Multi-Channel Wireless Mesh Networks with TCP Proxies

by

Adam Kohn

Bachelor of Engineering, Ryerson University, 2007

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Masters of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Multi-Channel Wireless Mesh Networks with TCP Proxies

Masters of Applied Science 2010

Adam Kohn

Electrical and Computer Engineering

Ryerson University

Wireless mesh networks based on 802.11 technology could potentially be an inexpensive means of constructing large-scale wireless infrastructure networks. Wireless mesh networks attempt to capitalize on multiple hop communication to achieve transmissions over relatively larger distances. One fundamental concern is that multi-hop wireless networks may suffer heavily from co-channel interference. If multiple channels from the 802.11 spectrum are employed across adjacent links of communication, the interference effects can be mitigated. In practice, either overlapping channels or independent orthogonal channels can be assigned to the different links with varying effects. Topology control can be used to help manage these channels to limit the interference effects while providing for the necessary capacity and scalability requirements. By means of analyses and testbed experiments, I have validated that the introduction of multiple channels can improve overall system performance. With respect to the end-users, end-to-end performance over multiple wireless hops should be the primary concern. Under UDP-based communication sessions, network congestion is not the main contributor to transport layer performance degradation. Upon further investigation, TCP performance degrades exponentially with hop count, because it incorrectly interprets lost packets as a sign of network congestion. Since TCP performance weakens for connections with more wireless hops, I further evaluate if network performance can be improved by adding an n-hop TCP proxy service. These proxies have the effect of breaking long connections into shorter connections with tighter transport layer control. A trade-off between the number of proxies and the hop count between proxies becomes evident through testbed evaluation. Analyzing various mesh characteristics and the relationships between MAC and transport layers can help establish a suitable protocol for future work.

Acknowledgements

This MASc thesis would not have been possible without the guidance and unending support from many people. First, and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Eddie Law, for his guidance, encouragement and patience throughout the years, starting from a mere summer research assistant to a full-fledged graduate student. I would like to further thank my dearest friends(Tal Tom, Vlad Pinkhasov, Vlad Boldyrev, Danny Levi, Allan Carswell, Leora Schlanger, Will Jin) for providing the much needed moral support and laughs which helped keep me relatively sane throughout the turmoil associated with graduate studies. I would also like to thank my colleagues (Barry, Richard, Tom) for making me jealous that I have yet to be a PhD student such as themselves. Despite the support from friends and family, I could not have endured those late night coding sessions without my closest friend coffee. Lastly, I am indebted to my loving family who bared with me as I attempted to reach this point. I apologize for the late nights strolling in after working late. I apologize for the all the missed dinners. For sticking with me, I cannot thank all these people enough.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The wide-spread growth of portable devices, such as laptops, portable digital assistants (PDAs), smart phones, etc, have stimulated the desire to access the Internet any time and any where. Wireless access technology has already been readily deployed throughout our homes, schools, offices and numerous hot-spot locations. The affinity towards the Internet will continuously grow as time progresses, which will consequently put a greater strain on the underlying network in the future.

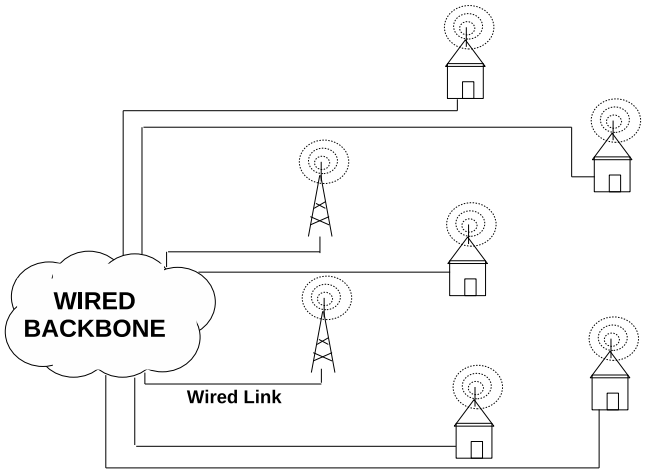
Accommodating for this any time any where philosophy, commonly referred to as ubiquitous computing, requires coverage of a significantly large area. Traditionally, wireless networks require strategically placed access points such that each of them is connected directly to the wired backbone network. For complete coverage of a wide area, a large number of these access points may be necessary to overcome the limited transmission range, line-of-sight constraints, and other factors that can attenuate the wireless signal. Laying the necessary cabling to extend the wired backbone is usually expensive, difficult, and time consuming. For a complete system, this inevitably leads to high installation and maintenance costs to properly accommodate for users spreading across large coverage areas.

