TRAFFIC-AWARE CHANNEL ALLOCATION AND ROUTING IN MULTICHANNEL, MULTI-RADIO WIRELESS NETWORKS

BY

VIJAY RAMAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Nitin Vaidya, Chair
Assistant Professor Matthew Caesar
Assistant Professor Sayan Mitra
Professor Klara Nahrstedt
Professor David Nicol

ABSTRACT

Modern day wireless network applications exhibit varying service demands to satisfy user requirements, while differing in the nature of traffic they generate. Future wireless networks should, therefore, be capable of adapting to the heterogeneous traffic characteristics, by efficiently utilizing the expensive radio resources. In this dissertation, we concentrate on three important problems in existing wireless networks and propose algorithms for addressing them.

As the first problem, we focus on the effect of rate-independent MAC overheads in random access protocols such as carrier sensing, backoff, and fixed rate header transmissions. These overheads become prominent in short packets that are transmitted at high data rates. We address this problem by partitioning the transmission spectrum 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 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.

Next, we address the problem of channel switching delay in multichannel, multihop wireless networks. Future networks can benefit from using multiple channels simultaneously within a network. However, to maintain connectivity between various wireless nodes, the wireless radios may have to switch between channels. Due to both software and hardware restrictions, switching channels incur significant delay, which can be detrimental to many delay-sensitive, real-time applications, such as VoIP and interactive gaming. To address this, we propose SHORT (*S*tatic-*H*ybrid approach for r*O*uting *R*eal *T*ime applications), a joint channel allocation and routing algorithm for finding delay efficient routes for real-time applications without significantly affecting the performance of non-real time applications.

Finally, we explore the possibility of using variable width channels in a multihop wireless network for efficient spectrum utilization. We propose a variable width channel allocation scheme that adjusts the channel widths for the nodes during routing proportional to the rate requirement of the flows. The nodes also perform an admission control mechanism to determine if there is enough spectrum to satisfy the rate requirement.

To my parents, for their love and support

ACKNOWLEDGMENTS

The last five years at Illinois has been a great experience, where I got to meet several influential people. Many of them played a significant role in encouraging my work and assisting me whenever I felt low. I would like to use this space to thank all of them to my heart's content.

First and foremost, I would like to thank my adviser Prof. Nitin Vaidya. His guidance and assistance, all through my research, provided me with all the positive energy whenever I needed. His straightforward remarks and timely suggestions proved critical throughout my research.

I would next like to thank Prof. Matthew Caesar for all the time he spent with me for my class research project. His encouraging words and able guidance helped me to come up with interesting ideas that eventually led me to my Ph.D. research topic that is presented in the Chapter 5 of this dissertation.

Next, I would like to thank my dissertation committee members, Prof. David Nicol, Prof. Klara Nahrstedt, and Prof. Sayan Mitra for agreeing to be part of my final exam process. Their comments and suggestions helped me improve my research topic and project it in a perspective suitable for a wider audience.

I thank Prof. Romit Roy Choudhury for giving me a chance to explore a wider research domain. I appreciate his SyNRG group for helping me to think differently and gain a different attitude on research. Visiting Duke University truly played an important role in developing the breadth of my research.

I wish to thank my collaborators Prof. Fan Wu, Dr. Nikhil Singh, Brian Proulx, and Rakesh Kumar for sharing their thoughts and ideas that proved vital to our research goals. I would like to thank my previous lab-mates and visitors, Dr. Vartika Bhandari, Dr. Cheolgi Kim, Dr. Helen Xi, Dr. Simone Merlin, Dr. Chun-cheng Chen, Lu-Chuan Kung, Anthony Halley, Thomas

Shen, Nistha Tripathi, and Rishi Bhardwaj, for helping me without reservations whenever I needed their assistance. I would like to specially thank Dr. Vartika Bhandari and Dr. Cheolgi Kim for being my mentors whenever I needed their suggestions. I also thank Dr. Pradeep Kyasanur and Chandrakanth Chereddi for taking time in answering my questions on the Net-X testbed. Special thanks to my lab mates Ghazale Hosseinabadi, Guanfeng Liang, Tae Hyun Kim, and Shehla Saleem for all the fun times and for providing a pleasant work environment.

I thank the ECE Publications Office for going through my thesis and for suggesting changes to improve its readability. I thank the National Science Foundation (NSF) and the US Army Research Office for funding my research.

Friends make an important part of a Ph.D. as without them it can be insanely difficult to go through the gloomy days. I made quite a few friends here who were really successful at making my every single day. This space may not be sufficient to express all my gratitude to them. However, I would like to extend my humble thanks to my following friends, who I would like to rather call my family: G3, Kutti, Dunjan, Phi-man, RG, Dorky, Topa, Bumpkin Latte, JK, PG, Mugsu, JD, DPK, Mush-man, Photon, (Bed) Bug Fixer, and Meta.

Last but not the least, I thank my mom, dad, sister, and brother-in-law for their encouraging words and support throughout my academic life away from home.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Today's internet is dominated by heterogeneous traffic characterized by a variety of applications that include both real-time and non-real-time applications. Real-time applications include voice over IP (VoIP), video, and online gaming that are delay sensitive. Non-real time applications, on the other hand, are delay tolerant, such as those transmitted over transmission control protocol (TCP) that require reliable data delivery. Each of these kinds of traffic behaves differently with respect to the packet characteristics and wireless bandwidth requirements. For instance, VoIP traffic is characterized by short packets, of the order of few 100 bytes, that may be generated at a constant rate. Such low rate traffic requires only very little wireless bandwidth, but the packets need to be delivered to the destination within a time deadline as otherwise the receiver may not be able to reproduce the voice without much distortion. Video traffic consists of variably sized packets and demands a higher transmission bandwidth. Finally, the traffic due to online gaming may involve a lot of packets and have to be delivered quickly to the gaming server to minimize user frustration. Moreover, these packets, similar to those of video, can be of varying sizes.

The real-time applications discussed above have to co-exist in the same network with the non-real-time traffic, which exhibits different characteristics. For instance, TCP packets are typically fixed size packets generated at a varying rate depending on the bandwidth availability. These are typically best-effort traffic requiring reliable packet delivery, but are not time-constrained, unlike the real-time applications. Examples of non-real-time applications include e-mail, file transfer, and normal web browsing.

The relative heterogeneity in the traffic characteristics and the service requirement of the applications has motivated extensive research on algorithms for adapting the radio resources to suit the application needs. Because wireless is a shared medium, a significant portion of this research has been dedi-

cated to exploring the means for effectively sharing the spectrum among the contending users or applications. Many of the existing wireless standards, such as IEEE 802.11, IEEE 802.15.4, etc., provision multiple frequency channels, where a channel is a band of frequency spectrum. Realizing wireless networks using these standards allows for a natural way of sharing the spectrum amongst the users. By equipping the wireless nodes with multiple radio interfaces [1, 2, 3] and by utilizing appropriate routing, channel allocation, and scheduling strategies [3, 4, 5, 6], substantial performance benefits can be harnessed using these multichannel standards.

In this thesis, we explore variable-width channelization in multichannel wireless networks and focus on three important problems that are significant in this perspective. We briefly summarize each of these three works below:

Channelization to Reduce Protocol Overheads: In random access protocols every packet transmission is preceded by a considerable amount of time spent in assessing the channel to be free. The time spent in these overheads is independent of the actual length of the packet transmitted or the data rate used for transmitting the packet. These are, therefore, termed as bandwidth independent or rate independent overheads [7]. The ratio of the time spent on these overheads to the time spent on an actual packet transmission, suggests an inefficiency in the underlying protocols. This is because, while the time spent on the overhead can be justified when considerable time is spent on the packet transmission, it may seem unfair when the actual packet transmission itself lasts for just a fraction of the time spent on the overheads [7]. The inefficiency can be substantial when the packet payloads are small or the transmission data rates are large. The situation can be worse when a wider spectrum is allocated for packet transmissions, as wider channels can achieve larger data rates.

Present day communication networks predominantly involve packets of smaller sizes. For instance, a 2008 study [8] 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 packets, such as those generated by VoIP or the ACKs generated by TCP (used commonly by the HTML traffic), are small packets of the order of 100 bytes. To minimize the inefficiency, it makes sense to use a lower rate of transmission for shorter packets. However, simply reducing the rate of transmission may result in the short packets

occupying the channel for a longer time, which may be unfair for any long packets that are to be transmitted. Furthermore, the channel will not be used efficiently. In this work, we propose to instead partition a channel into a narrow channel, for sending short packets, and a wide channel, for sending long packets. We intend to use two wireless radios,[1] one each for the two channel partitions. Narrow channels have a reduced capacity, which lowers the maximum transmission rate. We provide protocols for dynamically deciding the appropriate percentage of bandwidth for the short packets.

Managing System Latencies Due to Channel Switching Delays: To ensure connectivity between multiple wireless nodes that are allocated different channels, the wireless interfaces must be allowed to switch across channels. However, due to software and hardware constraints, the delay involved in switching the channels can be significant. This may result in latencies in packet transmissions that can be prohibitive for delay sensitive applications, such as VoIP. We propose an algorithm that can adapt the underlying channel allocation mechanism dynamically based on the type of traffic routed through the network. We consider two popular channel allocation strategies followed in multichannel wireless networks, namely static channel approach (in which the channels for all the radios of a node are fixed), and hybrid channel approach (in which the channels for only a subset of radios are fixed a priori and those for the remaining radios are varied dynamically during communication). The hybrid channel approach is optimized for achieving higher throughputs, but has poor delay performance due to the associated dynamic channel switching. While there are no such latencies in a static channel allocation scheme, which results in a better delay performance, they do not have a good throughput performance. In this work, we propose a mechanism that exploits the benefits of a static channel approach for providing lower delay paths for real-time applications, while at the same time utilizing the flexibilities of a hybrid channel approach for providing higher throughputs for non-delay sensitive applications. Using actual implementations on a multichannel mesh testbed, we show that the end-to-end delays of real-time applications are significantly lower in our proposed approach when compared to a purely hybrid approach. Furthermore, we show that the

[1]The terms 'wireless radio' and 'wireless interface' are used interchangeably in this thesis and both mean a wireless network interface card.

throughput of non-delay sensitive applications is not degraded significantly as a result of our approach.

Variable Width Channelization Based on Traffic Demand: Most of the existing multichannel protocols propose to use fixed width channels. For instance, the channel width in the case of IEEE 802.11a is fixed at 20 MHz. These fixed width channels may either result in the spectrum being used inefficiently when there is not enough traffic to utilize the spectrum or in an insufficient spectrum when the traffic requires more than the available bandwidth. If the spectrum widths, instead are allowed to be variable, then the flows that require less spectrum can use a narrow spectrum, thereby allowing the remaining spectrum to be used by flows requiring more spectrum. While certain standards allow for variable channels widths, as in the case of IEEE 802.11n where channel widths of 40 MHz are allowed, there are no existing provisions to allow for dynamically varying the spectrum widths based on traffic needs.

In this work, we exploit the notion of channel width adaptation for tuning the available spectrum in a multihop network dynamically during route selection, based on the bandwidth requirement of the flows. We provide ideas on how the bandwidth can be adjusted per flow during route selection and study the performance of our protocols using simulations.

1.1 Outline

The dissertation is organized as follows:

1. In Chapter 2, we present the overall system architecture that provides a perspective on all the protocols proposed in this dissertation with respect to the popular open systems interconnection (OSI) model. We also provide the network model that is assumed by our algorithms.

2. In Chapter 3, we motivate the problem of partitioning a channel into sub-channels of varying widths based on packet sizes. We then provide simulation results for various network scenarios to elucidate the performance of our protocol.

3. In Chapter 4, we propose a mechanism for fixing the wireless radios on pre-defined channels to minimize the latency due to channel switching while routing real-time applications, such as VoIP. We also provide results obtained from a prototype implementation of our protocol on a multichannel testbed, and a set of simulation results to understand its limitations.

4. In Chapter 5, we discuss an algorithm for adapting the spectrum widths based on the traffic load, and propose protocols for adapting the channel widths during routing based on the requirement of the flows.

5. Finally, in Chapter 6, we summarize our work and conclude the thesis.

CHAPTER 2

PROTOCOL ARCHITECTURE AND SYSTEM MODEL

In this chapter we develop a framework to accommodate the three problems addressed in this thesis with an overall system perspective. The protocol-level details of each problem are addressed specifically in the corresponding chapters. We also develop the system model and the relevant background on multichannel protocol operation that are used by the subsequent chapters.

2.1 System Architecture

The system architecture is shown in Figure 2.1. The architecture follows the generic open systems interconnection (OSI) model along with the proposed protocol additions. Each additional layer is color coded to indicate the protocol interactions with the OSI model. As we can observe from the figure, all of our proposed algorithms are built at the networking and the medium access control (MAC) layer. However, they utilize the attributes from other layers for their operation. The interactions between the layers are indicated using colored arrows. The protocol layers along with their interactions are described as follows:

1. The green blocks correspond to the problem of channelization to overcome protocol overheads. This protocol requires an estimate of the mix of packet sizes generated by the application. Therefore, the corresponding protocol operating at the network layer utilizes the packet size information along with the rates at which they are generated from the upper layers. This is shown using a green arrow from the Application and Transport Layer block in the figure. The packet size information is later used to partition the channels appropriately at the MAC layer. This process of channelization based on packet sizes, however, has nothing to do with the route selection process. We therefore do not show a

green arrow pointing to the Route Selection block.

2. The blocks corresponding to the problem where the channel switching latencies are controlled are shown in red. Here, the protocols decide on a suitable channel allocation strategy depending on the traffic class of the application to be routed. This information can be obtained from the application layer. Furthermore, in addition to performing channel allocation, the channel switching process of the wireless radios on the entire route of the flow has to be controlled to minimize latency. Routing, naturally, becomes an important part of this problem.

3. Next, the blocks corresponding to traffic demand-based channel width selection are shown in blue. The corresponding protocols have to estimate the arrival rate of packets as generated from the upper layers. Furthermore, route selection is an integral part of the traffic load-based channel width selection problem. This is because, as will be discussed later in Chapter 5, the channel widths are determined based on whether a particular flow can be admitted at every single hop depending on the bandwidth requirement. Hence, we show blue arrows drawn to the Application and Transport Layer block and Route Selection blocks.

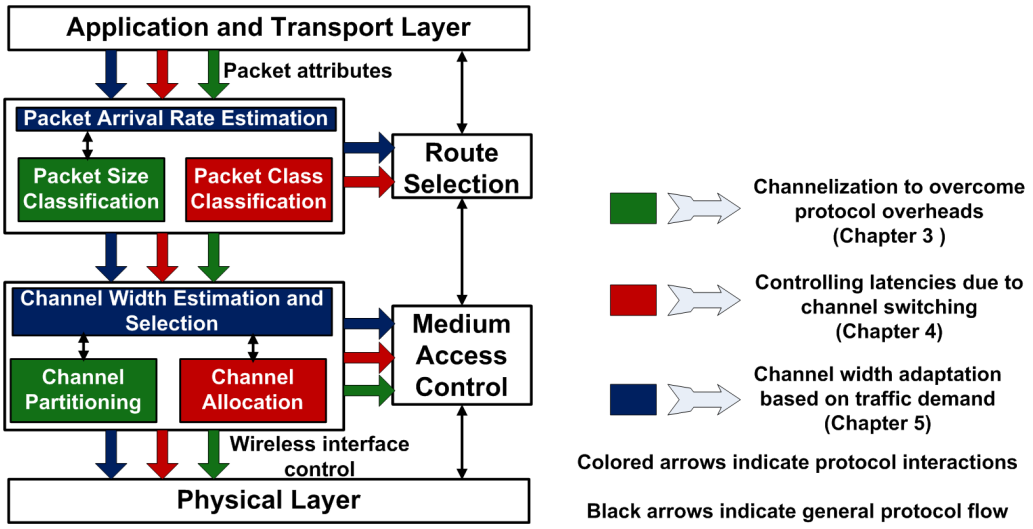


Figure 2.1: System Architecture.

All three problems addressed in this thesis deal with the MAC layer. This is because all three problems require means for adapting both the channels

allocated to a radio and their widths. The MAC layer should therefore be adapted to correspond to the current channel settings. The channel information should also be passed on to the wireless interface drivers to tune their RF chains and change their center frequencies appropriately. All of the newer protocol blocks, therefore, have their appropriately colored arrows directed to the respective blocks.

2.2 Network Model

We consider both a single-hop, infrastructure network and a multihop, ad-hoc network. In the case of an infrastructure network, a set of nodes are distributed around an access point (AP). In the case of a multihop, ad-hoc network, nodes are distributed evenly across a geographical area, and nodes communicate with each other using multiple single hop transmissions. (Two nodes are said to be one hop away if they are within the communication range of each other.) In both the infrastructure and multihop setting, all the nodes are assumed to be static. All the nodes and APs are equipped with two radios. All the radios are capable of either transmitting or receiving at any time, but not both simultaneously. Each of the radios within a wireless node is assigned to one of several possible channels (where the number of channels available depends on the wireless technology). The channels assigned to a wireless radio can be switched dynamically. In the infrastructure setting, the channels for the radios of an AP are determined using the algorithm in [9], and the radios of all the clients served by an AP are assigned the same channels as those of the AP. In the case of a multihop setting, the channels for the multiple radios are allocated and controlled by the multichannel protocol discussed in the next subsection.

2.3 Multichannel Protocol

The hybrid multichannel protocol (HMCP) proposed in [10] is used to ensure connectivity between nodes during multichannel operation (in Chapters 4 and 5). This is made possible by allowing the wireless interfaces on the wireless nodes to switch across channels as required. In this section, we

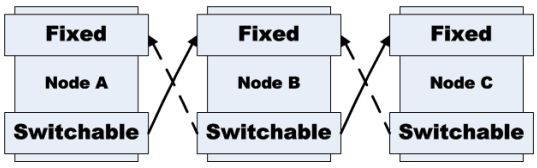


Figure 2.2: Example multi-channel protocol operation.

present a brief overview of the multichannel protocol.

As described earlier, all the wireless nodes in the network are equipped with two interfaces. Among the two wireless interfaces, one may switch across multiple channels whenever required, while the other remains fixed on a channel as long as the channel is perceived to be good. We call the interface that may switch often across channels the *switchable* interface and the interface that operates on a fixed channel the *fixed* interface. The fixed interface is used for data reception. However, data transmission can be from any of the two interfaces, fixed or switchable; this depends on the channel of the fixed interface on the neighboring node to which a multihop flow is directed. In general, if a neighboring node is operating on the same fixed channel as the current node, then the transmission can be through the fixed interface, else the switchable interface is used for transmission after switching its channel to the fixed channel of the neighboring node. Thus, a node can potentially transmit and receive simultaneously, if the channels on which it transmits and receives are different. Because the channel on which a switchable interface operates depends on the fixed channel of a neighboring node, it is clear that we need to allocate channels only to the fixed interface of a node. Figure 2.2 shows an example of our protocol operation, where the solid arrows indicate a data transmission from node A to node C, with node B as an intermediate node and the dotted arrows indicating a traffic from node to C to A via node B. Here nodes A and C switch their switchable interface to node B's fixed channel, while node B switches its switchable interface to the fixed channel of node C for forwarding packets from A, and that of node A for forwarding packets from node C.

To ensure connectivity between the nodes, every node should be aware of the channels on which their neighboring nodes are listening. This is made possible by the exchange of a broadcast `hello` message that contains the channel information. Every node periodically sends out a hello message on

all the channels so that all its neighbors that may be listening on any of the channels may receive the `hello` message. To help in channel management, the `hello` messages are propagated over two hops. This allows every node to be aware of the channel information of all the neighbors that are up to two hops away.

The routing mechanism used currently in the testbed is a modified AODV protocol [11]. The modifications to the original AODV protocol involve finding a channel-diverse route that avoids bottlenecks and reduces the expected transmission time. These modifications are incorporated into the route metric, called MCETT (multichannel expected transmission time) used by the routing protocol [10]. More details on the multichannel protocols can be found in [12] and [3].

While the channel allocation and routing algorithms used by HMCP provide a generic functionality, we modify these algorithms based on the requirements of our problem, as elaborated in the remaining chapters.

CHAPTER 3

OVERCOMING PROTOCOL OVERHEADS USING PACKET SIZE DEPENDENT CHANNEL WIDTHS

Present day communication networks have to carry traffic with different packet sizes. For instance, VoIP traffic and TCP acknowledgments (that are generated by web browsing and HTML traffic) are short packets with the VoIP packets typically being a few hundred bytes long, while the TCP acknowledgment (ACK) packets are 40 bytes long. File transfers and other TCP sessions, on the other hand, may involve data packets of the order of 1000 bytes [13]. An internet traffic analysis by CAIDA in 2008 shows that over 55% of the packets in the internet are smaller than 100 bytes and this statistic has not changed over the past ten years [8]. However, the traffic analysis also shows that the percentage of bits from large packets (> 1400 bytes) has grown by 15 - 20% over a ten year duration. This suggests that a variety of packet sizes will exist in the future communication networks. Even though the transmission time associated with the short packets is small, the channel waste due to bandwidth-independent overheads of the MAC protocol 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 many wireless random access schemes, 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 back off until the channel becomes free again. The associated overhead due to the time spent in backoff or channel sensing is independent of the packet size or the transmission rate, and is hence said to be rate/bandwidth independent. 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 [7]. Observe that the channel waste is higher when the packet

payload size is small or when we use higher rates of transmission.

Prior approaches for reducing the bandwidth independent overhead include frame aggregation [14, 15], where multiple MAC frames are combined into a larger 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 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. This may result in 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 usage of VoIP calls over the internet, this would imply a significant waste 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 lowers the maximum transmission rate achievable on these channels. As a result, the channel waste in rate-independent overhead can be reduced. However, an algorithm is needed to determine how much bandwidth to allocate for short packet transmissions. This is because, if a node predominantly transmits long packets with very few short packets, then the capacity lost for the long packets while partitioning the channel may have a negative effect on their throughput. In this case, it may be instead beneficial to send the long packets on both the channel partitions. The same may be true if the packets are predominantly short in the network.

To decide the appropriate channel partitions, we develop a protocol 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 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. Moreover, we show that we can achieve even more performance gains when we use our WiSP approach along with frame aggregation. Our results suggest that the WiSP protocol can complement the frame aggregation in reducing the MAC overheads in situations where just the frame aggregation cannot be used.

3.1 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 [16]. Frame aggregation is a popular approach that is currently being used to address the bandwidth independent overhead problem [14].

Sadeghi et al. [17] 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. In [18], 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 [19], 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 thought of as a time-based approach, where frames are aggregated across time, 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.

3.2 Problem Motivation

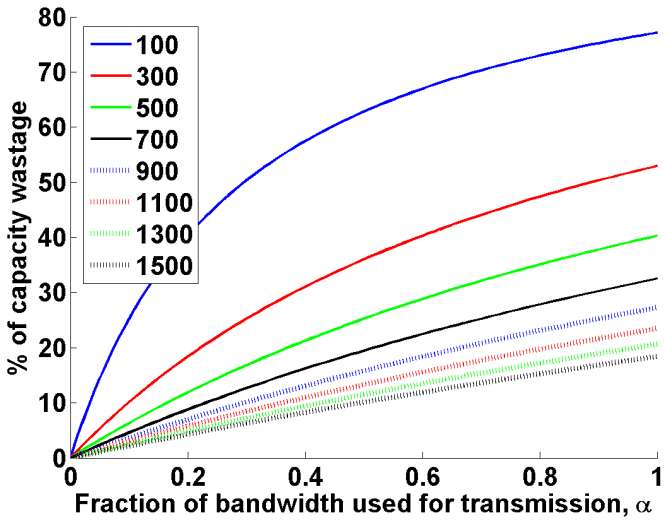


Figure 3.1: Percentage of capacity loss as a function of fraction of bandwidth used. All packet sizes are in bytes.

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 802.11a (chosen to be 34 μs and 16 μs respectively, considering a slot duration of 9μs), 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 (802.1la 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 3.1. 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 a lower capacity loss when they are sent at narrower bandwidths. This suggests that choosing bandwidth based on packet sizes can lower capacity loss.

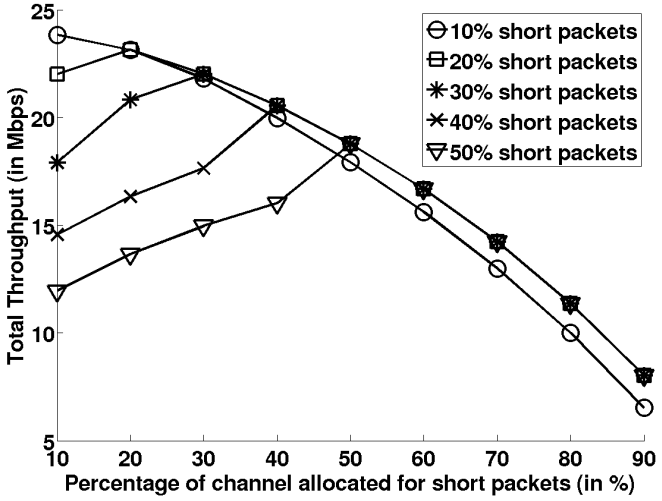


Figure 3.2: Illustration on technique used for partitioning the channels. All packet sizes are in bytes. Percentage values indicate the percentage of bits from short packets.

Next, we show that the percentage of channel allocated for the short packets should be proportional to the corresponding bytes of short packets in the network. For this, we used ns-2 to simulate an 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 flows generates 1000 byte packets at the rate 24 Mbps and is sent on one part of the channel. The other UDP flow generates 100 byte packets and is sent on the other part of the channel using a different radio. The data generation rate (in Mbps) of the 100 byte packets is varied so that the percentage of bits from short packets (PBSP) in the network is in the range of 10% up to 50% in steps of 10%, where the PBSP is defined as follows:

- **Percentage of bits from short packets (PBSP):** It is the number of bits from short packets over a certain period of time (in seconds)

divided by the total number of bits from all packets over the same period of time (in seconds).

We counted the number of bits from all the packets over a period of 3 seconds to estimate the PBSP. Accordingly, the data generation rates for the 100 byte packets are approximately 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. The throughput values are plotted in Figure 3.2. We first observe that the throughput values peak at the channel percentage value that is same as the PBSP value. 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.

3.3 Channel Partitioning Mechanism

We consider both an infrastructure mode network and a multihop, ad-hoc network setup. We assume that the available spectrum can be split into multiple sub-channels of varying widths. The center frequency of the sub-channel depends on the width of that channel. For all of our evaluations in this chapter, we only consider situations where a channel is split into two sub-channels. Figure 3.3 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 two 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.

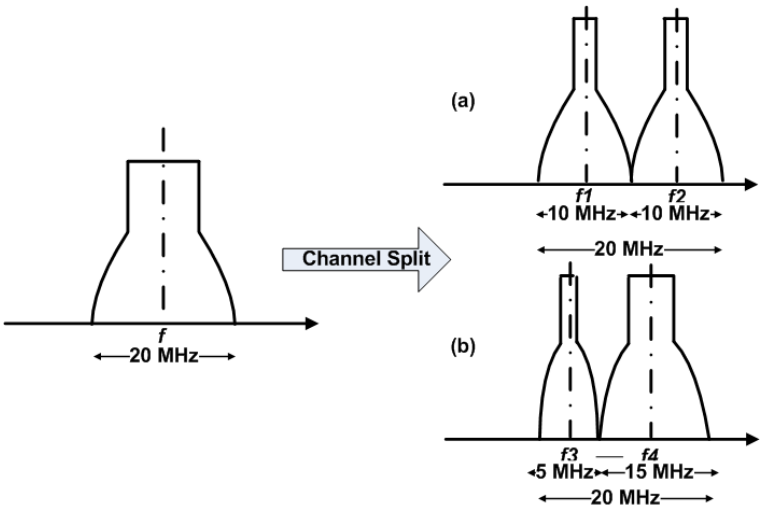


Figure 3.3: An example showing a 20 MHz channel split into (a) two 10 MHz channels, and (b) a 5 MHz and a 15 MHz channel.

