# AN ABSTRACT OF THE THESIS OF

Erin Sullivan for the degree of Master of Science in Electrical and Computer Engineering presented on June 2, 2015.

Title: 802.11 Channel Switching for High-Definition Video Streaming

Abstract approved: _____

Ben Lee

As screen resolution and video decoding capability have increased, high-definition (HD) video in resolutions as high as 1920 x 1080 is rapidly becoming the standard. Ad-hoc streaming of HD video over 802.11 wireless networks, e.g., streaming from a mobile device to a television, is convenient for users, but is hampered by bandwidth restrictions of 802.11g and 802.11n. In particular, performance suffers if the 802.11 channel is crowded with many transmitting nodes or contains a hidden terminal. In this paper, we present a method for dynamically selecting an 802.11 channel frequency for transmitting HD video between a transmitting and receiving node in an ad-hoc connection. We consider both the selection of the initial 802.11 channel and the need to change the channel if video quality begins to suffer, while keeping the overhead of channel changes to a minimum. To the best of our knowledge, our work is the first to address ad-hoc channel selection on 802.11 with respect to the demands of HD video. Simulations conducted using OEFMON (Open Evaluation Framework for Multimedia Over Networks) have yielded preliminary but promising results.

# 802.11 Channel Switching for High-Definition Video Streaming

by

Erin Sullivan

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 2, 2015
Commencement June 2015

Master of Science thesis of Erin Sullivan presented on June 2, 2015.

APPROVED:

---

Major Professor, representing Electrical and Computer Engineering

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

---

Erin Sullivan, Author

# ACKNOWLEDGEMENTS

I would like to begin by thanking my mentor, Dr. Ben Lee, for his continual guidance and encouragement during the graduate school process. With his help I was able to become a focused and professional researcher and complete a thesis that I am proud of. I deeply appreciate his patience and continuing belief in me.

I would also like to thank my research teammates: Mohammed Sinky, Kevin Gatimu, and Peter Zhao for their help with the QualNet software, and especially Taylor Johnson who provided a great deal of assistance with copyediting.

Finally, I would like to thank my family and friends who offered me kindness and support during this process, especially my parents Daniel and Danielle Sullivan and my husband Josh Schonstal.

# TABLE OF CONTENTS

Page

1 Introduction 1

2 Background 5

2.1 Effect of Packet Loss on Video . . . . . 5

2.2 Carrier Sense Interference . . . . . 7

2.3 Hidden Nodes . . . . . 8

2.4 Dynamic Channel Switching . . . . . 9

3 Related Work 12

3.1 Hidden Node Detection . . . . . 12

3.2 Dynamic Channel Switching . . . . . 14

3.3 Video streaming over MANETs . . . . . 15

4 The Proposed ASDCS Method 17

4.1 Channel Evaluation . . . . . 17

4.2 Channel Selection . . . . . 21

4.2.1 Active Link Detection Message Exchange . . . . . 25

5 Simulation study 29

5.1 Simulation Environment . . . . . 29

5.2 Simulation Results . . . . . 35

6 Conclusion and Future Work 40

Bibliography 40

Appendices 45

A Interference-based implementation details . . . . . 46

# LIST OF FIGURES


| Figure | | Page |
|---|---|---|
| 1.1 | A typical home environment contains a variety of wireless devices. | 2 |
| 2.1 | A typical GOP structure consisting of I-, P- and B-frames. | 5 |
| 2.2 | Macroblock prediction for I-MB, P-MB and B-MB. [1] | 6 |
| 2.3 | Effect of packet loss on received video. | 7 |
| 2.4 | Two methods of error concealment (EC). | 8 |
| 2.5 | Carrier sensing node scenario. | 9 |
| 2.6 | Hidden node scenario. | 9 |
| 2.7 | 802.11 channel frequencies [2] | 10 |
| 3.1 | Passive hidden node detection. | 13 |
| 4.1 | Block diagram of the proposed ASDCS method. | 17 |
| 4.2 | Outgoing IP queue size and PSNR for an HN scenario (*African Cats*, 6 Mbps). HN interference occurs at 6 seconds, which increases the queue size and decreases PSNR. | 18 |
| 4.3 | Outgoing IP queue size from 7 second to 8 second for an HN scenario (*African Cats*, 6 Mbps). | 18 |
| 4.4 | Nodes surrounding a streaming node-pair. | 22 |
| 4.5 | Average video PSNR of carrier sense and hidden node scenarios. | 24 |
| 4.6 | Packet exchange diagram for active link detection. | 27 |
| 5.1 | The general structure of OEFMON. | 30 |
| 5.2 | Hidden node scenario with one hidden node pair. | 31 |
| 5.3 | Carrier sensing scenario with three carrier sense node pairs. | 32 |
| 5.4 | Mixed scenario with one hidden and two carrier sense node pairs. | 32 |

# LIST OF FIGURES (Continued)


Figure | Page

5.5 Multichannel scenario 1 with two hidden node pairs. . . . . 33

5.6 Multichannel scenario 2 with two hidden and one carrier sense node pairs. 33

5.7 Predicted vs. actual video quality for the *African Cats* video in three interference scenarios. . . . . 36

5.8 Predicted vs. actual video quality for the *Life of Pi* video in three interference scenarios. . . . . 36

5.9 Predicted vs. actual video quality for the *Star Wars* video in both multichannel scenarios. . . . . 38

5.10 Predicted vs. actual video quality for the *Life of Pi* video in both multichannel scenarios. . . . . 39

# LIST OF TABLES


Table Page

4.1 Nodes visible by the RX node. . . . . 22

4.2 Nodes visible by the TX node. . . . . 23

4.3 Node neighborhood list of hidden nodes and carrier sense nodes. . . . . 23

4.4 Node neighborhood list with weak-signal nodes removed. . . . . 24

4.5 Format of PROBE_PKT. . . . . 27

4.6 Format of NODE_LIST. . . . . 28

4.7 Format of other control packets. . . . . 28

5.1 Bitrates for the five scenarios. . . . . 34

5.2 ASDCS default values. . . . . 35

5.3 Node neighborhood lists for multichannel scenarios. . . . . 37

# LIST OF APPENDIX FIGURES

Figure Page

A.1 Interference-based TX node state machine. . . . . 46

A.2 Interference-based RX node state machine. . . . . 47

# LIST OF APPENDIX TABLES

Table Page

A.1 Format of TX_SCAN_PKT. . . . . 48

A.2 Format of other Interference-based control packets. . . . . 48

A.3 Interference-based default values. . . . . 49

# Chapter 1: Introduction

The IEEE 802.11 standard has become the predominant networking protocol used in homes, offices, and public areas, largely displacing wired networks. Fig. 1.1 shows a typical home or office environment that contains a multitude of wireless devices – desktop and laptop computers, smartphones, tablet devices, game consoles, and television set-top boxes, and wireless HD video transmission between multiple pairs of devices on the wireless network has become an important enabling technology. This includes Apple's AirPlay [3], which provides HD screen mirroring and streaming from Apple devices (Mac, iPhone, iPod Touch, and iPad) to the Apple TV set-top box over a standard WiFi network, Intel's Wireless Display (WiDi) [4] technology provides largely the same feature set as AirPlay, but is only compatible with a limited number of consumer electronics and adapters using Intel's wireless chipsets, and Google's Chromecast [5], a relatively new arrival to this space, is primarily intended to stream Internet content from sites such as Netflix and YouTube, but also offers screen mirroring. In addition to these proprietary systems, there are also several proposed industry standards competing for the wireless HD video space; these systems support ad-hoc high-speed transfer over short distances by operating in the 20–60 GHz range. They include WirelessHD [6], Wireless Home Digital Interface (WHDI) [7], and Wireless Gigabit Alliance (WiGig), which is also known as 802.11ad [8]. Future enhancements to wireless HD video transmission include *N-screens* systems, where multiple multimedia feeds and other data streams can be shared and displayed among multiple devices.

The nature of video (as well as other multimedia files) requires data to be sent and consumed continuously, though data rates can vary during transmission. Any interruption of service due to network interference adversely affects user experience. Furthermore, bandwidth demands are increasing as HD video has become the norm, e.g., a 1080p video at 60 fps encoded using H.264 can require as much as 50 Mbps. These bandwidth requirements put a strain on older 802.11g networks, which can only achieve a maximum throughput of 54 Mbps. Newer revisions of 802.11 have increased bandwidth significantly in response to growing demand. The widely-used 802.11n standard can achieve

Figure 1.1: A typical home environment contains a variety of wireless devices.

145 Mbps, and the relatively recent 802.11ac standard can achieve at least 500 Mbps throughput on a single link [9].

As the amount of available bandwidth continues to grow, its demand is also increasing. A recent survey found that the average household contained five networked devices (70% using a wireless connection) and that 6% of households contained 15 or more networked devices [10], and these numbers are only expected to grow. In a densely packed apartment complex or office, the number of devices competing for the wireless medium can be very high indeed. Using N-screens technology, multiple media streams may be mirrored, exchanged or synchronized between devices [11]. New video technologies such as 3D television [12] and 4K Ultra-HD (containing four times the pixels of 1080p) [13] increase the necessary bandwidth for each video stream, which puts further strain on wireless networks. With these bandwidth-demanding technologies on the horizon, there is still motivation to combat wireless network congestion through intelligent methods, even as more bandwidth becomes available via new wireless standards.

As the number of nodes increases in a network, the amount of interference also increases. For example, heavy carrier sense interference among peer-to-peer connections on the channel causes delays and packet loss that affect visual quality. A more serious problem is the *hidden node* (or *hidden terminal*) scenario, which occurs when an another transmitting node is visible by the receiver of a sender/receiver node pair, but not by the sender of the node pair. Therefore, the sender transmits without the knowledge of packet collisions at the receiving node. Techniques such as the use of Request-To-Send (RTS) and Clear-To-Send (CTS) packets [14], and active detection of hidden nodes [15]

have been explored as ways of addressing the hidden node problem. These methods can be effective at dealing with hidden nodes, but at the cost of additional bandwidth.

To avoid or minimize the effect of channel interference, a node pair can be deliberately assigned to different channels. This assignment can be done manually by using WiFi channel scanning software, such as inSSIDer [16], and selecting the channel with the fewest number of interfering nodes. However, channel conditions and required bandwidth vary over time due to node mobility and activity. Therefore, manually finding the "best channel" is insufficient. Instead, a *dynamic channel switching* scheme with periodic channel re-evaluation is needed to take full advantage of multiple channels. There are numerous previous works on such dynamic channel switching methods [17–29]. Many of these methods require all nodes in the system to coordinate channel usage, making them difficult to incorporate into the real-world wireless environment. To the best of our knowledge, none of the existing methods incorporate a real-time Quality of Service measurement to determine when to change the channel. The "best channel" is determined by a single measurement such as SINR or 802.11 backoff window size, which do not directly correlate to better real-time streaming performance. The outgoing IP queue size was considered as a channel metric by some authors [20] but was ultimately rejected as being too volatile. Instead, our channel evaluation method uses changes in outgoing IP queue size as a metric for estimating video streaming quality.

This thesis presents the *Active Scanning-based Dynamic Channel Switching* (ASDCS) method for 802.11 aimed specifically at improving the quality of peer-to-peer HD video streaming. The proposed method is based on identifying potentially interfering devices on each channel and is compatible with existing WLANs. This is achieved by using active scanning and promiscuous-mode packet detection of 802.11 to obtain a list of potentially interfering devices surrounding the video transmitter and receiver pair on each channel, and uses this information to select the "best" channel within a reasonable timeframe. To determine whether the current channel has become unsuitable for video transfer, the proposed ASDCS evaluates the rate at which packets are leaving the outgoing IP queue and uses this information to predict packet loss for the upcoming video frame. If the predicted packet loss exceeds the acceptable packet loss for that frame type indicating unacceptable video quality, the transmitter and receiver will search for a new channel.

The thesis is organized as follows. Chapter 2 provides background information to better understand the motivation behind ASDCS. Chapter 3 discusses the relevant re-

lated work on hidden node detection, dynamic channel switching, and wireless video streaming. Chapter 4 presents the proposed ASDCS method. Chapter 5 presents the results of our simulation study. Finally, conclusion and future work are presented in Chapter 6.

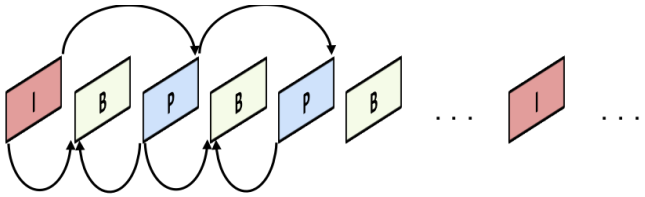



Figure 2.1: A typical GOP structure consisting of I-, P- and B-frames.