The drive for inexpensive wireless networking solutions that are capable of providing a reliable, large coverage area has prompted the development of alternative wireless infrastructure networks. One of which is wireless mesh networks (WMNs), which employs a multiple hop approach. There are different wireless technologies available that can be used to construct wireless mesh networks. The prevalent wireless local area network (WLAN) standard of IEEE 802.11 [1] is a possible candidate, and will be the focus in this thesis.

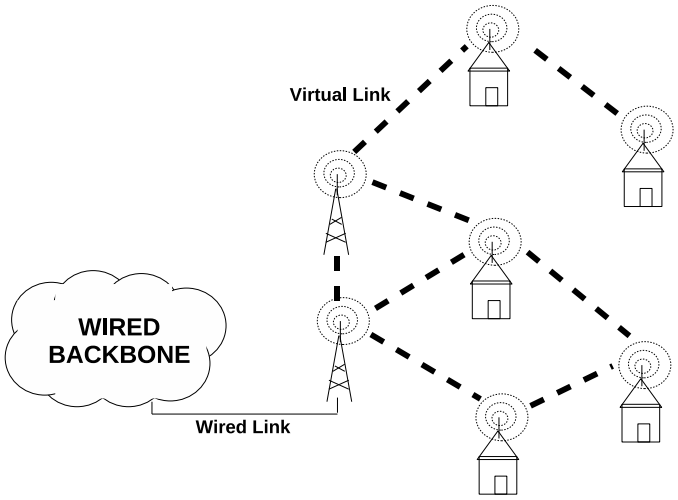
1.1 WMNs and Multi-Channel WMNs

Wireless mesh networking is a paradigm that all stations are connected through wireless links, without the need of having each access point connected to the wired backbone network. The terms wireless stations, routers, and nodes will be used interchangeably. In general, wireless stations can be set up rapidly to cover a wide geographical range. In this sense, packets may have to traverse along a path of wireless routers to communicate with corresponding nodes in the network that are outside of its transmission range. This multi-hop communication transforms the typical access point into additionally operating as packet relaying routers. As a result of this cooperative means of communication, only a selected few nodes are required to have a wired connection to the Internet, or wired backbone network. This concept is further exemplified by Fig. 1.1. In Fig. 1.1(a), the traditional access point infrastructure is shown, where each station requires direct cabling to the wired backbone network. This infrastructure design may be reliable, but the extensive cabling that is required can make it difficult to scale to larger networks. Achieving an infrastrucutre in this fashion will inevitably lead to high installation and maintenance costs. Alternatively, in Fig. 1.1(b) a typical mesh infrastructure is shown, where only a few selected stations are connected to the wired backbone network. It should be noted that the dashed lines are not actual links; they merely indicate that the two neighbouring nodes are within transmission range of each other. They are only capable of communicating if they each have a radio tuned to the same frequency channel. These links are thus referred to as virtual links. The reduced cabling makes the system less expensive. Additionally, new nodes can be installed into the mesh network quickly and with ease, even within rough terrain areas. They are a cost-efficient way to achieve the scalability, flexibility, and reliability necessary to build a wireless infrastructure network. Therefore, wireless mesh networks offer a promising compliment to existing wired backbone networks, that may be capable of providing last-mile Internet access.

Wireless mesh network takes its roots from traditional ad-hoc networks, but the two should not be confused. Unlike the paradigm of its ad-hoc network cousin, stations in mesh networks remain fairly stationary and are not constrained by limited power. Not being hindered by these constraints allow the wireless mesh networks to further exploit their resources. However, the need to operate as an infrastructure network often entails a different set of design parameters. In general, large-scale wireless mesh networks with many wireless hops are vulnerable to different communication problems [2, 3] such as bandwidth degradation, radio interference, high packet loss, long network latency, etc. These disadvantages have been limiting the deployment



(a) Traditional infrastructure



(b) Mesh infrastructure

Figure 1.1: Wireless infrastructure networks.

of these large-scale wireless mesh networks. Ideally, an effective mesh network should sustain a low latency and a high throughput performance between source-destination pairs that may be a few hops or many hops away.

The inherent mutual interference effect of multi-hop communication is the main cause of these known disadvantages. Assuming omni-directional antennas, the interference caused by adjacent nodes can be detrimental to any wireless communication, but the problem is exacerbated in the multi-hop communication scenario. Among the multiple channels defined in the specification [1] for wireless local area networks, a trivial method is to assign one frequency channel for each of the infrastructure stations operating within the wireless mesh network. Consequently, all wireless stations then contend for the same shared wireless medium. Furthermore, an infrastructure of wireless nodes may have large overlapping coverage area, which can cause an unwanted number of packet collisions.