3.4 Throughput Analysis

Before proceeding to discuss our algorithm, we wish to first provide an approximate analytical formulation of our problem. For our analysis, we assume that all the nodes follow the IEEE 802.11a DCF (distributed coordination function) mechanism for channel access, and use the throughput analysis technique in [20]. We assume the basic access scheme of 802.11 without the RTS/CTS mechanism. If $E[P]$ denotes the average duration of the packet payload, then the normalized system throughput S is given by (see Eq. 13 in [20])

$$S = \frac{P_s P_{tr} E[P]}{(1 - P_{tr})\sigma + P_{tr} P_s T_s + P_{tr}(1 - P_s) T_c} \tag{3.1}$$

where P_{tr} is the probability that there is at least one transmission in a slot and P_s is the probability of a successful transmission given that there is at least one transmission. Furthermore, σ is the duration of an empty slot time, T_s and T_c are the average time the channel is sensed busy because of a successful transmission and a collision, respectively, and their values for a basic DCF scheme (without RTS/CTS) are given by [20]

$$\begin{cases} T_s &= H + E[P] + SIFS + \delta + ACK + DIFS + \delta \\ T_c &= H + E[P^*] + DIFS + \delta \end{cases} \tag{3.2}$$

where H is the duration of the packet header, δ is the propagation delay, and $E[P^*]$ is the average duration of the longest packet payload involved in a collision. We now obtain expressions for the throughput achieved for a scheme in which different bandwidths are allocated for different packet sizes. For simplicity, we consider the case when there are only two packet size classes, depending on a threshold P_{th}. Any packets that are smaller than P_{th} are considered short packets and those that are larger than P_{th} are considered long packets. Let α denote the fraction of bandwidth allocated to the short packet class and $(1-\alpha)$ the fraction of bandwidth allocated to the long packet class. We assume that the transmission rate for each of the packet classes is also scaled by α or $(1-\alpha)$ as the case may be.

In Eq. (3.2), the interframe spaces ($SIFS$ and $DIFS$) and δ are bandwidth independent overheads. The remaining parameters have to be scaled appropriately depending on the bandwidth allocated and the transmission rates. Let $E[P_s]$ and $E[P_l]$ denote the average packet sizes of the short and long packets, respectively, and $E[P_s^*]$ and $E[P_l^*]$ denote the average duration of the longest packet payload that is involved in collision in each of the packet classes. Furthermore, let T_s^{small} and T_s^{long} denote the average time of successful transmission for short and long packets, respectively, and T_c^{small} and T_c^{long} denote their respective time the channel is sensed busy during a collision, which are obtained by appropriately scaling the packet duration, the header duration, and the ACK duration by α (for short packets) or $(1-\alpha)$ (for long packets) accordingly. The modified expressions are as follows:

$$\begin{cases} T_s^{small} &= (H + E[P_s] + ACK)/\alpha + SIFS + \delta + DIFS + \delta \\ T_c^{small} &= (H + E[P_s^*])/\alpha + DIFS + \delta \\ T_s^{long} &= (H + E[P_l] + ACK)/(1-\alpha) + SIFS + \delta + DIFS + \delta \\ T_c^{long} &= (H + E[P_l^*])/(1-\alpha) + DIFS + \delta \end{cases} \tag{3.3}$$

The normalized saturation throughputs of the short and long packets, S^{small} and S^{long}, are respectively given by

$$\begin{cases} S^{small} &= \frac{P_s P_{tr} E[P_s]}{(1-P_{tr})\sigma + P_{tr} P_s T_s^{small} + P_{tr}(1-P_s) T_c^{small}} \\ S^{long} &= \frac{P_s P_{tr} E[P_l]}{(1-P_{tr})\sigma + P_{tr} P_s T_s^{long} + P_{tr}(1-P_s) T_c^{long}} \end{cases} \tag{3.4}$$

Observe that the probability values in the above equations are assumed to remain unaffected due to bandwidth splitting as they are only dependent on bandwidth independent parameters and packet size [20]. (In effect, we assume similar contention on both channels.) The ratio of the throughputs of the short and long packets is given by

$$\frac{S^{small}}{S^{long}} = \frac{(P_s P_{tr} E[P_s])((1-P_{tr})\sigma + P_{tr}P_s T_s^{long} + P_{tr}(1-P_s)T_c^{long})}{(P_s P_{tr} E[P_l])((1-P_{tr})\sigma + P_{tr}P_s T_s^{small} + P_{tr}(1-P_s)T_c^{small})} \quad (3.5)$$

If we assume that the terms due to collision (containing the probability value $(1-P_s)$) and idle times (containing the probability value $(1-P_{tr})$) are negligible compared to the term involving successful transmissions (the middle terms in the numerator and the denominator), we get

$$\frac{S^{small}}{S^{long}} \approx \frac{E[P_s]}{E[P_l]} \times \frac{T_s^{long}}{T_s^{small}} \quad (3.6)$$

Using the expressions for T_s^{long} and T_s^{small} from Eq. (3.3) and neglecting (as an approximation) the terms other than the packet size, we can simplify Eq. 3.6 as

$$\frac{S^{small}}{S^{long}} \approx \frac{\alpha}{(1-\alpha)} \quad (3.7)$$

The objective of our algorithm is to make sure that both the short and long packets obtain a share of channel proportional to their corresponding load. When both the short and long packets are saturated and are transmitted independently of each other (by using two radios) over a single channel, we get:

$$\frac{S^{small}}{\beta} = \frac{S^{long}}{(1-\beta)} \quad (3.8)$$

where β denotes the fraction of bits from short packets and $(1-\beta)$ denotes the fraction of bits from long packets. Upon taking the ratio of throughputs in Eq. (3.8) and using Eq. (3.7), we have:

$$\frac{S^{small}}{S^{long}} = \frac{\beta}{(1-\beta)} = \frac{\alpha}{(1-\alpha)} \quad (3.9)$$

From Eq. (3.9), we have that $\alpha = \beta$. Because the value of α is derived

assuming that the collision and idle periods are negligible, its value may not be accurate. However, to compensate for this approximation, we allow the packets belonging to one of the partitions to be sent on the other partition whenever possible. Using this understanding, we develop our protocol that is discussed in the next section.

3.5 The WiSP Protocol

In the WiSP protocol, the bandwidth partition values are decided by the access point (AP) based on packets received from all its clients. The partition values are such that a certain percentage of channel is used for the short packets, and the remaining channel is used for the long packets (after discounting for a guard band of W_{guard}). The mechanism used for deciding the percentage of channel partitions will be discussed shortly.

Partitioning a channel into varying widths will affect the timing parameters of the 802.11 as observed in [21]. Accordingly, the maximum transmission rate achievable on each of the channel partitions will also be different, as the number of data bits per symbol will not change with the channel width [21]. For instance, the duration of an 802.11a OFDM symbol, which is 4 μs when transmitted over a 20 MHz channel, becomes 40 μs when only 10% of the channel (2 MHz) is used. Accordingly, the maximum data rate of 54 Mbps achieved using a 64-QAM modulation using a 3/4 coding rate on a 20 MHz channel, will be reduced to 5.4 Mbps on the 2 MHz channel with same 216 data bits per symbol in the both the cases. The data rate within each channel partition can be adapted between a minimum and a maximum rate automatically based on the channel conditions. One such algorithm has been proposed by the authors of [21]. We do not scale the slot size, DIFS, SIFS, and other system parameters as these values are bandwidth independent.

The authors of [21] also observed that narrower channels have a longer transmission range than that of a wider channel. One of the key observations in this regard is that for a given total transmit power, the wireless radios can transmit at a higher power per unit Hz on a narrow channel. Observe that this effect may result in different length transmission links for each of the channel partitions. This is not desired for our system as this may cause a link asymmetry between the clients and the AP for the short and

long packets. For example, the short packets may end up contending over a larger area than the long packets. In order to overcome this asymmetry, we scale the transmission power by the same factor as the fraction of channel allocated. Thus, narrower channels transmit at a lower power than wider channels to provide the same SNR at the receiver. We also scale the carrier sense thresholds of the wireless radios accordingly.

Because the AP decides the channel partition for all the clients in the network, all the clients use the same channel widths. As a result, the clients can carrier sense each other independently on each of the partitions. Specifically, a client sending short (long) packets carrier senses only that sub-channel used for sending the short (long) packets. We now proceed to our algorithm description.

Algorithm The channel partitions are decided by the AP based on the overall knowledge of the packet size mix in the network. The WiSP protocol achieves this by letting the clients individually estimate the overall incoming arrival rate of the packets from the application, in addition to the percentage of bits that are from packets smaller than a certain threshold, P_{th} (in bytes). The clients then periodically transmit the arrival rate estimate and the PBSP to the AP. An alternative approach will be for the AP to estimate the PBSP based on its local packet receptions. However, this estimate may not be accurate as some of the packets may be lost due to collisions, and a few others may be lost due to buffer overflows at the clients. These lost packets will not be accounted in the AP's estimate.

The PBSP along with the overall arrival rate of packets at each client enables the AP to estimate the individual arrival rates of short and long packets at each client. Ideally, the AP can use this information to propose individual channel partitions to each client proportionate to their packet mixes. This may, for instance, result in a scenario where a client sending only long packets or only short packets will be transmitting on a whole un-partitioned channel, while those that send an equal mix of both the packet sizes will be using a half of the channel for each packet sizes. While this scenario can intuitively provide a significant benefit with respect to minimizing the overheads, implementing this protocol in reality may be hard. This is because, 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), to establish proper frequency lock and synchronization in the RF hardware. In order to achieve this, the AP has to timeshare across clients, each time using a different channel partition for the short and long packets. This approach may require stringent time synchronization across nodes. Furthermore, when each client uses a different channel partition, there can be additional difficulties with respect to carrier sensing.

The node responsible for estimating the partitions starts by estimating the network-wide PBSP using $\beta^* = \frac{\sum_i \beta_i \lambda_i}{\sum_i \lambda_i}$ x 100, where β_i and λ_i (in bps) are the fraction of bits from short packets and the overall arrival rate of packets in bits per second, respectively, at client i. Thus, for instance, if at a client A the packets arrive at a rate of 40 bps of which 20% of the bits are from short packets, and at another client B packets arrive at a rate of 60 bps of which 30% of the bits are from short packets, then β^* can be evaluated to be $\frac{(0.2\text{X}40)+(0.3\text{X}60)}{(40+60)}\text{x}100 = 26$. The node evaluates and communicates the percentage of channel to be used for short and long packets based on β^*. Intuitively, our mechanism recommends that the percentage of channel chosen be proportional to the actual arrival rate of packets.

The pseudocode of our algorithm is shown in Algorithm 1.

In this algorithm, each of the segments (demarcated by a line) is executed at different instants of time. The algorithm starts by sending both the short and long packets on the same channel (using a single radio) using the full channel width W_{total}. Every client i then estimates the arrival rate of packets on its uplink flows, λ_i and computes the fraction of bits from short packets, β_i using the formula,

$$\beta_i \ = \ \frac{\text{No. of bits from packets of size less than or equal to } P_{th}}{\text{Total no. of bits from all packets}},$$

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 (λ_i) and the fraction of bits from short packets (β_i) to the AP periodically every T_{cl} seconds. The AP, after receiving the λ_i and β_i values from all the clients, calculates the aggregate PBSP in the

Algorithm 1 WiSP: Channel Partitioning Algorithm

Parameters: Total channel width - W_{total}
Small packet Threshold - P_{th}
Guard band - W_{guard}

Algorithm executed at client i

Initialization

a. $W_{curr} \leftarrow W_{total}$

Steps repeated every T_{cl} seconds by each client

1. Start interval T_{cl}
2. if $W_{curr} = W_{total}$ OR $W_{curr} = 0$
3. Send packets using W_{total} to AP
4. Go to Line 12.
5. // `size`($packet$) gives the size of packet in bits
6. if `size`($packet_i$) $\leq P_{th}${
7. Send packets using ($W_{curr} - W_{guard}$) to AP
8. $numShortBits_i +=$ `size`($packet_i$)
9. } else {
10. Send packets using ($W_{total} - W_{curr} - W_{guard}$) to AP
11. $numLongBits_i +=$ `size`($packet_i$)}
12. End interval T_{cl}
13. $\lambda_i = \frac{(numShortBits_i + numLongBits_i)}{T_{cl}}$
14. $\beta_i = \frac{numShortBits_i}{(numShortBits_i + numLongBits_i)}$
15. `sendToAP`(λ_i, β_i)
16. return

Steps performed when receiving the channel partitions from the AP

17. `recvFromAP`(w)
18. if($W_{curr} \neq w$)
19. $W_{curr} \leftarrow w$
20. return

Algorithm executed at the AP

1. Repeat for each client j,
2. `recvFromClient`(λ_j, β_j) //receive λ_j and β_j
3. // Calculate the network wide % of short packets
4. $\beta^* = \frac{\sum_i \beta_i \lambda_i}{\sum_i \lambda_i}$ x $100, i \in$ set of clients
5. $w_{percent}$ = smallest multiple of integer $k \geq \beta^*$
6. `sendToClient`($w_{percent} * W_{total}$)

network, using $\beta^* = \frac{\sum_i \beta_i \lambda_i}{\sum_i \lambda_i}$ x 100. The AP then chooses the percentage of channel for the short packets to be the smallest multiple of an integer k that is greater than or equal to β^* (we use $k = 5$ in our simulations), and broadcasts the channel width to all the clients (using `sendToClient`()). Thus, for the previous example where we evaluated β^* to be 26%, the percentage of channel for short packets will be chosen as 30% for $k = 5$. Once the new channel widths are received by the clients (in `recvFromAP`()), they use the appropriate percentage of channels for the short and long packets. The percentage of channel partition for the short packets is chosen at the granularity of 5% to simplify our evaluation. However, in reality this will depend on the capability of the wireless hardware and the driver.

Observe that when all the packets in the network are short, then W_{curr} will be correctly estimated to be 100%, in which case the nodes do not partition the channel. The same will be the case when all the packets are long in the network (see Line 2). 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 one of the channel queues is full at a client, then the client can choose to send any additional packets arriving at that channel queue through the other channel. Finally, the AP also includes the estimate of short and long packets in any downlink flows while evaluating the channel partitions. For this, the AP estimates the total number of bits of downlink packets to be sent and finds the percentage of those bits that are from short packets.

3.6 Performance Results

As the first part of our performance evaluation, we validate our WiSP algorithm to show that the algorithm quite accurately estimates the percentage of channel to allocate to short packets. We provide simulation 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 1, the main purpose of our algorithm is 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 an IEEE 802.11a access protocol, and unless specified otherwise, the packet transmission rate for 100% bandwidth is fixed at 54 Mbps to provide a fair evaluation of our protocol. However, our protocol does not restrict the use of any rate control algorithms, like the one proposed in [21] within each channel width. We use a guard band of 5% between the sub-channels while partitioning the channels. We also tried simulating other guard band sizes, and we found that the results do not vary significantly. We use a packet size threshold P_{th} of 128 bytes to determine whether a packet is short or long in all our simulations. This choice is motivated by the study in [8], where the authors have identified that most of the packets in the internet are either of the order of 100 bytes or 1000 bytes. Along with the transport protocol headers, 128 bytes seemed to be a good option for discriminating a short packet from a long packet.

3.6.1 Algorithm Validation

To validate our WiSP algorithm, we first simulated two wireless nodes and generated two constant bit rate UDP flows from one of the nodes to the other. One UDP flow generates 1000 byte packets at the rate 54 Mbps. The other UDP flow generates 100 byte packets. The data generation rate of the 100 byte packets is varied so that the PBSP in the network (calculated by dividing the bit rate of packets generated in the 100 byte UDP flow by the total bit rate of packets from both the UDP flows) is approximately in the range of 10% up to 50% in steps of 10% (accordingly, the data generation rates for the 100 byte packets are evaluated to be 6 Mbps, 14 Mbps, 23 Mbps, 36 Mbps, and 54 Mbps). Furthermore, to vary the PBSP in the network from 60% to 90%, we set the rate at which the short packets (100 bytes) are generated to be 54 Mbps and varied the rate of long packets to be 36 Mbps, 23 Mbps, 14 Mbps, and 6 Mbps. In each case, we measured the combined throughput of the two flows for three cases, namely:

- **No Partition**: In this case, we use only one channel (without partitioning) that is shared by both the short and long packet flows. We use a single radio for transmitting both the flows.

- **Fixed Partition**: For this scenario, we choose the best partition of channels manually. We vary the percentage from 10% to 90% in steps of 10 and choose the partition that has the maximum throughput. We then record the maximum throughput obtained across all the percentage values.

- **WiSP**: We use our WiSP protocol to estimate the percentage of channel to be allocated for the short and long packet flows.

In the following set of simulations, we do not apply the optimization for WiSP suggested in Section 3.4, where we send the long packets in the partition meant for the short packets (and vice-versa). We have used this optimization in our simulations later in this chapter starting from Section 3.6.2.

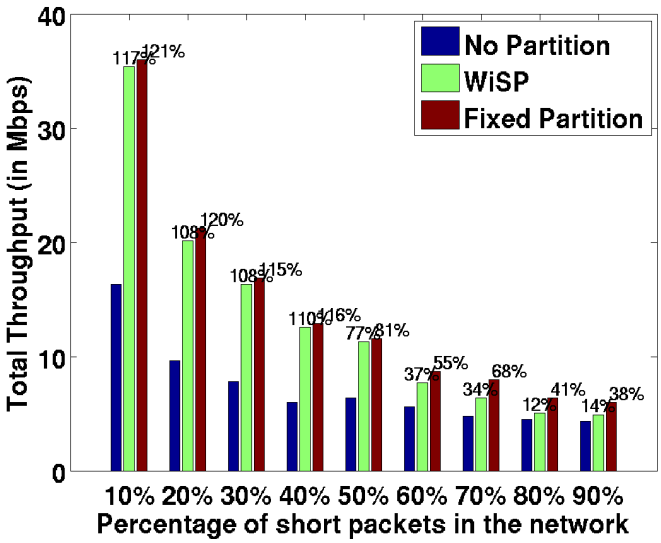


Figure 3.4: Validation of WiSP algorithm.

We plot the throughput values for the three cases listed above in Figure 3.4. We observe from the plot 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. However, the percentage of improvement decreases as the PBSP in the network is increased. This is because the benefit from partitioning the channel can be useful only when there is a balanced mix of long and short packets in the network. When a network has predominantly short packets (greater than 50%), there are not many 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 we do not use an optimal split of channels due to the use of a heuristic. Moreover, 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 PBSP and reporting them to AP, they start to use different sub-channels for the two packet sizes. The associated latency involved in estimating the PBSP and getting the amount of channel from AP, therefore, creates a throughput difference.

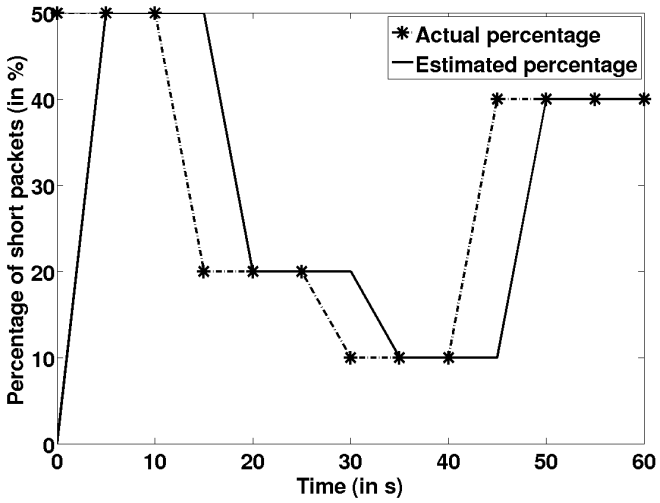


Figure 3.5: Effectiveness of WiSP in estimating the PBSP.

Next, we show that our algorithm can correctly estimate the PBSP 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 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 PBSP as evaluated using these rates, and that estimated by our algorithm in Figure 3.5. The interval at which the clients send the reports on PBSP is set to 5 seconds. We therefore observe that, except for a latency of 5 seconds, our algorithm correctly tracks the PBSP.

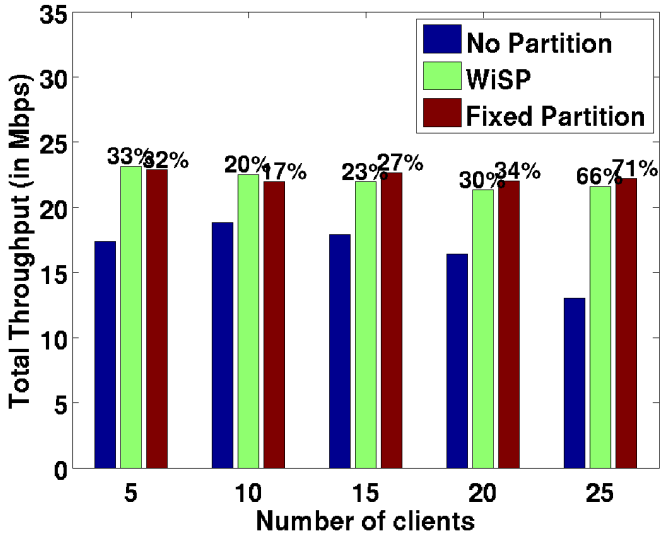


Figure 3.6: Performance of WiSP for TCP flows.

Next, we further validate our protocol using TCP flows. For this, we consider a network where the number of clients is 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 for the three cases: No Partition, Fixed Partition, and WiSP. The plots are shown in Figure 3.6. 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 at least 20%. We also observe that the throughput improvement using our scheme becomes significant as the number of TCP flows in the network increases. This is mainly due to two factors: (a) decreased throughput for the No Partition case due to increased contention (as the number of clients increase), and (b) more ACKs

in the network as the number of TCP flows increase, which can benefit by the channel partitioning.

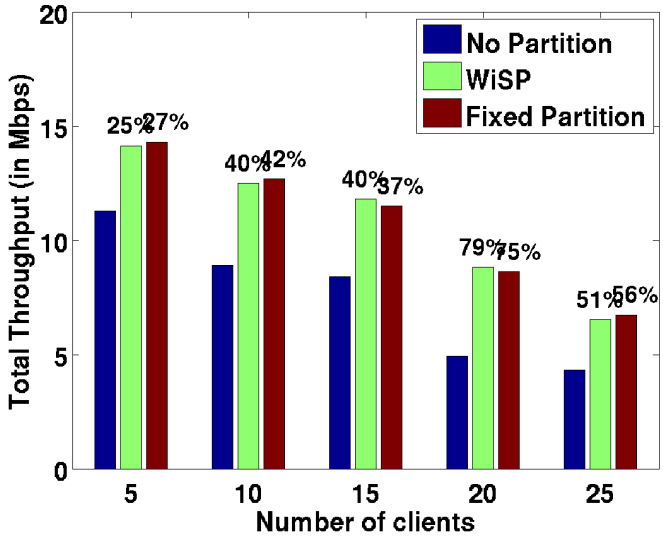


Figure 3.7: Performance of WiSP for UDP flows with varying packet rates.

Finally, we validate our protocol for the case of UDP flows with varying rates of packets generated. 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 another UDP flow consisting of 100 byte packets. The rate for the 100 byte traffic is chosen uniformly at random from 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 3.7 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 PBSP, as we can observe from the plots. Furthermore, we observe that we achieve throughput improvement of at least 25% and up to 79% using our WiSP algorithm. This shows that a significant percentage of channel capacity has been saved using our algorithm.

3.6.2 Comparison with Frame Aggregation

We now provide throughput results for a scenario where frame aggregation (FA) cannot be used effectively for reducing the impact of bandwidth independent overheads. The IEEE 802.11n standard proposes two approaches for frame aggregation, namely the MAC service data unit aggregation (A-MSDU), and the MAC protocol data unit aggregation (A-MPDU) [22]. Each of these schemes is explained below:

- **A-MSDU**: In this algorithm, the packets belonging to the same destination are aggregated into a single 802.11 frame with a common MAC header and checksum. This scheme requires that all the aggregated packets are of the same service type. The maximum size of an aggregated frame is restricted to 7935 bytes (which is 256 bytes less than the maximum PHY layer frame size of 8191 bytes). The packets are aggregated until either the maximum frame size is reached or the time since the first packet in the aggregated frame is within an allowable delay. Because an aggregated MSDU has a common MAC header, a packet failure will require the whole aggregated MSDU to be re-transmitted.

- **A-MPDU**: This scheme concatenates multiple 802.11 MAC frames each having its own MAC header and checksum. 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 sub-frames are acknowledged separately, which allows the re-transmission of only those sub-frames in error. The maximum aggregated frame size in this scheme is 64 kilobytes. Unlike the A-MSDU scheme, packets are not delayed during aggregation, but instead the algorithm simply aggregates whatever packets are present in the interface queue limited by the maximum frame size.

We compare our WiSP algorithm with both the variants of the frame aggregation approach. However, because the A-MPDU scheme incorporates a block ACK scheme that allows for selective re-transmission of erroneous frames, it is a preferred approach to A-MSDU. Additionally, we also provide results for WiSP used along with frame aggregation (henceforth termed as FA+WiSP), which is described below:

- **FA+WiSP**: In this scheme, we perform frame aggregation to aggregate multiple MAC frames whenever possible. However, we additionally run the WiSP algorithm to partition the channel so that the effect of overheads on any unaggregated short packets can be reduced. The flavor of frame aggregation used depends on the scenario discussed in the subsequent sub-sections.

For the case of WiSP and FA+WiSP, in the subsequent set of simulations, we also allow for the long packets to be sent in the channel partition meant for the short packets (and vice-versa), whenever the partition is empty.

Scenario I - Comparison with A-MSDU scheme

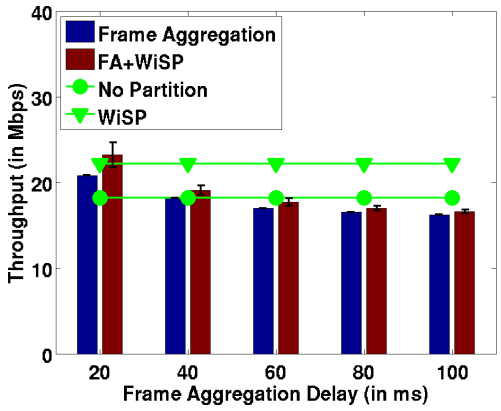


Figure 3.8: Throughput comparison for TCP flows across various A-MSDU aggregation delays.

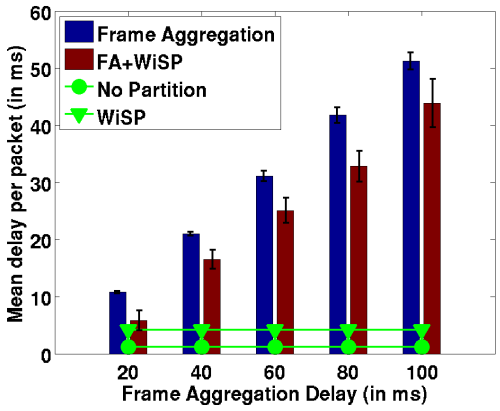


Figure 3.9: Mean packet delay comparison for TCP flows across various A-MSDU aggregation delays.