# Chapter 2: Background

Developing a dynamic channel switching method for wireless HD video transmission presents an interesting intersection of problems. This section will present the background information necessary to understand the proposed channel switching method.

## 2.1 Effect of Packet Loss on Video

In 802.11, traffic is sent over one of several wireless channels. Bandwidth on a channel is affected by numerous factors, including channel interference, distance between nodes, and number of active nodes contending for the channel. HD video has a high bitrate and a relatively low tolerance for dropped or delayed packets, making it particularly susceptible to interference on the channel. Modern HD video codecs, such as H.264, anticipate that packets will be lost during streaming and use Error Concealment (EC) techniques, such as Frame Copy, Weighted Pixel Averaging (WPA), and Motion Vector Recovery, to conceal the data loss. In the case of a severely noisy channel, however, these may prove insufficient.

In order to stream HD videos (up to 1080×1920 pixels per frame and 60 frames per second) at a reasonably low bitrate, H.264 uses various compression techniques to decrease the size of the video. A video is comprised of a series of ordered frames, which are arranged in a sequence known as a *Group of Pictures* (GOP) consisting of intra-picture frames (I-frames), unidirectional predicted frames (P-frames), and bidirectional predicted frames (B-frames) as shown in Fig. 2.1.

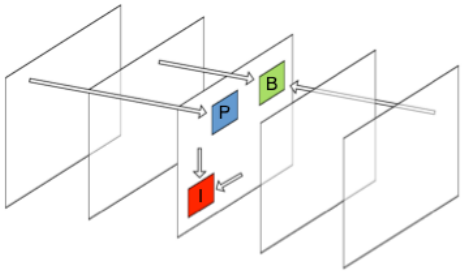



Figure 2.2: Macroblock prediction for I-MB, P-MB and B-MB. [1]

Pixels within a frame are arranged into 16×16 (or smaller) groups called *macroblocks* (MB). The three macroblock types are shown in Fig. 2.2. For compression purposes, most macroblocks are *predicted* from other macroblocks, so that only the difference or *residual* between the two blocks needs to be encoded. *Intra-coded* macroblocks (I-MB) are generated using samples of neighboring blocks on the same frame, whereas *inter-coded* macroblocks are generated using either one past reference frame (P-MB) or one past and one future frame (B-MB). I-frames contain only I-MB, P-frames contain I-MB and P-MB, and B-frames contain I, P and B-MB.

Each frame is divided into several *slices*, each of which contains a roughly equal number of macroblocks. Packet loss can result in a partial or entire missing slice (if the *slice header* is lost). Fig. 2.3b shows a video frame with two slices lost. The choice of how to conceal missing slices is up to the decoder – the simplest method is simply to copy from the previous frame, as shown in Fig. 2.3c. Because inter-predicted macroblocks are generated from neighboring frames, errors can propagate several frames later (Fig. 2.3d).

Fig. 2.4b shows an attempted frame recovery using WPA and Motion Vector Recovery. As can be seen, the result is a blurred "vertical lines" effect because many slices were lost. Missing inter-MBs can be concealed using motion vector recovery (Fig. 2.4d), which attempts to recover the *motion vector* (relative movement of the MB) by averaging motion vectors from adjacent frames. This helps to preserve the overall motion of the video even when parts of the picture are missing.

Because error concealment relies on the presence of spatially and temporally neighboring macroblocks, there is only so much that can be done to recover macroblocks or frames when many consecutive packets are lost – especially those containing I-MBs.

(a) Original frame.

(b) Frame with two missing slices due to dropped packets.

(c) Error concealment using duplication

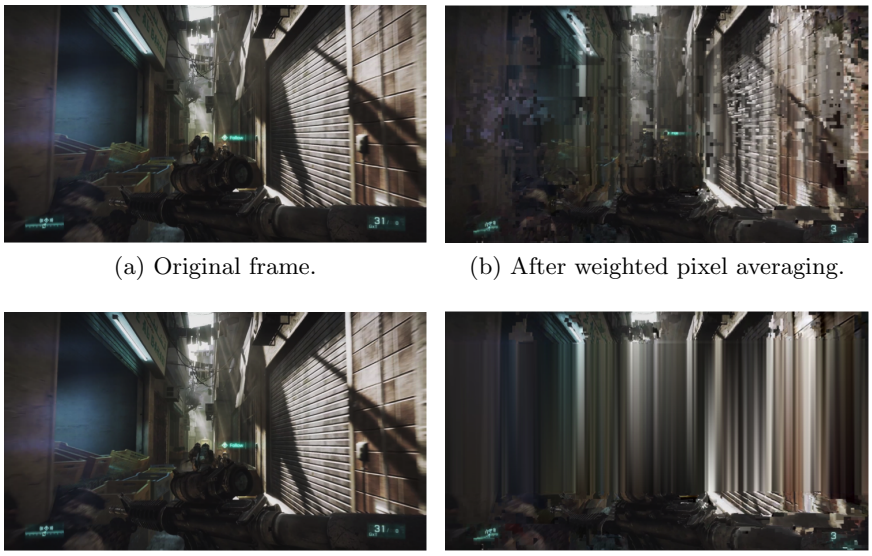
(d) Error propagation 5 frames later.

Figure 2.3: Effect of packet loss on received video.

Other methods are needed to prevent packet loss and prioritize more valuable packets.

## 2.2 Carrier Sense Interference

*Carrier sensing* is a part of the 802.11 Distributed Coordination Function (DCF). When a node wishes to transmit on the wireless medium, it first senses if the medium is busy (i.e.,) a nearby node is transmitting on the same channel). If so, it will defer its transmission and try to gain access again after a random backoff period. A *carrier sensing node* is any node within the carrier sensing range of the transmitter that could potentially contend for the wireless medium. Fig. 2.5 depicts a transmitter (TX) and receiver (RX) node pair with two carrier sense (CS) nodes: CS1, which is visible by both RX and TX, and CS2, which is visible only by the TX. While carrier sensing nodes do not cause a high rate of collisions like hidden nodes do, they can cause delays in packet transmission that effectively result in dropped packets for real-time applications.



(a) Original frame. (b) After weighted pixel averaging.

(c) Original frame. (d) After motion vector recovery.

Figure 2.4: Two methods of error concealment (EC).

## 2.3 Hidden Nodes

As discussed in Sec. 1, the presence of even a single *hidden node* can have a severe impact on the wireless link quality. Fig. 2.6 depicts a typical hidden node scenario. The TX is sending packets to the RX. The hidden node (HN) is outside of the TX's carrier sensing range, but is visible by the RX. The TX node and HN cannot perform the usual contention for the wireless medium and may unknowingly transmit packets at the same time, causing packet collisions at the RX.

The traditional method of dealing with hidden nodes is to use the carrier sense multiple access with collision avoidance (CSMA/CA) specified in the 802.11 standard [14]. Specifically, the collision avoidance portion specifies two additional control packets, Ready-to-Send (RTS) and Clear-to-Send (CTS). In the scenario shown in Fig. 2.6, the transmitter would send an RTS packet before attempting to send a data packet to the receiver. The receiver would then respond with a CTS packet, indicating that the medium is clear (i.e., there is no interference from hidden nodes) and that the transmitter is free to send. Any node that hears the receiver's CTS packet will refrain from sending for a set period of time thereby resolving the hidden node problem.

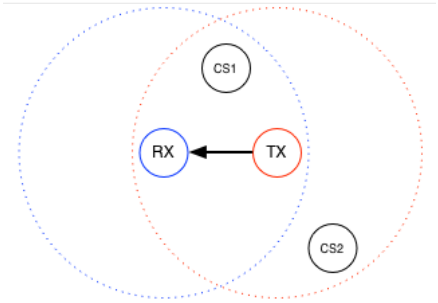



Figure 2.5: Carrier sensing node scenario.

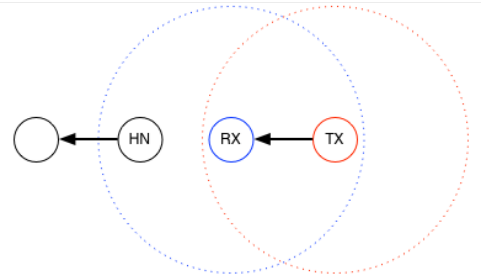



Figure 2.6: Hidden node scenario.

While RTS/CTS is somewhat effective at solving the hidden node problem, it is not without its issues. The additional overhead incurred by sending RTS and CTS frames can lead to a significant drop in overall throughput. Furthermore, the fact that other nodes are required to wait can lead to increased packet delay. This is especially a problem for multimedia streaming with strict playout deadlines.

## 2.4 Dynamic Channel Switching

The original 802.11 specification defines fourteen channel frequencies around 2.4 GHz. Later versions of the standard also include support for channels at 5 GHz. Each channel (except for channel 14) is spaced 5 MHz apart and is 22 MHz wide. In the United States, only channels 1–11 are permitted for use; this leaves channels 1, 6 and 11 as the non-overlapping channels. These channels will not have cross-channel interference.

Implementing a dynamic channel switching method on 802.11 has several challenges.

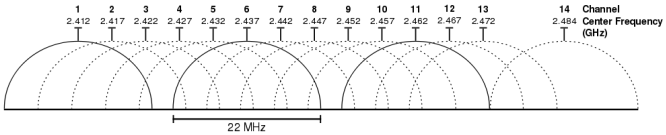



Figure 2.7: 802.11 channel frequencies [2]

Changing the channel on a station or an Access Point (AP) is not instantaneous since the device's antenna must be tuned to a new frequency. More importantly, the device initiating the channel switch must be able to communicate with the other devices it's communicating with and inform them to change channels. It must also account for the fact that these communication packets may be lost; this is even more likely if poor channel conditions have triggered the channel switch.

Because of the channel switching overhead, it is important for any channel switching algorithm to consider the cost of changing channels versus the potential benefit of moving to a new channel. This is even more important for real-time multimedia applications such as video streaming, where every second the viewer is left waiting leads to reduced Quality of Experience (QoE). In order to determine whether the current channel is "bad" and should be changed, multi-channel protocols incorporate a *load metric* [20] to determine the relative load on the channel. If the method is to operate on the lower layers of the 802.11 stack, such as the MAC and PHY layers, it must use some measurement that is visible to these layers. A commonly used measurement is the *Signal to Interference plus Noise Ratio* (SINR), which is the ratio of received signal power to background noise and interference from other devices as defined as follows:

$$SINR = \frac{Signal\ Power}{Background\ Noise + Interference\ Power},$$

where *Signal Power* is power of the incoming signal of interest – in this case, the power of the transmitter's signal, *Background Noise* is the environmental noise due to fading and RF interference, and *Interference Power* is the power of other concurrently transmitting nodes.

On an 802.11 network, a SINR of at least 20 dB is acceptable and 40 dB is excellent [30]. SINR and interference power provide a reasonable measure of surrounding carrier sense interference and many existing channel switching methods [17–19, 22] have used

SINR/interference thresholds both to trigger channel switches and to determine the best new channel to switch to. Others have used the 802.11 MAC backoff window size [23] and overall MAC delay [20] to determine when to switch.

# Chapter 3: Related Work

There are a number of related works on dealing with hidden nodes, dynamic channel switching methods, and HD video streaming on mobile ad-hoc networks (MANETs). The following subsections discuss the state-of-the-art.

## 3.1 Hidden Node Detection

There have been several efforts to mitigate the hidden node problem that go beyond the RTS/CTS solution provided by the 802.11 standard.

Li *et al.* proposed a method for passive detection of hidden nodes in which nodes enter *promiscuous mode* and listen for a neighboring node to send a data packet without the corresponding ACK packet, or vice versa [15]. Fig. 3.1 depicts the passive hidden node detection, where Node 1 is listening to the channel in promiscuous mode and can sense data packets sent from Node 2 to Node 3. However, Node 1 cannot detect the ACK packet from Node 3, and concludes that Node 2 must be communicating with a hidden node. The passive method has limited usefulness because the hidden node can only be detected if it is directly communicating with its neighbor; in practice this is often not the case. To rectify this, the authors proposed an active variant on the method that uses probe packets. In the above scenario, Node 1 would send a probe packet to Node 2, which would force Node 2 to send probe packets to all its neighbors, and wait for ACK responses. The active method is better at finding all the hidden nodes, but still has a couple of shortcomings. First, it requires modification to all the nodes in the network so that they respond to probe packets. By contrast, our proposed ASDCS only requires modifications to node pairs involved in video streaming. Second, it is possible that some of the hidden nodes detected may be "false positives". The active hidden node detection method will find all hidden nodes, even those whose signal strength is low enough to be considered negligible interference. ASDCS uses a SINR threshold to disregard these "false positive" hidden nodes, which are not likely to cause packet collisions at the receiver.