The problem associated with single-channel networks can be further explained by referring to a chain of nodes, as depicted in Fig. 1.2. Nodes are arranged such that each node is only capable of transmitting to its one-hop neighbour, as shown by the transmission range R_{tx}. More specifically, from a system design perspective, the R_{tx} indicates the range that a frame can be successfully received with a signal strength higher than the reception power threshold. It implies that a station may be capable of decoding the received signal. In reality, this range depends on many factors such as, transmission power, receiver sensitivity, and radio propagation properties. However, the capability of properly receiving a packet is also dependent upon the interference range R_{ir}. It is described as being the range that a station in receive mode is interfered by another transmitter, and consequently suffers a packet collision. The interference range is always longer than the transmission range [4]. Based on different communication conditions, different values of R_{ir} can be assigned in terms of R_{tx}. The assumption of $R_{ir} = 2.2 \times R_{tx}$ is widely used in many simulators. With this setting, the interference ranges for nodes s_4 and s_5 are shown in Fig. 1.2. As a direct result of this, noise is created as nodes outside of its transmission range, but within its interference range, are unable to properly interpret messages. As a result, these parameters can help determine the amount of interference a wireless card may experience.

If using only one channel throughout the infrastructure and nodes s_4 and s_5 are in the process of communicating, then their transmissions may potentially impact the transmissions of six other nodes. This is shown by the interference ranges of both these nodes. Another frame corruption scenario occurs when two or more nodes are incapable of sensing each other, but attempt to transmit at the same time to the same

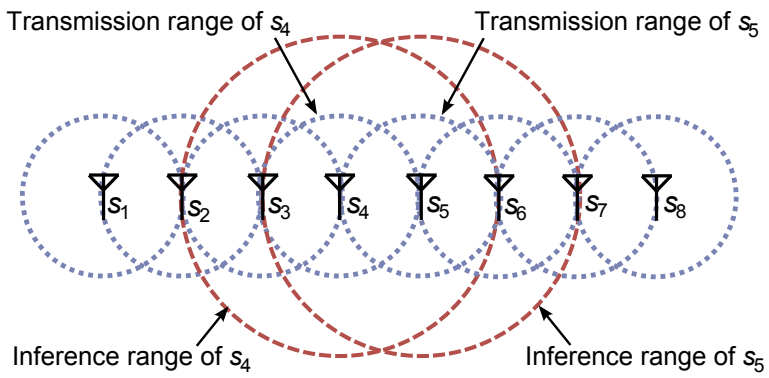


Figure 1.2: Interference ranges

node; frames are then corrupted with one another. For example, nodes s_2 and s_4 both attempt to send data to station s_3 simultaneously, but both frames collide and are corrupted at s_3. This scenario is known as the hidden terminal problem, and is one of the major issues associated with multi-hop wireless networks. In addition to hidden terminals, packets are susceptible to noises from uncontrollable external sources. These external sources refer to transmitters that are within the interference range but are not part of the cooperative mesh network. Excessive collisions and corruptions require nodes to perform a proportional number of packet retransmissions. Consequently, the duration and throughput performance is affected for packets successfully traversing the multiple hops. Even in the simplified chained network topology as shown in Fig. 1.2, we can observe how the small overlapping coverage area can affect the performance of the network.

Using a single channel in a wireless mesh network inevitably results in high levels of co-channel interference. To mitigate the effects of interference between adjacent mesh routers, the use of multiple channels is a promising solution. This design aims at assigning different channels to different links. Ideally, the minimum requirement is to have all links within an interference range assigned a different channel, in order to achieve the optimal levels of performance. With the wireless medium being a limited resource, there are only a limited number of channels that are available. It may be difficulat, and often impossible, to have enough channels available for the ideal state, since interference ranges can encompass many virtual links. The usage of multiple channels has the effect of increasing the theoretical capacity of the network. Much of the multi-channel capacity analyses, such as in [5], tend to focus on single hop, ad-hoc, wireless communication. They fail to take into consideration the implications of multi-hop communication. It is thus important to build a mesh network that can properly handle the limited number of available channels, while limiting the effects

of interference.