We first compare our WiSP algorithm with the A-MSDU frame aggregation scheme. For this, we generate a network consisting of a single AP. The number of clients connected to the AP is chosen uniformly at random between 5 and 15. The time delay for aggregating the MSDU frames is varied from 20 ms to 100 ms, in steps of 20 ms. First, we generate a TCP flow from each client to the AP. We then measure the combined throughput and the mean delay per packet for the flows from all the clients. We repeat the simulation 10 times, each time choosing a different number of clients and their positions around the AP. The throughput and the delay per packet averaged over 10 runs are plotted across the frame aggregation time delays

in Figures 3.8 and 3.9, respectively. The throughput and packet delay values for WiSP and No Partition are not dependent on the frame aggregation delays and are hence plotted as a straight line across all the X-axis values. We observe from Figure 3.8 that WiSP has a higher throughput than No Partition. The improvement mainly comes from the fact that in the case of WiSP, the TCP ACK packets are sent on a narrower channel than the one used for sending the data packets. This helps in overcoming the bandwidth-independent overheads for the ACK packets. We also observe that except for the 20 ms case, FA+WiSP has a throughput performance that is almost same as the frame aggregation approach. This is because, when the packets are delayed for aggregation, most of the packets become larger than the packet size threshold of 128 bytes. This results in the channel not being partitioned at all, which is similar to that of the frame aggregation scenario. In the case of 20 ms, the packets are not delayed significantly to result in such a scenario. Furthermore, both FA+WiSP and frame aggregation perform comparably to or worse than the No Partition case. This is due to the fact that data and ACK packets are delayed for aggregation, bringing down the throughput. This is evident from the mean packet delay plot in Figure 3.9. This plot shows that packets suffer a significantly higher delay on average when frame aggregation is used when compared to the cases where it is not used (i.e., No Partition and WiSP). Additionally, the mean delay per packet goes up as the frame aggregation delay increases. However, the packets sent using the FA+WiSP scheme suffer a lower delay than the frame aggregation approach as the ACK packets are aggregated and sent on a different channel than the data packets in the case of FA+WiSP, resulting in lower delays due to contention and collision.

Next, using the same network setup as the TCP simulations, we generate a 40 kbps UDP flow consisting of 100 byte packets from each client. This traffic is intended to simulate VoIP traffic in the network. Because we do not generate any long packets, we partition the channel into two halves and use them both simultaneously for the WiSP scenario. The throughput and mean packet delay, averaged over 10 runs as before, are plotted across the frame aggregation delays in Figures 3.10 and 3.11, respectively. In this case the throughput for the WiSP case is only slightly higher than the No Partition case as the benefit from using two channels is not significant owing to the lower rate of traffic generated. Unlike the previous scenario involving TCP

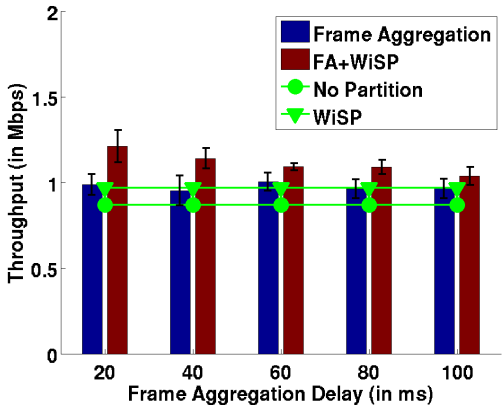


Figure 3.10: Throughput comparison for UDP flows across various A-MSDU aggregation delays.

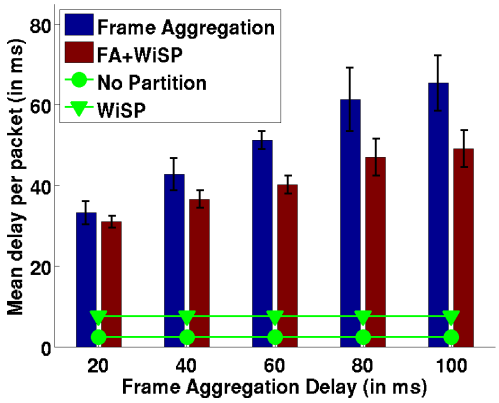


Figure 3.11: Mean packet delay comparison for UDP flows across various A-MSDU aggregation delays.

packets, frame aggregation performs better than the No Partition case as multiple packets are delivered in a single transmission. The trend otherwise is very similar to that of the TCP scenario discuss in the previous paragraph.

Scenario II - Comparison with A-MPDU scheme

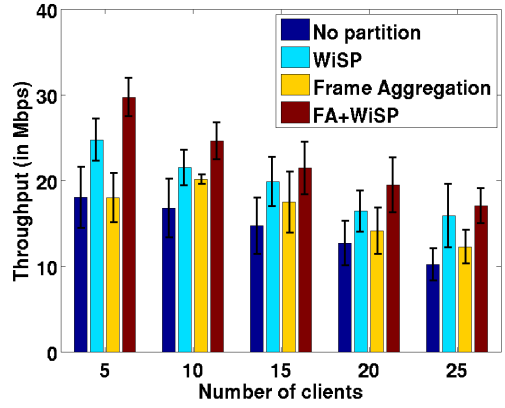


Figure 3.12: Performance comparison (with A-MPDU FA) for UDP flows with every client sending both short and long packets.

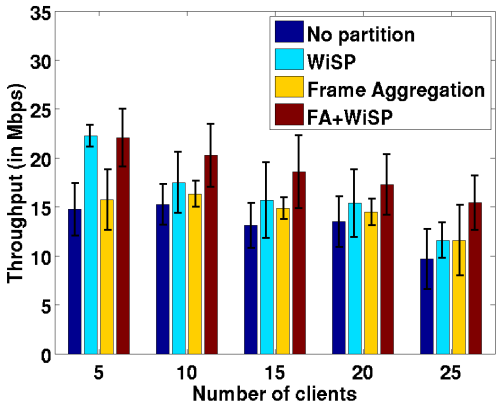


Figure 3.13: Performance comparison (with A-MPDU FA) for UDP flows with some clients sending only short packets and the rest sending only long packets.

We now compare our algorithm with the A-MPDU frame aggregation technique. For the following set of simulations, we vary the number of clients in

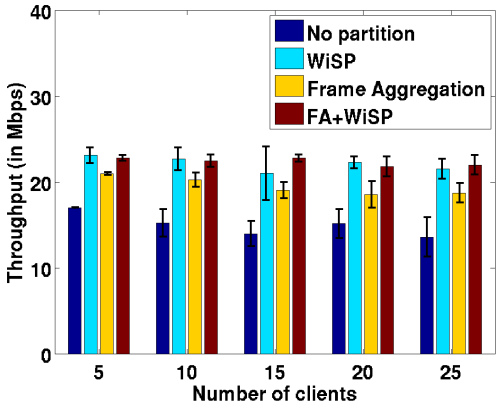


Figure 3.14: Performance comparison (with A-MPDU FA) for TCP flows.

the network from 5 to 25 in steps of 5, and each client is made to generate two UDP flows. One of the flows consists of 100 byte packets at a rate of 1 Mbps, and the other flow generates 1000 byte packets. The rate at which the 1000 byte packets are generated is chosen randomly from {16 Mbps, 24 Mbps, 36 Mbps, 48 Mbps}. We simulated several instances of this network, each time varying the location of the clients around the AP and the rate at which the 1000 byte UDP flows are generated. In each case, we measure the throughput obtained for No Partition, WiSP, A-MPDU frame aggregation (we used the code shared with us by the authors of [23]), and FA+WiSP mechanisms. Because the largest packet size used in this simulation is 1000 bytes, we set the maximum frame size for frame aggregation also to be 1000 bytes to get a fair comparison. We evaluate the throughput for bigger frame sizes later in Section 3.6.2.

We plot the average throughput across all the runs for the four scenarios described above. The corresponding plot is shown in Figure 3.12. We observe from the plot that, despite there being a reduction in throughput as the number of clients increase (possibly due to increased collisions in the network), the WiSP protocol always performs better than the frame aggregation and no partition scenarios. This suggests that WiSP can be useful for scenarios where a client generates different mix of short and long packets with the rate of short packets being relatively small. Furthermore, we observe that the FA+WiSP always has the best performance among all the four scenarios as it combines the benefit of both frame aggregation and WiSP.

Next, we simulate a network in which about half the clients send a 1000 byte packet flow, at a rate of 24 Mbps each, while the remaining clients send a 100 byte packet flow at a rate of 1 Mbps each. Thus, unlike the previous case, every client sends just a single UDP flow to the AP. The number of clients in the network is once again varied between 5 and 25. We once gain measure the throughput for the four scenarios averaged over multiple runs. During each run, we vary the number of clients generating short packets and long packets in addition to varying their positions around the AP. Figure 3.13 shows the corresponding results. First we observe that, though WiSP outperforms frame aggregation on average for all the number of clients chosen in this simulation, the improvement decreases as the number of clients increase. Upon analyzing the data we observe that the reason for this is mainly due to a reduction of the long packet throughput in the case of WiSP when compared to that of frame aggregation. This may be because of increased contention on both the channel partitions. We also observe that FA+WiSP, as before, always has the best performance in all cases.

Finally, we simulate a scenario where a user attempts to open multiple web sessions; web sessions 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 as in the case of UDP flows. Every client in the network is made to simulate the same number of TCP connections as explained. We then plot in Figure 3.14 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, frame aggregation, and FA+WiSP. We observe that WiSP always performs better than frame aggregation. This is because the ACKs and data packets are sent using different radios on different channels in the case of WiSP, resulting in a performance benefit. We observe, however, that WiSP and FA+WiSP have almost the same performance, suggesting that there is not much performance benefit from aggregating ACKs compared to just sending them on a separate channel.

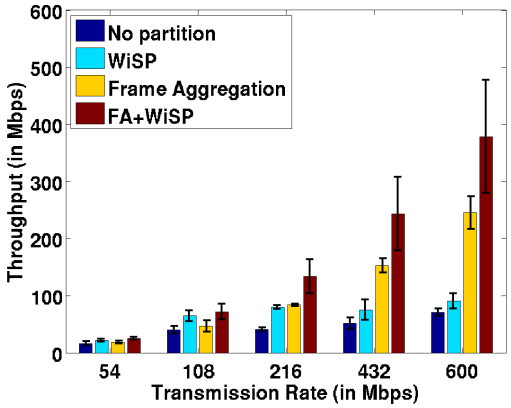


Figure 3.15: UDP flows with every client sending both short and long packets (with A-MPDU FA).

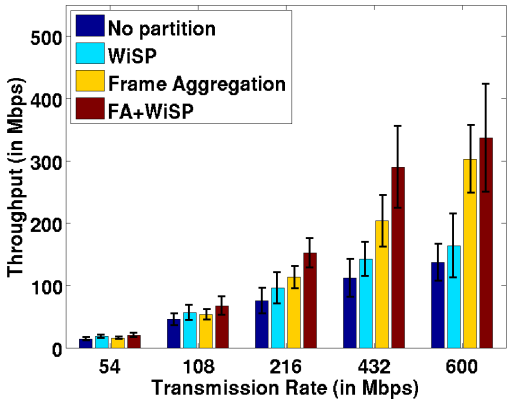


Figure 3.16: UDP flows with some clients sending short packets only while the rest sends long packets only (with A-MPDU FA).

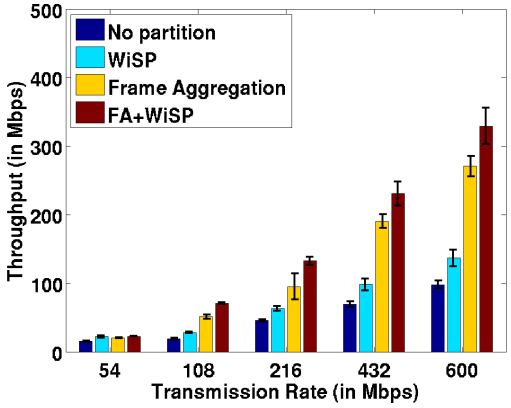


Figure 3.17: Performance comparison (with A-MPDU FA) for TCP flows at higher rates.

Scenario III - Effect of higher data rates and bigger frame size (using A-MPDU FA)

Newer wireless network standards, such as the IEEE 802.11n, are capable of supporting higher data rates of up to 600 Mbps. Furthermore, the aggregated frame sizes at these rates can be up to 64 kilobytes long [14], which is much longer than 1000 bytes used in the previous set of simulations. However, the packet sizes generated by existing applications are still limited to 1500 bytes. Therefore, in addition to just the short packets, even the long packets may be aggregated using bigger frame sizes. Evaluating the performance of our WiSP

protocol is therefore important both for completeness of our performance study, and to bring out the limitations of our approach. To address this issue, we repeated the simulations from Scenario II for comparing with the A-MPDU scheme, for four data rates: 108 Mbps, 216 Mbps, 432 Mbps, and 600 Mbps (when 100% of the channel is used). The maximum frame size used depends on the data rate used. For 54 Mbps, we use a frame size of 5000 bytes; for 108 Mbps, we use 10000 bytes; for 216 Mbps, we use 20000 bytes; for 432 Mbps, we use 40000 bytes; and for 600 Mbps, we use 64000 bytes. The network consists of a single AP with the number of clients chosen uniformly at random between 5 and 15. Figures 3.15, 3.16 and 3.17 show the corresponding throughput values averaged over multiple runs for the two UDP cases and the TCP case described in the Scenario II. We also included the plots for the 54 Mbps case for comparison.

From the plots, it is obvious that though WiSP performs better than the No Partition case, frame aggregation performs much better. Furthermore, the performance of the frame aggregation increases with higher data rates owing to the corresponding increase in the maximum frame size allowed. We, however, observe that FA+WiSP has the best performance in all scenarios. This clearly brings out the limitation of the WiSP approach, when it is not performed along with frame aggregation. This suggests that WiSP cannot be used as a stand alone approach for improving performance, but can rather be used as a complement to frame aggregation to further improve network performance.

Scenario IV - Effect of packet size threshold (using A-MPDU FA)

The packet sizes generated in simulations so far are either 1000 bytes or 100 bytes long. We therefore used a fixed packet size threshold of 128 bytes to distinguish between a short packet and a long packet. While these packet sizes are typical of the voice and TCP flows in the internet [8, 13], video traffic has more varied packet sizes [24, 25]. The authors of [26] have estimated that the packet sizes for video traffic may be Weibull or exponentially distributed, depending on the codec. A different packet size threshold may have to be chosen when the packet sizes are varied.

In this subsection, we evaluate our protocol for more varied traffic and analyze its performance for various packet size thresholds. We created a single

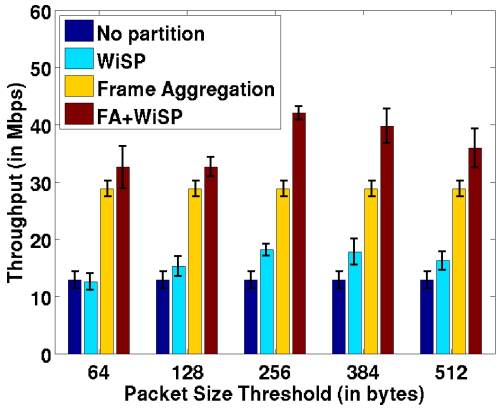


Figure 3.18: Performance comparison (with A-MPDU FA) across various packet size thresholds.

network consisting of an AP to which a number of clients (chosen uniformly at random from 5 to 15) are connected. A third of the clients generate a TCP flow consisting of 1000 byte frames, another third of the clients generate VoIP-type UDP flows consisting of 100 bytes packets at 40 kbps rate. The remaining third of the clients generate a video traffic consisting of exponentially distributed packet sizes with a mean packet size of 750 bytes. We measured the overall throughput of this network using packet size thresholds of 64, 128, 256, 384, and 512 bytes. The corresponding plot for all the four schemes evaluated is shown in Figure 3.18. The maximum frame size for the A-MPDU frame aggregation approach is chosen to be 5000 bytes. We observe from the plot that, as before, WiSP performs better than No Partition and FA+WiSP performs better than frame aggregation for all packet size thresholds. This shows that the performance benefit of our protocol is significant even for a varied traffic scenario. We also observe from the WiSP and FA+WiSP plot that, for the packet size distribution in this simulation, the performance benefit is highest when the packet size threshold is 256 bytes, after which it decreases. This suggests that choosing the right packet size threshold is vital to achieving good performance benefits from our protocol. A threshold that is too large or too small may result in the channel being partitioned inappropriately, such that it may result in more packets queued in one of the partitions with very few packets in the other.

3.6.3 Performance in a Multiple Network Scenario

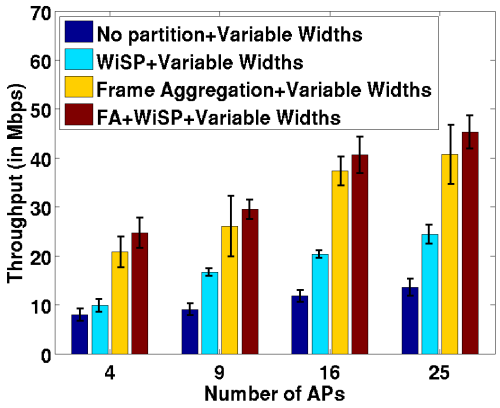


Figure 3.19: Performance comparison (with A-MPDU FA) for a multiple network scenario.

The performance results so far evaluated a single network scenario consisting of just a single AP. The performance results hold as is for a multiple network scenario when each AP (in multi-AP scenario) is allocated a unique channel, as there is no inter-dependence between the APs with respect to our protocol operation. Accordingly, the performance achieved by one AP can be extended to every AP that operates on a different channel. For the case where two APs are allocated the same channel, the performance may be affected. However, the performance in such a scenario will be similar to the case when a single AP is used to serve the clients both in its network and from the adjacent AP's network (that is allocated the same channel).

In this section, we evaluate the performance of our protocols for a scenario where all the APs are allocated channels of varying widths based on their individual client load. For this purpose, we used a load-aware spectrum allocation algorithm called Greedy Rising described in [9]. In this algorithm, the APs with the largest client load are allocated the widest spectrum possible with the constraint that every AP is allocated at least a minimum width spectrum. The reasoning behind this approach is that APs with higher load require more spectrum bandwidth to service all of their client loads fairly. However, when a larger chunk of spectrum is allocated to the APs, their data rates also increase, which provides an opportunity to employ our WiSP protocol to minimize the resulting protocol overheads. The Greedy Rising

algorithm also makes sure that no two APs within one hop of each other are allocated overlapping portions of the spectrum. Because no two neighboring APs share a channel, the carrier sensing mechanism is simplified, as the clients are only required to carrier sense individually the channel partitions on which they are sending the packets.

To evaluate this scenario, we let the Greedy Rising algorithm choose channel widths ranging between 2 MHz and 40 MHz (in steps of 2 MHz). The corresponding data rates using a certain spectrum width are scaled accordingly. Thus, a data rate of 54 Mbps at a 20 MHz spectrum will be scaled to 27 Mbps at 10 MHz and 108 Mbps at 40 MHz. We then measure the throughput for a network consisting of 4, 9, 16, and 25 APs. The number of clients per AP is chosen uniformly at random between 5 and 15. We then generate both a 100 byte UDP flow at 1 Mbps and a 1000 byte UDP flow at 24 Mbps from each of the clients to the AP. Because every AP potentially has a different number of clients, the load for each AP is also different. The throughput results, averaged over 10 runs, where in each run a different number of clients is chosen for each AP, are presented in Figures 3.19. For the case of frame aggregation, while it may be ideal to use a variable aggregated frame size corresponding to the width of the spectrum chosen, we could not find any such approach in the literature. We therefore use a A-MPDU scheme with a fixed frame size of 5000 bytes. We observe from the figure that the trend is very similar to that observed in the previous sections. Particularly, we observe that WiSP performs better than the No Partition case, while FA+WiSP performs the best. The performance benefits are mainly due to the fact that increased channel widths transmit at higher data rates, creating further room for improvement due to a reduction in the protocol overheads.

3.6.4 Performance in a Multihop Network Scenario

We now evaluate our protocol for an ad-hoc, multi-radio, multihop network setting. We generate a network consisting of 100 nodes distributed uniformly at random over a 150 m x 150 m area. The transmission range for each node is set at 30 m. We then choose a set of source and destination nodes uniformly. We simulate both a single channel and a multichannel network. In the case of multichannel network, the channels are allocated statically to

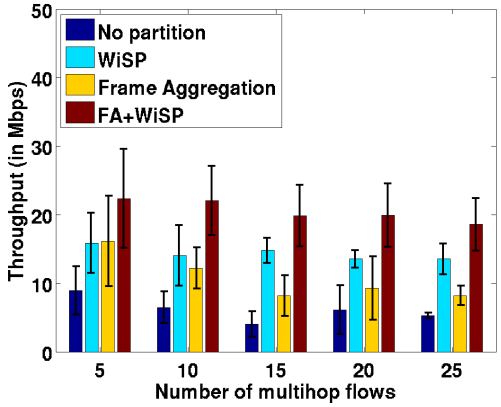


Figure 3.20: Performance comparison (with A-MPDU FA) of TCP flows over a single channel, multihop network.

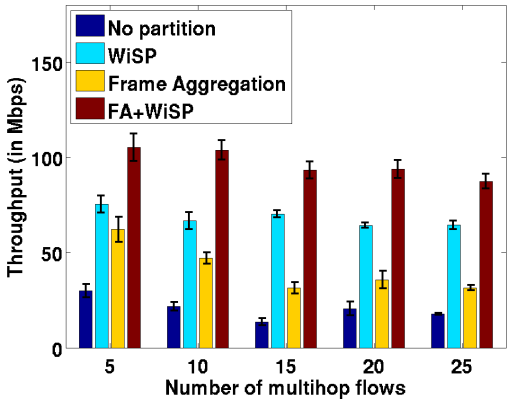


Figure 3.21: Performance comparison (with A-MPDU FA) of TCP flows over a multichannel, multihop network.

the nodes based on the algorithm in [27]. We used 12 channels for allocation. In the case of a single channel allocation, the nodes are equipped with two radios, one each for the two channel partitions realized using WiSP. For a multichannel scenario, each node is equipped with four radios. Two of the radios are used for receiving packets, and two radios are used for forwarding the packets to the next hop. Ideally, the channel partitions on the receive channel of a node must be decided by the receiving node based on the short and long packet mixes received from its senders. Likewise, the partitions for the transmit channel of a node must be decided by the next hop node that receives packets from this node. However, this will require a complicated carrier sensing mechanism, as every node may be required to carrier sense on every possible channel width partitions possible for a given channel. This is because, the mix of short and long packets may be different at different nodes. Therefore, to simplify carrier sensing in our evaluations, we choose a globally common channel partition for each channel, as detailed below.

For both the single channel and multichannel cases, we use the AODV protocol for routing the packets from the source to the destination. The route request messages (RREQ) in the AODV protocol are broadcast on all possible channel widths (as decided by the network wide mix of short and long packets, described below). Additionally, in the case of multichannel networks, the RREQ messages are also broadcast on all channels. We evaluated our protocol using both UDP and TCP flows. We varied the number of source

nodes that originate a flow in the network from 5 to 25, in steps of 5. In each case, for simplicity of evaluation, we ensured that the packets are generated with a fixed proportion of short and long packets. For UDP, we generate 100 byte short packets at 6 Mbps and 1000 byte long packets at 12 Mbps from every source node, so that the network wide PBSP is 33%. In the case of TCP, the PBSP (which are TCP ACKs) per flow turned out to be 10%. Using a fixed proportion of short and long packets allows us to set the percentage of channels for short packets in WiSP manually to be 10% for simulations using TCP flows and 33% for simulations using UDP flows. In each case, a node will carrier sense only the sub-channel on which the packets are to be sent. In other words, a node sending long packets will only carrier sense the sub-channel used for sending the long packets and vice-versa for short packets. We compare the throughput results using WiSP with that obtained using the No Partition, A-MPDU frame aggregation, and the FA+WiSP schemes. The maximum frame size for the frame aggregation scheme is chosen to be 5000 bytes.

The throughput results for TCP flows are plotted in Figure 3.20 for the single channel case and Figure 3.21 for the multichannel case. In either scenario we observe that, except for the case of 5 flows, both WiSP and FA+WiSP perform better than No Partition and frame aggregation. The performance benefit in the case of WiSP comes from the fact that the short and long packets are sent on separate channels, and the savings from the overhead. The throughput results for UDP flows are shown in Figures 3.22 and 3.23 for the single channel and multichannel cases, respectively. We observe that the trend is similar to that of the TCP flow scenario.

From the figures, we observe a big variance in throughputs in the case of single channel plots. This variance is due to the fact that we evaluate the throughput of different topologies individually and plot the average throughput across all the topologies. The plots, therefore, capture the performance comparison that is averaged across multiple topologies. We have also plotted the performance of our protocols for individual topologies along with the performance improvements averaged across the topologies in Figures 3.24 and 3.25 for TCP and UDP, respectively. We observe from the figures that while in general WiSP and FA+WiSP perform better than No Partition and frame aggregation in most of the topologies, there are cases where frame aggregation performs slightly better than WiSP (for example,

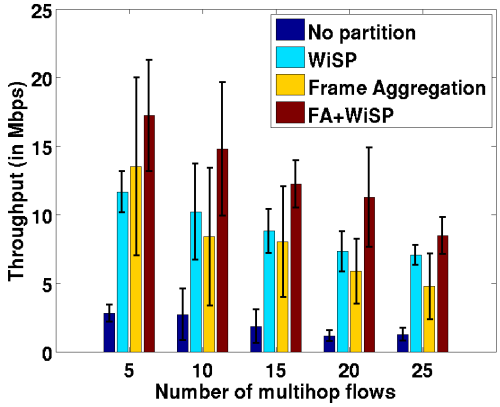


Figure 3.22: Performance comparison (with A-MPDU FA) of UDP flows over a single channel, multihop network.

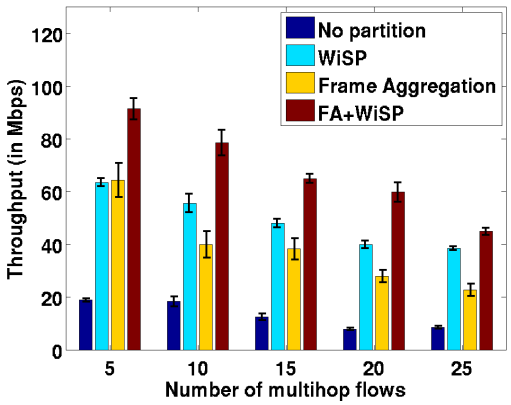


Figure 3.23: Performance comparison (with A-MPDU FA) of UDP flows over a multichannel, multihop network.

topology 5 in the case of 5 TCP flows). This is because, when there are fewer flows in the network, the aggregated packets in the case of frame aggregation experience less contention and therefore, less packet collisions. Note that the benefit from WiSP is also due to the reduced contention per sub-channel. However, because the contention in the network is already small, there is not much opportunity for WiSP to provide a better performance than frame aggregation. This suggests that there can be cases where frame aggregation may be more beneficial. But, WiSP can still provide a good performance improvement in most cases.

3.7 Discussion

In this chapter, we have proposed to partition a channel into a narrow and a wide sub-channel for overcoming bandwidth-independent 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 protocol called WiSP for determining the channel partitions. 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 cannot provide a significant improvement. Furthermore, we have also discussed a mechanism where the WiSP protocol can be used along with frame ag-

gregation to obtain significant performance benefits. We have evaluated our protocol both for infrastructure and ad-hoc network scenarios.

While we have shown that the WiSP protocol is beneficial in reducing the MAC independent overheads in a variety of scenarios, there are three important complexities involved with respect to a practical implementation of this protocol. We enumerate them as follows:

1. The WiSP protocol relies on the fact that the transmission spectrum can be split in to multiple channels of varying widths with arbitrary center frequencies. While some of the newer drivers already allow for variable width channels, they do not allow for arbitrary center frequencies. However, with the advent of software defined radios that allow for such flexibilities, we expect that the future communication systems will allow for different center frequencies.