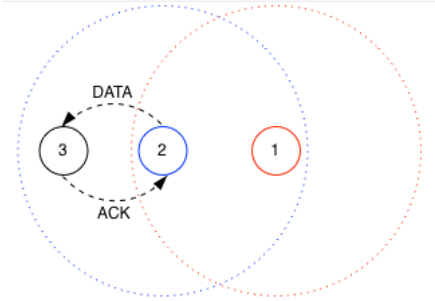



Figure 3.1: Passive hidden node detection.

Hidden node interference may also be inferred indirectly. If the channel is selected using SINR or interference at the receiver, rather than at the transmitter, this gives a better estimation of overall interference, including interference from hidden nodes. Some of the prior studies on dynamic channel switching [18, 22] measure SINR at the receiver for exactly this reason. SINR does not distinguish between sources of interference, so hidden node interference cannot be distinguished from (much less disruptive) carrier sense interference. Because hidden nodes are so disruptive to video streaming, it would be preferable to use a method that identifies them definitively. SINR measured at the receiver may also miss interference from carrier sense nodes that are only visible by the transmitter.

Disproportionate packet loss or delay compared with the measured SINR may also be an indicator of a hidden node. The existence of hidden nodes can also be inferred by combining signal strength or MAC-layer statistics with knowledge of the approximate loss due to carrier-sense nodes [31, 32]. For example, a high number of lost ACKs at the transmitter is a strong indicator that a hidden node may be present, although this may not always be the case. However, these methods are inappropriate for a dynamic channel switching method because they would require spending a significant amount of time on each channel sending data and analyzing the results. Any amount of time spent on "bad" channels contributes adversely to the user Quality of Experience (QoE). A good dynamic channel switching method must determine the best channel with as little interruption to the user experience as possible.

If a hidden node is detected, it can be dealt with directly. A multi-hop protocol, such as the one proposed by Kortebi *et al.* [31], can adjust the effective bandwidth of a link

accordingly and route around the hidden node. But, this is not useful when links are single-hop, as in the case of peer-to-peer video streaming. Link-adaptation protocols [32] can adjust the data rate or turn on RTS/CTS when the loss caused by hidden nodes is high enough to justify the increased overhead. For high-bitrate traffic such as HD video, changing the channel is a better option because it does not decrease the available bandwidth.

Our proposed ASDCS method improves upon existing methods in three ways. First, it uses a combination of active scanning and promiscuous-mode listening on each channel to identify all visible actively transmitting nodes and APs – hidden or not. Second, in addition to detecting the existence of hidden nodes, the proposed method takes into account the strength of the received interference signal to determine whether it will truly cause a collision at the receiver. Third, because it uses channel switching to avoid hidden nodes, it does not reduce channel bandwidth, unlike existing protocols that reduce the data rate or turn on RTS/CTS.

## 3.2 Dynamic Channel Switching

Many existing multichannel methods are *cooperative* in nature [17, 18, 22, 24, 25, 27–29]. In these methods, every node in the network shares information about its current state on a *control channel* reserved for inter-channel communication. Using this information, each transmitting node attempts to select a channel that will benefit itself but not cause too much interference to surrounding nodes. The control channel may use a separate antenna [17,18,22], or be accessed during synchronized time windows [24,25,27–29]. The former solution requires additional hardware, while the nodes in the latter method are unable to transmit data during control window periods. This reduces overall bandwidth, which is undesirable given the high bandwidth requirements of HD video.

In contrast, with *distributed* channel selection methods, nodes on different channels do not share information. Instead, individual nodes must make their own channel switching decisions based on observed channel quality. Existing distributed channel selection methods use receiver SINR [19] or packet delay [20, 26] as a metric of channel quality. When this measurement drops below the channel switch threshold (i.e., SINR is too low or packet delay is too high), the node initiates a channel switch, analyzes each channel, and picks a new one with better SINR or average packet delay. Channel SINR and packet

delay are indirectly related to video quality, but determining an appropriate switching threshold is difficult. For new channel selection, packet delay is a strong indicator of overall channel load, but calculating it accurately requires sending test packets or long periods of time (up to 10 seconds) [20] on each channel, which is highly undesirable for real-time video streaming. On the other hand, measuring receiver SINR requires sender and receiver to "hop" to each channel together and send test packets on each, which adds an overhead for coordinating each hop. SINR-based methods cannot account for the performance impact of carrier sensing nodes, because these nodes coordinate access and therefore don't interfere with the transmitter. Also, SINR-based methods cannot distinguish between multiple weaker nodes vs. a single stronger node, even though these two channel scenarios impact video performance differently. Our proposed ASDCS does use both SINR and packet delay, but in a way that addresses the shortcomings of these existing methods.

To the best of our knowledge, there are no existing channel switching methods that specifically address the requirements for wireless video transmission. These requirements include the need for a fast channel selection and switching time to minimize QoE disruptions, a channel quality measurement that accurately reflects video quality, and avoidance of potential interfering nodes, especially hidden nodes. Our proposed method predicts the amount of packet loss in upcoming H.264 video frames to accurately determine when video quality has become poor and the channel should be changed. After initiating the channel switch, the ASDCS method attempts to identify surrounding hidden and carrier sense nodes on each channel using active link detection and communication between transmitter and receiver. This information is used to select the new channel with the fewest interfering nodes, with higher priority given to avoiding of hidden nodes. While the proposed method does not directly account for the impact the channel switch will have on other nodes, moving a high-bandwidth video node pair from a busy channel to a less populated one indirectly benefits the other nodes on the busy channel.

## 3.3 Video streaming over MANETs

Many techniques have been developed to deal with loss of video quality when transmitting over a wireless network. Several methods relied on adaptation of the PHY link when video quality drops, including the techniques presented in [33, 34] that lower the

PHY data rate to decrease the overall error rate. These strategies were tested on low-resolution videos (QCIF and CIF size) and attempt to transmit at the highest data rate possible while keeping the packet error rate (PER) below a certain threshold. While these methods proved effective at improving video quality, it would be preferable to find a method that does not reduce the data rate for high bitrate HD video. Krishnamachari *et al.* used the Point Coordination Function (PCF) for managing real-time data, which yielded some positive results [35]. However, PCF also has additional bandwidth requirements as it must periodically secure the wireless medium. It is also not implemented on the majority of hardware devices on the market.

A related technique is to change the quality of the video transmission at the application layer. Setton *et al.* proposed a method that changes the quantization level and GOP length of a QCIF video based on available bandwidth [36]. This however cannot be done in real-time. Qin and Zimmermann used a multi-layer encoding where videos have a base layer and multiple enhancement layers, which may be added or dropped depending upon available bandwidth [37]. This technique is effective at preventing service interruptions and dealing with brief congestion on the network, but may still "break down" under prolonged heavy channel interference – in that case, channel switching may be the only viable option.

Another method that has been used with some success is a priority-based scheme that gives an advantage to real-time packets. Oh and Chen used 802.11e Enhanced Distributed Channel Access (EDCA) [38], and Fiandrotti *et al.* used the same functionality to give priority to packets containing an I-frame [39]. These techniques show promise at reducing packet loss in real-time systems due to MAC delay, but do nothing to protect against packet loss from hidden node collisions or general channel noise. Moreover, priority-based methods are orthogonal to channel switching, so they can be used together.

It is very possible that a channel may be so congested that any of the previous methods, effective as they may be, will not be enough to produce a video with acceptable QoE. The use of our dynamic channel switching method provides an opportunity to avoid heavy interfering traffic on a channel altogether rather than having to compensate for it. Using dynamic channel switching *in conjunction* with existing encoding or priority-based methods may provide the best overall result.

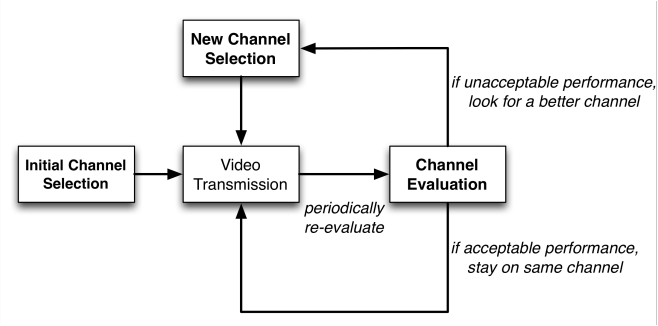



Figure 4.1: Block diagram of the proposed ASDCS method.

# Chapter 4: The Proposed ASDCS Method

Fig. 4.1 shows the dynamic channel selection process of ASDCS, which consists of Initial Channel Selection, Channel Evaluation, and New Channel Selection.

In order to determine whether or not to perform channel switching, *Channel Evaluation* is performed periodically during video transmission to determine if the current channel condition is still acceptable for the video being delivered. During the *Channel Selection* process, the node pair collects information necessary to determine the "best" channel (i.e., the channel with the least amount of disruptive interference to the streaming node pair) within a reasonable timeframe. The following subsections discuss the Channel Evaluation and Channel Selection steps in detail.

## 4.1 Channel Evaluation

Previous studies [20, 21, 26] have shown that IP queue size and packet delay are related to channel quality. The proposed method expands on this by using channel packet delay, which is derived from IP queue measurements, to predict *video quality* in upcoming frames.

Fig. 4.2 shows PSNR and outgoing IP queue size as function time for a sample video in a hidden node interference scenario (see Sec. 5.1). The video is encoded at a variable

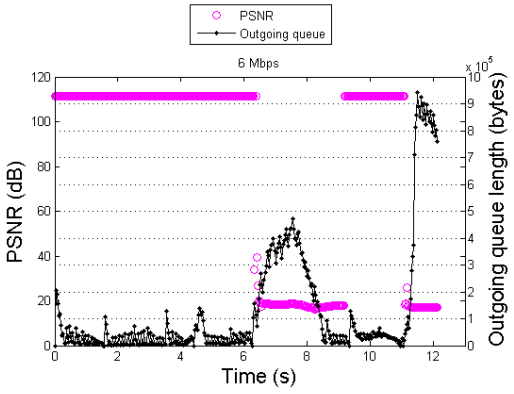



Figure 4.2: Outgoing IP queue size and PSNR for an HN scenario (*African Cats*, 6 Mbps). HN interference occurs at 6 seconds, which increases the queue size and decreases PSNR.

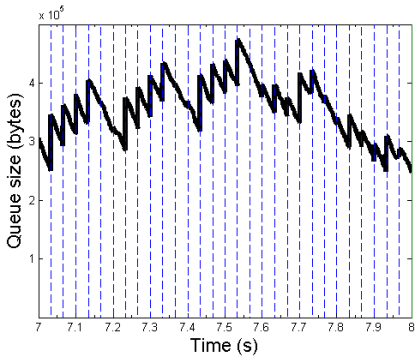



Figure 4.3: Outgoing IP queue size from 7 second to 8 second for an HN scenario (*African Cats*, 6 Mbps).

bit rate (VBR) with an average rate of 6 Mbps. At first, the video bit rate is low enough to withstand the hidden node interference, but at around 6 seconds, and again at around 11 seconds, there are high-motion sections of the video that increase the bit rate. The interfering hidden node causes packets to collide, and a backlog forms on the IP queue which increases delay and decreases PNSR of the received video frames. Therefore, there is a correlation between IP queue size and the quality of the received video frames (i.e., PSNR). Fig. 4.3 shows the magnified portion of the outgoing IP queue size from 7 sec. to 8 sec. of Fig. 4.2. Note the "jagged" appearance, which is the result of placing the data for an encoded frame into the IP queue every $1/fps$ seconds denoted by the blue

dashed lines. Between frames, packets in the IP queue are transmitted until the packets for the next frame are enqueued. For frames in the video that require more bandwidth than the network can handle, packets will be enqueued in the IP queue faster than they can be transmitted causing it to fill up. At this point, any additional packets added will be dropped and thus considered lost.

