodes on two bandwidths. Thus, our distributed algorithm requires that every node is equipped with at least four radios.

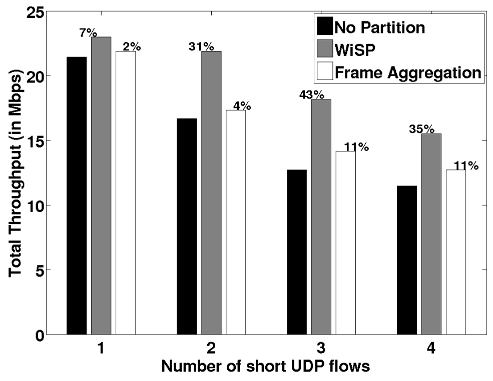

Fig. 8. Performance comparison between WiSP and frame aggregation for UDP flows.

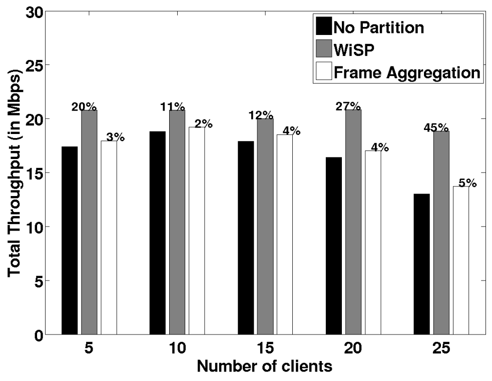

Fig. 9. Performance comparison between WiSP and frame aggregation for TCP flows.

Few interesting problems arise in the case of a distributed setting as summarized below:

c) Carrier sensing across different bandwidths:: The nodes in the case of a distributed setting may be using different bandwidths on the same channel spectrum. An important aspect, therefore, that need to be considered in a distributed setting is the means for carrier sensing transmissions on all the possible bandwidth pairs. One straightforward heuristic will be to carrier sense every possible bandwidth pair before initiating a transmission. The carrier sense thresholds, of course, have to be scaled according to the bandwidth, as mentioned in the centralized case. This mechanism, however, can be expensive due to the associated latencies. We wish to explore effective alternatives to this simple approach.

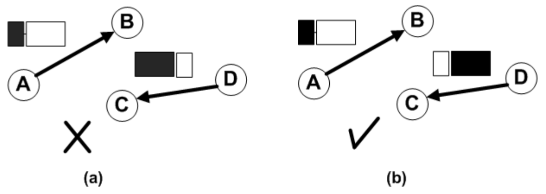

Fig. 10. An example to illustrate interference-aware bandwidth selection. The portion of bandwidth used by the transmissions is shaded black. A cross mark indicates that the transmissions cannot take place simultaneously and a check mark indicates that the two transmissions can be scheduled simultaneously.

d) Interference-aware bandwidth selection:: We illustrate this problem using the example in Figure 10. The figure (a) shows two transmissions on the same frequency spectrum, one from node A to B, and the other from node D to C. The transmissions, however use only the bandwidths that are shaded black. These two transmissions cannot take place simultaneously (indicated by a cross mark), as otherwise they will interfere with each other since their bandwidths overlap. If however, the bandwidth are chosen as in (b), then the two transmissions can be scheduled simultaneously without interfering with each other, as their bandwidths do not overlap. Thus, (b) can achieve a higher system throughput than (a). We wish to explore more on interference-aware channel width selection algorithms.

VII. Related Work

The bandwidth independent MAC overheads limit the maximum achievable throughput despite the various physical layer approaches used to improve the wireless network performance [7]. Frame aggregation, is a popular approach that is currently being used to address the bandwidth independent overhead problem [3]. In this section, we describe some of the approaches used to improve system throughput using frame aggregation.

The IEEE 802.11n standard proposes two approaches to frame aggregation, namely the MAC service data unit (MSDU) aggregation, and the MAC protocol data unit (MPDU) aggregation [8]. MSDU aggregation is the more efficient of the two aggregation methods, where the packets, belonging to the same destination, are aggregated into a single 802.11 frame with a common MAC header and checksum. This scheme is useful for aggregating multiple small user packets such as TCP ACKs or other control oriented data.

MPDU concatenates normal 802.11 MAC frames each having its own MAC header and checksum. Each of these subframes is separated by a MAC delimiter, which includes a length, checksum, and delimiter signature. The MAC delimiter allows a receiver to robustly separate each subframe, even in the case where some errors occur in the individual subframe. The MPDU approach is less efficient than MSDU because of the added overhead of the individual MAC headers of the constituent 802.11 frames. However, MPDU supports a block ACK scheme by which individual subframes are acknowledged separately, which allows the re-transmission of only those subframes in error. A restriction of both MSDU and MPDU is that all of the constituent frames must be of the same quality of service (QoS) level. It is not permitted to mix voice frames with best-effort frames, for example.