2. The WiSP protocol requires that every node be equipped with multiple radios. The number of radios depends on the number of channel partitions. Thus, when we have two channel partitions every node requires two radios, in the case of an infrastructure network. Twice as many radios are required in the case of a multihop network. Having multiple radios within a node may increase the complexity and cost of the system. However, reducing hardware costs may make this feasible in the future systems.

3. Finally, the WiSP protocol also requires that the radio can switch between multiple channel widths frequently. We do not factor the associated latencies involved in switching the channel widths. However, we expect that the future hardware will be capable of adjusting the channel widths at a fast time scale.

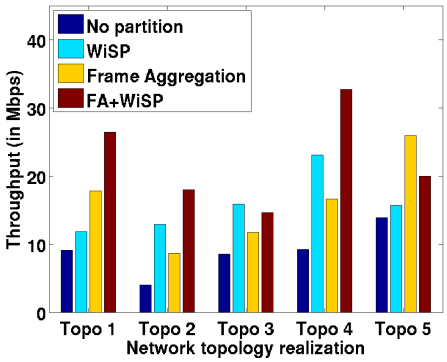
3.8 Future Work

All our evaluations of the WiSP protocol assume a single packet size threshold and split the channel into two. An interesting future work would be to evaluate the protocol with multiple packet threshold, thereby splitting the channel into more than two sub-channels. However, exploring this direction

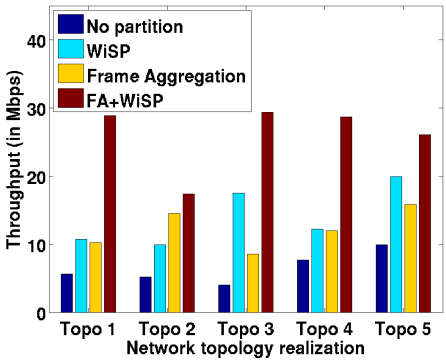
will require each nodes to be equipped with more than two radios. Another direction will be to estimate the appropriate packet size threshold dynamically based on the current packet mixture in the network.

In the case of multihop network, the nodes have to employ complex carrier sensing mechanisms. This is because the nodes in a multihop network may be using different partitions on the same channel spectrum. One straightforward heuristic will be to carrier sense every possible channel-width pair before initiating a transmission. However, this will require different carrier sense thresholds for each of the channel widths. This mechanism, however, can be expensive due to the associated latencies. An interesting future work will therefore be to explore efficient means for carrier sensing across channel widths.

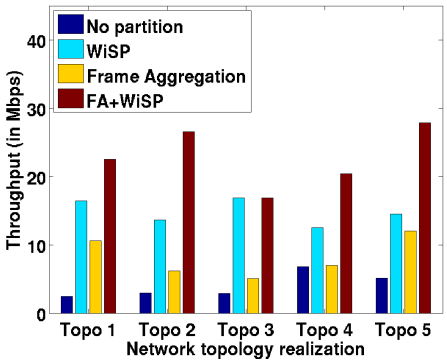
Finally, we developed a heuristic algorithm for partitioning the channels based on the packet sizes. It may, however, be possible to formulate this as an optimization problem, which is likely to be NP hard. Moreover, we need significant knowledge of the network and traffic parameters to arrive at an optimal solution. However, as done by the authors in [9], an approximate version of the problem can be solved and we leave this as a future work.



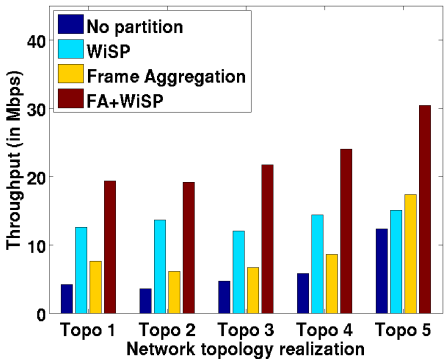
(a) 5 Flows



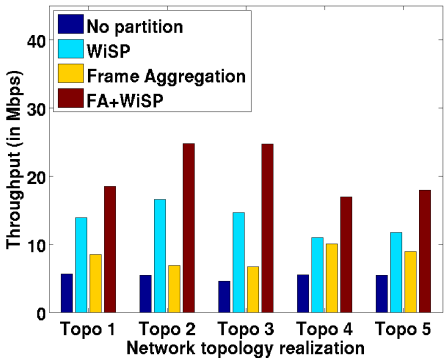
(b) 10 Flows



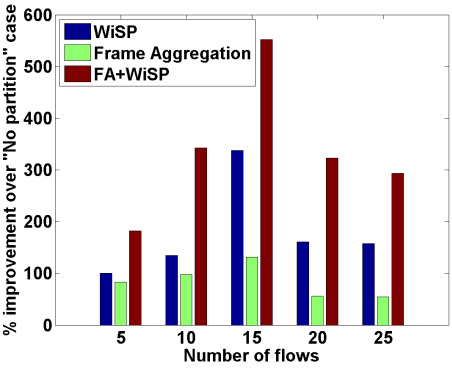
(c) 15 Flows



(d) 20 Flows

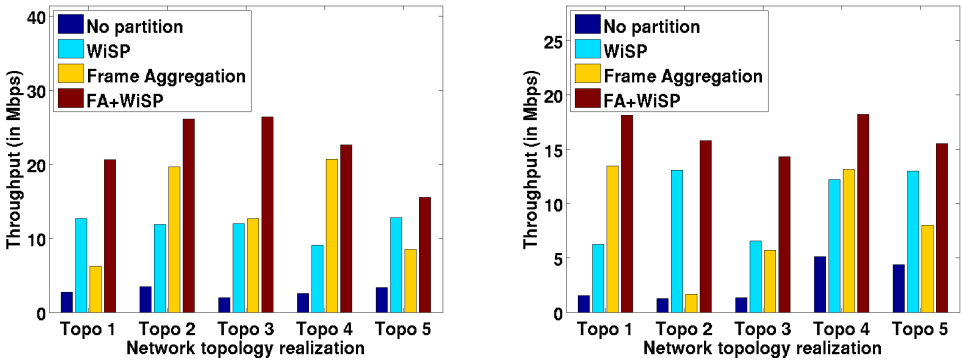


(e) 25 Flows



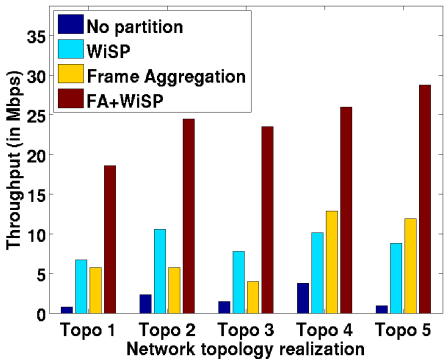
(f) Percentage improvement over No Partition

Figure 3.24: Comparison across multiple multihop topologies - TCP flows.

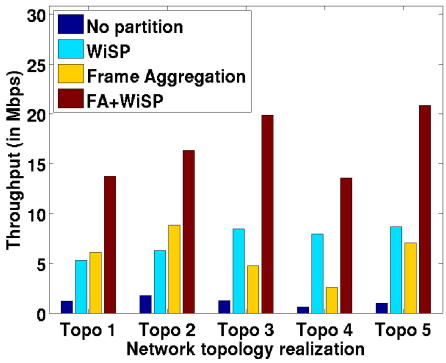


(a) 5 Flows

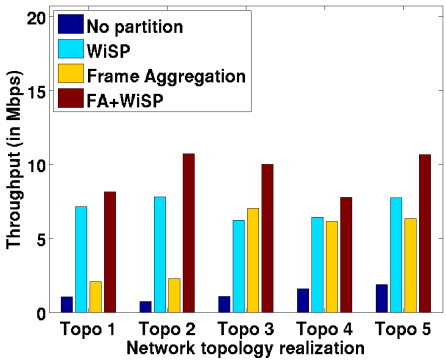
(b) 10 Flows



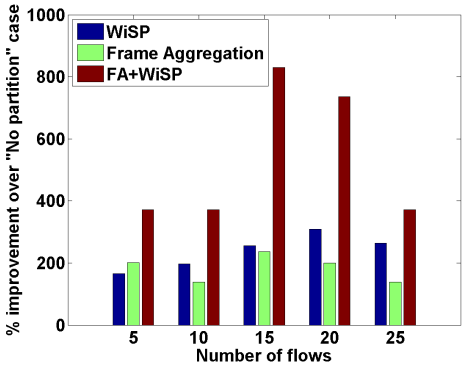
(c) 15 Flows



(d) 20 Flows



(e) 25 Flows



(f) Percentage improvement over No Partition

Figure 3.25: Comparison across multiple multihop topologies - UDP flows.

CHAPTER 4

MANAGING HARDWARE LATENCIES FOR DELAY SENSITIVE APPLICATIONS

In a multichannel wireless network, multiple nodes that operate on different channels have to be coordinated to ensure network connectivity. The coordination between the nodes can be built as part of the channel allocation algorithm that chooses channels for the wireless interfaces. Three popular channel and interface allocation strategies exist in the literature, namely common control channel approach [5], static channel approach [27, 28], and hybrid channel approach [10]. Among these three channel allocation strategies, the common control channel and hybrid channel approaches exploit the channel switching capabilities of the wireless radios.

In the common control channel approach [5], a set of nodes, before initiating a communication, exchange control messages on a common channel, mostly using a radio dedicated for this purpose, to agree upon the channel to use before a data transmission. The radios used for transmission are then switched to the channel agreed upon. The disadvantage with this scheme is that it wastes significant time (used for deciding the transmission channel) and spectrum (used for the dedicated control channel). In the static channel approach [27, 28], the radios in a node are assigned channels such that any two neighboring nodes share at least one common channel to to ensure network connectivity. In this scheme, none of the wireless radios switch their channels during communication. However, this scheme offers limited flexibility in channel allocation and may not use all the available channels for allocation [29].

In the case of the hybrid channel and interface allocation strategy [10], a subset of radios in a node are assigned channels that do not switch channels, as in the static approach, and the remaining set of radios are allowed to switch channels when required. Only the radios that have fixed channels are used for receiving data. Furthermore, the radio with the fixed channel is used for transmitting data that are intended for neighbors with the same

fixed channel. The radios that are capable of switching channels are used for transmissions to neighbors that are on a different channel. Thus, network connectivity can be maintained more easily than in the static approach.

Among these three approaches, the hybrid multichannel protocol has been shown to be efficient in providing higher system throughput [10]. However, the hybrid channel allocation approach is not optimized for providing low delays for real-time applications, such as VoIP. This is because, while a static channel allocation achieves network connectivity by ensuring common channels with neighboring nodes [27], a hybrid channel allocation relies on the radios of a node to switch across channels to maintain network connectivity. The latency associated with channel switching can be prohibitive for delay sensitive applications such as VoIP or interactive gaming [30], especially in the case of a multihop operation. Because no such channel switch delays are incurred in a static channel approach, a static scheme may be beneficial for delay sensitive applications. However, a purely static channel-based approach is not suitable for providing higher throughput values for non-delay sensitive applications [29]. Moreover, a pure-static channel approach is not suitable in a network where the link characteristics keep varying that can make the network topology change with time (as in a mobile network). Therefore, we need a new scheme that can exploit the advantages of both the static and the hybrid channel allocation schemes.

Routing real-time applications over multichannel wireless networks has been handled in several different ways in the literature. Most of the existing approaches concentrate on provisioning QoS in multichannel wireless networks [31, 32]. The authors in [32], for instance, propose a topology control and QoS routing approach with the goal of providing bandwidth aware routing for real-time flows. However, the approach requires significant topology information for its execution, and hence not wholly suitable for an unmanaged network. In [31], the authors provide a QoS-aware multichannel scheduling algorithm for providing higher priorities for VoIP packets, by which they are scheduled more often than non-real-time packets. A similar approach for scheduling delay sensitive flows more often than non-delay sensitive flows is proposed in [33]. In [34], the authors propose a gateway controlled channel allocation scheme, where the channel allocation to the nodes is determined by the gateway based on the flows in the network. However, the scheme does not differentiate between real-time and non-real-time flows. In this work, we

propose a mechanism that can provide low delay routes for real-time applications and high throughput routes for non-real-time applications, which can complement any of the existing QoS mechanisms. The goal is to consider practical difficulties (such as hardware delays) that may exist in a network, which many of the existing QoS mechanisms overlook.

We propose a mechanism called SHORT that exploits the benefits of a static channel approach for providing lower delay paths for real-time applications, while at the same time utilizing the flexibilities of a hybrid channel approach for providing higher throughputs for non-delay sensitive applications. According to this approach, we design a protocol that can, depending on the type of traffic being routed, control the channel allocation strategy of the nodes. More specifically, when routing a delay sensitive flow, the routing protocol, after determining the route to be taken for the flow, forces the nodes on the path to behave as in a static channel approach. In other words, the radios in the nodes are controlled in such a way to prevent them from switching across channels for the duration of the real-time flow. A hybrid channel allocation scheme is used for routing non delay-sensitive flows. We modify the multichannel AODV routing protocol proposed in [10] for this purpose. Note that, while our protocol enables the nodes on a real-time flow's path to behave as in the static channel mechanism, the actual path is not determined by our approach and is taken care of by the multichannel routing protocol [10], discussed briefly in Section 4.1, that is complemented by the hybrid channel allocation protocol (hence the name static-hybrid approach).

Using actual implementations on a multichannel mesh testbed called Net-X [3], we show that the end-to-end delays of real-time flows are significantly lower in SHORT when compared to a purely hybrid approach. Furthermore, we show that the throughput of non-delay sensitive applications is also not degraded too much.

4.1 Background

The hybrid channel allocation protocol [10] is discussed briefly in Chapter 2 (see Section 2.3) and we do not repeat it in this chapter. In this section, we provide a brief overview of the multichannel routing protocol that is used in the testbed on which we carry out our experiments. In the discussion that

follows, we assume that every node is equipped with two radios or interfaces (the terms *interface* and *radio* are used interchangeably in this work and both mean a wireless radio).

4.1.1 Routing Protocol

The routing mechanism used currently in our testbed is an AODV protocol, modified for multichannel operation. The modifications to the original AODV protocol include incorporating a mechanism for finding a channel diverse route, avoiding bottlenecks, and reducing the overall expected transmission time in addition to reducing the number of hops [3]. More specifically, to utilize the benefit of using multiple channels, it is necessary to make sure that a flow experiences minimum intra-flow interference (interference due to transmissions of the same flow on adjacent hops). This requires that the route taken by the flows is such that the adjacent hops are on different channels as much as possible. Furthermore, it is preferable to avoid routing multiple flows through a single node, as this may result in the node requiring to switch its transmission channel frequently for routing the flows, which may possibly be targeted at neighbors on different channels. These requirements are incorporated in the form of a routing metric, called the MCR metric [3], as the traditional routing metric based on hop count is not suitable. The MCR metric, in brief, uses the statistics of channel usage, such as the time spent by the wireless radio transmitting or receiving on a particular channel and the number of bytes sent or received on that channel, from the interface drivers. The metric then uses the channel statistics to calculate the cost for switching the channels for routing a flow, which is essentially the time taken to switch a radio from one channel to another. Channels that are heavily utilized (i.e., have more data sent on them) will have a higher switching cost than those that do not have as much data. Additionally, the cost of a link per channel is estimated using the popular ETT metric [35] on every channel, which when coupled with the switching costs and summed up over the entire path results in the MCR routing metric. If $SC(c_i)$ is the channel switching cost of channel c_i used in the i^{th} hop of transmission, and ETT_i is the estimated transmission time in the i^{th} hop, then the MCR metric is given by

$$\text{MCR} = (1-\beta) * \sum_{i=1}^{h} (ETT_i + SC(c_i)) + \beta * \max_{1 \le j \le c} X_j$$

where β is a weight between 0 and 1, h is the number of hops on the path, and c is the total number of channels. X_j is the total ETT cost for all links using channel j on the route, and is given by

$$X_j = \sum_{\forall i \text{ such that } c_i = j} ETT_i.$$

The ETT of a link is given by $ETT = ETX * \frac{S}{B}$, where ETX is the expected number of transmission attempts (including re-transmissions) required to transmit a packet, S is the average packet size and B is the data rate of the link. The expected number of transmissions is estimated based on the loss in the link.

The multichannel protocol also incorporates a few other modifications [3]. For instance, when a routing entry is created for a node, it is also necessary now to indicate the channel and the actual interface to use for reaching the next hop. The multichannel routing protocol incorporates the appropriate mechanism for creating the route entries. Furthermore, route caching, available in the original AODV protocol, is not performed as the channel allocations and the corresponding costs may change frequently, which can be estimated accurately only at the destination. Finally, the multichannel routing protocol incorporates a procedure called "Route Refresh", by which a source node initiates a route discovery periodically (currently every 30 seconds in our testbed) for learning routes with better costs or for updating the costs of the current route.

4.2 Problem Statement

A pure-hybrid channel allocation approach (such as HMCP [10]) is optimized for providing higher system throughputs for non-delay sensitive applications. However, a main drawback with the hybrid channel allocation approach is the channel switching delay associated with the wireless radio hardware and software. For instance, the channel switching delay T_s in our hardware is approximately 5 ms. This includes several components such as delays due

to stopping interrupt service routines of the driver, tuning to the new frequency, re-starting the interrupt service routines and sensing the medium. To compensate for the higher switching delays, it is advisable to spend a non-trivial amount of time in a channel, before switching to another channel for amortizing the switching costs. Additionally, consider a scenario where there are multiple packets to be sent by a node, each on a different channel. In this case, while sufficient time has to be spent transmitting packets on the current channel before switching to the next channel, there has to be a limit on the time spent on any single channel. In the implementation used for our experiments, the minimum time spent on a channel, T_{min}, is 20 ms, and the maximum time spent on a channel, T_{max}, before switching to another channel that has packets waiting to be sent is 60 ms. The relevance of these parameters and the procedure used for choosing these values are discussed in more detail in [36]. Thus, the channel switching delay T_s along with T_{min} and T_{max} together may add to the overall transmission time of a packet.

To illustrate more on the switching delays, we discuss the following simple experiment.

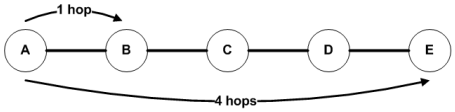


Figure 4.1: Topology used for ping experiments.

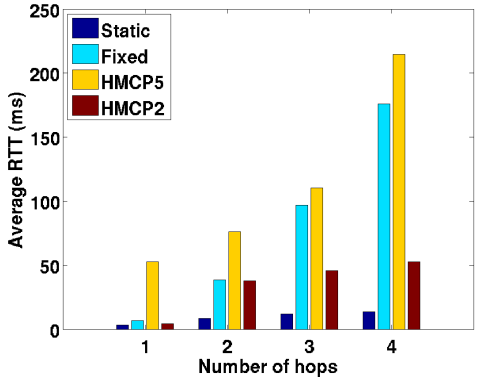


Figure 4.2: Results for pinging the nodes in flooding mode.

4.2.1 Ping Experiment

In this experiment, we use up to five wireless nodes that are placed across different offices in our lab and arranged in a linear topology as shown in the Figure 4.1. We initiate one hop, two hop, three hop, and four hop pings in flooding mode with 1500 byte packets. In flooding mode, consecutive

ping requests are sent after a user specified interval without awaiting a ping reply. We used an interval of 0 seconds (which is the default in our system) between consecutive ping packets. We plot in Figure 4.2 the resulting average round trip time (RTT) returned by ping when all the nodes use four different schemes:

- **Fixed:** All the nodes are allocated the same fixed channel. This makes the network behave like a single channel network. However, the switchable radios are free to switch across the remaining channels for sending broadcast `hello` packets. The fixed radio is used for sending the `hello` packets on the fixed channel.
- **Static:** The nodes are assigned channels using the centralized static channel allocation scheme proposed in [27] with five channels. No broadcast `hello` packets are sent in this scheme.
- **HMCP5:** The nodes are assigned channels using the hybrid multichannel protocol with five channels. Like the Fixed scheme, both the fixed and the switchable radios are used for sending the broadcast `hello` packets on all the five channels.
- **HMCP2:** The nodes are assigned channels using the hybrid multichannel protocol with only two channels. Note that in the case of HMCP2, the switchable radios do not switch channels as there is no other channel to switch (the other channel is allocated to the fixed radio). The broadcast `hello` packets are sent only on these two channels.

We can readily observe from Figure 4.2 that the average RTT in the case of HMCP5 is significantly higher than the other channel allocations. Furthermore, we observe that the RTTs become worse as the number of hops increase. Finally, we also observe that the RTTs in the case of HMCP2 are much lower than HMCP5 and the Fixed cases, but higher than the Static case. This is because, in HMCP2, there are only two channels used for allocation versus five channel in the case of Static. Therefore, Static has lower contention resulting in a lower delay than HMCP2. Furthermore, HMCP2 has to transmit a periodic `hello` every 5 seconds on both the channels. Additionally, in the case of HMCP (both with 2 channels and 5 channels), our system can schedule transmissions on only one channel at a time [36]. This

is because, the HMCP protocol makes use of a special functionality of the Linux kernel, called a 'bonding driver' (Section 4.1.1 in [36]). A feature of the bonding driver is to 'bond' multiple physical wireless interfaces into a single virtual interface. Furthermore, the driver creates a single network address common for all the physical wireless interfaces (Section 4.2 in [36]). As a result, the packets can be sent on only one interface at a time. Thus, when there are transmissions scheduled on a switchable radio, the packets to be sent on a fixed radio have to be delayed (and vice-versa). This can add more delays in the case of HMCP. The bonding driver feature is also used in our implementation of the Fixed scheme, and as a result similar delays will be experienced by the packets using this scheme. Additionally, in the case of Fixed scheme, the adjacent hops of a flow have to contend for channel access as they are both transmitted on the same channel, increasing the delay further.

In the case of HMCP5, few other reasons contribute to the higher delays, as enumerated below.

1. A transmission from one node to another that are on different fixed channels requires channel switching. This can take place at every single hop of the path taken by the flow.

2. Because a periodic broadcast message, such as `hello` or a route refresh has to be sent on every channel, the associated switching delay adds up, at every hop, to the end-to-end delay. Any ping packets queued on a channel other than the one in which the broadcast packets are currently being sent have to wait until the corresponding channel is scheduled, resulting in a delay.

Assuming a channel switching delay of 5 ms, and a $T_{min} = 20$ ms, which is the amount of time spent on each channel, a message broadcast on multiple channels will incur a switching delay for every channel that it switches to (other than the fixed channel), and a delay due to the time spent on each of the switchable channel. Thus, for a broadcast on 5 channel, a delay of $((5-1) \times 5 + (5-1) \times 20 = 100$ ms) will be experienced and for 2 channels, a delay of $((2-1) \times 5 + (2-1) \times 20 = 25$ ms) will be incurred (we subtract a 1 in each case to account for the fixed channel, as the packet sent on the fixed radio does not incur a channel switching delay). Thus, the periodic broadcast messages, such as `hello` packets alone can cause round trip delays of up to $2 \times 100 = 200$ ms and $2 \times 25 = 50$ ms, respectively. This is the

reason for HMCP2 to have a lower delay when compared to HMCP5.

The resulting delays, in the case of HMCP5, are prohibitive for real-time, delay sensitive applications such as VoIP or interactive gaming, and therefore alternate mechanisms have to be formulated for routing such applications. However, we should also ensure sufficient throughput for non-delay sensitive applications that may co-exist in the network. This motivates a routing approach that can improve the delay and throughput performance depending on the type of application. From Figure 4.2, we see that a static channel allocation may be advantageous for real-time applications, as it results in the least RTTs among the four mechanisms compared. Our proposed protocol exploits the advantages of this allocation. In this work, we assume a dense network scenario that has a predominantly non-delay sensitive traffic with fewer delay sensitive applications. In fact, this mimics a real network scenario, as most of the flows in the present day internet are HTTP or FTP-type best effort traffic.

4.3 Proposed Approach

Motivated by our initial ping experiments, we develop a new routing strategy, called SHORT, for controlling the wireless radios and the underlying channel allocation mechanism. The idea is to make the wireless radios behave as in a static channel allocation mechanism for real-time applications and to follow the hybrid channel allocation mechanism for non-real-time applications. Accordingly, the nodes in the network operate on one of two modes as follows:

- **Normal mode:** The normal mode of operation is exactly as explained in Section 2.2 and shown in Figure 2.2, wherein only the fixed radio is used for receiving data and either the fixed or the switchable radio is used only for transmitting data. The fixed radio is used for transmissions when the next hop node's fixed channel is the same as this node. Otherwise, the switchable radio is used after switching to the next hop's fixed channel. This mode of operation is used for non-delay sensitive traffic.

- **Static mode:** This mode is used for delay sensitive flows. In this mode, the switchable interface is not allowed to switch channels for the duration of the delay sensitive flow. Rather, after the route for the flow is determined, it remains fixed on the channel of the previous hop's[1] fixed interface. Furthermore, both the fixed and the switchable radios are allowed to receive and transmit. In other words, the switchable interface also behaves like a fixed interface for the duration of the delay sensitive flow.

Note that only those nodes that lie in the path of a delay sensitive flow operate in static mode. The remaining nodes in the network continue to behave as in the normal mode. Furthermore, the nodes that are in static mode revert back to normal mode of operation once the delay sensitive flow ends. (A flow ends if there are no packets with the same flow identity for a certain duration of time.) The associated protocol steps for getting back to normal mode involve restoring the channel switching capabilities of one of the radios and updating the neighbors on the current fixed channel used. While the channel allocations in our protocol are based on HMCP to simplify implementation, any existing dynamic channel allocation can be used, in general.

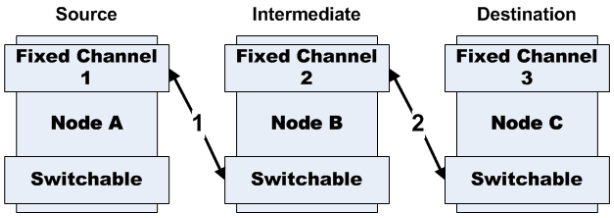


Figure 4.3: SHORT protocol operation.

We now explain this concept more clearly using the illustration in Figure 4.3. The figure shows a traffic flow from node A to C via node B. Let the channels allocated to the fixed radios of the nodes A, B, and C be labeled 1, 2, and 3, respectively. Accordingly, during the static mode of operation, the switchable radio of node C is fixed to channel 2, which is the fixed channel of node B. Similarly, the switchable radio of node B is fixed to the channel 1,

[1]The terms 'previous hop' and 'next hop' imply the appropriate nodes in the path as seen by a node in the 'source to destination' direction of the flow.

which is the fixed channel of node A. Thus, traffic from A to B flows on channel 1, and that from B to C flows on channel 2. Moreover, the switchable radios on nodes B and C receive the traffic on these channels transmitted by the fixed radios of nodes A and B, respectively. Observe that any traffic from C to A can be routed using the same configuration, except that the switchable radios will be sending traffic that will now be received by the fixed interface of the nodes B and A. Thus, this setting enables a bi-directional flow without requiring any channel switching. We would like to point out that the switchable interface of node A is not required to be fixed on any channel in this example, and is free to switch across channels as in the normal mode. Because in this example the nodes B and C behave as in a static channel allocation (both the radios are non-switchable and every node on the path shares a channel with the adjacent hop nodes), we call this static mode.

In the static mode of operation, a node does not send a broadcast message on all the channels (unlike the normal mode of operation, see Section 2.2). Instead, it simply forwards them on the channels to which the two radios are fixed. Note that this may result in a few nodes not being aware of the channel used by their neighboring nodes. We require in our protocol that two nodes involved in a direct communication be aware of each other's channels (as otherwise the nodes cannot decide on which channel to transmit). This may result in a node losing connectivity with the nodes that are on a channel different from those on which the broadcast messages are sent. When several such nodes lose connectivity with each other, this can eventually result in a network partition. To avoid such a scenario, we propose a channel re-selection mechanism that works in tandem with the routing protocol. More details of the channel re-selection mechanism are explained later in this section.