1.2 IEEE 802.11 Networks

The 802.11 wireless access technology, commonly associated with WLANs, has become the popular standard for the last-mile Internet access. Being widely available, they offer the necessary low cost, ease of installation, and fast setup desired for wireless mesh infrastructures. Therefore, the 802.11 may serve as a possible underlying technology for building wireless mesh networks. Recently, the IEEE standardization body has been formalizing a draft of the 802.11s protocol, as the forthcoming standard that expands on the WLAN standard to allow for the necessary operations of a wireless mesh network [6]. The protocol in the draft provide operations such as neighbour discovery and routing to achieve self-configuring and self-healing attributes. At the moment, it is still quite simplistic and is not expected to perform well under certain scenarios. As it currently stands, it does not provide much support for the usage of multiple channels and multiple radios, which makes it quite limited in achieving the necessary throughput requirements of next generation systems. In other words, they lack the ability to scale to larger capacities. There are some research results which study the effects and behviour of large-scale wireless networking projects and implementations. For example, the MIT's Roofnet 802.11b mesh network has helped prove its validity as an infrastructure network, but has also demonstrated its unfortunate shortcomings [7]. The aptly named carrier sense multiple access/collision avoidance (CSMA/CA) techniques used in the 802.11 protocol attempts to prevent packet collisions by limiting hidden terminals from simultaneous communication. By itself, the 802.11 medium access control (MAC) protocol operates poorly in a multi-hop setting.

It is the responsibility of the Distributed Coordination Function (DCF) of the IEEE 802.11 MAC protocol to handle the carrier sensing mechanism. The operation of the DCF protocol is illustrated in Fig. 1.3. Referring back to the aforementioned hidden terminal problem, stations are incapable of sensing nodes that are outside of the transmission range. The goal of the DCF protocol is to prevent the hidden terminal problem from occurring. In order to reduce these effects, the 802.11 protocol uses request-to-send (RTS) and clear-to-send (CTS) messages as a handshake mechanism prior to granting access to a specific channel. When a node senses another node transmitting on the same channel by means of the RTS message, it runs its backoff algorithm and defers its transmission until it can sense that the channel is no longer busy. A neighbouring node must also defer its transmission when it senses that neighbouring nodes are in the process

of transmitting. However, a station may hold its transmission upon finding that its neighbour is also sending a frame, even though the destination of these two frames do not corrupt each other at their respective receiving nodes. In fact, both frames can be sent simultaneously. Referring to Fig. 1.2 as an example, node s_4 sends to s_3, while s_5 sends to s_6. But, one of them may hold its transmission in 802.11; this is better known as the exposed terminal problem. Based on its virtual carrier sensing mechanism, only neighbouring nodes are aware of a station's channel access activities. The standard and its corresponding protocol were designed specifically for single hop communication; it was never intended to operate over multiple hops. Nodes that are multiple hops away but still within the interference range may attempt to transmit at the same time, which can cause a level of co-channel interference that the DCF protocol cannot prevent.

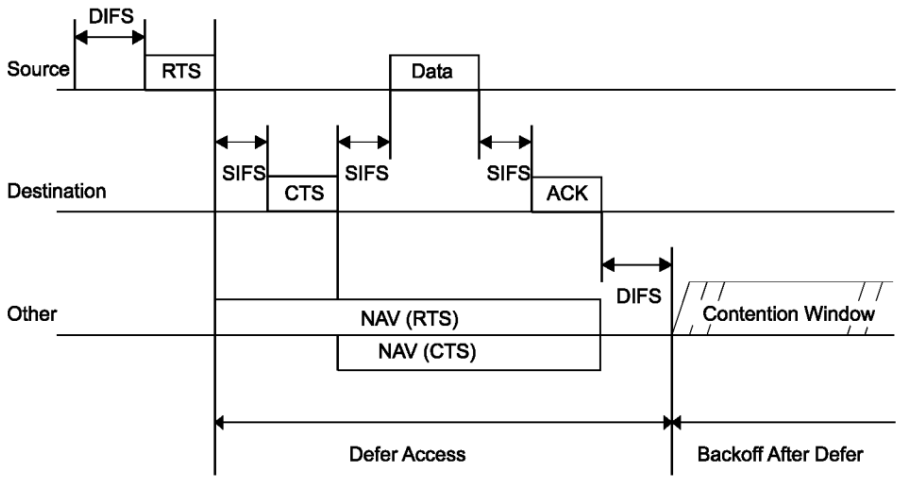


Figure 1.3: DCF operation of IEEE 802.11

When designing a wireless mesh network, it is important to consider the effects of collisions. This safety mechanism that prevents packet collision tends to have a negative impact on multi-hop networks. Upon experiencing a collision, a node will defer its transmission for a random time uniformly distributed between 0 and $2^z W_{min}$, where z is the transmission attempt and W_{min} is the minimum backoff window size. If multiple collisions occur, the possible length of backoff time increases exponentially. Excessive delays are created in the network to prevent corruption as a result of the hidden and exposed terminal problems. If one channel is used in the entire mesh network for a particular transmission between nodes, all adjacent nodes that are capable of sensing the transmission will be unable to perform any of their own transmissions. As a result, the backoff delay at each hop may add up and drastically increase the round trip time (RTT)

of a Transmission Control Protocol (TCP) session, as described in the later sections. Since the number of collisions affect the total backoff period, increased collisions negatively impact the per-hop performance. The end-to-end, or transport layer, performance is consequently degraded as excessive delays incur at each hop within the transmission. Therefore, by reducing the number of collisions that can occur by means of multiple channels, we can possibly achieve a reduced delay and higher per-hop throughput performance. When different channels are assigned to adjacent links, these adjacent links can have concurrent transmissions, which overcomes the delays experience from co-channel interference.