There are a couple of other reasons packets can be lost. First, video streaming applications use a parameter called *jitter* to indicate the acceptable variance from the fixed playout deadline (i.e., every 1/fps seconds for a video frame). If a packet sent by the transmitter at time $t_0$ arrives at the receiver later than $t_0 + jitter$, where *jitter* is typically 150 ms, that packet is considered outdated and discarded by the receiver. Packets may also be dropped at the MAC layer if they exceed the number of allowed retransmissions. This situation is very infrequent, but these packets can be delayed long enough that they would have been outdated even if they are delivered.

In order to estimate the available channel bandwidth based on the IP queue size, suppose a frame is enqueued at time $t_0$, resulting in total of $Q_0$ bytes in the IP queue. Afterwards packets leave the queue as they are transmitted. Then, at time $t_1 = t_0 + 1/fps$, the transmitter enqueues $f_1$ bytes of the next frame, after which there are $Q_1$ bytes in the IP queue. This means that $Q_0 - (Q_1 - f_1)$ bytes successfully left the queue during a period of $1/fps$. If the IP queue did not empty before time $t_1$ (i.e., $Q_1 - f_1 > 0$),then the rate at which bytes were sent from the queue ($BW_{est}^{Tx}$, the *estimated transmitter bandwidth* during the $1/fps$ period) can be calculated as:

$$BW_{est}^{Tx} = \frac{Q_0 - (Q_1 - f_1)}{1/fps}. \tag{4.1}$$

If the IP queue does empty at some time $t_{empty}$, (before time $t_1$), it means that all $Q_0$ bytes were sent from the queue during time period $t_{empty} - t_0$, in which case $BW_{est}$ can be calculated as:

$$BW_{est}^{Tx} = \frac{Q_0}{t_{empty} - t_0} \tag{4.2}$$

In practice, $BW_{est}^{Tx}$ can fluctuate from frame to frame, and a brief downward spike in estimated available bandwidth does not necessarily indicate that network conditions have deteriorated significantly. Therefore, a simple moving average of several recent estimated

bandwidth samples can help avoid false positives, as shown below:

$$avgBW_{est}^{Tx} = \frac{1}{n}\sum_{i=1}^{n}(BW_{est}^{Tx})_i, \tag{4.3}$$

where $n$ represents the number of past $BW_{est}^{Tx}$ measurements. The default value for $n$ is set to $fps/2$, or 0.5 seconds worth of samples. This is enough samples for accuracy while still being reasonably responsive.

Since the transmitter enqueues all the packets for a frame at once, the estimated delay for each packet in the frame can be calculated individually before the frame is sent. This is achieved by using $avgBW_{est}^{Tx}$ to estimate end-to-end delay for individual packets $Delay_{est}^{pkt}$, which is equal to the amount of time required to transmit the packet itself *plus* the amount of time to transmit all the packets ahead of it in the IP queue:

$$Delay_{est}^{pkt} = \frac{Q + p_i}{avgBW_{est}^{Tx}}, \tag{4.4}$$

where $Q$ is the number of bytes currently ahead of packet $p_i$ in the queue and $p_i$ is the size of the $i^{th}$ packet to be transmitted within a frame. $Delay_{est}^{pkt}$ can then be compared against the jitter parameter to predict whether or not packet will be lost as shown below:

$$Delay_{est}^{pkt} > jitter. \tag{4.5}$$

The proposed ASDCS performs *Video Quality Prediction* based on the estimated packet delay given in Eq. 4.4. The video quality is predicted to be unacceptable under the following circumstances:

1. *Any* packets containing a Sequence Parameter Set (SPS) or Picture Parameter Set (PPS) are predicted to be lost. These NAL units contain information that will be used in many subsequent frames, so it is important that they are properly transmitted.

2. The percentage of packets predicted to be lost from an I-frame NAL unit exceeds the threshold $I_{lost}$. This threshold should be fairly low because loss from I-frames will propagate through many subsequent frames.

3. The number of consecutive P- or B- frame NAL units predicted to be completely

lost exceeds the threshold $P_{lost}$ or $B_{lost}$. H.264 error correction techniques can reasonably compensate for loss of macroblocks or even an entire P- or B- frame. However, multiple frame loss cannot be easily compensated for and is a good indicator that the IP queue has become full.

Approximation of user QoE (a subjective determination) from quantitative measurements is an ongoing research problem [40, 41]. The user's experience is affected by both network conditions (which result in packet loss and delay) and the video content itself [42, 43]. Based on our observations of frame loss and its relation to PSNR we have selected $I_{loss} = 20\%$ and $P_{lost} = B_{lost} = 2$ frames as default threshold values.

In addition, when video quality becomes poor due to the above situations, it is predicted to remain poor until reception of the next I-frame, which effectively resets the picture. When the Video Quality Prediction determines that the current channel is unacceptable, ASDCS will initiate a channel change and the Channel Selection process begins.

## 4.2 Channel Selection

ASDCS utilizes *Active Link Detection* to obtain information about the interfering nodes on each channel. Since both AP and non-AP nodes can interfere with video transmission, Active Link Detection uses active scanning to detect AP nodes and passive listening to detect non-AP nodes, to come up with the *visible node list*. Active scanning is normally used when a node is trying to find an AP to associate with. A node wishing to associate with an AP sends a probe request packet to each channel, and APs that receive the probe request respond with a probe response packet. In this way, the node can obtain a list of APs on each channel. Since active scanning only receives probe responses from APs, not from non-APs, the node performing the active scan will enter promiscuous mode to passively listen for the MAC headers of visible packets from transmitting non-AP nodes.

As discussed in Sec. 2, there are two types of potentially interfering nodes: *hidden nodes* (HN), which are visible by the receiver but not the transmitter, and *carrier sense* (CS) nodes, which are all nodes visible by the transmitter. If the transmitting and receiving nodes compare their visible node lists, they can identify the HN and CS nodes. In ASDCS, this process involves the following steps: First, the transmitting node initiates

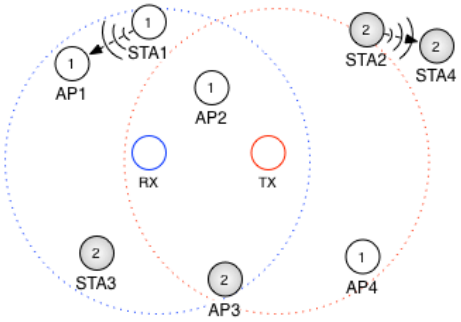



Figure 4.4: Nodes surrounding a streaming node-pair.

| Channel | Node ID | Signal Strength | AP? |
|---|---|---|---|
| 1 | AP1 | -65 dBm | Yes |
| 1 | AP2 | -40 dBm | Yes |
| 1 | STA1 | -80 dBm | No |
| 2 | AP3 | -65 dBm | Yes |

Table 4.1: Nodes visible by the RX node.

a channel scan by sending a message to the receiving node. The transmitter temporarily suspends video playback (if it has started), then both nodes perform the Active Link Detection to generate visible node lists. After both nodes have finished scanning channels, the receiving node sends its visible node list back to the transmitting node.

In order to illustrate the Active Link Selection, Fig. 4.4 depicts nodes surrounding the RX and TX node pair on channels 1 and 2 (the only available channels in this example scenario). Nodes on channel 1 are shaded white, and nodes on channel 2 are gray. Channel 1 contains three APs (AP1, AP2, and AP4) and one non-AP node (STA1), while Channel 2 contains one AP (AP3) and three non-AP nodes (STA2, STA3, and STA4). STA1 is currently transmitting packets to AP1, and STA2 is transmitting to STA4. The TX node will initiate the active link detection by sending a control packet to the RX node. Both TX and RX nodes then perform the active link detection as described in detail in Sec. 4.2.1, and each will generate a visible node list, which are shown in Table 4.1 and Table 4.2, respectively.

Note that AP1 and STA1 on channel 1 are visible to the RX node, but not to the TX node. This means they are hidden nodes to the TX node. The non-AP node STA3 was

| Channel | Node ID | Signal Strength | AP? |
|---|---|---|---|
| 1 | AP2 | -40 dBm | Yes |
| 1 | AP3 | -70 dBm | Yes |
| 2 | AP4 | -65 dBm | Yes |
| 2 | STA2 | -85 dBm | No |

Table 4.2: Nodes visible by the TX node.

| Channel | Node ID | Type | AP? |
|---|---|---|---|
| 1 | AP1 | Hidden | Yes |
| 1 | AP2 | Carrier Sense | Yes |
| 1 | AP4 | Carrier Sense | Yes |
| 1 | STA1 | Hidden | No |
| 2 | AP3 | Carrier Sense | Yes |
| 2 | STA2 | Carrier Sense | No |

Table 4.3: Node neighborhood list of hidden nodes and carrier sense nodes.

not transmitting during the passive detection, so it does not appear on the RX node's visible node list even though it is within range of the RX node. After the TX and RX nodes have finished building their visible node lists, the RX node sends its list to the TX node. By comparing the two lists, the TX node can determine which nodes are hidden nodes and which are carrier sense nodes. The resulting *node neighborhood list* for the above scenario is shown in Table 4.3.

Some hidden nodes will have too weak of a signal to impact the video transmission. For example, consider the hidden node STA1 on channel 1, which has a relatively weak signal strength of -80 dBm (for comparison, -53 dBm or higher would be considered a strong signal [30]). The impact of the hidden node can be evaluated using the following SINR equation:

$$SINR_{thr} < \frac{P_{TX}}{I_{HN} + N}, \tag{4.6}$$

where $P_{TX}$ is the received signal power from the transmitter on the initial channel, $I_{HN}$ is the signal strength of the hidden node, and the background noise $N$ can be sensed per channel. Hidden nodes will only affect transmission if they cause SINR at the receiver to fall below the SINR threshold. For example, suppose that the signal power from the transmitter measured at the receiver is -40 dBm and the background noise is -92 dBm, which is a typical amount of noise for an 802.11 network. Now consider the two hidden

| Channel | Node ID | Type | AP? |
|---|---|---|---|
| 1 | AP1 | Hidden | Yes |
| 1 | AP2 | Carrier Sense | Yes |
| 1 | AP4 | Carrier Sense | Yes |
| 2 | AP3 | Carrier Sense | Yes |

Table 4.4: Node neighborhood list with weak-signal nodes removed.

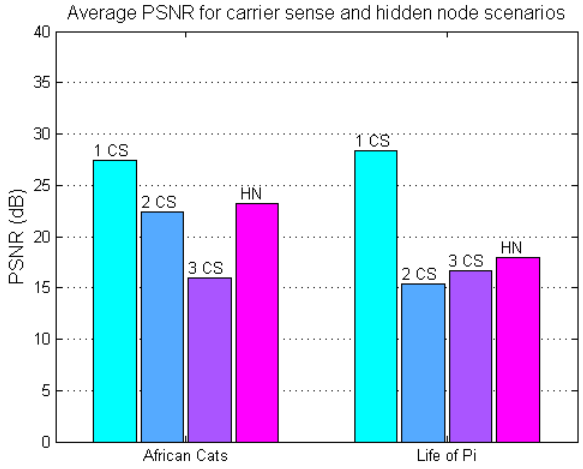



Figure 4.5: Average video PSNR of carrier sense and hidden node scenarios.

nodes on channel 1. If AP1, with a signal strength of -65 dBm, sends a packet at the same time as the video transmitter, the resulting SINR at the receiver would be 15 dB. A SINR below 20 dB is considered weak [30], so the hidden node interference from AP1 is significant. The other hidden node, STA1, has a signal strength of -80 dBm. If STA1 is transmitting, SINR at the receiver will be 40 dB – well above the threshold of 20 dB. Because STA1 has a weak signal, it can be removed from the hidden node list.

The TX node will negotiate with carrier sense nodes whose signal strength exceeds the carrier sense threshold, $P_{CSthr}$. In QualNet, the default threshold is -69 dBm for 802.11a/g transmitting at 54 Mbps [44]. Carrier sense nodes with a lower signal strength will be removed from the list. The updated list with weak-signal nodes STA1, STA2 and AP4 removed is shown in Table 4.4.