4.3.1 SHORT Protocol Operation

We now discuss the details of the SHORT protocol. We assume that the information whether the flow being routed is delay-sensitive or not is available at the routing layer of the source node. Such information can be passed on from the upper layers by, for instance, using the ToS (type of service) field in the IP header. Some of the VoIP applications such as session initiation protocol (SIP) have the service type information in-built in the packet headers [37]. We now present the protocol sequence executed for a delay sensitive

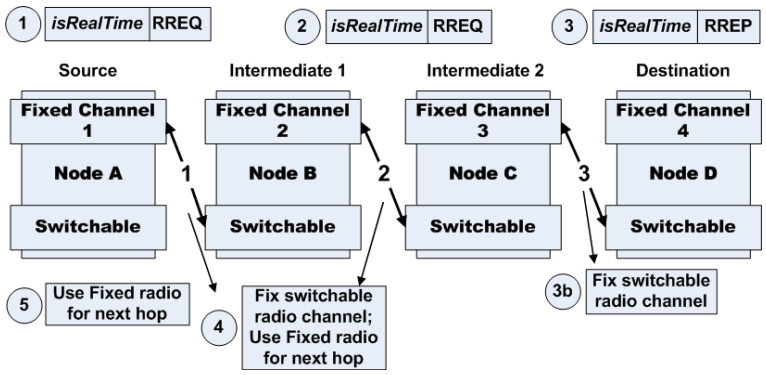


Figure 4.4: SHORT protocol steps.

flow. The sequence of procedures carried out for a non-delay sensitive flow is as done in the multichannel routing protocol, explained in [12, 3, 38], and is not reproduced here. The protocol mechanism described for delay sensitive flows, however, is a modification of the multichannel routing protocol and to avoid duplication of work, we present only the relevant modifications to the multichannel routing protocol. The protocol steps discussed below are illustrated using the example in Figure 4.4.

Once the source node determines that its flow is a delay sensitive flow, the following steps are performed:

1. The source node (node A in Figure 4.4) initiates a route request message (RREQ) along with a special flag, `isRealTime`, to indicate that the request is for a real-time flow and broadcasts it on all the channels, if it is in normal mode, or on the two fixed channels if it is in static mode.

2. Any intermediate node (nodes B and C in Figure 4.4), that is not the destination, simply re-broadcasts the RREQ message on all channels, if it is in normal mode, or on the two fixed channels if it is in static mode.

3. The destination (node D in Figure 4.4), upon receiving the RREQ, creates a route response (RREP) message and unicasts the RREP along with the `isRealTime` flag (copied from RREQ) to the node from which the corresponding RREQ was received. Additionally, it takes the following actions only if the channel on which the RREP is unicast (which

is the fixed channel of the next node to whom the RREP is to be sent) is different from its own fixed channel:

(a) The node sends a broadcast `hello` message on all the channels. However, in this case, the node includes the flag `isRealTime` along with two channel identifiers. One is the fixed channel that it has been operating on, and the other is the channel over which the above RREP is unicast. (Note that the original `hello` message described in Section 2.2 contains only the fixed channel information.) The cost associated with this broadcast is considered as part of the route setup cost, which does not affect the delay experienced by the delay sensitive packets after the route has been setup.

(b) The node fixes its switchable interface on the channel over which the RREP message is unicast and uses this interface for sending any data in this direction. In the example in Figure 4.4, node D fixes its switchable radio to channel 3 (fixed channel of node C) over which it unicasts the RREP.

(c) The switchable interface is also informed to start receiving packets on this channel.

4. Any intermediate node, upon receiving the RREP along with the `isRealTime` flag, forwards the RREP message to the node from which it received the corresponding RREQ message. Furthermore, the intermediate node, in addition to performing the set of operations described in Step (3) when the RREP is unicast on a channel different from its own fixed channel, also performs the following:

   (a) The node creates a routing entry for the next hop node and uses the fixed interface for sending data in this direction of flow (forward direction). Thus, nodes C and B in Figure 4.4, in addition to fixing their switchable radios to channels 2 and 1, respectively, use their fixed radios to send packets in the forward direction.

5. The source node, upon receiving the RREP, starts sending the packets after creating the routing entry for the next hop node through its fixed

interface. In Figure 4.4, this operation is performed by node A, which after receiving RREP starts sending packets through its fixed radio.

Once the radios of the nodes in the real-time flow's path are fixed based on the above steps, any transmissions by these nodes (including broadcasts) are restricted to the two fixed channels. Observe that any non-real-time existing in the chosen real-time path prior to the arrival of the real-time flow may be affected because of this protocol. In particular, an existing non-real-time flow may be dropped during the above process as the radios on the corresponding path will no longer be allowed to switch across channels. We handle this situation by initiating a RERR message, which gets forwarded to the source. The source can then re-initiate a new RREQ message to find a new route.

4.3.2 Channel Re-selection Mechanism

The channel re-selection mechanism is introduced to maintain network connectivity in spite of nodes in static mode restricting their broadcast to only the two channels that their interfaces are fixed on. The channel re-selection mechanism is only executed by those nodes that lie adjacent to the path chosen for the real-time applications and are in the normal mode. For this purpose, the nodes make use of the broadcast `hello` message with an `isRealTime` flag broadcast by a node in the path of a real-time flow before switching to the static mode (see Section 4.3.1 step 3a.). Upon receiving the broadcast message with an `isRealTime` flag, the nodes initiate a channel re-selection mechanism and perform the following steps:

1. The node first checks if both of the channels contained in the `hello` message are different from its fixed channel. If its fixed channel is the same as one of the channels in the `hello` message, the node discards the message and takes no further steps.

2. If both the channels in the `hello` message are different from the node's fixed channel, then the node selects one of the two channels, chosen uniformly at random, as its new fixed channel.

3. If more than one `hello` message with an `isRealTime` flag is received (which may happen when a node is adjoining two nodes that lie in

the path of a real-time flow), then the node first tries to choose the channel that is common to a majority of the `hello` messages. Thus, the channel re-selection mechanism is designed to improve the chance of maintaining connectivity with more nodes in the network. If none of the channels is common to all the `hello` messages, then the node just selects one of the channels contained in the `hello` messages, uniformly at random, as its fixed channel.

When many of the flows in the network are real-time, the channel re-selection mechanism will tend to make the overall network behave as in a pure-static approach, while when most flows in the network are non-real-time, the network tends to behave as in a pure-hybrid approach.

4.3.3 Implementation Details

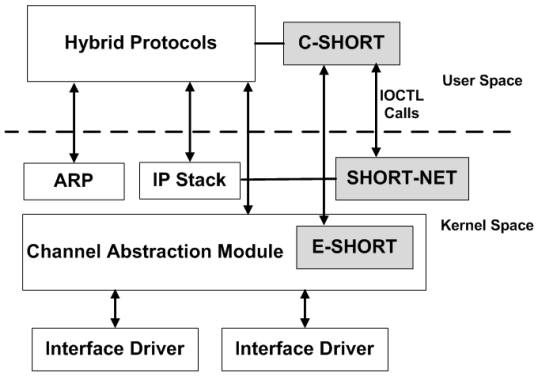


Figure 4.5: Protocol architecture with SHORT-specific components in gray.

The architecture of our multichannel protocol along with the SHORT implementation is shown in Figure 4.5. The SHORT protocol consists of two main components, namely the SHORT controller or *C-SHORT* and the SHORT executor or *E-SHORT*. The *C-SHORT* is implemented in the user level and interacts with the multichannel routing protocol for creating routing entries compatible with the static mode of operation whenever a real-time flow is to be routed. Furthermore, it is also responsible for setting the `isRealTime` flag when a new route discovery for a real-time flow is initiated. Finally, *C-SHORT* indicates to the *E-SHORT* component, through a special

IOCTL control message, whether to transition to static mode or revert back to normal mode. (IOCTL messages are used commonly in Linux for any interaction between the user space and kernel space code.) If the message is for transitioning to static mode, then the channel to which the switchable radio has to be fixed from now on is also specified.

The *E-SHORT* component, on the other hand, is implemented as a kernel module and resides as part of the Linux 'bonding' module.[2] The *E-SHORT* component is responsible for fixing the switchable radio to the channel supplied by the *C-SHORT* component and for restoring the switchable radio back to normal mode, depending on the message from the *C-SHORT* component.

In addition to the two main components, SHORT protocol also consists of a smaller third component, called SHORT-NET, which interacts with the Linux netfilter hooks for making the switchable interface behave like a fixed interface for real-time flows. In normal mode, the netfilter hook is designed to drop any incoming packets on the switchable radio. The SHORT-NET overrides this and lets the switchable radio accept the packets while in static mode. The relevant control messages are passed on from the *C-SHORT* as an IOCTL message.

4.4 Experimental Results

In this section, we present the experimental results to illustrate the performance benefits of the SHORT protocol. Before proceeding further, we first present an overview of our testbed and the associated hardware.

4.4.1 Testbed Overview

We use a multi-channel, multi-interface, and multi-hop wireless testbed called Net-X, developed by the Wireless Networking Group at the University of Illinois at Urbana-Champaign (UIUC). The testbed consists of more than 20 Soekris net4521 boxes distributed across various offices on the fourth floor of the Coordinated Science Lab (CSL) at UIUC. Each testbed node has a 133 MHz microprocessor, a compact flash (CF) card slot, two PCMCIA slots, and one mini-PCI slot. We run Linux kernel 2.4.26-based operating system

[2]The Linux bonding module has been modified to enable multi-radio operation [3].

on each of these boards. For our experiments, we equip the test nodes with one mini-PCI and one PCMCIA wireless card. These wireless cards are based on Atheros chipsets and are driven by madwifi drivers. The cards operate in the IEEE 802.11a mode. The mini-PCI cards make use of a pair of external antennas, and the PCMCIA card has its own internal antenna for communication.

4.4.2 Experimental Methodology

Traffic Details:

For evaluating our protocol, we used different traffic sources for generating real-time and non-real-time traffic. For real-time traffic, we used a tool called D-ITG [39] for generating G.711 codec type VoIP packets for 50 seconds. The tool generates approximately 100 byte VoIP packets every 20 ms. The same D-ITG tool is used for generating non-delay sensitive TCP and UDP type packets. The UDP flows are generated at a rate of 6 Mbps and the packet sizes are fixed at 512 bytes. The size of the TCP packets on the other hand is fixed at 1000 bytes, and the packets are generated at the rate of 1000 packets per second. Both UDP and TCP packets are generated for a duration of 50 seconds. Every wireless radio transmits at the maximum power and the physical rate of transmission is fixed at 6 Mbps. For all the experiments we use five orthogonal 802.11a channels for allocation, namely the channel numbers 36, 48, 64, 149, and 161.

Protocols Compared:

We compare the performance of our SHORT protocol with HMCP [3] and two other protocols as described below:

Static channel allocation: For this case, we allocate channels to the radios using a centralized static channel allocation methodology. In other words, knowing the connectivity graph among the nodes, we allocated channels to the two radios in each node such that every node has at least one channel in common with their neighbors. The channel allocation technique is

based on the scheme proposed in [27]. Any broadcast packets in this scheme are sent only on the two static channels.

Fixed channel for real-time traffic: This is a protocol similar to, but simpler than, SHORT. In this protocol, while generating RREPs for route discovery for a real-time flow, the source node also includes its current fixed channel in the RREQ message. Every intermediate node re-broadcasts the RREQ message, as usual. However, while forwarding the RREP message the corresponding node changes its own fixed channel to that of the source node (which is embedded in the RREP message). Thus, all the nodes in the path of a real-time flow use their fixed interface for routing and tune the fixed interface on the same channel. The advantage of this scheme is that the switchable radios in the nodes need not be fixed and can remain switchable as in normal mode of NetX (and can continue sending the `hello` messages periodically). As a result, unlike with SHORT, there will be no loss of connectivity. We call this the fixed mode of operation. Figure 4.6 illustrates this protocol. Here, we show a real-time flow from node A to C via node B. All the three nodes use their fixed interface for this flow, and tune the interface to channel 1.

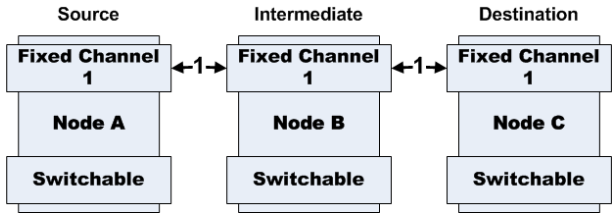


Figure 4.6: Fixed mode operation.

Performance Metrics:

The D-ITG tool is capable of generating per flow statistics on the minimum, maximum, and average delays, average jitter, and throughput achieved. Because throughput is not a concern for real-time flows, and delays are not important for non-real-time flows, we measure the average and maximum delays and jitters (which is the variance in time of arrivals of adjacent packets at the destination) for the real-time traffic and the throughput for the

non-real-time traffic.

Time Synchronization:

For measuring the delays, it is important to have a common notion of clock between the traffic sources and destinations. However, the wireless nodes used have imperfect clocks and proper time synchronization is necessary for measuring time delay values between the sender and receiver. We therefore use `ntpdate` periodically on these nodes for synchronizing their time values. Because the nodes are not connected to the internet, we use a desktop computer as the network time protocol (NTP) server and synchronize all the nodes relative to this server. We use the local wired LAN connectivity for time synchronization between the nodes and the desktop NTP server. Using this scheme we get an accuracy close to 50 μs.

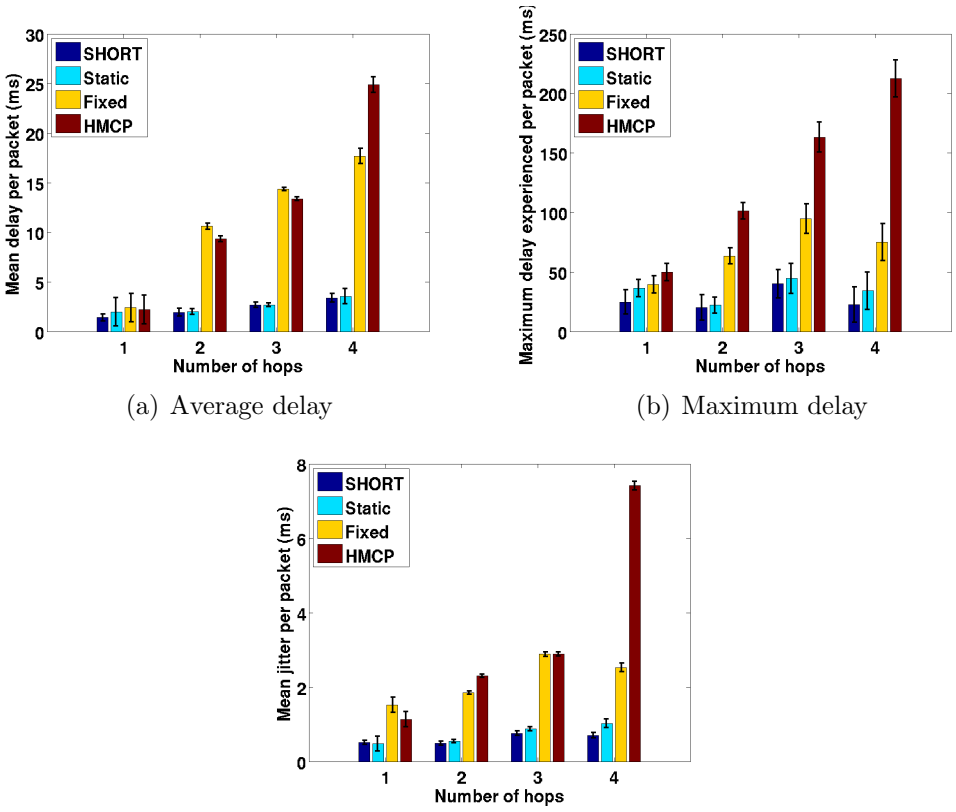
4.4.3 Results for a Multihop Setting

The first set of experiments are conducted to demonstrate the performance of our protocol in a multihop setting.

Unidirectional flows:

For this experiment we first generate a VoIP flow between a source and a destination that is located one hop away from the source. We then repeat this for destinations that are two hops, three hops, and four hops away from the same source node. In each case, we measure the average delay, the maximum delay, and the mean jitter experienced by the packets. We repeated the same set of experiments 10 times, each time choosing different source and destination nodes. The results averaged over the 10 different one hop, two hop, three hop, and four hop routes are plotted in Figure 4.7. In all the figures presented in this section, the plots corresponding to the static channel allocation are labeled as Static and those obtained for the case where we use the fixed channel for real-time flows are labeled as Fixed.

From Figure 4.7(a), we first observe that the average delays experienced by the VoIP packets in the case of SHORT and Static are always lower than 5 ms, irrespective of the number of hops. (As a comparison, a one hop transmission using 802.11a experiences a typical delay of 2 ms.) We also observe that the



(a) Average delay

(b) Maximum delay

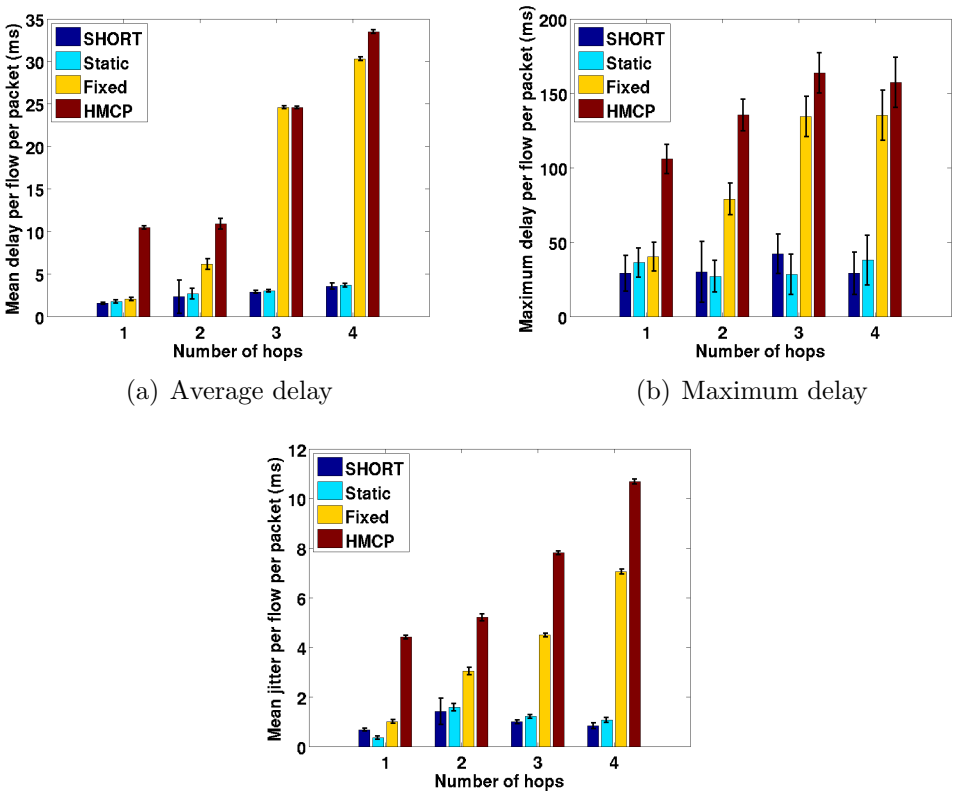
(c) Mean jitter

Figure 4.7: Performance comparison for multihop unidirectional VoIP flows.

delays in the case of Fixed and HMCP allocations are much higher than SHORT or Static allocation, and the difference increases significantly as the number of hops increase. As mentioned in Section 4.2, the main reason for higher delays in the case of HMCP is the need to switch the channels at every hop along the multihop path. In the case of Fixed channel allocation, the flows experience similar or lower delay than HMCP. However, the delays are still high when compared to SHORT or Static. The high delay in the case of Fixed scheme is for two reasons. First, in the Fixed scheme consecutive hops transmit on the same channel when routing a real-time flow. Because of the resulting contention, one hop of the flow has to wait for an adjacent hop to finish transmission. The second reason for this is that the fixed radio has to share its transmission opportunity with that of the packets in the switchable radio, as explained in Section 4.2. The switchable radio is used for sending the periodic `hello` packets, and due to hardware restrictions, only one radio can transmit on any one channel at a time in our system.

These two reasons cause the delays to be, sometimes, higher than HMCP. In the case of HMCP, though the average delay of about 38 ms for the 4 hop case is acceptable for VoIP packets, the rate at which the delay grows with the number of hops is significant and the delays may become unacceptable in the case of real multichannel deployments, where more than 4 hops may be common. Furthermore, as we can observe from the maximum delay values in Figure 4.7(b), some of the packets in HMCP experienced delay of more than 200 ms, which is certainly unacceptable for VoIP. We also observe that the maximum delay values for SHORT and Static cases are well under 50 ms and do not increase significantly with the number of hops. Finally, the jitter values shown in Figure 4.7(c) suggest that packets using HMCP undergo a significant increase in jitter as the number of hops increases, which may make it unsuitable for some of the real-time applications, such as video.

Bidirectional flows:



(a) Average delay

(b) Maximum delay

(c) Mean jitter

Figure 4.8: Performance comparison for multihop bidirectional VoIP flows.

In this case, we generate two VoIP flows, one from a source to a destination

and the other from the destination to the source. The mean and maximum delay values, and the mean jitter averaged over all the flows and over 10 different pairs of nodes (each pair constituting a different run), chosen from different locations in the network for each scenario, are plotted in Figure 4.8. We first observe that the delays in the case of SHORT and Static mechanisms are similar to those in the unidirectional case. This is because, in the case of SHORT protocol, once a route is established between two nodes, the same route is used both for the forward and reverse traffic. The same is true in the case of Static mechanism. The delay values for both the Static and SHORT cases, however, are higher than the unidirectional case. This is once again due to the fact that only one radio can transmit in our system at any instant. Similarly, the delays in the case of Fixed mechanism are also higher for the same reason. The increase in the delays in the case of HMCP from that in the unidirectional case is mainly because significant time is spent by the switchable radio in switching between the forward and reverse traffic.

VoIP with UDP and TCP (non-delay sensitive)

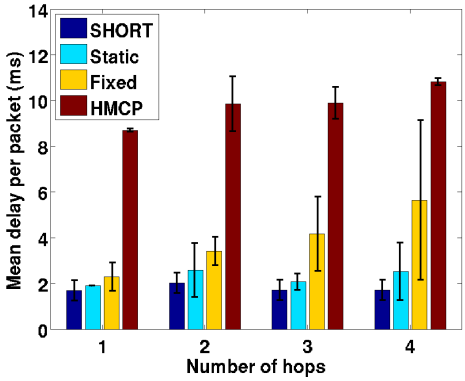


Figure 4.9: Average delay of VoIP packets sent with a UDP flow.

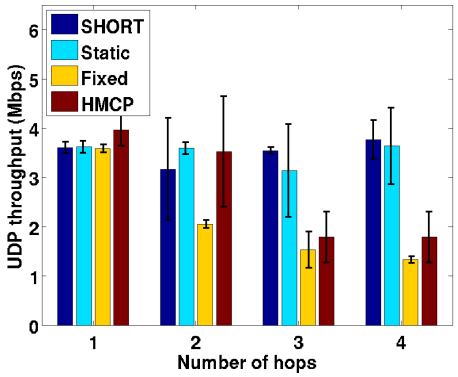


Figure 4.10: Throughput of UDP flow sent with a VoIP flow.

For this experiment, we first generate a VoIP flow along with a UDP flow, both from the same source and targeted at the same destination. Figure 4.9 shows the average delay experienced by the VoIP packets, and Figure 4.10 shows the throughput achieved by the UDP packets, all averaged over 10 different source-destination pairs. Next, we generate a VoIP flow along with a TCP flow as before, and the delay and throughput values of the VoIP and TCP packets, respectively, are plotted in Figures 4.11 and 4.12.

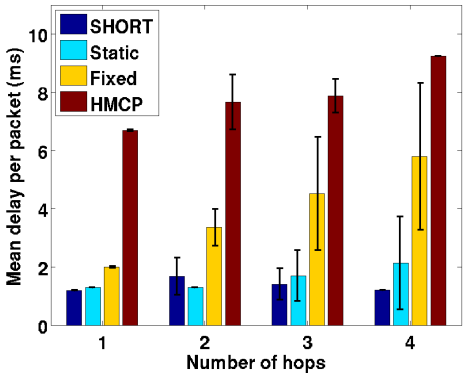


Figure 4.11: Average delay of VoIP packets sent with a TCP flow.

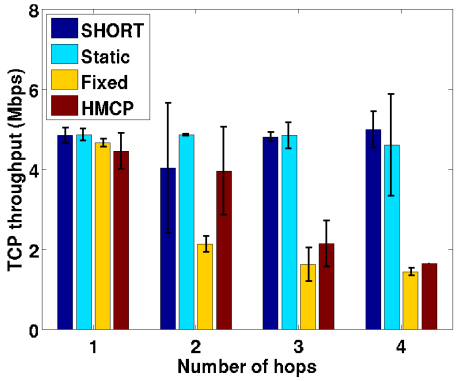
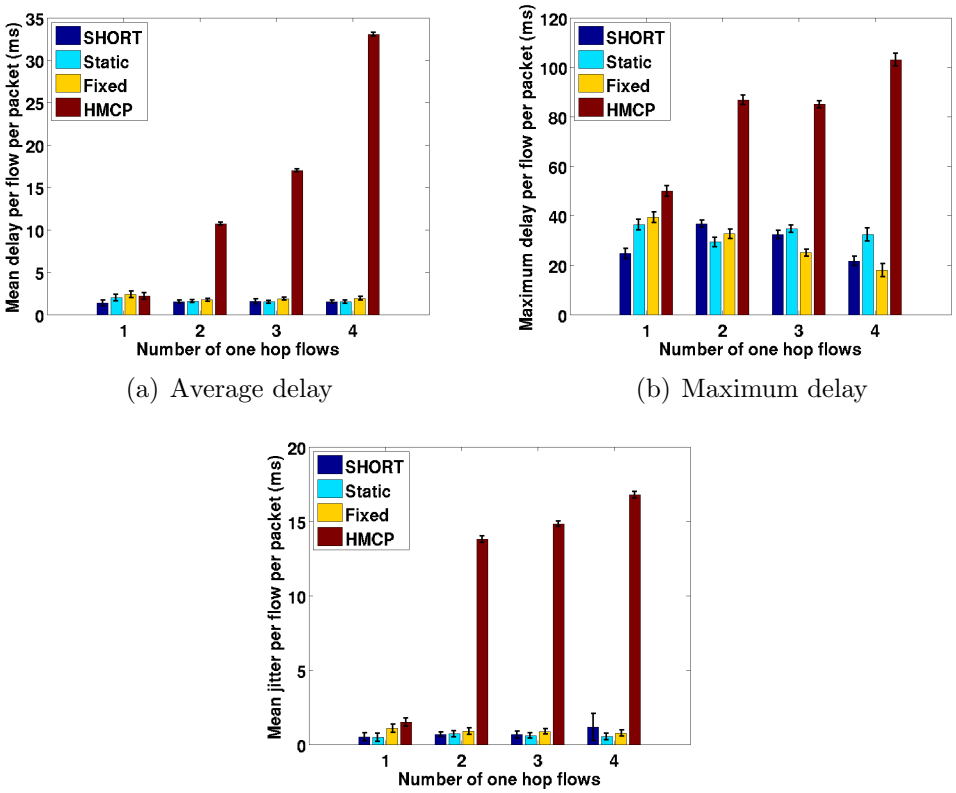


Figure 4.12: Throughput of TCP flow sent with a VoIP flow.