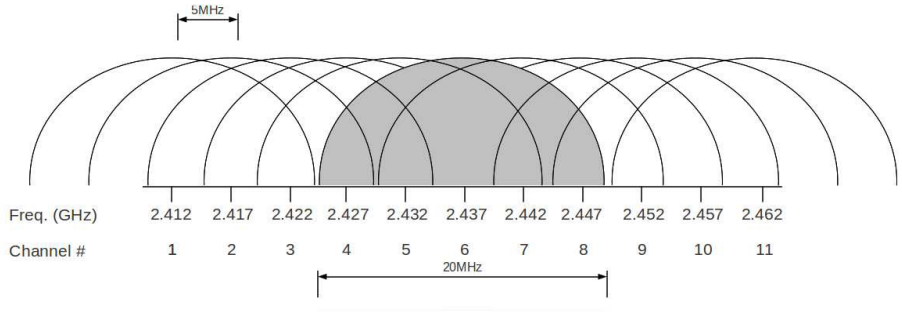
Nowadays, there are different underlying physical technologies associated with 802.11 wireless communication, which include 802.11a/b/g/n wireless boxes. The different physical transmission attributes results in differing bandwidths, data rates, interference constraints, and channel availability. Commodity wireless devices for each of these standards are readily available; hence, each one can serve as possible candidates for establishing the multi-channel wireless mesh network. The 802.11b/g standards operate on the 2.4 GHz frequency band using direct-sequence spread spectrum (DSSS) and orthogonal frequency division multiplexing (OFDM) techniques respectively. As a result of the different transmissions, the 802.11b standard has a maximum of 11 Mbps, while 802.11g has a maximum data rate of 54 Mbps. The frequency spectrum of these systems is shown in Fig. 1.4. Both the 802.11b/g standards carry 11 channels in North America. Typically, each channel occupies approximately 20 MHz worth of bandwidth, as stated by the filtering requirements of the standard [1]. Unfortunately, separation between two neighbouring channels is a mere 5 MHz; hence the frequency bands, of some channels will overlap each other. For example, channels 1 and 2 are centred at the frequencies 2.412 GHz and 2.417 GHz, respectively. Even though some channel overlap exists, there are three channels, 1, 6, and 11, that are non-overlapping. These non-overlapping channels are often called orthogonal channels. The filtering at the wireless transceivers ensure that these orthogonal channels do not significantly interfere with one another. When using orthogonal channels, none of the neighbouring links interfere with one another and is not constrained by transmission from other nodes within the network. With the onset of various devices sharing the noisy 2.4 GHz band, 802.11a was introduced to operate on the 5 GHz frequency spectrum. Also using an OFDM-based transmission, it is capable of achieving a maximum data rate of 54 Mbps. In turn, greater capacities can possibly be reached, as 12 orthogonal channels are available. The relatively newest standard of IEEE 802.11n [8] further attempts to increase network throughput and capacity using multi-input, multi-output (MIMO) technology. This standard attempts to combine aspects of its predecessors by operating at both the 2.4 GHz and 5 GHz spectra to provide 15 non-overlapping chan-

nels. As a baseline analysis of worst case scenario, the 802.11b standard was used in subsequent analyses and experimentation. With a scaled down wireless mesh network testbed in Section 4, the limited number of available channels is sufficient for out tests.