After the final node neighborhood list obtained, the TX node can select the best channel. The "best" channel is the channel with the lowest *node score*. Each hidden

node counts for 2 points, and each carrier sensing node counts for 1 point. If there is a tie between multiple channels, the channel with the fewest carrier sense nodes will be selected, because the PSNR for two carrier sensing nodes is slightly lower than the PSNR for one hidden node. Finally, if there is still a tie, the channel with the lowest average background interference at the receiver will be selected. In the above scenario, channel 1 has one hidden node and two carrier sense nodes, yielding a node score of 3. Channel 2 has one carrier sense node, yielding a node score of 1. Therefore, Channel 2 is selected as the new transmission channel. The relative node score for hidden and carrier sensing nodes was determined by observing how many carrier sense nodes were required to cause a similar drop in video quality to one hidden node; such a comparison can be seen in Fig. 4.5. The African Cats and Life of Pi videos (each encoded at 6 Mbps) were run in network scenarios containing one, two or three interfering carrier sense nodes. The average PSNR of the output videos was calculated and compared to the PSNR of the same videos run in a hidden node scenario with the hidden node transmitting at the same bitrate. The results show that while a single carrier sensing node does not affect PSNR as much as a single hidden node, two carrier sensing nodes result in an output video with slightly worse PSNR than one hidden node. Therefore, for our method two carrier sensing nodes are considered to be roughly equivalent to a hidden node, and a channel with a hidden node will be picked over a channel with two or more CS nodes in the event of a tie.

(Note: For the Life of Pi videos, the 3 CS node scenario has a slightly higher average PSNR than the 2 CS node scenario. The 3 CS scenario did result in more packet loss than the 2 CS scenario – 355 frames were lost from the 2 CS scenario and 412 were lost from the 3 CS scenario. Depending on which frames are lost, PSNR can be slightly higher even for a video with more packet loss.)

### 4.2.1 Active Link Detection Message Exchange

This subsection provides implementation details for the active link detection method.

The application-layer component of ASDCS active link detection uses TCP for all communication between the TX node and RX node. TCP, unlike UDP, guarantees *reliable delivery*, but does not provide any estimated timeframe for packet delivery. This

can be a problem for time-sensitive nature of channel switching in ASDCS. For example, there is a chance that a TCP control packet is sent to initiate a channel selection, and then the channel is changed before the packet is actually delivered. Therefore, active link detection in ASDCS uses a series of timeouts, acknowledgement packets, and state changes to ensure proper communication. ASDCS control packets are sent with a higher priority than video data packets to prevent them from being delayed.

Fig. 4.6 shows the packet exchange diagram for active link detection. Initially, the TX and RX nodes are in TX_IDLE and RX_IDLE state. Channel selection is initiated by the TX node at the beginning of the transfer during Initial Channel Selection as described in Sec. 4 or when the channel state is deemed unacceptable for video transfer during Channel Evaluation. When these conditions occur, the TX node pauses any currently transmitting video and sends PROBE_PKT to the RX node. It then enters the TX_PROBE_WFACK state, and waits for a response from the RX node. Ideally, the RX node should receive the packet, transition to the RX_PROBE_ACK state, send PROBE_ACK back to TX, delay for RX_Probe_ACK_Delay, and begin its active link detection in the RX_PROBING state. This delay is to ensure that PROBE_ACK is sent before RX changes channels. The TX node will then transition into the TX_PROBING state and perform its active link detection. If PROBE_ACK is not received from the RX node after TX_Probe_WfACK_Timeout, the TX node assumes that it cannot communicate with the RX node and transitions to the TX_PROBING state. After completing the active link detection and returning to the initial channel, the TX node will again attempt to communicate with the RX node and wait in the TX_PROBE_WFRX state until it has received the visible node list from the RX node.

After acquiring both node lists, the TX node will compare the two node lists to generate the node neighborhood list as shown in Table 4.4. If the current channel is chosen as the "best" channel, TX will remain on the same channel and return to the TX_IDLE state. Otherwise, TX will send the new channel information to RX using the CHANGE_PKT message. After receiving CHANGE_PKT from TX, RX will send a CHANGE_ACK message on the current channel, delay for RX_Change_ACK_Delay, change to the new channel, and send a VERIFY_ACK message on the new channel before returning to the RX_IDLE state. TX will wait for a CHANGE_ACK message on the current channel until TX_Change_WfACK_Timeout. TX then moves to the new channel and waits for a VERIFY_ACK message. If VERIFY_ACK is not received on

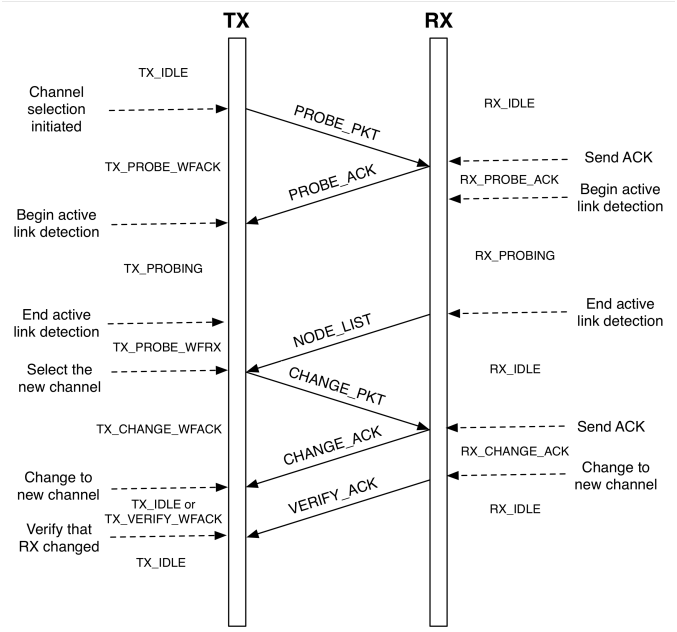



Figure 4.6: Packet exchange diagram for active link detection.

| **PROBE_PKT Format** | | | |
|---|---|---|---|
| **Fields** | **Size** | **Bit(s)** | **Meaning** |
| PROBE_PKT ID | 1 byte | 15 - 0 | Probe Packet ID |
| Channel Mask | 2 bytes | 15 - 14 | Reserved |
| | | 13 - 0 | $n^{th}$ bit corresponds to $n + 1^{st}$ channel; 0 = Do not scan this channel, 1 = Scan this channel |
| Flags | 1 byte | 7 - 1 | Reserved |
| | | 0 | Initial Channel Switch Identifier: 0 = Initial channel switch, 1 = Mid-stream channel switch |

Table 4.5: Format of PROBE_PKT.

the new channel before TX_Verify_WfACK_Timeout, TX returns to the previous channel. TX will continue checking both channels until it receives either CHANGE_ACK or VERIFY_ACK; upon receiving one, it returns to the TX_IDLE state on the new channel.

Table 4.5 contains information about the PROBE_PKT control packet sent by the TX node when it wishes to initiate a channel change. The first byte identifies the packet as PROBE_PKT, the second two bytes comprise the channel switching mask, and the final byte contains the flag bits. The channel switching mask designates the channels to

| NODE_LIST Format | | | |
|---|---|---|---|
| **Fields** | **Size** | **Bit(s)** | **Meaning** |
| NODE_LIST ID | 1 byte | 15 - 0 | Node List ID |
| RX MAC address | 6 bytes | 47 - 0 | MAC address of RX node |
| Node count | 2 bytes | 31 - 0 | Number of node list entries |
| Node list entry | 16 bytes | 127 - 119 | Channel |
| | | 119 - 71 | MAC address |
| | | 70 - 8 | Signal strength |
| | | 7 - 1 | Reserved |
| | | 0 | isAP identifier: 0 = not an AP, 1 = is an AP |

Table 4.6: Format of NODE_LIST.

| CHANGE_PKT Format | | | |
|---|---|---|---|
| **Fields** | **Size** | **Bit(s)** | **Meaning** |
| CHANGE_PKT ID | 1 byte | 15 - 0 | Probe Packet ID |
| New channel | 1 byte | 15 - 0 | New channel |
| **ACK PKT Format** | | | |
| ACK Type | 1 byte | 15 - 0 | ACK type identifier |

Table 4.7: Format of other control packets.

scan and select from. The *Initial Channel Switch Identifier* flag denotes whether this is an initial channel switch or a mid-stream channel switch.

Table 4.6 shows the information in the NODE_LIST packet sent from the RX node when it has finished active link detection, which consists of a header followed by a variable number of node list entries. The header contains the NODE_LIST packet identifier, the MAC address of the RX node (which is used by the TX to identify the RX in its visible node list), and a count of the number of nodes discovered. Each node list entry contains the channel the node is on, the node's MAC address, the node's signal strength (stored as an 8-byte double precision floating point value), and an identifier of AP or non-AP node.

The format of the CHANGE_PKT message is shown in Table 4.7, which consists of the CHANGE_PKT identifier and the new channel. All ACK packets are a single byte that identifies the packet as a PROBE_ACK, CHANGE_ACK, or VERIFY_ACK.

# Chapter 5: Simulation study

This section evaluates the effectiveness of the proposed ASDCS in a number of scenarios.

## 5.1 Simulation Environment

The proposed ASDCS was implemented and evaluated using *Open Evaluation Framework for Multimedia Over Networks* (OEFMON) [45], which integrates Microsoft multimedia framework DirectShow [46] and network simulator QualNet 7.3 [47]. Together, they provide visualization of the underlying network details and on-the-fly display of sent and received videos.

A simplified diagram of OEFMON is shown in Fig. 5.1. OEFMON requires the following inputs: A video file in YUV (raw) or H.264 (encoded) format, a QualNet scenario file, a QoS mapping parameter file, and a DirectShow graph. The three outputs generated are the received raw video file, a sender log, and a receiver log, which are used for on-line analysis to compute PSNR, throughput, delay, and packet loss ratio among other metrics. Streaming video transmission trials are run in OEFMON to test how video streams react to hidden node and carrier sensing interference. Video quality can only be directly evaluated on a video that has finished streaming, using a measurement such as Peak Signal-to-Noise Ratio (PSNR). One of the goals when developing ASDCS was to identify a metric that can be measured in real-time and closely correlates with PSNR measured afterward.

QualNet simulations model the entire network protocol stack, from the application layer to the physical layer. This allows for the analysis of many different network scenarios. OEFMON adds the Video traffic generator to QualNet, allowing for the simulation of a real-time video stream between two nodes and outputting the received raw video, sender log and receiver log.

The first video used for simulation was a 13-second clip from the African Cats video ($1920 \times 1080$ @30 fps) containing 365 frames. The video was encoded at a variable bit rate averaging 6 Mbps. The second test video was a 43-second clip of the *Life of Pi* trailer

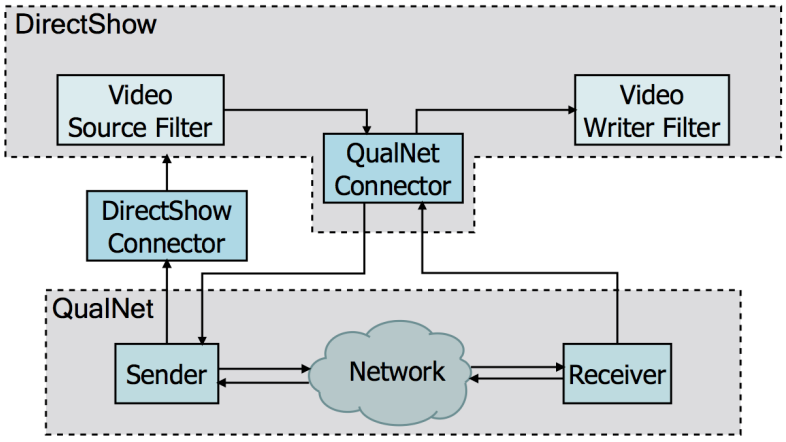



Figure 5.1: The general structure of OEFMON.

(1920 × 1080 @24 fps) containing 1,081 frames encoded at the same variable bitrate as the first test video. The third test video was a 45-second clip of the *Star Wars: The Force Awakens* trailer (1920 × 1080 @24 fps) containing 1150 frames encoded at a variable bit rate averaging 4 Mbps. Figs. 5.2, 5.3, 5.4, 5.5, and 5.6 show the five network scenarios created to test the proposed method. The hidden node scenario (Fig. 5.2) contains the primary node pair transmitting video and a hidden node pair that is continually transmitting constant bit rate (CBR) data (see Table 5.1). The carrier sensing scenario (Fig. 5.3) replaces the hidden node pair with three carrier sense node pairs. The mixed scenario (Fig. 5.4) includes one hidden node pair and two carrier sense node pairs. These three basic scenarios were used to observe how accurately the ASDCS Video Quality Prediction is able to detect declines in video quality under different kinds of interference conditions. As described in Sec. 4.1, video quality is predicted to become bad when the percentage of packets lost from an I-frame exceeds $I_{loss}$ or when more than $P_{lost}$ P-frames or $B_{lost}$ B-frames are lost consecutively. In each of the three basic scenarios, all node pairs transmit on channel 6.