We observe from the plots that the throughputs for both UDP and TCP flows remain almost the same, irrespective of the number of hops, in the case of SHORT and Static protocols. However, we observe that the throughputs reduce with the number of hops in the case of Fixed and HMCP. In the case of HMCP, one reason for this is that in our system, when a radio switches from one channel to another any unsent packets in the packet queue are dropped. This is explained in more detail in [36]. Additionally, in the case of TCP flows, the data and the ACK packets, sent in opposite directions, require that the switchable radio be switched between the two directions, worsening the throughput further. When the Fixed scheme is used, the VoIP packets, TCP or UDP data, and ACK packets (in case of TCP) are all sent on the same channel, and the increase in contention reduces the overall throughout. In the case of SHORT protocol, the TCP and the UDP flows benefit from the channel assignment made for the VoIP flows, which does not incur any delays or packet losses due to channel switching. The throughput performance in the case of SHORT, therefore, is similar to that of Static.

We observe that the delay values in Figures 4.9 and 4.11 are lower than those observed in the unidirectional flows scenario. The reason for this may be because the delay values are only averaged over the VoIP packets that are successfully received at the destination. When there are additional packets from TCP or UDP, some of the VoIP packets are dropped as a result of a buffer overflow. These dropped packets are not accounted in the delay measurements; we believe this to be the reason for the lower delay values compared to the unidirectional case.

4.4.4 Results for Multiple Single Hop Flows



(a) Average delay

(b) Maximum delay

(c) Mean jitter

Figure 4.13: Performance comparison for multiple one hop VoIP flows from a node.

We now evaluate the capability of the protocols in supporting multiple flows from the same source node, as such a scenario may usually involve several channel switches when each flow is sent on a different channel. For this purpose, we choose a source node and four other nodes that are within one hop from the source node. We then generate multiple VoIP flows (varied from one to four) between them. Once again, we choose 10 different source nodes and a set of four nodes situated at different locations and distances from each other (but still within one hop from the source) in our network and present the average values across the different realizations. The average and maximum delay values per flow along with the mean jitter values are plotted in Figure 4.13. We observe from the delay plots that the average and maximum delay value does not vary much with number of flows in the case of SHORT, Static, and Fixed mechanisms, while it increases significantly for

HMCP. This is because all the flows are sent over just a single hop, and in the case of SHORT and Fixed schemes, all the transmissions take place on the same channel. This is due to the fact that all the receive nodes in the case of SHORT tune their switchable radio to the fixed channel of the source node; and in the case of Fixed scheme, all the fixed radios are assigned the same channel. For the Static scheme, the source node and all the destination nodes share at least one channel in common. In any case, there is no channel switching required in any of these three schemes. When HMCP is used, however, the switchable radio of the source node has to switch across the fixed channel of the receiving nodes. This results in an increase in the delay as the number of flows increase.

This shows that HMCP is not capable of carrying multiple real-time flows from the same source to neighbors on different channel, as it requires significant channel switching. From the jitter values, we observe that HMCP performs poorly while handling multiple flows. Higher jitter values mean that the amount of jitter buffer at the receiving node should also be higher, so as to prevent packet losses. The jitter performance for SHORT and Static are fairly stable irrespective of the number of flows.

4.5 Simulation Results

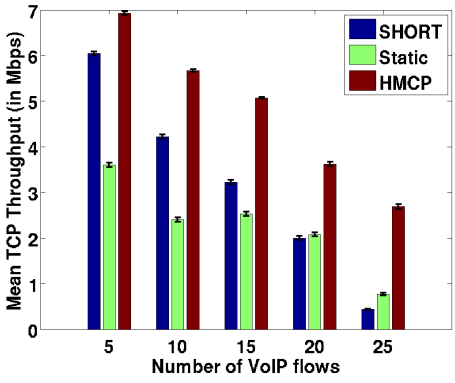


Figure 4.14: Throughput comparison with more VoIP flows in a 100 node network.

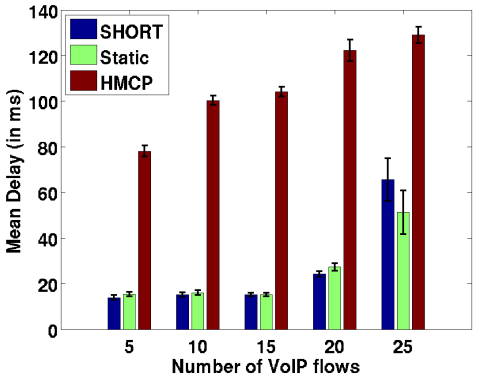


Figure 4.15: Delay comparison with more VoIP flows in a 100 node network.

We performed simulations using ns-2 to evaluate our protocol on a bigger

network to better understand its scalability. We generated 100 nodes distributed uniformly at random on a 150m x 150m area. The transmission range of the nodes is fixed at 30 m. We then generated a combination of TCP and VoIP flows from 50 different source nodes chosen uniformly at random to 50 different destination nodes, again chosen uniformly at random. The number of VoIP flows is varied from 5 to 25 (in steps of 5), and the remaining flows are TCP. The TCP flows have a frame size of 1000 bytes, while the VoIP flows have a packet size of 100 bytes generated at a constant rate of 1 Mbps (over an UDP transport).

We simulated the SHORT protocol, the static channel allocation, and HMCP channel allocation schemes and in each case calculated the overall throughput achieved by the TCP flows and the end-to-end delay of the VoIP flows. Figure 4.14 shows the TCP throughput values, and Figure 4.15 has the VoIP delay values. We observe that because HMCP protocol is optimized for achieving better throughputs, it results in a higher throughput for the TCP flows. In the case of SHORT protocol, the throughout performance is better than the static channel allocation when the number of VoIP flows is small. However, the throughput gets worse than that of static channel allocation scheme when the number of VoIP flows in the network increases (more than 20 flows in our simulations).

Similarly, we observe from Figure 4.15 that HMCP has the worst delay performance of the three schemes. However, the VoIP flows using SHORT protocol start exhibiting higher delays as the number of flows increases. The reason for decreased TCP throughput and higher delays is that, as the number of VoIP flows increases in the network, more nodes end up not switching their radios. This results in the flows taking a longer route. This is illustrated in Figure 4.16. Here, we have plotted the average number of hops taken by the TCP flows. We observe from the plot that the flows in HMCP have the least number of hops. In the case of SHORT, the number of hops keeps increasing as the number of VoIP flows increases resulting in a route longer than the static channel allocation scheme in the case of 25 VoIP flows. The per-hop delay in the case of SHORT is still smaller than HMCP, as there is no channel switching delay associated with SHORT. The number of hops in the case of Static is high because of the requirement that two nodes share at least one common channel, while utilizing all the available channels may result in a longer route. This is also observed in [29]. These results suggest

that the SHORT protocol, though it may be useful when the network is not dominated by real-time flows, may start performing poorly when most of the flows in the network are real-time.

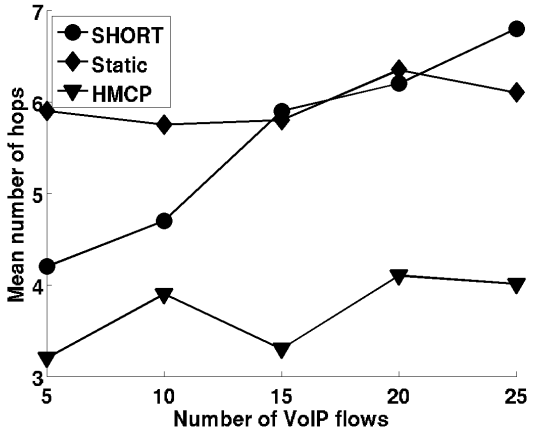


Figure 4.16: Number of hops taken by TCP flows.

4.6 Discussion

In this chapter, we proposed SHORT, a routing approach that exploits the benefits of both static and hybrid channel allocation strategies. We have implemented the protocol on a real multichannel testbed and using experimental data we have demonstrated the performance benefits of the SHORT protocol over a hybrid channel allocation protocol, called HMCP. All our experimental results illustrate that the SHORT protocol can provide low delay multihop paths for real-time traffic, while not significantly affecting the throughputs of non-real-time traffic when there are only few real-time flows in the network. Our results also show that the delay performance of real-time flows using SHORT protocol is comparable to that of a static channel allocation method. We also simulated the protocols using ns-2 to better understand their performance in a bigger network. The simulation results illustrate the limitations of the SHORT protocol, as they suggest that the performance of the protocol deteriorates when the number of real-time flows in the network increases.

In addition to the benefits and limitations of the SHORT protocol discussed

in the previous sections, there are two important complexities associated with a practical realization of this protocol, as enumerated below:

1. The SHORT protocol requires that the network is capable of re-selecting a channel during route setup. This relies on the fact that the broadcast `hello` messages sent by the nodes for this purpose are received loss-free by the neighbors. If a significant number of these `hello` messages are lost, then the network may get disconnected for reasons discussed in Section 4.3.2.

2. The SHORT protocol may not be suitable in a multichannel network where some nodes have only one radio. In this case, the radios may have to be switched to maintain network connectivity, and implementing SHORT will result in the network tending towards a single channel network.

4.7 Future Work

The SHORT protocol reallocates channels based on static channel allocation starting from an initially throughput optimized hybrid channel allocation. One possible future direction to explore will be to come up with an initial hybrid channel allocation that requires fewer channel re-selections when the static scheme is introduced. This can be realized by allocating the same channel to multiple neighbors of a node dynamically based on their traffic usage. Achieving such a scheme requires stringent interference prediction and an efficient neighbor discovery mechanism.

Another possibility will be to develop a routing protocol that jointly chooses the channel allocation during route setup, which is optimized for both throughout and delay. Such a joint channel and route selection may experience higher route setup delays. However, it may be beneficial for flows that are sustained for a long time.

CHAPTER 5

VARIABLE WIDTH CHANNEL ALLOCATION BASED ON TRAFFIC DEMAND

In the previous chapters we assumed that the width of the channel allocated to a node is fixed. This is true with most of the existing multichannel protocols. For instance, the channel width in the case of IEEE 802.11a is fixed at 20 MHz. When the traffic load is low, it may be sufficient to use only a fraction of the fixed width channel to guarantee the required data rate. Alternatively, when the traffic load is large, then more than the spectrum available in the fixed width channel may be required. However, the current standard method of allocating fixed width channels does not allow for this flexibility. If, instead, the spectrum widths are allowed to be variable, then the flows that require less spectrum can use a narrow spectrum width, allowing the remaining spectrum to be used by flows that have higher loads. While some wireless standards, such as IEEE 802.11n, allow for variable channels widths, there are no existing provisions to allow for dynamically varying the spectrum widths based on traffic needs.

In this chapter, we introduce a protocol for variable width channel allocation. The motivation of our protocol is to admit a flow only if there is sufficient spectrum to satisfy its rate requirement. We perceive that our protocol may be useful in a distributed wireless sensor network [40], where every node has a certain amount of data to be routed. Based on the availability of the spectrum, the nodes can decide whether or not it is possible to send the data at the required rate.

The channel width is selected jointly with an appropriate channel during routing. The technique used for partitioning a channel based on packet size, as introduced in Chapter 3, can still be applied in addition to choosing a variable width channel.

Using variable width channels has gained significant interest recently. For instance, the authors of [9, 21] have leveraged channel width adaptation for the purpose of load balancing. The authors of [41] have considered variable

width channels to minimize the adjacent channel interference in the network by allocating narrow width channels to the nodes whenever the load is small. In [42], adapting the channel widths is shown to offer efficient spectrum utilization.

In the subsequent sections of this chapter, we start by exploring the related literature. We then present the network and interference model, followed by our variable width channel allocation protocol. We then present simulation results to validate the protocol benefits.

5.1 Related Work

To the best of our knowledge, most of the existing work on QoS provisioning for wireless networks has focused on fixed channel widths. In [32], Tang et al. have proposed separate optimization problems for channel allocation and QoS routing for multi-channel, fixed-width wireless networks. In [43], Xu et al. have presented a QoS framework over LANMAR routing with a single fixed width channel per node. They propose a probing-free call admission control (CAC) mechanism and thus claim lower admission delays. In [44], the authors discuss a link-state approach coupled with a core node set extraction. In [45], Perkins et al. have presented QoS extension for the Ad Hoc On demand Distance Vector (AODV) routing protocol [11]. While this work focuses primarily on signaling and path setup, we extend this work by developing algorithms to allocate spectrum widths to achieve QoS goals. In [46], Liao et al. have attempted to provide a probe-ticket based approach for provisioning QoS. The unique aspect of their route discovery mechanism is what they call a ticket-splitting approach. QoS provisioning is achieved by allocating bandwidth to every sub-ticket on every intermediate node during route-discovery. This approach may present an interesting solution in the absence of a contiguous spectrum at certain nodes.

The notion of bandwidth adaptability in wireless networks has been recently researched in [9] and [21]. In this work, the authors have demonstrated bandwidth adaptability in 802.11a/b wireless networks using a wireless emulator and a few experiments. The authors show that narrow bandwidth transmissions can have a greater communication range and experience reduced interference when compared to wide-channel transmissions. On the

other hand, they also show that wide-channel communication can achieve higher data rates and increase the overall spectrum utilization in the network. We propose to use the inherent tradeoff involved between narrow and wide channel communication for differentiating the traffic flows with the goal to provide QoS depending on the application needs.

5.2 System Description

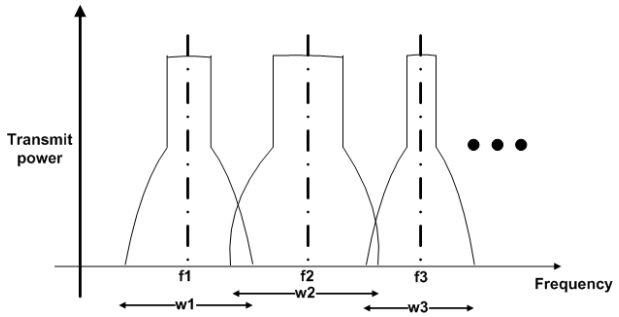


Figure 5.1: An illustration of a spectrum with different center frequencies and channel widths.

We use the multichannel, multi-radio model developed in Chapter 2. To distinguish amongst the possibilities of a channel having multiple widths, we use the term 'channel' to describe just the center frequency of operation. The width of the channel can be one of k values centered around this center frequency. We assume that the center frequencies of the channels are fixed and their widths are adapted around these fixed center frequencies. However, our protocols are applicable to scenarios where the center frequency can also be varied.

As an illustration, if we consider the spectrum shown in Figure 5.1, the center frequencies $f1, f2, \ldots$ indicate the channel of operation and the widths $w1, w2, \ldots$ indicate the width of the channels. If a channel l, currently operating with a width $w1$, is to be expanded, it is achieved by occupying a portion of channel $(l+1)$ (if available) and an equal portion of channel $(l-1)$ (if available), thereby maintaining the symmetry around the center frequency. Therefore, the width of a channel depends on the availability of spectrum from its neighboring channels as elaborated later in Section 5.2.1.

Note that according to our system architecture discussed in Chapter 2 (see Section 2.2), every node is equipped with two radios, namely a fixed radio and a switchable radio. The fixed radio is used for transmitting and receiving data on the fixed channel, and the switchable radio is used for transmitting data on a channel other than the fixed channel. The channel on which a node transmits depends on the fixed channel of the next hop node. When variable channel widths are used, the radio used for transmitting in a node must use the same channel width, in addition to the center frequency, as that of the next hop's fixed radio.

5.2.1 Interference Model

Because the channels can be operated on different widths, the amount of overlap between the adjacent channels can significantly affect the cross-channel interference between the channels. The effect of partial overlap between channels has been studied in the literature [47, 48, 49]. As concluded by [49], the degree of interference between two partially overlapping channels depends on the actual extent of overlap and the spatial separation of the nodes that use the channels. Based on this conclusion, we assume that adjacent channels cannot choose arbitrary widths, if they are allocated to nodes that are separated by less than or equal to two hops. (The term "hop" is defined as follows: If two nodes can have a direct communication link between them, then they are said to be within one hop from each other. If a transmission from one node to another requires h one-hop transmissions, then the nodes are said to be h hops away from each other.) The reason for considering nodes within a two hop neighborhood follows from the protocol model of interference [50], which requires a receiver to be separated by at least a two hop distance from an interfering transmitter.

More specifically, we assume that the adjacent channels (if used within a two hop neighborhood of each other) can reliably operate on their 'default' channel widths. By 'default' channel width, we mean the actual widths as specified by the corresponding technology standards. For instance, IEEE 802.11a specifies the channels to be of 20 MHz wide and IEEE 802.11b specifies a 22 MHz channel. However, if one of the channels has to expand its channel width (based on the flow requirements), then its neighboring chan-

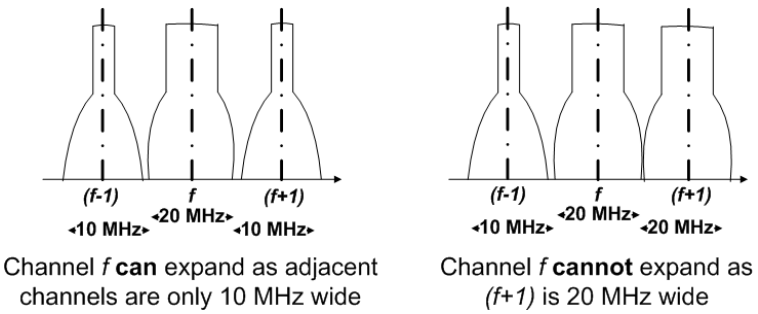


Figure 5.2: An illustration of allowed and disallowed channel width expansions. A check mark indicates that the expansion is allowed and a cross mark indicates that it is disallowed. (Figure not drawn to scale.)

nels have to use a narrower width. As an illustration, consider for instance that some node using an IEEE 802.11a channel, say f, is currently transmitting on its default width of 20 MHz. If this channel is required to expand to a width of 30 MHz, then as explained in Section 2.2, it can do so by occupying 5 MHz from channel $(f + 1)$ and another 5 MHz from channel $(f - 1)$, depending on availability. By availability, we mean that the flows on channels $(f + 1)$ and $(f - 1)$ are using widths that are smaller than their default widths. Thus, the channel width expansion in this example is possible only if both the channels $(f + 1)$ and $(f - 1)$ are using at the most 10 MHz (as the channel width has to be symmetric around the center frequency). This is illustrated in Figure 5.2.

In addition to the criteria on channel width adaptation, to avoid a potential co-channel interference, we also require that no two nodes that are within two hops from each other use a channel with the same center frequency (irrespective of their channel widths).

5.3 Proposed Mechanism

The goal of our protocol is to determine whether a flow can be admitted at a node based on the rate requirement of the flow and the possibility of expanding a channel, if required, based on the channel used by its neighbors. All the nodes are allocated an initial channel of the same width. Every node maintains a data structure containing the center frequency, the width of channel, and the traffic load at each of its neighbors that are within two

hops. This data structure can be built by exchanging this information with each of the neighbors periodically.

When a new flow is to be routed, every node checks to see if the flow can be admitted based on the flow's rate requirement and the available spectrum at a node. (We assume that we can estimate the rate requirement of the flows from its traffic pattern using existing techniques in the literature, such as [51, 52].) A node denies admission to a flow if its rate requirement cannot be satisfied. The steps involved in routing along with the routing metric used and the admission control mechanism are discussed in the following subsections.

5.3.1 Routing Protocol

We modify the popular AODV routing protocol [11] for determining the routes based on the rate requirements of the flows. We can estimate the rate requirement of the flows from its traffic pattern using techniques proposed in [51, 52]. The rate requirement of a flow can then be converted to the channel width required to satisfy the rate. The steps involved in routing a flow are as follows:

1. The source node constructs a 'route request' (RREQ) message containing the destination node's IP address and the channel width required for the flow, *reqdWidth*. The source then broadcasts the RREQ message using all allowed channel widths on all channels. This is done to ensure that every node receives the RREQ message irrespective of the channel and the width being used.

2. Every intermediate node receiving the RREQ performs an admission control mechanism, as follows. Let c be the current fixed radio channel used by a particular intermediate node.

   (a) The node checks to see if the channel c is used by any of the nodes within its two hop neighborhood (interfering nodes).

      i. If the channel c is in use, the node scans through all the channels until a free channel that is not used by any of the interfering nodes is found.

ii. If no free channel can be found, the node drops the RREQ message.

(b) The node then checks to see if the current channel width is sufficient to admit the flow.

i. If the current width is not sufficient, the node sees if the channel width can be expanded. This depends on whether there is sufficient spectrum in the adjacent channels that is unused.

A. If a channel width expansion is not possible and not all channels are tried, then repeat Step 2(a) with c being a channel that is not tried yet.

B. If all the channels have been tried and none of them have sufficient spectrum, drop the RREQ message.

ii. If the channel width expansion is possible, the node records the required channel width for this flow.

(c) If the RREQ is not dropped, the node increments the hop count in the RREQ headers. Additionally, if the node is required to switch to a different channel for accommodating this flow, it also increments a counter, C_{sw}, in the header. Thus, C_{sw} is the number of nodes that are required to switch their channels in a route. The node then forwards the RREQ.

3. A destination node may receive multiple RREQs. After receiving a fixed number of RREQs, the destination sorts the routes in the increasing order of the routing metric, $h + C_{sw}$. The destination then sends out a 'route response' message including the *reqdWidth* from RREQ via the first route in the sorted list. The destination then starts a timer expecting a data packet from the source.

4. An intermediate node receiving the RREP checks to see if the required channel width can still be satisfied, as the availability of channel may have changed since the RREQ has been forwarded.

(a) If there is not sufficient channel spectrum, the node just drops the RREP.

(b) If sufficient channel is still available, the node forwards the RREP and changes its channel and channel width (and informs other neighbors of the channel change using a broadcast message), if required, as recorded previously.

5. If the timer at destination node expires without receiving any data packets, it re-sends another RREP via the next best route, until all the available routes are exhausted and restarts the timer.

6. The source node, upon receiving the RREP, starts sending the data packets. If no RREPs are received, the source can re-send the RREQ message with either the same rate requirement or a reduced rate requirement.

When a flow ends, every node involved in the routing reduces its channel width by the amount it expanded for admitting this flow, until the minimum allowed channel width is reached. An end of a flow can be detected when there are no packets available to be forwarded with the same flow identity for a certain duration of time.

5.4 Simulation Results

We developed a simulator using MATLAB to evaluate our protocol. Our simulation setup consists of 200 nodes distributed uniformly at random over a 150 m x 150 m network. The transmission range for each node is set as 30 m and the interference range is chosen to be twice of the transmission range (60 m). The total spectrum available for allocation is 240 MHz, which corresponds to twelve 20 MHz channels. Each node is allocated a 20 MHz channel, chosen uniformly at random from the 12 different possibilities. The nodes are allowed to expand or contract its channel width as chosen from the set {5, 10, 15, 20, 25, 30, 35, 40} MHz. The maximum data rate at which node can transmit with a 20 MHz channel width is set as 54 Mbps. The maximum data rates at other channel widths are scaled accordingly. Thus, the data rate at 5 MHz is 13.5 Mbps, and at 40 MHz is 108 Mbps.

We generated a number of flows, each with a different source-destination pair chosen uniformly at random from the set of nodes generated. The number of flows generated is varied from the set {25, 50, 100, 150, 200}. Each

flow generated chooses a rate requirement uniformly at random from the set {13.5, 27, 40.5, 54, 67.5, 81, 94.5, 108} Mbps. These rates correspond to the allowable channel widths discussed earlier. The flows are started one at a time and last until the simulation ends. For each set of flows generated, we measure the following metrics averaged over 50 runs:

- **Average number of flows accepted:** This is the average across the number of flows admitted in each run.
- **Average data rate of flows accepted:** This is the average of total rate requirement of all the admitted flows divided by the number of admitted flows in each run.

We compared the above two metrics for four different schemes:

1. **Fixed Width Restricted (FWR):** In this scheme the channel widths are not changed from the fixed 20 MHz. Any flow requesting a rate greater than 54 Mbps is not admitted. A flow requesting a rate of at the most 54 Mbps is admitted if the requirement can be fully satisfied.
2. **Fixed Width (FW):** In this scheme, similar to FWR, the channel widths are fixed at 20 MHz. However, a flow requesting more than 54 Mbps may also be considered for admission. When the requested rate is more than the available spectrum at a node, then the flow is admitted at whatever rate is possible with the available spectrum (unless the rate possible is zero).
3. **Variable Width (VW):** According to this scheme, the channel width is adapted, if required, to incorporate a new flow requiring more than the available spectrum at a node (up to a maximum of 40 MHz). However, when a channel width expansion is not possible in the current channel, the nodes will not search for a different channel. In other words, the Step (2.(b).i.A) of the protocol in Section 5.3.1 is not performed.
4. **Variable Width and Channel (VWC):** This scheme follows all the protocol steps, including the Step (2.(b).i.A), described in Section 5.3.1.

5.4.1 Effect of Number of Flows

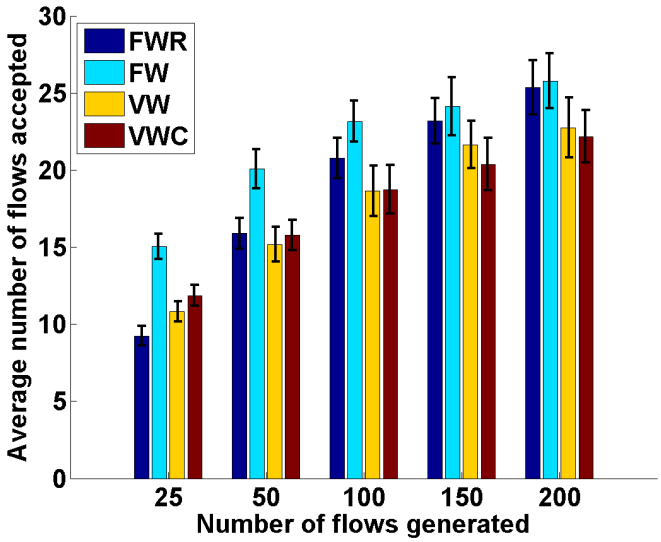


Figure 5.3: Average number of accepted flows.

First, we compare the number of accepted flows and the data rate of the accepted flows for all the four schemes for different number of flows generated. Figure 5.3 shows the corresponding plot for the accepted flows. We observe from this figure that the FW scheme always admits the highest number of flows. This is because the FW scheme does not guarantee the required data rate of the flows and admits them if a non-zero rate can be provided. The FWR scheme performs better than the variable width schemes, except when the number of flows generated is 25. The FWR scheme admits a flow only if its rate requirement can be satisfied by a fixed width channel. Accordingly, only flows that have a requirements of at most 54 Mbps are even considered for admission. The variable width schemes (VW and VWC), on the other hand, expand the channel width, if required, to admit a flow. Because a channel is not re-used within a two hop neighborhood, expanding a channel leaves less spectrum available for other flows when compared to the fixed width schemes.