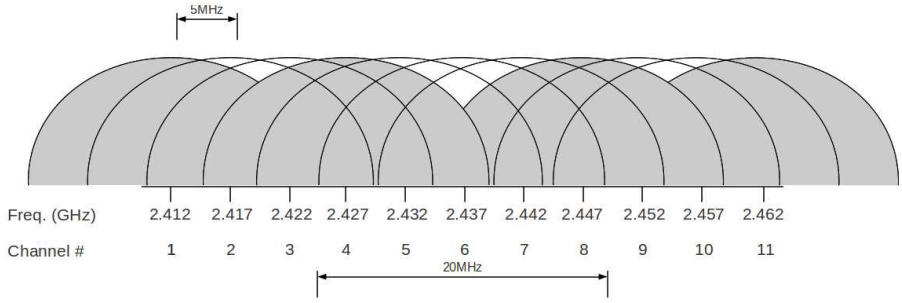
Although using all available channels in 802.11 may reduce co-channel interference, network performance may still be hindered by the presence of adjacent channel interference. The adjacent channel interference is attributed to the overlapping of channel's power spectrums. Based on the experiments performed in [9], it is not recommended in using two neighbouring channels on two adjacent wireless links. Doing so would create an excessive amount of noise and packet corruption, due to the large overlapping regions. But, it may be possible to use overlapping channels which are relatively further apart. For example, channels 1 and 4 which are 15 MHz apart may be able to provide improvements over the single channel scenario shown in Fig. 1.4(a). Hence, it may be possible to achieve suitable performance gains from the use of channels 1, 4, 8, and 11, as shown in Fig. 1.4(b). Given a network topology and known number of available channels, channel assignment algorithms as discussed in [10, 11, 12, 13] can be used to set channels among different links. Interference is still present upon using partially overlapped channels, but the level of interference is much less than that of the single channel case. As each link does not have to share the medium with as many other links, the network may experience fewer packet corruptions, shorter waiting times, and subsequently higher per-hop throughputs. To reduce the effects of adjacent channel interference completely, orthogonal channels can be used. Orthogonal channels are not constrained by transmission from other nodes within the network; hence, none of the neighbouring links interfere. Unfortunately, using only orthogonal channels reduces the number of available channels to the mesh network system. When there are fewer available channels, these channels will have to be re-used more frequently as the system scales to larger networks. This causes a greater chance for co-channel interference.

1.3 Contributions

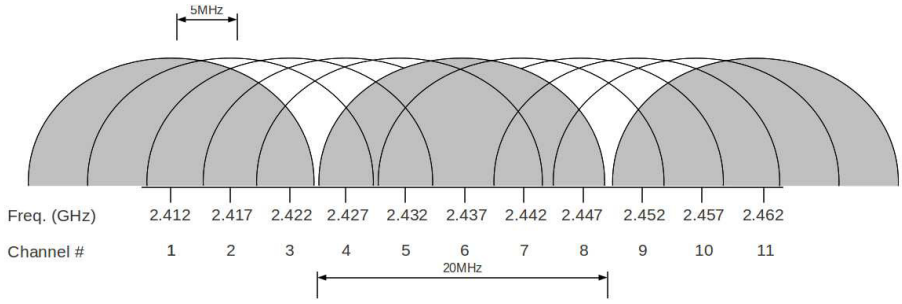
By the nature of the underlying 802.11 MAC layer, the protocol is only concerned with the performance related to its neighbouring nodes. But for end-users using wireless mesh networks, a satisfactory end-to-end performance across multiple wireless hops must be offered. In other words, the performance of transport layer traffic should be evaluated. This does not infer that the MAC layer, per-hop performance is insignificant. Instead, the cross-layered interactions should be thoroughly investigated. The per-hop



(a) Single Channel



(b) Overlapping Channels



(c) Orthogonal Channels

Figure 1.4: IEEE 802.11 b/g frequency spectrum

performance related to the MAC layer could play a critical role in evaluating the end-to-end performance of transport layer. It is commonly known that wireless mediums are more prone to errors and packet loss. When operating with multiple hops, we are no longer dealing with losses that can occur over a single link; wireless mesh networks are concerned with packet losses over every link amidst the end-to-end connection. Transmission Control Protocols (TCP) [14] and its corresponding congestion control algorithms have been designed specifically for wired networks. Many existing algorithms such as NewReno [15] may operate poorly over wireless mesh networks. These congestion control algorithms in TCP incorrectly assume that link layer errors and delays are directly attributed to the presence of congestion in the network. Consequently, congestion windows improperly resize to compensate, resulting in an inefficient use of network resources. If congestion is not a concern the User Datagram Protocol (UDP) [16] can be used to bridge the gap in understanding the relationship between the MAC and TCP layers. Therefore, understanding the delays imposed across transport layer connections can help distinguish the frailties of typical wireless mesh networks.