The first multichannel scenario (Fig. 5.5) contains a video transmitting node pair initially transmitting on channel 1, a hidden node pair on channel 1, and a hidden node pair on channel 6; channel 11 contains no transmitting nodes. The second multichannel

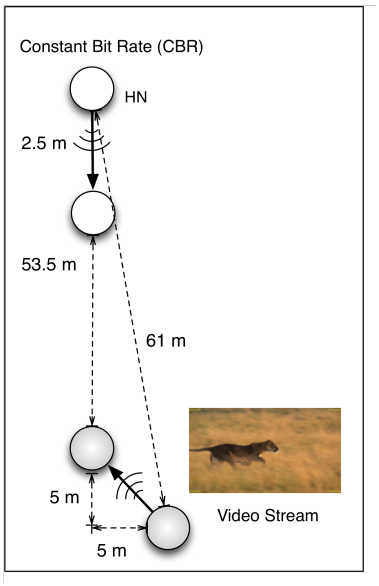



Figure 5.2: Hidden node scenario with one hidden node pair.

scenario (Fig. 5.6) is the same as the first, but one carrier sensing node pair is added on channel 11. Nodes 6 and 10 in the multichannel scenarios are APs and all other nodes are non-AP. These two multichannel scenarios were used to evaluate the performance of Active Link Detection (Sec. 4.2).

The H.264 encoded video streams were sent via Real-time Transfer Protocol (RTP) over UDP. RTP is frequently used for streaming media and includes timestamp functionality for stream synchronization. Each H.264 NAL unit is directly encapsulated into one or more RTP packets for delivery. UDP is typically used instead of TCP for real-time applications because TCP can incur long delays when retransmitting lost data. ASDCS control packets were sent over TCP (as described in Sec. 4.2.1) to guarantee reception. The video transmitting node uses two outgoing IP queues: a high-priority queue for ASDCS control packets and a low-priority queue for video data. The packet scheduler is Strict Priority, meaning that packets from the higher priority queue will *always* be sent before packets from the lower priority queue. This prevents ASDCS control packets from being backed up behind many video packets. When packets are lost within frames, the H.264 decoder will attempt to compensate using WPA and motion vector recovery as described in Sec. 2.1. When entire frames are lost, the previous frame is duplicated,

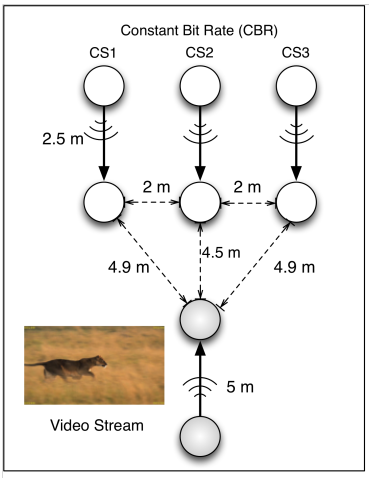



Figure 5.3: Carrier sensing scenario with three carrier sense node pairs.

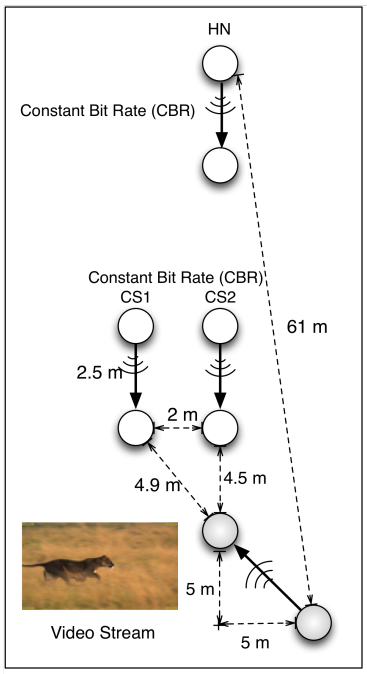



Figure 5.4: Mixed scenario with one hidden and two carrier sense node pairs.

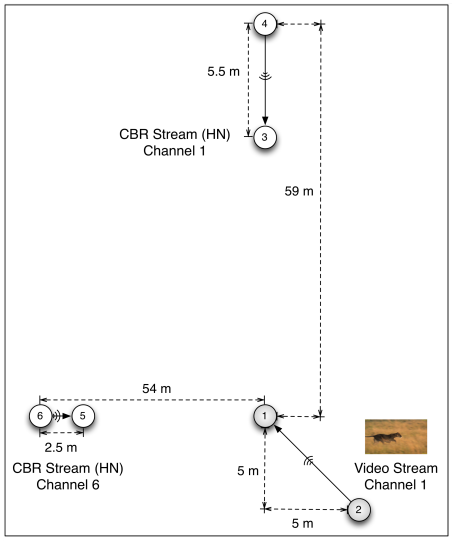



Figure 5.5: Multichannel scenario 1 with two hidden node pairs.

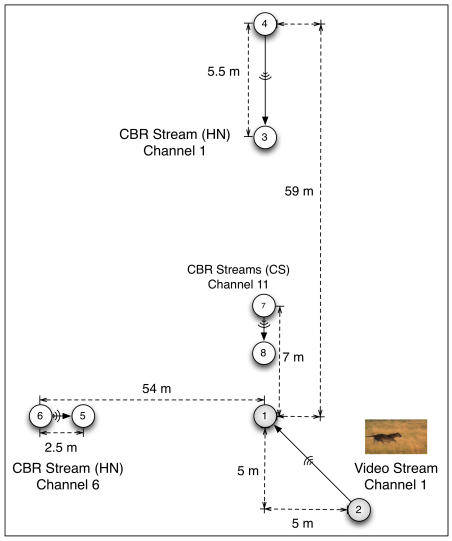



Figure 5.6: Multichannel scenario 2 with two hidden and one carrier sense node pairs.

which is a standard decoder behavior.

The default values for various ASDCS parameters are shown in Table 5.2. $I_{lost}$, $P_{lost}$, and $P_{lost}$ are set to default threshold values for Video Quality Prediction (see Sec. 4.1).

| **Hidden node scenario** | | | |
|---|---|---|---|
| **Video** | **HN CBR bitrate** | | |
| *African Cats* (6 Mbps) | 13.8 Mbps | | |
| *Life of Pi* (6 Mbps) | 13.8 Mbps | | |
| **Carrier sensing scenario** | | | |
| **Video** | **CS CBR bitrate** | | |
| *African Cats* (6 Mbps) | 13.8 Mbps | | |
| *Life of Pi* (6 Mbps) | 13.8 Mbps | | |
| **Mixed scenario** | | | |
| **Video** | **HN CBR bitrate** | **CS CBR bitrate** | |
| *African Cats* (6 Mbps) | 6.9 Mbps | 6.9 Mbps | |
| *Life of Pi* (6 Mbps) | 6.9 Mbps | 6.9 Mbps | |
| **Multichannel scenario 1** | | | |
| **Video** | **HN Ch.1 CBR** | **HN Ch. 6 CBR** | |
| *African Cats* (6 Mbps) | 13.8 Mbps | 13.8 Mbps | |
| *Life of Pi* (6 Mbps) | 13.8 Mbps | 13.8 Mbps | |
| **Multichannel scenario 2** | | | |
| **Video** | **HN Ch.1 CBR** | **HN Ch.6 CBR** | **CS CBR** |
| *Star Wars* (4 Mbps) | 15.7 Mbps | 15.7 Mbps | 15.7 Mbps |
| *Life of Pi* (6 Mbps) | 13.8 Mbps | 13.8 Mbps | 13.8 Mbps |

Table 5.1: Bitrates for the five scenarios.

MinChannelTime and MaxChannelTime are channel scanning times used in the 802.11 standard. The timeout (TX_Probe_WfACK_Timeout, TX_Change_WfACK_Timeout, TX_Change_WfACK_Timeout) and delay (RX_Probe_ACK_Delay, RX_Change_ACK_Delay) parameters were established by running simulation trials. $P_{CSthr}$ of -69 dBm is the default carrier sensing threshold in QualNet's 802.11g model [44], and $SINR_{thr}$ of 20 dB is a standard threshold for "acceptable" signal strength at an 802.11 node [30]. A channel switch backoff time of 10 seconds is used to limit the frequency of channel changes. The channel mask is set to scan the non-overlapping channels 1, 6 and 11 when searching for a new channel. (The QualNet simulator does not simulate the effects of interference from overlapping channels, and ASDCS does not account for it during channel selection, so we have opted to omit overlapping channels for these simulations.)

| Name | Value | Description |
|---|---|---|
| $I_{loss}$ | 20% | Maximum acceptable loss from I-frame |
| $P_{lost}$, $B_{lost}$ | 2 | Maximum acceptable number of P- or B- frames lost |
| MinChannelTime | 6 ms | Minimum time to scan one channel |
| MaxChannelTime | 24 ms | Maximum time to scan one channel |
| TX_Probe_WfACK_Timeout | 1000 ms | TX timeout when waiting for PROBE_ACK |
| TX_Change_WfACK_Timeout | 200 ms | TX timeout when waiting for CHANGE_ACK |
| TX_Change_WfACK_Timeout | 200 ms | TX timeout when waiting for VERIFY_ACK |
| RX_Probe_ACK_Delay | 5 ms | RX delay after sending PROBE_ACK |
| RX_Change_ACK_Delay | 5 ms | RX delay after sending CHANGE_ACK |
| $P_{CSthr}$ | -69 dBm | Minimum signal strength for an interfering carrier sense node |
| $SINR_{thr}$ | 20 dB | Minimum SINR for an interfering hidden node |
| Channel switch backoff time | 10 s | Minimum time between channel switches |
| Channel mask | 1, 6, 11 | Channels to scan |

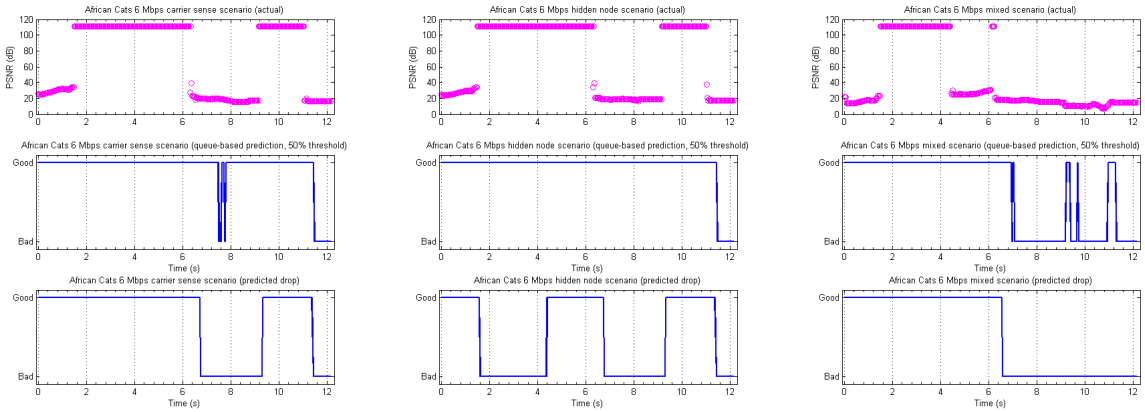
Table 5.2: ASDCS default values.

## 5.2 Simulation Results

The first set of simulations demonstrate the accuracy of the Video Quality Prediction method used for Channel Evaluation discussed in Sec. 4.1. The purpose of these simulations was to evaluate how Video Quality Prediction performed with varying video bitrates, types of interference (hidden or carrier sense) and video lengths. Predicted video quality, as estimated from the transmitter's perspective during the course of a simulation, is compared to a simple queue-based quality prediction method and the actual video PSNR in Figs. 5.7 and 5.8. For the simple queue-based method, video quality is predicted to become "bad" if the number of bytes in the queue exceeds a fixed threshold (in this case, if 50% or more of the queue is filled).