We also observe from Figure 5.3 that VWC admits fewer flows than VW when the number of flows is large. To illustrate this difference, consider the example network shown in Figure 5.4, where every node is within two hops from every other node. Let there be 5 channels for allocation. The spectrum

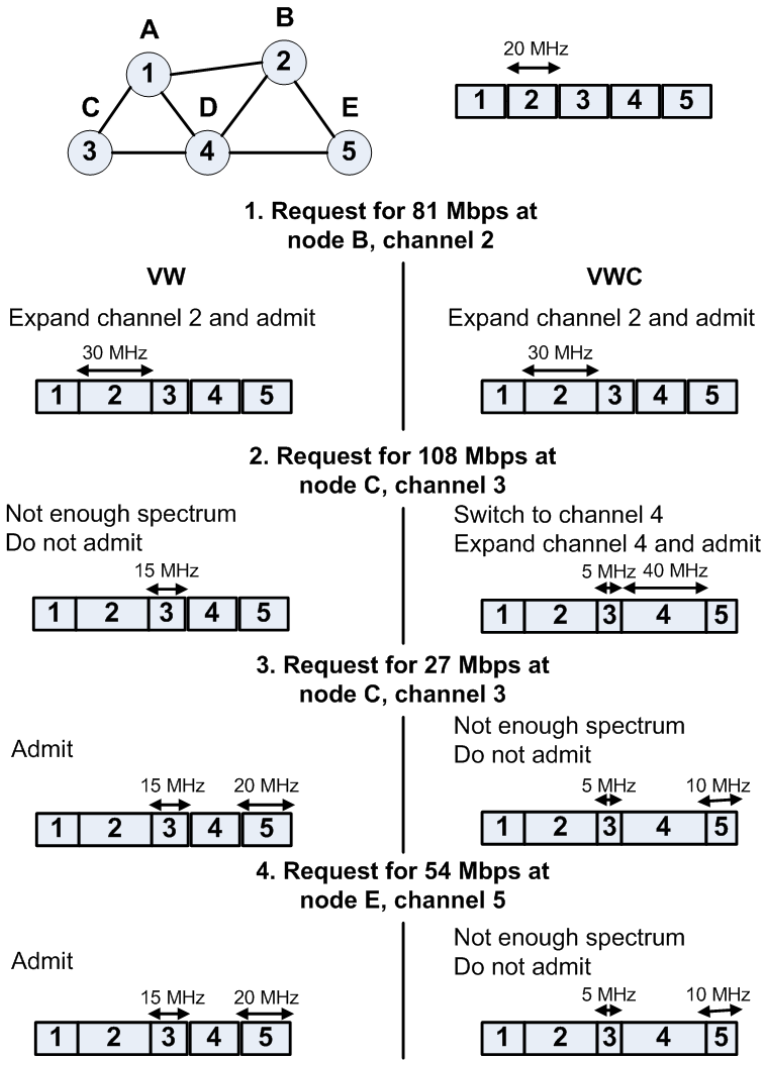


Figure 5.4: Illustration to compare number of accepted flows in VW and VWC.

is shown adjacent to the network in Figure 5.4. Initially, all channels are of equal width of 20 MHz. First, let there be a request from a flow for rate 81 Mbps at node B, allocated to channel 2. This flows requires a channel width of 30 MHz. Therefore, both in the case of VW and VWC, channel 2 is expanded to admit this flow. Next, there be a request for 108 Mbps (requiring 40 MHz) at node C, which is on channel 3. In the case of VW, this flow will not be admitted as there is not enough spectrum and as channel 3 cannot be expanded any further, symmetrically. VWC, on the other hand, searches for a different channel when a width expansion is not possible in the current

channel. It therefore switches to channel 4, expands the channel to 40 MHz, and admits the flow. Later, when there are subsequent requests for channels 3 or 5, VWC will not be able to admit these flows if the requests are greater than the available spectrum. Thus, as in steps 3 and 4 in Figure 5.4, the flows requesting 27 Mbps (requiring 10 MHz) of channel 3 and 54 Mbps (requiring 20 MHz) of channel 5 are not admitted by VWC, while they are admitted by VW. The total number of flows admitted by VW in this illustration comes up to 3, while VWC admits only 2 flows. However, this scenario is more likely to happen when there are many flows in the network as is evident from Figure 5.3.

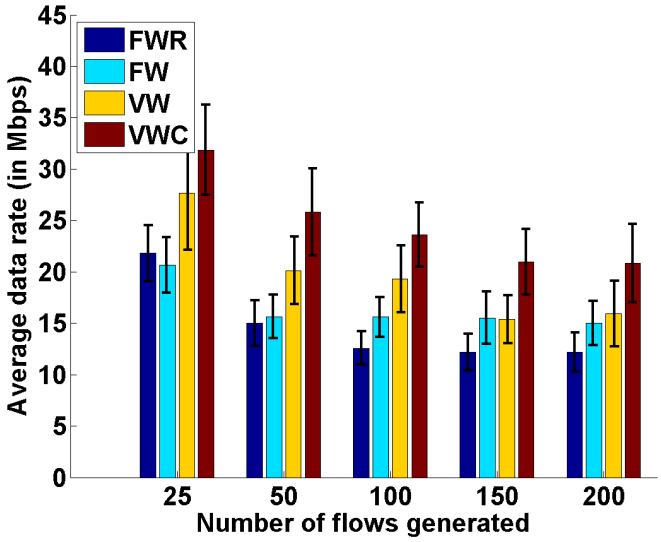


Figure 5.5: Average data rate of accepted flows.

The fixed width schemes, though admitting more flows than the variable width flows, have a smaller net data rate of admitted flows than the variable width schemes. This is shown in Figure 5.5, where we plot the total data rate of all admitted flows divided by the number of admitted flows. Here, we observe that VW and VWC always have a higher average data rate than FW or FWR. This is because the FWR scheme only admits flows that have lower data rate requirements, while the FW scheme can only offer a data rate of at most 54 Mbps to any flow. We also observe that, except for the case of 25 flows, FW always has a higher average data rate than FWR, as FW can even admit flows that request more than 54 Mbps. In the case of 25 flows, we found that the total rate of all admitted flows for FW is higher

than that of FWR. But, because FW admits more flows than FWR, the ratio of the total rate to the number of admitted flows for FW turned out to be slightly smaller than FWR. Next, we observe that VWC has a higher data rate than VW, as VWC can accommodate more high rate flows by switching channels than VW. This is also evident in the previous illustration in Figure 5.4, where the total rate of admitted flows for VWC is 189 Mbps and is just 162 Mbps for VW. Finally, we observe that the average data rate decreases as the number of flows increases in the network, since the spectrum available per flow decreases.

5.4.2 Effect of Percentage of Flows Requesting Lower Rates

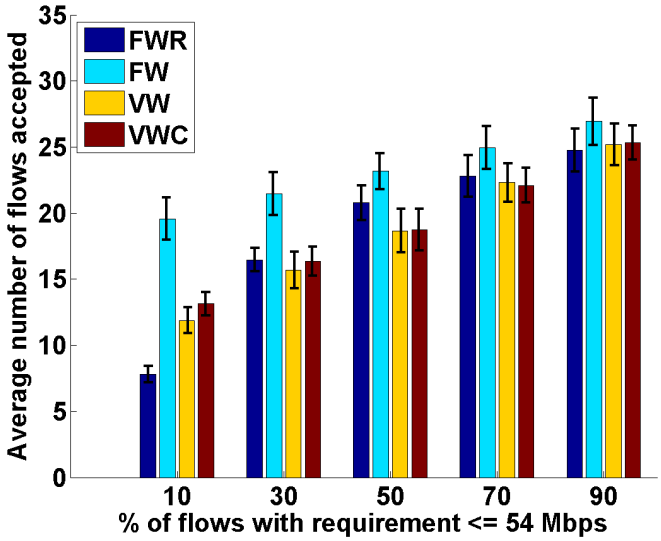


Figure 5.6: Average number of accepted flows with different percentage of low rate flows.

In the previous simulation scenario, the rate requirements for the flows are selected uniformly at random from the possible data rates. Now, we study the effect of varying the distribution of the rate requirements. For this we generated 100 flows in the network, and varied the percentage of flows that request a rate smaller than or equal to 54 Mbps (henceforth termed as low rate flows) from 10 to 90 (in steps of 30). We observe from Figure 5.6 that VW and VWC schemes admit more flows than FWR, when there are only

10% of low rate flows in the network. However, this trend quickly changes as evident from FWR admitting more flows starting from 30% case. The FW scheme still admits more flows than any other scheme as it does not guarantee the rate requirements. Finally, as more low rate flows are in the network, the number of admitted flows in the case of FWR gets almost close to that admitted in VW and VWC schemes, as seen in the plot for the 90% case.

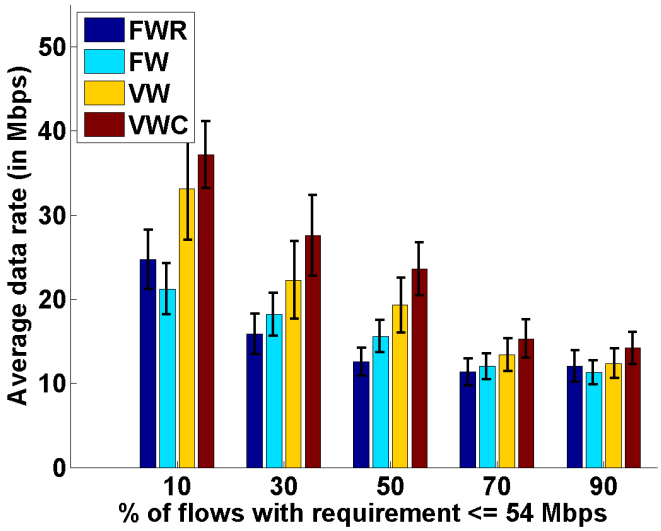


Figure 5.7: Average data rate of accepted flows with different percentage of low rate flows.

The average data rate values shown in Figure 5.7. As expected, the VW and VWC schemes have higher average data rates than the fixed schemes. But, the average data rate values become almost the same for all the schemes as the percentage of low rate flows approaches 90%.

5.4.3 Effect of Number of Nodes

In the next of simulations, we compare the performance of the protocols with different number of nodes in the network. For this, we varied the number of nodes in the network from 100 to 350 (in steps of 50). The area of the network is held to be the same at 150 m x 150 m. We generated 100 flows in each case, and the distribution of the rate requirement for the flows is left

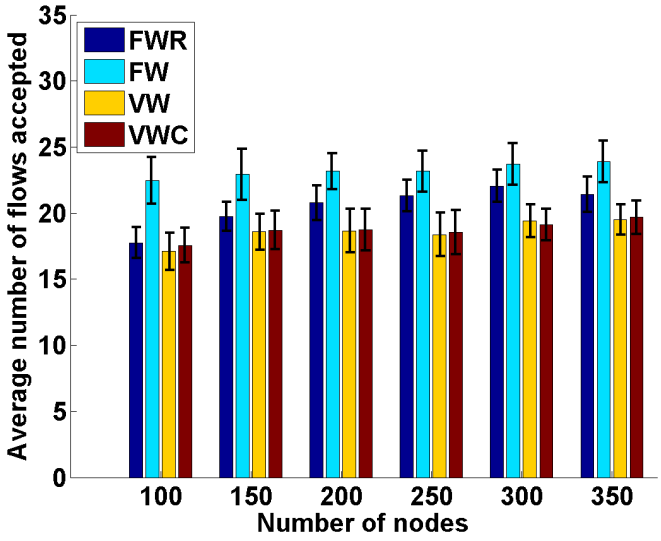


Figure 5.8: Average number of accepted flows with different number of nodes.

to be uniform across the possible rate values. We observe from Figure 5.8 that the number of accepted flows increases only marginally as the number of nodes increases. This is because the number of flows generated is 100 in all the cases. We can also see from Figure 5.9 that the average data rate increases with the number of nodes. Because the number of accepted flows increases only marginally, as the number of nodes increases, more high rate flows are accepted resulting in an increase in the average data rate. This can be illustrated using the following scenario. Consider the case where there are 150 nodes distributed uniformly over the 150 m x 150 m area, and the case where 300 nodes are distributed uniformly over the same area. For simplicity of discussion, let us assume that the transmission regions are square (instead of a circle). If we consider a transmission area of 30 m x 30 m (note that 30 m is the transmission range of the nodes), there are on an average 6 nodes in this area for the 150 node case, and 12 nodes in this area for the 300 node case. Note that there are twelve, 20 MHz channels, and a flow requiring 40 MHz of channel requires two 20 MHz channels. Therefore, for the same number of accepted flows, the 300 node scenario can accept as many large rate flows as the small rate flows accepted by the 150 node scenario (6 flows in this example). This results in the 300 node case having a larger average data rate than the 150 node case. However, there is no increase in data rate

as the number of nodes increases beyond 300, (which can be observed from the bars corresponding to 350 nodes) as the maximum number of number of channels used is only 12.

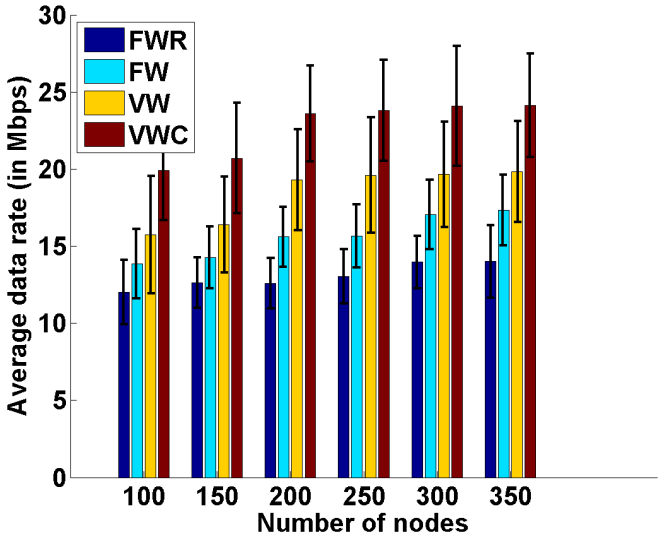


Figure 5.9: Average data rate of accepted flows with different number of nodes.

5.4.4 Effect of Number of Channels

In the final set of simulations, we vary the default spectrum width per channel, while keeping the total available spectrum constant at 240 MHz. This will result in fewer than 12 channels when the width is greater than 20 MHz, or more than 12 channels when the width is smaller than 20 MHz. The number of nodes in the network is set as 200, and we generated 100 flows with their requirements distributed uniformly at random from the possible rate values. We compared the performance of the protocols across per channel spectrum widths of 5 MHz, 10 Mhz, 20 Mhz, and 40 Mhz. This results in 48, 24, 12, and 6 channels respectively. Because comparing the fixed width schemes at a smaller channel width is unreasonable, we only compare the variable width schemes in this simulation.

First, we observe from Figure 5.10 that the number of accepted flows decreases as the number of channels decreases (or as the per channel width

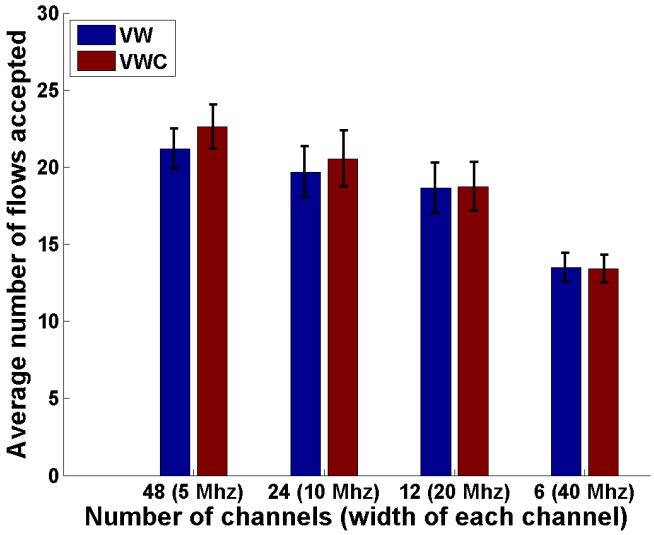


Figure 5.10: Average number of accepted flows with different number of channels.

increases). This is because more channels allow for more flexibility, allowing for more flows to be admitted. However, we observe from Figure 5.11 that the average data rate increases as the per channel width increases, as a wider channel allows more high rate flows to be admitted readily. However, the data rate drops at 40 MHz channel width in the case of VWC. This is because the behavior of VWC becomes similar to that of VW, in that a channel expansion is never performed (as the maximum allowed channel width is limited to 40 MHz in our simulations). Therefore, the additional benefit in data rate obtained by switching channels when required is not available anymore.

We also attempted a simpler version of the channel width adaptation protocol with a distance-based propagation model that does not account for interference between nodes. The protocol uses a different traffic model and routing metric. We have summarized the protocol and the simulation results for this variant in the Appendix A.

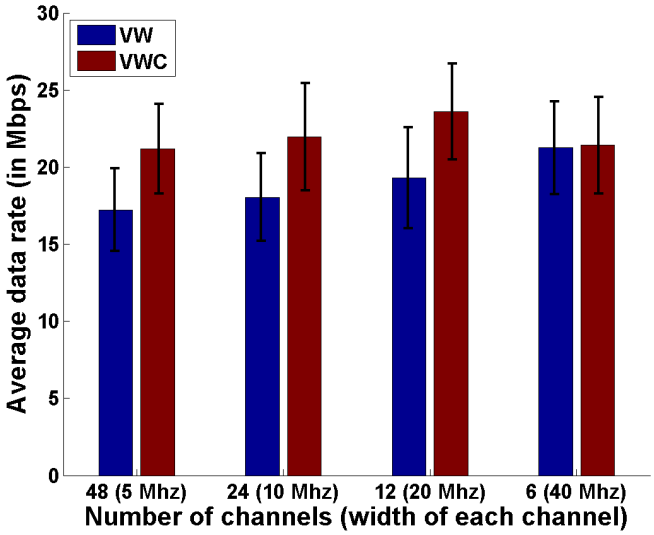


Figure 5.11: Average data rate of accepted flows with different number of channels.

5.5 Discussion

In this chapter, we explored the concept of variable width channel allocation in a multi-hop, multichannel wireless network to satisfy a flow's rate requirement. Such a scheme may be beneficial in a wireless sensor network, where a sensor node can decide whether the available spectrum will be sufficient to send an amount of data at a desired rate. We proposed a channel width adaptation protocol executed during routing and studied its performance using extensive simulations. Our simulations show that there can be a significant throughput advantage in adopting a variable width channel allocation scheme. Furthermore, our results show that determining the channels dynamically, in addition to the channel widths, can be more beneficial.

While variable width channels are supported by present day wireless hardware, there are some complexities that need to be considered while implementing our protocol:

1. Our protocol requires that a source node has an estimate of the data rate required by the flow to be routed. While some applications, such as VoIP or video, may have this information as part of the protocol headers, in general, additional mechanisms are needed for this estimation.

2. Every node in the network has to be aware of the channel and the width used by its neighbors. Exchanging this information during routing may incur route setup delay. Furthermore, broadcasting this information periodically on channel and on all widths may result in additional overhead.

3. Switching channels or changing channel widths may incur a non-negligible delay, both at the hardware and software level. These delays should be considered as part of the route setup time, which may require a more sophisticated routing metric.

5.6 Future Work

Our simulation results in this section are intended to evaluate the potential benefits of channel width adaptation. Future work can address many other challenges. Here, we enumerate a few of them:

1. We assumed that a channel cannot be reused by any of the two hop neighbors. This allows for the use of simple access protocols. However, it will also be beneficial to study the performance of the protocols by allowing for time sharing of the spectrum.

2. The routing metric we considered does not consider the effect of multiple channels. It may therefore be beneficial to analyze the protocols using more sophisticated routing metrics, such as MCETT [12].

3. Finally, with the possibility of commodity hardware supporting variable widths in the future, it may be beneficial to implement our protocols on a testbed to measure the performance with real traffic.

CHAPTER 6

CONCLUSION

In this thesis, we have presented a methodology for spectrum management in multichannel wireless networks. Our approach deals with exploiting the flexibilities either available in existing commodity hardware or expected in future devices, with respect to adapting the spectrum widths dynamically. We have developed channel allocation and routing protocols for dynamically varying the widths based on traffic characteristics and demand. We have also identified the practical difficulties that need to be addressed while implementing the multichannel protocols with a perspective on minimizing system latencies for real-time applications, and have presented algorithms for addressing them. We now provide a brief chapter-level summary in the following paragraphs.

In Chapter 3, 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 an algorithm called WiSP for determining the channel partitions. We have studied the performance of our algorithm using simulations and showed that our algorithm can provide significant improvements even in cases where frame aggregation performs poorly. Furthermore, we also show results where our mechanism can be used with frame aggregation to obtain significant performance benefits. We have also presented results for a multiple network scenario where the APs are allocated variable width spectrum proportional to their client load, and show that our protocol provides a performance that is better than just doing frame aggregation.

In Chapter 4, we proposed SHORT, a routing approach that exploits the benefits of both static and hybrid channel allocation strategies. We have implemented the protocol on a real multichannel testbed, and using experimental data, we have demonstrated the performance benefits of the SHORT protocol over a hybrid channel allocation protocol, called HMCP. All our

experimental results illustrate the abilities of SHORT protocol in providing low delay multihop paths for real-time traffic, while not significantly affecting the throughputs of non real-time traffic. Our results show that the delay performance of SHORT protocol is comparable to that of a static channel allocation method.

In Chapter 5, we discussed a method for provisioning spectrum in a multichannel wireless network by adapting spectrum widths of wireless channels depending on the traffic demand. We proposed a joint algorithm that performs routing and channel resource allocation with the goal to maximize the overall data rate of the admitted flows. Our algorithm uses a modified AODV routing protocol to coordinate allocation decisions across wireless nodes. Using simulations, we evaluated the performance of our algorithm by comparing it with other alternatives. Our results show that allocating both the channels and the channel widths dynamically during routing provides the best performance in terms of the network wide data rate of the admitted flows.

The protocols proposed in this dissertation can be helpful in realizing a system that can efficiently utilize the wireless resources.

APPENDIX A

VARIABLE WIDTH CHANNEL ALLOCATION WITH DISTANCE-BASED PROPAGATION MODEL

Before evaluating the protocol proposed in Chapter 5, we evaluated another version of the protocol with a different traffic model. This variant of the protocol does not consider interference between nodes that are allocated the same channel. We do, however, ensure that a channel does not overlap with its neighboring channel when its width is expanded, as discussed in Section 5.2.1. Here, we summarize the model and the simulation results of this variant of the protocol.

A.1 Traffic Model

We consider two traffic classes in the network, namely real-time flows and best-effort flows. Real-time flows are those that require end-to-end rate guarantees, and best-effort flows are those that do not require such guarantees. Examples of real-time flows include VoIP, streaming video, interactive gaming, etc. Traffic generated by web browsing, file transfer, and HTTP sessions is classified as best-effort traffic. We assume that the rate requirements of the real-time flows are known at the routing layer, which can later be converted to an appropriate rate requirement. Any of the existing literature on estimating the rate requirements of a flow can be used for this purpose [51, 52].

In this variant of the protocol, a node attempts to admit as many real-time flows as possible such that their rate requirements are satisfied. To accommodate a real-time flow, a node performs an admission control mechanism where it first checks to see if there is enough channel resource to accommodate the flow. If not, it checks to see if a channel width expansion is possible. If the rate requirement of the real-time flow cannot still be satisfied, it drops any of the best-effort flows that it is currently serving one after another until there is enough spectrum to admit the flow. A real-time will not be admitted

if its rate requirement cannot be satisfied even after dropping all the best-effort flows. Note that our protocol does not guarantee the rate requirement of the best-effort flows.

The objective of our protocol is to maximize the number of real-time flows that are admitted while minimizing the number of best-effort flows dropped.

A.2 Protocol Architecture

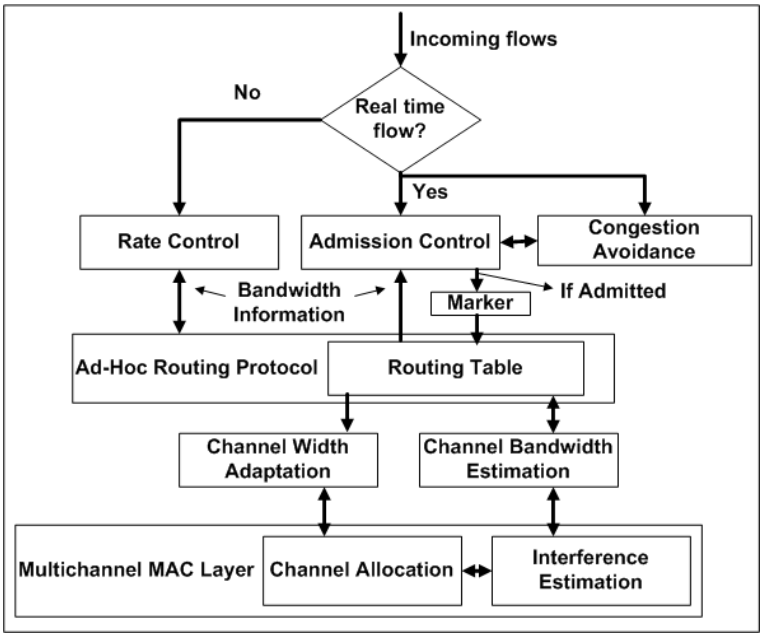


Figure A.1: Proposed protocol architecture.

The protocol architecture is motivated by that in [43] and is modified to accommodate multiple channels and variable channel widths. The protocol architecture is shown in Figure A.1. When an incoming flow arrives at a node (either source or an intermediate node), a packet classifier determines whether it is a real-time or a best-effort flow. If it is a real time flow, the packets go through an admission control mechanism. If not, the packets are sent to a rate-control mechanism. The admission controller is responsible for estimating the rate requirement of the flows and for deciding whether the flow can be admitted based on the spectrum used by any of the the existing flows (obtained from the routing table) and the channel widths used by the

neighboring nodes (obtained from the interference estimation function in the MAC layer). Once the real-time flows are admitted, the packets are marked as real-time (by setting a flag) and are sent to the routing layer for route selection. The transmission channel and the channel width to use (based on the flow arrival) are also determined as part of the route selection process.

In the case of a best-effort flow, the rate controller chooses the maximum rate possible for the flows after discounting the spectrum used by any real time flows in the channel. If there is no spectrum is available for the best-effort flows (which can happen when entire available spectrum is being utilized by real-time flows) at an intermediate node, the flow requests are dropped (not shown in the figure). This is an attempt to ensure that the real-time flows are not affected by a potential bandwidth hungry best-effort flow (such as a large file transfer). An ongoing best-effort flow may also be dropped (and the source informed of this), when a new incoming real-time flow requires the channel. Whenever a best-effort flow is dropped, the source is informed about this so that it can initiate a new route discovery for finding an alternative route.

A.3 Proposed Mechanisms

In this section, we explain in detail the various components that form our protocol architecture. We start with a discussion on the channel allocation algorithm, followed by the routing protocol and the routing metric.

A.3.1 Channel Allocation Algorithm

We assume that five channel are available, with initial channel widths of 20 MHz. A channel width expansion is performed as explained in Chapter 5, Section 5.2.1. The algorithm described here is a variant of our earlier work discussed in [53]. We replicate the salient features of the algorithm from [53] for completeness. The main difference here is that the channel usage information exchanged between the nodes contains the width of the channels used by the nodes, in addition to their center frequencies. An IEEE 802.11a MAC is assumed for explaining the algorithm in this section. However, the algorithm itself is more general. The channel allocation algorithm decides

the center frequencies of the channels and is run periodically at every node in a distributed fashion. According to this algorithm, every fixed interface at a node initially starts up on `currChan`, which is one of the `K` channels used for allocation. The nodes then exchange their chosen channels (center frequencies) and the spectrum usage of each of the channels, $recvbwUsage$, with their one and two hop neighbors, `neighList`, using broadcast `hello` messages. The spectrum utilization of a channel is essentially the amount of channel spectrum used by a flow transmitted on that channel. Because the nodes in the neighborhood may be listening on different channels and widths, the broadcast of the `hello` has to be performed on all the channels and possible widths (by switching the transmit radio on each of these channels and widths) to ensure that all the nodes in the neighborhood receive the information. Every node, after receiving the broadcast messages, counts the number of one and two hop neighbors that are assigned a particular channel, denoted $chanCount$, and calculates the fraction of spectrum bandwidth used on each of the channels, denoted $bwUsage$, from the $recvbwUsage$ values in the broadcast message. The nodes then calculate a weighted sum of the number of nodes using a particular channel as well as the fraction of spectrum bandwidth used on each channel, called the 'load' on that channel, denoted by $load$. The nodes also calculate the average, $meanLoad$ load across all the channels. A node then probabilistically (with a probability p) decides to switch its channel if the load on the current channel, $currChan$, is at least one above the $meanLoad$. The current channel is then switched to one of the channels that has the minimum load, denoted by the list `minChanList`. While picking the channel to switch from `minChanList`, our algorithm attempts to pick the spectrally farthest channel, `maxDistChan`, from the ones that are in use by its neighbors. This is done to allow for future channel expansions when needed. The spectral distance of a channel i, is given by

$$distance_i = \sqrt{\sum_{\substack{j \in neighChannels, \\ j \neq i}} (i-j)^2}, \ \forall i \in K \tag{A.1}$$

where $neighChannels$ is the set of channels assigned to the one and two hop neighbors of a node and K is the set of all channels that are considered for allocation. The spectrally farthest channel i is then given by $\arg\max_i \{distance_i\}$. The channel allocation and selection algorithms are

shown in Algorithm 2.