In this thesis, we will address the performance of 802.11-based wireless mesh networks with respect to the MAC and transport layers. As our intent is to focus on link layer contentions and possible TCP layer improvements, routing and its corresponding overhead will not be considered in this evaluation. Using a chained topology of varying hop counts and differing channel assignments, the channel contention problem is studied through analysis and testbed-based experimentation. Since the performance of the transport layer is highly dependent upon the MAC layer performance, it is crucial to initially analyze the influence of collisions and packet losses. With the goal of trying to lessen these collisions, the concept of multiple channels is introduced. Using UDP and TCP based analyses and tests, it becomes apparent that the multiple channel solution is capable of improving the network's performance only to certain degree. UDP based tests help enforce the assumption that TCP congestion is of little concern and that the performance at the MAC layer is the main cause of the poor transport layer performance. By attempting to combine TCP proxies with multi-channel wireless mesh networks, long error-prone connections could be broken into shorter ones with a tighter transport layer control. The possible gains achievable with TCP proxies, on the other hand, may consequently be hindered by the excessive queueing delay imposed at each of the proxies. By observing the relationships between MAC and transport layer protocols, we can see how a suitable protocol should be designed in order to properly operate a reliable multi-channel wireless mesh network.

In order to design an 802.11 based multi-channel wireless mesh network, we need to design the infrastructure that can properly accommodate and handle the multiple channels while limiting the interference

effects. To do this we suggest in Chapter 2 a multi-channel model to control the usage of multiple channels along with a topology design that can help limit the interference effects. This description will help explain the reasoning behind the tests and analyses performed in the subsequent chapters. But, a multi-channel system and topology is meaningless without understanding the behaviour of such a network. In order to design the 802.11 based mesh network, we need to understand why the network behaves the way it does. To do this, Chapter 3 provides an analysis of the physical effects of multiple channels, an analysis of the 802.11 MAC operation, and eventually their effects on an end-to-end transport layer connection. In general, the analysis allows us to further understand how nodes interact with one another and where problems exist with respect to the multi-hop nature of wireless mesh networks. Based on this analysis and understanding of the network, it can be seen how the network performance can be improved via TCP related solutions. As a result of this, TCP proxies are introduced in the network to overcome some of the problems witnessed in the transport layer analysis. Using a scaled down testbed of a multi-channel wireless mesh network as described in Chapter 4, the behavior was further tested to discover the effects of a multiple channel network that are difficult to witness in a mathematical analysis. It was possible to further observe how the network reacts under certain real-life conditions. Implementing the TCP proxy mechanism in the testbed provided a clear observation of the inclusion of an extra TCP sublayer connection. It is this behavior which can help define tunable parameters for the necessary algorithms of a distributed multi-channel wireless mesh network in future work described in Chapter 5.

Chapter 2

Topology Designs for WMNs

When dealing with multiple channels, channel management becomes a pertinent issue. The broadcast nature of the wireless medium emphasizes the importance of maintaining a controlled level of interference throughout the networking environment. Interference of this kind depends on factors such as traffic on neighbouring links and topology of the network. This interference consequently has an impact on the achievable capacity in a multi-hop setting. Paths between two communicating end-users need to operate with low interference and high network capacity. Interference can be caused by inter-path communication, where transmissions from separate paths corrupt one another. More importantly, interference can occur based on intra-path communication, wherein transmission on different links in a single path interfere. A more channel diverse multi-hop path has less intra-flow interference which increases the throughput along the path, as more links can be active simultaneously. This leads us to evaluate the possibility of an interference associated topology design for multi-channel wireless mesh networks [17].

Ideally, an effective wireless mesh network should be a low latency network that can dedicate separate wireless bandwidth links for ingress and egress traffic, which is similar to the operation of cellular networks. However, the system cost of a full duplex system can be very expensive. Using the half duplex 802.11 technology, it is possible to establish a framework for a multi-channel system.

2.1 Multi-Channel System Model

To operate a multi-channel system, the station can be equipped with a single radio or multiple radios. Multi-channel systems operating with a single radio transceiver, such as the Multichannel MAC (MMAC) protocol [18] or the Slotted Seeded Channel Hopping (SSCH) protocol [19], require precision scheduling to avoid the risk of permanently partitioning the network. As a reminder, two nodes are only capable of communicating if they both have a radio tuned to the same channel. Even if nodes are within transmission range, the network can become disconnected when neighbouring nodes do not have radios on the same channel. It can become quite difficult to properly schedule and coordinate neighbouring nodes to be tuned to the same channel at the same times. The alternative option is to use multiple radios at each node. These radios can thus be preset or dynamically tuned at runtime. Our system model opts for the multi-radio approach in the handling of multiple channels.

In our system model, we define κ as the set of available channels in a system for mesh packet relaying. The corresponding number of channels is denoted by $K = |\kappa|$. To accommodate for the multiple channels, each node is thus equipped with r number of radios. Stations do not need to be equipped with enough radios to access the complete system bandwidth. In a multi-radio multi-channel wireless network, stations can be installed with different numbers of radio ports such that $1 \leq r \leq K$. Certainly, no two radio ports in the same station should be tuned to the same channel at the same time, as this will cause an undesirable self-inflicted interference. Generally speaking, it is often unnecessary to have a station with $r = K$, as they may remain idle for large portions of time. This property will be shown in the following section discussing topology. If $r = K$, the radios can be preset to each of the specific channels. The problems arise when $r < K$, which requires the system to dynamically switch the radio's frequency in order to properly use all the available channels.