Several variants of the basic MSDU and MPDU scheme have been proposed in the literature. For instance, Skordoulis et al [4] proposed a two-level frame aggregation scheme that mixes the two aggregation methods. In the first stage, the MAC aggregates user packets from the upper layer into an MSDU with a MAC header and checksum. Then, a series of these MSDUs are concatenated into an MPDU with each MSDU separated by a MAC delimiter. This scheme increases the maximum aggregation size compared to using MSDUs and reduces MAC header overheads compared to using MPDUs. It allows the block ACK scheme to be applied to the MSDUs. Kim et al. [9] proposed a multi-layer scheme that provides aggregation at both the MAC and PHY layers. The MAC aggregates multiple MAC frames into an MPDU, and then the PHY aggregates a series of MPDUs into a single physical frame. Within the physical frame, an additional physical delimiter precedes each of the MPDUs. The physical delimiter contains modulation and coding scheme information for each MPDU, and thus allows each MPDU to be transmitted at a different rate. Unlike the other existing approaches, this scheme also supports multi-destination aggregation because each MPDU can be addressed to a different destination.

Sadeghi et al. [10] proposed the opportunistic autorate (OAR) method, which uses frame aggregation to take advantage of favorable channel conditions. When the underlaying rate adaptation algorithm shows that a frame can be sent at higher than base-rate, the MAC attempts to aggregate frames so that the time spent sending the frame at the higher rate equals the time to send a single frame at base-rate. This preserves the basic fairness capabilities of the 802.11 MAC while taking advantage of higher rates and the overhead reduction of frame aggregation. In [11], the authors propose a cross-layer approach for frame aggregation by which both broadcast and unicast packets can be aggregated into a single frame. The authors use this approach for combining ACK packets (which are considered to be broadcast frames as they do not require link level ACKs) with TCP data packets traveling in the opposite direction. In [12], the authors propose to use frame aggregation, not just to improve the TCP throughputs, but also to improve fairness and reduce the end-to-end delays in the network.

While frame aggregation can be though of as a time-based approach, where frames belonging to different time instants are aggregated, the bandwidth partition approach that we propose is a frequency-based approach. Our scheme can therefore be used to complement the frame aggregation scheme. Furthermore, our scheme can benefit from frame aggregation, as multiple short packets sent on the narrow channel can be combined to a single large frame and sent on the wide channel whenever the bandwidth allocation for the short packets is not sufficient. However, we do not exploit this possibility in our current approach.

VIII. Conclusion

In this work, we have proposed to partition a channel into a narrow and a wide sub-channel for overcoming MAC overheads. The narrow sub-channel is used for sending short packets and the wide channel is used for sending long packets. We have proposed a centralized algorithm for partitioning the channel and have discussed some interesting problems when extending this to a distributed algorithm. We have studied the performance of our algorithm using extensive simulations and show that our algorithm can provide significant improvements even in cases where frame aggregation performs poorly.

References

[1] (2009) Packet size distribution comparison between internet links in 1998 and 2008. CAIDA Research. [Online]. Available: http://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml.

[2] X. Yang and N. Vaidya. (2006, March) Spatial backoff contention resolution for wireless networks. [Online]. Available: http://www.crhc.uiuc.edu/wireless/groupPubs.html.

[3] Y. Kim, S. Choi, K. Jang, and H. Hwang, "Throughput enhancement of IEEE 802.11 WLAN via frame aggregation," in *IEEE VTC*, 2004.

[4] D. Skordoulis, Q. Ni, H.-H. Chen, A. Stephens, C. Liu, and A. Jamalipour, "IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput WLANs," *IEEE Wireless Communications*, 2008.

[5] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," *ACM Sigcomm*, August 2008.

[6] S. Frohn, S. Gubner, and C. Lindemann, "Analyzing the effective throughput in multi-hop IEEE 802.11n networks," in *IEEE HotMESH Workshop*, 2010.

[7] Y. Xiao, "Throughput in wireless LANs," *IEEE Wireless Communications*, vol. 12(6), pp. 82–91, December 2005.

[8] *HT MAC*, Enhanced Wireless Consortium Std., 2006.

[9] S. Kim, S. Choi, Y. Kim, and K. Jang, "MCCA: A high-throughput MAC strategy for next-generation WLANs," *IEEE Wireless Communications*, 2008.

[10] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, “Opportunistic media access for multirate ad hoc networks,” in *ACM MobiCom*, September 2002.

[11] W. Kim, H. Wright, and S. Nettles, “Improving the performance of multi-hop wireless networks using frame aggregation and broadcast for TCP ACKs,” in *ACM CoNEXT*, 2008.

[12] J. Karlsson, A. Kassler, and A. Brunstrom, “Impact of packet aggregation on TCP performance in wireless mesh networks,” in *IEEE HotMESH Workshop*, 2009.