A.3.2 Routing Metric

The routing metric used by our approach is a variant of the multichannel routing metric (MCR) [10] and the expected transmission time (ETT) metric [35]. The ETT, in seconds, is specified per link and is given by $ETT = ETX * \frac{S}{D}$, where ETX [35] is the expected number of transmission attempts (including retransmissions) required for transmitting a packet, S is the average packet size, and D is the data rate of the link. The data rate D of the link depends on the channel width chosen on that link. The proposed routing metric combines the ETT metric with the hardware delay involved in switching the channel, namely C_{sw}, and the delay involved in adapting the channel width, namely C_{bw}. Additionally, we associate two penalty metrics, C_{drop} and C_{demand}. C_{drop} is the number of best-effort flows to be dropped for admitting a real-time flow, and C_{demand} is called the demand factor, which is used to keep track of number of route requests that a node received over a certain duration of time. The details of these two penalty metrics are discussed in Sections A.3.4 and A.3.5. If c_i is the channel used in the i-th hop of route and w_i is the corresponding channel width used, the end-to-end routing metric for a path involving h hops, namely QOSAR (QoS-based ad-hoc routing metric) is given by

$$QOSAR = \sum_{i=1}^{h} [ETT(i) + C_{sw}(i) + C_{bw}(i) + C_{drop}(i) + C_{demand}(i)]$$

In their definition of the MCR metric, the authors of [10] introduce a factor for the interference in the network. We ignore that factor in our definition for simplicity.

A.3.3 Routing Protocol

We modify the popular AODV routing protocol [11] for determining the routes based on the rate requirements of the flows. As mentioned in Section A.1 we assume that we can estimate the rate requirement of the flows

Algorithm 2 Channel allocation algorithm.

ChanAlloc(`currChan, K, neighList`):

1. for i in `K`
2. $chanCount[i] \leftarrow 0$
3. $bwUsage[i] \leftarrow 0$
4. end for
5. for $neigh$ in `neighList`
6. $chanCount[neigh \rightarrow channel] + +$
7. $bwUsage[neigh \rightarrow channel] \leftarrow$
8. $bwUsage[neigh \rightarrow channel] + recvbwUsage[neigh]$
9. $//recvbwUsage[j]$ is the received bandwidth usage from neighbor j
10. end for
11. $sumLoad \leftarrow 0$
12. for j in `K`
13. $load[j] \leftarrow chanCount[j] * bwUsage[j]$
14. $sumLoad \leftarrow sumLoad + load[j])$
15. end for
16. $meanLoad \leftarrow sumLoad/$`K`
17. if($load[$`currChan`$] \geq meanLoad + 1$)
18. With probability $p = \frac{1}{chanCount[\texttt{currChan}]}$
19. // create list, `minChanList` of channels that have the $minLoad$
20. `currChan` $\leftarrow$ ChanSelect(`minChanList, neighList`)
21. end if
22. return `currChan`

ChanSelect(`minchanList,neighList`):

1. for i in `minchanList`
2. $sumDist \leftarrow 0$
3. for $neigh$ in `neighList`
4. $sumDist \leftarrow sumDist + [i - (neigh \rightarrow channel)]^2$
5. end for
6. $distance[i] \leftarrow \sqrt{sumDist}$
7. end for
8. `maxDistChan`$\leftarrow \max_i\{distance[i]\}, \forall i \in$ `minChanList`
9. return `maxDistChan`

from its traffic pattern using existing techniques in the literature, such as [51, 52]. We also assume that the rate requirement is known at the routing layer, which can be converted into the actual channel width required to satisfy the rate requirement. The routing protocol described here is common to both real-time and best-effort flows. The steps involved in routing a flow are as follows:

1. The source node constructs a 'route request' (RREQ) message containing the destination node's IP address. For real-time flows, the source node sets a special flag, $isRealTime$ to TRUE and includes the rate required for the flow. The source then broadcasts the RREQ message on all channels and widths.

2. Every intermediate node receiving the RREQ checks the $isRealTime$ flag. If it is set, then it performs admission control (see Section A.3.4). If the flow is not admitted (i.e., the rate requirement of the flow cannot be satisfied by this node), then the node just drops the RREQ message. If the real-time flow is admitted, then the node adds the following costs to the RREQ message as part of the routing metric discussed earlier in Section A.3.2:

   - The ETT for every outgoing channel.
   - The cost for switching the channel, C_{sw} for each channel.
   - The cost to adjust the channel widths, C_{bw} for each channel.
   - The penalty metrics, C_{drop} and C_{demand} for each channel.

   The node then broadcasts the RREQ message (on all channels). In case of a best-effort flow, the nodes do not add the penalty metrics.

3. The destination node, upon receiving the RREQ messages (note that the destination node may receive multiple RREQs), compares the routing metrics across the received messages and generates a 'route response' (RREP) message for the route that has the least metric.

4. The destination node may choose to respond to multiple RREQ messages if a newly received RREQ has a smaller metric than the one for which it may have replied earlier.

5. The RREP message is forwarded to the source along the path used by the corresponding RREQ. The intermediate nodes also add a routing table entry for the flow. Additionally, if it is a real-time flow, the intermediate nodes check if any best-effort flow has to be dropped, as discussed in Section A.3.4.

A.3.4 Admission Control Mechanism

The proposed admission control mechanism is executed at a node to determine whether or not the rate requirement of an incoming flow can be satisfied on any of the links. In the discussion that follows, the available channel spectrum at a node is computed based on the amount of spectrum used by the neighbors, and the amount of spectrum used by any existing flows in the node. The available spectrum information of the neighbors can be obtained by exchanging, periodically the channel widths with the neighbors. Because there is no rate requirement associated with a best-effort flow, a node discounts 5 MHz (which is the smallest allowed channel width in our protocol) from its available spectrum for every best-effort flow that it is currently servicing.

The pseudocode for the admission control mechanism is shown below. If the available spectrum can satisfy the requirement of the incoming flow, then the new flow is admitted. If not, the node marks any best-effort flows, that are currently serviced, for dropping one after another (in some random order) and adds 5 Mhz to the available spectrum for every best-effort flow marked. Thus, the available spectrum in the Step 2 of the pseudocode below is the spectrum that will be available if the marked best-effort flows are eventually dropped. This is repeated until the available spectrum becomes sufficient for the new flow. If there is not sufficient spectrum even after all the best-effort flows are marked for dropping, the incoming flow is rejected (and the best-effort flows that were previously marked are unmarked). Otherwise, the flow is accepted and the marked best-effort flows are dropped.

At a node with an incoming flow,

```
1.  If available spectrum ≤ required spectrum
2.  Repeat until all best-effort flows are dropped:
      Mark a best-effort flow for dropping
```

```
      If available spectrum ≥ required spectrum
       Drop all the marked best-effort flows
       Admit the flow
       Exit
    end repeat
3.  Reject the flow
      Unmark all the marked best-effort flows
```

Admitting a flow at a node by itself may not guarantee that the flow will be eventually sent through this node. The decision on whether or not a node is chosen for forwarding a flow depends on the routing protocol used.

A.3.5 Note on Demand Factor

We will now explain how the demand factor in the $QOSAR$ metric is used for congestion control. Consider that a certain node, say node A, receives a RREQ with a rate requirement b_{f1} for a flow f_1. Let the available spectrum at node A be b_a and let $b_{f1} < b_a$. Node A will therefore broadcast the RREQ message along with spectrum b_a and other costs. Before this response propagates to the source of this flow, let us assume that another request arrives at A from a different source requiring a spectrum of $b_{f2} < b_a$ for a flow f_2. In this case, node A can either choose not to rebroadcast the RREP as it has already responded with its available spectrum for the flow f_1, or it can respond with a spectrum that is smaller than b_a. However, flow f_1 may not choose the route via node A. If f_1 did not choose the route via A, then dropping the RREP for f_2, or advertising a smaller available spectrum to f_2 may result in the flow f_2 not using the route via node A. As a result, a potentially usable route may end up being unused. To overcome such a situation, we propose to attach a demand factor, C_{demand}, to the routing responses, which is simply the number of routing requests at a node. Therefore, in the example above node A responds with the same spectrum b_a for the flow f_2, but includes a C_{demand} value that is incremented by 1 to account for the flow f_1. Because a node having a high demand factor can be a potential bottleneck node, including the demand factor information in the routing metric can help reduce, to some extent, the chance of multiple flows being routed through the same node.

A.4 Simulation Results

In this section we start by giving an overview of our simulation methodology in Section A.4.1, and then describe performance results in Section A.4.2.

A.4.1 Simulation Model

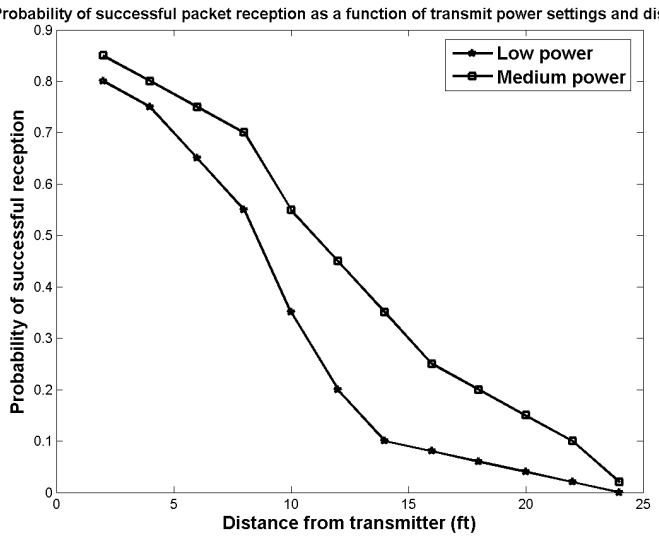


Figure A.2: Probability of successful packet reception as a function of distance and power.

In our event-driven simulations, we place 2500 nodes distributed randomly on a 150 m $\times$ 150 m area. The transmission range of the nodes is fixed at 30 m. Every node is equipped with two radios of which one is used for transmitting data and the other is used for receiving data. For modeling the link layer we have used the propagation model from [54], which is based on real world experiments. This model defines the network topology by associating with every link a probability value, which decides whether a transmission on that particular link will be successful or not. This is decided based on a coin toss for every packet sent on a link (including broadcast packets such as those used for route discovery). The probability values are defined as a function of distance and transmission power. We have shown this model in Figure A.2. The plot shows that the probability of successful packet reception decreases as the distance from the packet source increases. Furthermore, the

plot shows that the probability of successful reception of a packet improves as we increase the transmit power.

The real-time flows are introduced into the network at a rate modeled by a Poisson distribution of rate λ arrivals per second. The value of λ is varied from 0.02 to 0.1 arrivals per second in steps of 0.2 arrivals per second, and further from 0.2 to 1.0 arrivals per second in steps of 0.02 arrivals per second. Each of these flows are assumed to have a rate requirement, which is chosen uniformly at random from a set of five possible values, namely {13.5 Mbps, 27 Mbps, 54 Mbps, 81 Mbps, 108 Mbps}. Each of the five channels can be tuned to any of the following five channel widths: {5 Mhz, 10 MHz, 20 MHz, 30 MHz, 40 MHz}. The nodes are allocated channels chosen uniformly at random from the five channels before the start of the simulation. The nodes are also allocated a certain number of best-effort flows before the start of the simulation. The number of best-effort flows in a node is chosen uniformly at random between 1 and 5. Each of the best-effort flows is assumed to consume 5 Mhz worth of spectrum. Accordingly, the initial channel widths of the nodes are adjusted based on the number of best-effort flows. Thus, a node with just a one best-effort flow will have a width of 5 MHz, while the node with 5 best-effort flows will be using a channel width of $5 \times 5 =$ 25 MHz.

We assume that the network, traffic, and propagation characteristics are static for the period of the simulation. For each of the offered load characterized by the arrival rate λ, we run 100 randomly generated networks with the given parameters. The performance results presented are averaged across all the flows and all the 100 network realizations. For each of the runs, we compare the performance of our proposed approach based on the modified AODV protocol using the $QOSAR$ metric and a greedy approach, which is also a modified version of the AODV protocol described below. First, a list of potential routes between the source and destination is formed. This is done using a centralized algorithm knowing the source and destination locations, and the location of all the other nodes. Using this information we create the list of nodes that are located between the source and destination, and compute the set of all possible routes through these nodes. Then the route with the best ETT is picked from the list of potential routes using the following steps: the source node and every subsequent node that receives an RREQ, instead of broadcasting the RREQ, unicasts the message to the neighbor

(chosen from the nodes in the potential routes) with the minimum ETT and chooses that neighbor as the next hop. If the neighbor of a node is the destination, then the node simply forwards the RREQ to the destination. The destination node then sends back an RREP, which is forwarded all the way to the source node, to inform the source node that the route to destination exists. The source node re-sends an RREQ message to the neighbor with the next best ETT, if there is no RREP received within a certain duration of time. Thus in this approach, the route is decided locally at every node, based on the local ETT values, instead of the destination node.

A.4.2 Performance Results

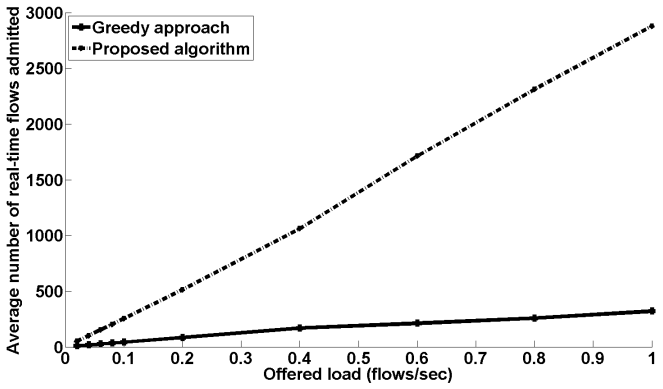


Figure A.3: Average number of real-time flows admitted.

We now discuss the various performance results obtained through simulations. We first plot the number of admitted flows in the network. Figure A.3 compares the number of admitted real-time flows for the greedy and the proposed algorithms. We observe that the difference in the number of admitted flows between our proposed approach and the greedy approach increases as the offered load increases. This shows that fewer flows are admitted in the network in the greedy approach.

Next, we compare the average transmission rate achieved per flow in the network. Figure A.4 shows the plots for the rates achieved per flow for the two algorithms. We can observe from that plot that our proposed algorithm can achieve higher rate than the greedy algorithm approach. For instance,

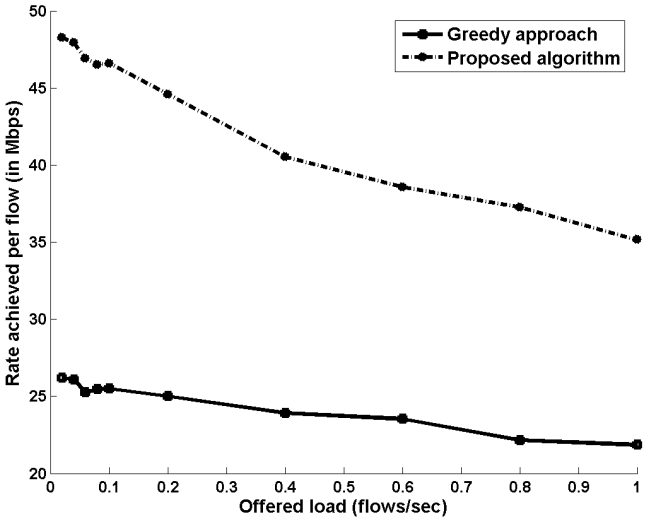


Figure A.4: Average rate achieved per flow.

the rate achieved per flow at an offered load of 0.1 is around 25.2 Mbps in the case of greedy approach, while it is 46.2 Mbps for our proposed approach.

Next, we compare our approach with the greedy approach as the network density is varied. For this purpose, we fix the network size as a 150 $m \times 150\,m$ square as before and the offered load to be 0.1 arrivals per second. However, we vary the number of nodes in the network from 1000 to 5000.

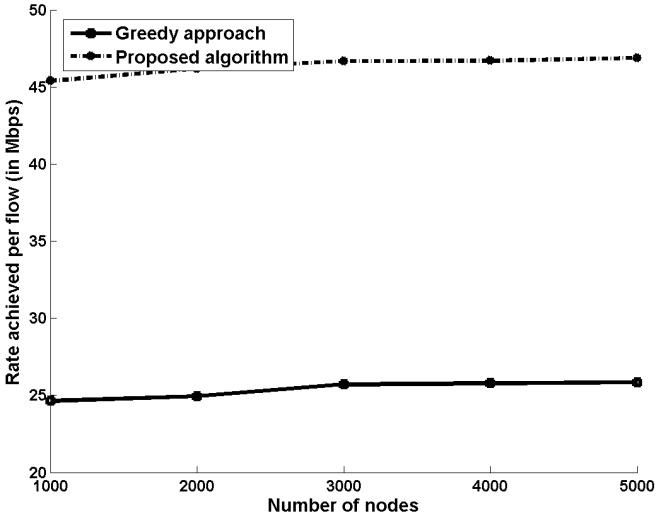


Figure A.5: Average rate achieved per flow as a function of network density.

We plot the rates achieved by the two approaches in Figure A.5 for the various network densities. As before, our approach achieves better average rate than the greedy approach.

A.5 Discussion

In this appendix, we presented a variable width channel allocation algorithm for a simpler propagation model. Our protocol uses the rate requirement of the real-time flows to estimate the required spectrum. During routing, every node decides on whether the real-time flows can be admitted or not base on the spectrum availability. Furthermore, the nodes attempt to admit a real-time flow by dropping any existing best-effort flows that it is currently servicing. Using simple simulations we show that our approach can admit more real-time flows and provide higher data rates for the admitted flows, compared to a greedy approach.

The network model used in our evaluations is simplistic, as we did not consider interference between the nodes that are allocated the same channel. Furthermore, we used a distance-based propagation model to determine if a packet can be received by a node or not. However, the results obtained in this appendix provided valuable insights in understanding the capabilities of channel width adaptation. For instance, our simulations suggest that more flows that request higher data rates can be admitted by adapting the channel widths. Furthermore, our results suggest that a scheme that chooses the route with a global knowledge of the network can be beneficial than a scheme that greedily decides the route based on local information. With this understanding, we have improved our protocol to incorporate joint routing and channel resource allocation, as presented in Chapter 5. Additionally, for the evaluations presented in Chapter 5 we have used a FDMA-based channel allocation and an improved interference model, which overcomes the shortcomings of the simulations in this appendix. Finally, in Chapter 5, we have compared our protocol with a variety of other alternative approaches.

REFERENCES

[1] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A multiradio unification protocol for IEEE 802.11 wireless networks," in *Broadnets Conference*, vol. 1, October 2004, pp. 344–354.

[2] P. Bahl, A. Adya, J. Padhye, and A. Wolman, "Reconsidering wireless systems with multiple radios," *ACM Sigcomm Communication Review*, vol. 34, no. 5, pp. 39–46, October 2004.

[3] P. Kyasanur, C. Chereddi, and N. Vaidya, "Net-X: System extensions for supporting multiple channels, multiple interfaces, and other interface capabilities," August 2006. [Online]. Available: http://www.crhc.illinois.edu/wireless/groupPubs.html.

[4] R. Maheshwari, H. Gupta, and S. Das, "Multichannel MAC protocols for wireless networks," in *IEEE Secon Conference*, September 2006, pp. 393–401.

[5] J. So and N. Vaidya, "Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *ACM MobiHoc Conference*, 2004, pp. 222–233.

[6] K. Ramachandran, I. Sheriff, E. Belding, and K. Almeroth, "A multi-radio 802.11 mesh network architecture," *Mobile Networks and Applications*, vol. 13, no. 1-2, pp. 132–146, April 2008.

[7] X. Yang and N. Vaidya, "On physical carrier sensing in wireless ad hoc networks," in *IEEE INFOCOM Conference*, March 2005.

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

[9] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan, "Load-aware spectrum distribution in wireless LANs," in *IEEE ICNP Conference*, October 2008, pp. 137–146.

[10] P. Kyasanur and N. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *ACM SIGMOBILE MC2R*, vol. 10, pp. 31–43, January 2006.

[11] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing."

[12] C. Chereddi, P. Kyasanur, and N. Vaidya, "Design and implementation of a multi-channel multi-interface network," in *ACM REALMAN Workshop*, vol. 5, May 2006, pp. 23–30.

[13] C. Phillips and S. Singh, "Analysis of wlan traffic in the wild," in *IFIP Networking Conference*, 2007, pp. 1173–1178.

[14] Y. Kim, S. Choi, K. Jang, and H. Hwang, "Throughput enhancement of IEEE 802.11 WLAN via frame aggregation," in *IEEE VTC Conference*, September 2004, pp. 3030–3034.

[15] 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*, vol. 15, no. 1, pp. 40–47, February 2008.

[16] Y. Xiao, "Throughput in wireless LANs," *IEEE Wireless Communications*, vol. 12, no. 6, pp. 82–91, December 2005.

[17] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media access for multirate ad hoc networks," in *ACM MobiCom Conference*, September 2002, pp. 24–35.

[18] 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 Conference*, 2008.

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

[20] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[21] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," in *ACM Sigcomm Conference*, August 2008, pp. 135–146.

[22] *HT MAC*, Enhanced Wireless Consortium Std. Specification v1.24, 2006.

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

[24] G. Al-Suhail, "Impact of packet size on the temporal quality of video transmission over wired-to-wireless network," in *International Conference of Advances in Mobile Computing and Multimedia*, 2008.

[25] Y. Lu, Y. Zhao, F. Kuipers, and P. Meighem, "Measurement study of multi-party video conferencing," in *IFIP Networking Conference*, 2010, pp. 96–108.

[26] D. B. Masi, M. Fischer, and D. Garbin, "Video frame size distribution analysis," *The Telecommunications Review*, pp. 74–86, 2008.

[27] A. Das, R. Vijayakumar, and S. Roy, "Static channel assignment in multi-radio multi-channel 802.11 wireless mesh networks: Issues, metrics and algorithms," in *IEEE Globecom Conference*, 2006.

[28] M. Marina and S. Das, "A topology control approach to using directional antennas in wireless mesh networks," in *IEEE ICC Conference*, June 2006, pp. 4083–4088.

[29] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *ACM Mobicom Conference*, September 2005, pp. 73–87.

[30] W. Wu, A. Arefin, Z. Huang, P. Agarwal, S. Shi, R. Rivas, and K. Nahrstedt, "I am the Jedi! - A case study user experience in 3D tele-immersive gaming," in *IEEE International Symposium on Multimedia*, December 2010, pp. 220–227.

[31] M. Castro, P. Dely, A. Kassler, and N. Vaidya, "QoS-aware channel scheduling for multi-radio/multi-channel wireless mesh networks," in *WinTech Workshop*, 2009.

[32] J. Tang, G. Xue, and W. Zhang, "Interference-aware topology control and QoS routing in multi-channel wireless mesh networks," in *ACM MobiHoc Conference*, May 2005, pp. 68–77.

[33] T.-Y. Shen, "Experiments on a multichannel multi-interface wireless network," M.S. thesis, University of Illinois at Urbana-Champaign, 2008. [Online]. Available: http://www.crhc.illinois.edu/wireless/groupPubs.html.

[34] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian, "Practical, distributed channel assignment and routing in dual-radio mesh networks," in *ACM Sigcomm Conference*, 2009, pp. 99–110.

[35] R. Draves, J. Padhye, and B. Zill, "Routing in multiradio, multihop, wireless mesh networks," in *ACM MobiCom Conference*, 2004, pp. 114–128.

[36] C. Chereddi, "System architecture for multichannel multi-interface wireless networks," M.S. thesis, University of Illinois at Urbana-Champaign, 2006. [Online]. Available: http://www.crhc.illinois.edu/wireless/groupPubs.html.

[37] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, Network Working Group Std. RFC:3261, June 2002.

[38] P. Kyasanur, "Multichannel wireless networks: Capacity and protocols," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2006. [Online]. Available: http://www.crhc.illinois.edu/wireless/groupPubs.html.

[39] A. Botta, A. Dainotti, and A. Pescape, "Multi-protocol and multi-platform traffic generation and measurement," in *IEEE Infocom demo abstract*, 2007.

[40] X. Liu, Q. Wang, W. He, M. Caccamo, and L. Sha, "Optimal real-time sampling frequency assignment for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 2, pp. 263–295, May 2006.

[41] R. Gummadi, R. Patra, H. Balakrishnan, and E. Brewer, "Interference avoidance and control," in *HotNets VII Workshop*, October 2008.

[42] Y. Yuan, P. Bahl, R. Chandra, T. Moscibroda, and Y. Wu, "Allocating dynamic time-spectrum blocks in cognitive radio networks," in *ACM MobiHoc Conference*, 2007, pp. 130–139.

[43] K. Xu, K. Tang, R. Bagrodia, M. Gerla, and M. Bereschinsky, "Adaptive bandwidth management and QoS provisioning in large scale ad hoc networks," in *IEEE MILCOM Conference*, October 2003, pp. 1018–1023.

[44] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: A core-extraction distributed ad hoc routing algorithm," *IEEE Journal on Selected Areas in Communication*, vol. 17, no. 8, pp. 1454–1465, August 1999.

[45] C. Perkins, "Quality of service for ad-hoc on-demand distance vector routing," *Mobile Ad Hoc Networking Working Group - Internet Draft*, July 2000.

[46] W. Liao, Y. Tseng, S. Wang, and J. Sheu, "A multi-path QoS routing protocol in a wireless mobile ad hoc network," in *ACM ICN Conference*, 2001, pp. 158–167.

[47] C.-M. Cheng, P.-H. Hsiao, H. T. Kung, and D. Vlah, "Adjacent channel interference in dual-radio 802.11a nodes and its impact on multi-hop routing," in *IEEE Globecom Conference*, November 2006, pp. 1–6.

[48] V. Angelakis, A. Traganitis, and V. Siris, "Adjacent channel interference in a multi-radio wireless mesh node with 802.11a/g interfaces," in *IEEE Infocom Conference*, Student poster abstract, 2007.

[49] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaughl, "Partially overlapped channels not considered harmful," in *ACM SIGMetrics Conference*, June 2006, pp. 63–74.

[50] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, March 2000.

[51] S. Shah, K. Chen, and K. Nahrstedt, "Dynamic bandwidth management in single-hop ad hoc wireless networks," in *IEEE PerCom Conference*, March 2003, pp. 195–203.

[52] K. Chen, K. Nahrstedt, and N. Vaidya, "The utility of explicit rate-based flow control in mobile ad hoc networks," in *IEEE WCNC Conference*, March 2004, pp. 1921–1926.

[53] V. Raman, "Dealing with adjacent channel interference effects in multichannel, multi-interface wireless networks," M.S. thesis, University of Illinois at Urbana-Champaign, 2008. [Online]. Available: http://www.crhc.illinois.edu/wireless/groupPubs.html.

[54] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex behavior at scale: An experimental study of low-power wireless sensor networks," UCLA Computer Science, Tech. Rep., 2002.