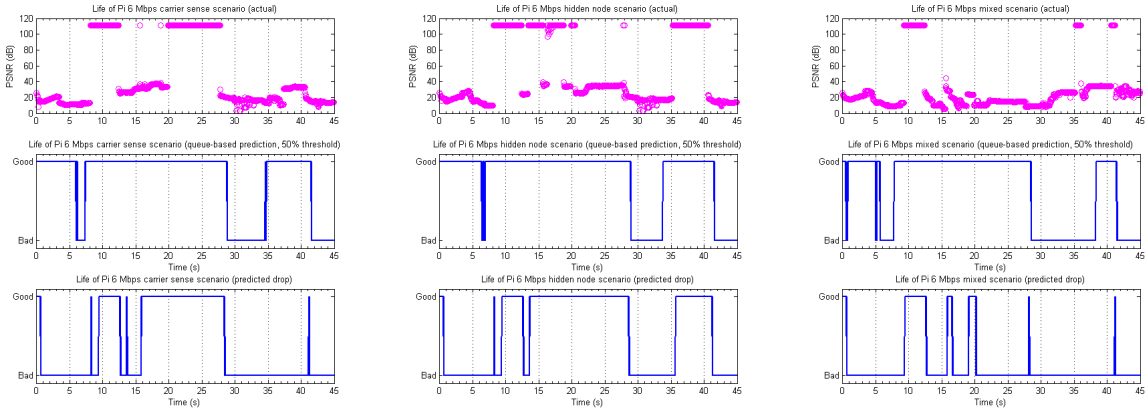
In all the video scenarios tested, Video Quality Prediction performed quite favorably, successfully predicting periods of reduced PSNR in most cases. It is important to note that Video Quality Prediction does not need to be completely accurate; it just needs to signal that video quality has dropped to an unacceptable level, so that the New Channel Selection process can begin. For example, in the Life of Pi carrier sense scenario, the PSNR drop around 12.7 seconds lasts longer than Video Quality Prediction predicts, but New Channel Selection would still be activated when quality was predicted bad.

Because Video Quality Prediction estimates bandwidth from measured samples, it cannot predict video quality loss at $t = 0$, when no measurements are available yet; this can be seen at the beginning of the three African Cats videos. The first I-frame (frame




(a) Carrier sense scenario (b) Hidden node scenario (c) Mixed scenario

Figure 5.7: Predicted vs. actual video quality for the *African Cats* video in three interference scenarios.




(a) Carrier sense scenario (b) Hidden node scenario (c) Mixed scenario

Figure 5.8: Predicted vs. actual video quality for the *Life of Pi* video in three interference scenarios.

0) of all three African Cats videos experiences packet loss that is not detected by Video Quality Prediction. Because the following P- and B- frames use the I-frame as a basis for decoding, they experience a drop in PSNR even though they are fully received. The

| **Multichannel scenario 1** | | | |
|---|---|---|---|
| **Channel** | **Node ID** | **Type** | **AP?** |
| 1 | Node 4 | Hidden | No |
| 6 | Node 6 | Hidden | Yes |
| **Multichannel scenario 2** | | | |
| **Channel** | **Node ID** | **Type** | **AP?** |
| 1 | Node 4 | Hidden | No |
| 6 | Node 6 | Hidden | Yes |
| 11 | Node 7 | Carrier Sense | Yes |

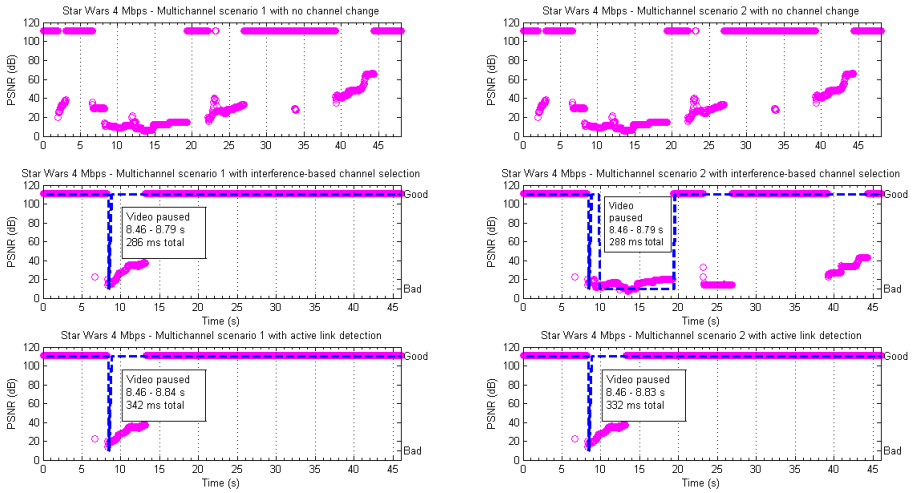
Table 5.3: Node neighborhood lists for multichannel scenarios.

result is a loss of PSNR until the next I-frame (frame 44, 1.46 seconds) is received. This problem can be mitigated by using Initial Channel Selection (Sec. 4.2), which attempts to select a channel before video begins transmitting.

Compared to the simple queue-based prediction method, Video Quality Prediction is generally more accurate and less volatile. In all of the scenarios, Video Quality Prediction identified sections of poor quality that the queue-based prediction did not. However, in the African Cats hidden node scenario, Video Quality Prediction had a false positive result, incorrectly identifying a section of poor quality at 1.6 seconds. In addition, Video Quality Prediction occasionally has "spikes" of predicted good channel quality in an otherwise bad section, which can be seen in the Life of Pi scenarios. This indicates that there are still adjustments to be made to the Video Quality Prediction thresholds.

The second set of simulations demonstrates Active Link Detection in two multichannel scenarios. The video TX node is initially transmitting on Channel 1, where a high bitrate hidden node is interfering with the transmission of the primary node pair. Channels 1, 6 and 11 are available as transmission channels. Predicted Video Quality is used to determine when the video quality drops and the channel should be changed. Initial Channel Selection is disabled for these scenarios in order to observe performance of the mid-stream channel switch. Active Link Detection produces the node neighborhood lists shown in Table 5.3. In each case, channel 11 is selected as the new channel. As anticipated, video quality remains low until the next I-frame. Setting a channel switching backoff time of 10 seconds prevents too many channel switch attempts in a short time period.

Simulation results for the *Star Wars* and *Life of Pi* videos in both scenarios are



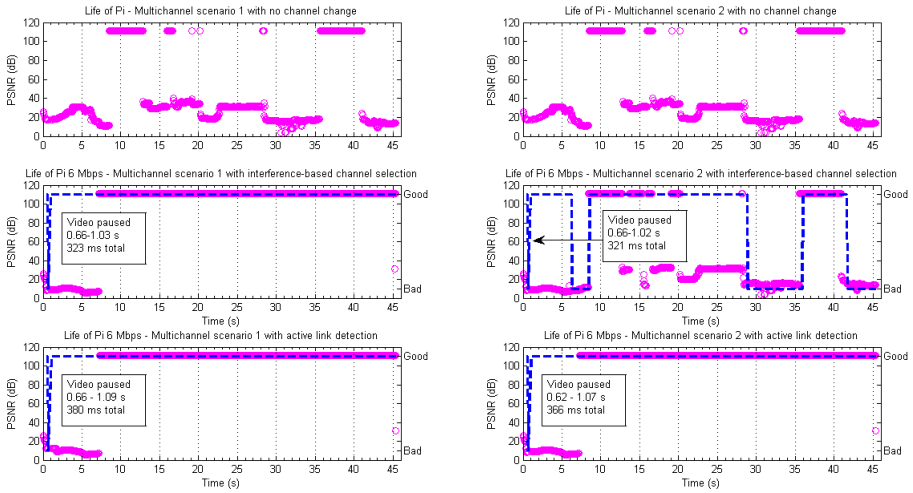

(a) Multichannel scenario 1 (b) Multichannel scenario 2

Figure 5.9: Predicted vs. actual video quality for the *Star Wars* video in both multichannel scenarios.

shown in Figures 5.9 and 5.10. Results from the Active Link Detection method are shown alongside the original PSNR (if the channel was not changed), and the PSNR using a simple interference-based channel selection method (Appendix A). Predicted Video Quality is overlaid as a dotted line. In the interference-based method, the RX node measures received interference power from interfering nodes on all available channels and simply selects the channel with the lowest measured interference.

In each graph, there are gaps in the PSNR measurements showing where the video was paused for interference-based channel selection and Active Link Detection. Each video was paused for 332-380 ms while Active Link Detection took place. Compared to the video quality degradation that was occurring on the initial bad channel, this delay represents a better QoE and would be perceived from the user perspective as an acceptable buffering event for a streaming video which can last several minutes. This pause time is kept close to minimum by placing control packets in a higher priority queue so that the channel switch is not delayed by the backlog of video packets.

Because the second multichannel scenario has significant interference on all three




(a) Multichannel scenario 1 (b) Multichannel scenario 2

Figure 5.10: Predicted vs. actual video quality for the *Life of Pi* video in both multichannel scenarios.

channels, it provides an interesting 'non-ideal' test case in which no channel is completely free of interference. In the interference-based method, the RX node listens for interference on both channels. It senses greater interference from the nearby carrier sensing node on channel 11 than the hidden node on channel 6, and selects channel 6 as the new transmission channel. On the other hand, the active link detection method identifies a hidden node on channel 6 and a carrier sensing node on channel 11, determines that channel 11 has a lower node score, and selects channel 11 as the new channel. Channel 11 is the best choice because a single hidden node will cause more packet loss than a single carrier sensing node. In this scenario, Active Link Detection's ability to positively detect hidden nodes allowed it to make a better channel selection.

The results from the multichannel scenarios demonstrate that ASDCS is capable of identifying poor video quality, scanning channels for interfering nodes, and switching to a better channel in an acceptable period of time. If none of the available channels are suitable for HD video transmission, ASDCS will periodically check until a better channel becomes available.

# Chapter 6: Conclusion and Future Work

In this thesis we propose ASDCS, a two-part method for dynamically switching channels during real-time wireless video streaming. ASDCS uses the IP queue to predict received video quality for upcoming frames and, if predicted video quality is poor, uses active link detection to select a new channel with the fewest interfering nodes in a reasonable timeframe. Simulation results found Predicted Video Quality to be very accurate and Active Link Detection to be reasonably fast and successful at discovering and avoiding interfering hidden nodes.

Results from ASDCS simulation are preliminary but promising. The next step is to continue refining and improving the method based on what was learned from simulation performance. In particular, the Video Quality Prediction thresholds may be fine-tuned by analyzing simulation data. Other potential avenues for future work include testing more advanced scenarios, incorporating node mobility, revisions for high-speed 802.11ac and implementing ASDCS on physical hardware.

# Bibliography

[1] I. E. Richardson, *The H.264 Advanced Video Compression Standard, Second Edition*. Wiley, Mar. 2010.

[2] List of WLAN channels - Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/List_of_WLAN_channels

[3] Apple - AirPlay — Play content from iOS devices on Apple TV. (Visited on 10/17/2013). [Online]. Available: http://www.apple.com/airplay/

[4] Intel® Wireless Display and Intel® Pro Wireless Display. (Visited on 10/17/2013). [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/intel-wireless-display.html

[5] Chromecast. (Visited on 10/17/2013). [Online]. Available: http://www.google.com/intl/ja/chrome/devices/chromecast/#netflix

[6] The WirelessHD Consortium serves to organize an industry-led standardization effort to define a next-generation wireless digital interface specification for consumer electronics and PC products. (Visited on 10/17/2013). [Online]. Available: http://www.wirelesshd.org/

[7] WHDI$^{TM}$- Wireless High Definition. (Visited on 10/17/2013). [Online]. Available: http://www.whdi.org/

[8] Wi-Fi Alliance. (Visited on 10/17/2013). [Online]. Available: http://www.wi-fi.org/

[9] IEEE P802.11 - TASK GROUP AC. [Online]. Available: http://www.ieee802.org/11/Reports/tgac_update.htm

[10] US Mobile Data Market Update Q3 2012. (Visited on 10/18/2013). [Online]. Available: http://www.chetansharma.com/usmarketupdateq32012.htm

[11] C. Yoon, T. Um, and H. Lee, "Classification of N-Screen Services and its standardization," in *Advanced Communication Technology (ICACT), 2012 14th International Conference on*, 2012, pp. 597–602.

[12] How Does 3D TV work - A Beginner's Guide — Amazon.com. (Visited on 10/18/2013). [Online]. Available: http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000492751

[13] Ultra HDTV: What is Ultra High Definition, Ultra HD, 4K, 8K? (Visited on 10/18/2013). [Online]. Available: http://www.ultrahdtv.net/what-is-ultra-hdtv/

[14] "IEEE Std 802.11n$^{TM}$-2009, IEEE Standard for Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements," 2009.

[15] F. Y. Li, A. Kristensen, and P. Engelstad, "Passive and active hidden terminal detection in 802.11-based ad hoc networks," in *IEEE INFOCOM*, 2006.

[16] inSSIDer for Home – Discover The Wi-Fi Around You. (Visited on 10/16/2013). [Online]. Available: http://www.metageek.net/products/inssider/

[17] N. Jain, S. Das, and A. Nasipuri, "A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks," in *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*, 2001, pp. 432–439.

[18] A. Nasipuri and J. Mondhe, "Multi-channel MAC with dynamic channel selection for ad hoc networks," *Techinical report, http://www. ece. uncc. edu/anasipur*, 2004.

[19] V. Kanodia, A. Sabharwal, and E. Knightly, "MOAR: A multi-channel opportunistic auto-rate media access protocol for ad hoc networks," in *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on.* IEEE, 2004, pp. 600–610.

[20] M. Ihmig and P. Steenkiste, "Distributed dynamic channel selection in chaotic wireless networks," in *13th European Wireless Conference, Paris, France*, 2007.

[21] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot, "Measurement-based self organization of interfering 802.11 wireless access networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE.* IEEE, 2007, pp. 1451–1459.

[22] M. Sarkar, S. Nagaraj, and I. H. Balsania, "A SINR based MAC layer protocol for multi-channel ad-hoc networks," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International.* IEEE, 2011, pp. 1889–1893.

[23] Q. Wang and M. Liu, "Throughput optimal switching in multi-channel WLANs," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2011 International Symposium on*, 2011, pp. 383–388.

[24] J. Mo, H.-S. So, and J. Walrand, "Comparison of Multichannel MAC Protocols," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 1, pp. 50–65, 2008.

[25] S. Wiwatthanasaranrom and A. Phonphoem, "Multichannel MAC protocol for ad-hoc wireless networks," in *Proc. National Computer Science and Engineering Conf*, 2003, pp. 115–120.

[26] G. Athanasiou, I. Broustis, T. Korakis, and L. Tassiulas, "LAC: Load-aware channel selection in 802.11 WLANs," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, 2008, pp. 1–6.

[27] R. Maheshwari, H. Gupta, and S. Das, "Multichannel MAC Protocols for Wireless Networks," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, vol. 2, 2006, pp. 393–401.

[28] J. So and N. H. Vaidya, "Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '04. New York, NY, USA: ACM, 2004, pp. 222–233. [Online]. Available: http://doi.acm.org/10.1145/989459.989487

[29] W.-H. Liao and W.-C. Chung, "An efficient multi-channel mac protocol for mobile ad hoc networks," in *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, vol. 2, Jan 2009, pp. 162–166.

[30] J. Geier. How to: Define Minimum SNR Values for Signal Coverage. [Online]. Available: http://www.wireless-nets.com/resources/tutorials/define_SNR_values.html

[31] R. M. Kortebi, Y. Gourhant, and N. Agoulmine, "On the use of SINR for interference-aware routing in wireless multi-hop networks," in *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*. ACM, 2007, pp. 395–399.

[32] M. Kim and C.-H. Choi, "Hidden-Node Detection in IEEE 802.11n Wireless LANs," *Vehicular Technology, IEEE Transactions on*, vol. 62, no. 6, pp. 2724–2734, 2013.

[33] M. H. Manshaei, T. Turletti, and T. Guionnet, "An evaluation of media-oriented rate selection algorithm for multimedia transmission in MANETs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2005, no. 5, pp. 757–773, 2005.

[34] G. C. Lee and H. Song, "An effective cross layer-based video streaming algorithm over mobile ad hoc network," in *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*. IEEE, 2009, pp. 1–5.

[35] S. Krishnamachari, M. van der Schaar, S. Choi, and X. Xu, "Video streaming over wireless LANs: a cross-layer approach," in *Proc. Packet Video Workshop*, 2003.

[36] E. Setton, X. Zhu, and B. Girod, "Minimizing distortion for multi-path video streaming over ad hoc networks," in *Image Processing, 2004. ICIP'04. 2004 International Conference on*, vol. 3. IEEE, 2004, pp. 1751–1754.

[37] M. Qin and R. Zimmermann, "An adaptive strategy for mobile ad hoc media streaming," *Multimedia, IEEE Transactions on*, vol. 12, no. 4, pp. 317–329, 2010.

[38] B. J. Oh and C. W. Chen, "Performance evaluation of H. 264 video over ad hoc networks based on dual mode IEEE 802.11 B/G and EDCA MAC architecture," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 3510–3513.

[39] A. Fiandrotti, D. Gallucci, E. Masala, and E. Magli, "Traffic Prioritization of H.264/SVC Video over 802.11e Ad Hoc Wireless Networks," in *Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on*, 2008, pp. 1–5.

[40] B. Ciubotaru, G. Muntean, and G. Ghinea, "Objective assessment of region of interest-aware adaptive multimedia streaming quality," *Broadcasting, IEEE Transactions on*, vol. 55, no. 2, pp. 202–212, June 2009.

[41] S. Chikkerur, V. Sundaram, M. Reisslein, and L. Karam, "Objective video quality assessment methods: A classification, review, and performance comparison," *Broadcasting, IEEE Transactions on*, vol. 57, no. 2, pp. 165–182, June 2011.

[42] G. Muntean, "Efficient delivery of multimedia streams over broadband networks using qoas," *Broadcasting, IEEE Transactions on*, vol. 52, no. 2, pp. 230–235, June 2006.

[43] G. Muntean and N. Cranley, "Resource efficient quality-oriented wireless broadcasting of adaptive multimedia content," *Broadcasting, IEEE Transactions on*, vol. 53, no. 1, pp. 362–368, March 2007.

[44] *QualNet 5.0.2 Wireless Model Library*, Scalable Network Technologies, Inc., 6100 Center Drive, Suite 1250 Los Angeles, CA 90045, March 2010.

[45] C. Lee, M. Kim, S. J. Hyun, S. Lee, B. Lee, and K. Lee, "OEFMON: An open evaluation framework for multimedia over networks," *Communications Magazine, IEEE*, vol. 49, no. 9, pp. 153–161, 2011.

[46] DirectShow. (Visited on 10/22/2013). [Online]. Available: http://msdn.microsoft.com/en-us/library/ms783323.aspx

[47] QualNet — SCALABLE Network Technologies. [Online]. Available: http://web.scalable-networks.com/content/qualnet

# APPENDICES

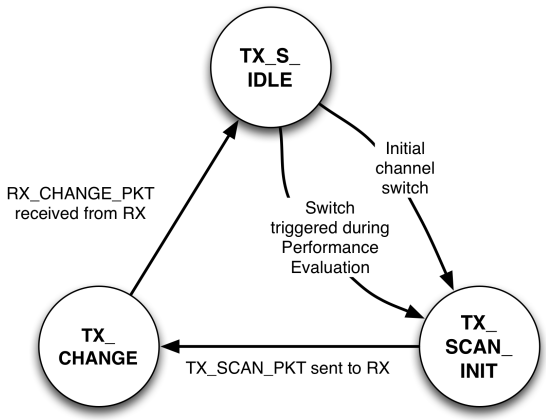



Figure A.1: Interference-based TX node state machine.

# Appendix A: Interference-based implementation details

Appendix A provides implementation details for the Interference-based channel switching method.

## A.1 Interference-based method overview

The Interference-based channel switching method was implemented to provide a comparison method to ASDCS. It is based on the principle of measuring the received interference from background noise and surrounding nodes at the RX node in order to find a clear channel. The channel with the lowest interference is considered to be the "best" channel. Interference measured at the receiver is used as a channel quality metric in previously existing channel switching methods [17, 18, 22], making it a good base of comparison for the new method.

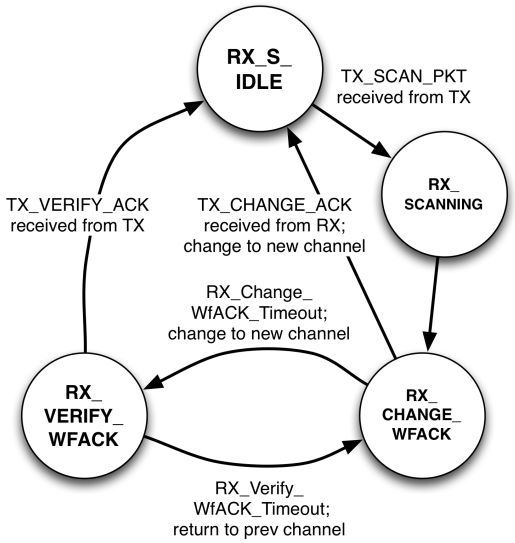



Figure A.2: Interference-based RX node state machine.

## A.2 Interference-based state machines

The Interference-based channel switching method uses TCP for communication between sender and receiver. During normal transmission, the TX and RX nodes are both in the idle state. The TX node initiates a channel switch either at the beginning of a transmission (Initial Channel Selection) or midway through transmission if the channel quality is deemed unacceptable (Performance Evaluation). To initiate the channel scan the TX node sends TX_SCAN_PKT to RX. Upon receiving the packet, RX will scan through all the channels selected in the channel mask for RX_Scan_Chan_Sample_Time per channel. RX will sample the total received interference power (i.e., noise plus node interference) on the channel every RX_Scan_Interval and determine the received interference power on each channel. The channel with the lowest received interference power will be selected as the new channel. This is equivalent to selecting the channel with the highest SINR, assuming received signal power from TX is the same on every channel.

After RX has determined the new channel, it will send its channel selection back to TX in RX_CHANGE_PKT. After receiving RX_CHANGE_PKT from RX, TX will send TX_CHANGE_ACK on the current channel, delay TX_Change_ACK_Delay, change to the new channel, and send TX_VERIFY_ACK on the new channel before return-

| **Information element** | **Size (bytes)** |
|---|---|
| TX_SCAN_PKT ID | 1 |
| Channel mask | 2 |
| **Channel mask bit** | **Definition** |
| $0-13$ | Each bit $n$ corresponds to the $n+1^{th}$ channel on 802.11 channels $1-14$<br>0 = Do not scan this channel<br>1 = Scan this channel |
| $14-15$ | Reserved |

Table A.1: Format of TX_SCAN_PKT.

| **RX_CHANGE_PKT information element** | **Size (bytes)** |
|---|---|
| RX_CHANGE_PKT ID | 1 |
| New channel | 1 |
| **ACK packet information element** | **Size (bytes)** |
| ACK type identifier | 1 |

Table A.2: Format of other Interference-based control packets.

ing to TX_S_IDLE. RX will wait for TX_CHANGE_ACK on the current channel until RX_Change_WfACK_Timeout. RX then moves to the new channel and waits for TX_VERIFY_ACK. If TX_VERIFY_ACK is not received on the new channel before RX_Verify_WfACK_Timeout, RX returns to the previous channel. The reason for this is because TCP delay is unknown to the application layer, so there is a possibility that RX_CHANGE_ACK had not yet been received by TX and TX had not changed channels. RX will continue checking both channels until it receives either TX_CHANGE_ACK or TX_VERIFY_ACK; upon receiving one it returns to RX_S_IDLE on the new channel.

## A.3 Interference-based packet format and default values

To initiate a channel scan, TX sends TX_SCAN_PKT (Table A.1) to RX. This packet consists of three bytes. The first byte identifies the packet as TX_SCAN_PKT. The second two bytes are the channel mask telling RX which channels should be scanned.

The format of the Interference-based method control packets are shown in Table A.2. RX_CHANGE_PKT, shown in Table A.2a consists of the RX_CHANGE_PKT identifier and the new channel. All ACK packets are a single byte which identifies the packet as

| Name | Value | Description |
|---|---|---|
| RX_Scan_Chan_Sample_Time | 24 ms | Time for RX to scan each channel |
| RX_Scan_Interval | 100 us | Sampling interval when scanning channel |
| RX_Change_WfACK_Timeout | 200 ms | RX timeout when waiting for TX_CHANGE_ACK |
| RX_Verify_WfACK_Timeout | 200 ms | RX timeout when waiting for TX_VERIFY_ACK |
| TX_Change_ACK_Delay | 5 ms | RX delay after sending TX_CHANGE_ACK |

Table A.3: Interference-based default values.

a PROBE_ACK, CHANGE_ACK or VERIFY_ACK.

The default values for various Interference-based parameters can be found in Table A.3. RX_Scan_Chan_Sample_Time is the time the RX spends sampling the interference on each channel; RX_Scan_Interval is the frequency with which the interference is sampled. RX_Change_WfACK_Timeout and RX_Verify_WfACK_Timeout are the timeouts while waiting for TX_CHANGE_ACK and TX_Verify_ACK, respectively. TX_Change_ACK_Delay is the time the TX delays after sending TX_CHANGE_ACK and before changing to the new channel.