Instead of precise scheduling, coordinated channel switching operations necessary for distributed algorithms require the presence of separate control channels. The control channel allows us to separate the data packets from control messages (i.e. resource reporting, routing information, etc). For example, when a node wants to transmit data to a node multiple hops away, the system must properly coordinate with one another to ensure that each of the nodes along the path has a radio tuned to a similar channel preventing any network partitioning from occurring. Additional radios are now needed such that they are permanently tuned to these control channels. Therefore, the network now needs $K + D$ available wireless channels, where

D represents the number of channels required for sending control messages. If the channel capacities for control messaging is C_c and the relay channels' capacities are C_d, then the total theoretical capacity C_{total} of the system becomes

$$C_{total} = DC_c + KC_d. \tag{2.1}$$

A question raised from this relates to the number of realy channels K that should be used in the system. This leads to a baseline design established on the existing 802.11b standards. Currently, the 802.11b standard has the fewest number of orthogonal channels. These channels are commonly picked for experiments in the various literature, but it can only support a maximum of $K = 2$ to accommodate for at least one control channel. Therefore, we have carried out experiments running overlapping channels with channels 1, 4, 8, and 11. Thus, we now have three channels available for relaying traffic between nodes $K = 3$, with one separate channel used for control messaging $D = 1$.

2.2 Topology Control

Unlike mobile ad-hoc networks, or other ad-hoc networks, the topology of a wireless mesh network can be controlled, since nodes remain fairly stationary. It was demonstrated in [20], that the throughput capacity per node reduces significantly when the node density increases. Abiding by these concepts, interference should be controlled by limiting the number of neighbouring nodes, N. When designing a multi-channel wireless mesh network with $K = 3$, it is important to ensure that the system can properly scale to larger networks if necessary. Ideally, the usage of multiple channels should allow us to have all links within any path to be active simultaneously. In fact, if a neighbouring node uses an identical channel, serious co-channel interference will occur. This entails a network where all adjacent links are always capable of operating over the different channels, even when scaled to larger networks. In order to do so, topology construction units are designed around the K available channels and the N neighbouring nodes to establish a system that limits the interference and delays. These are appropriately referred to as *K-channel N-station* topology units.

The simplest way to handle three channels is with a small wireless mesh network as shown in Fig. 2.1. The farthest distance between any two nodes can be set to the transmission range, R_{tx}. With $K = 3$ and $N = 3$, each node only needs two transceivers to operate ($r = 2$), such that each of the virtual links can be assigned to a different channel. In this *3-channel 3-station* unit, the relaying channels can be permanently fixed among these stations without introducing interference among relaying traffic. Hence, routing operations

are trivial in this particular topology. However, the channel bandwidth is heavily under utilized.

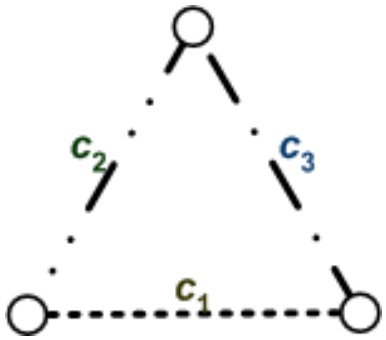


Figure 2.1: 3-channel 3-station triangular topology construction unit

A simple way to create larger mesh networks is to duplicate and expand the already suitably functional triangular base units. This design generates the aptly named *3-channel 7-station* hexagonal unit shown in Fig. 2.2(a). Upon using omni-directional antennas, the transmission range R_{tx} becomes the transmission radius encompassing all seven nodes within this base structural unit. To re-utilize the channel bandwidth, a station can use the same channel for communications to two other stations, at the two ends of the circle's diameter.

Upon considering Fig. 2.2(a), the topology now runs the risk of heavy intra-path interference for certain multi-hop paths of communication. For example, any activities between stations s_1 and s_2 prohibit the communication activities between stations s_1 and s_5, as per the operation of the 802.11 MAC protocol. The nodes s_5, s_1, and s_2 form a chained wireless mesh network which is similar to the one shown in Fig. 1.2. Based on this problem, the network may experience excessive delays as nodes may be required to defer their transmissions. Performance of a chained mesh network using the same channel under the TCP protocol deteriorates rapidly for paths longer than three hops [2]. This occurrence will be further investigated and verified in the subsequent chapters.

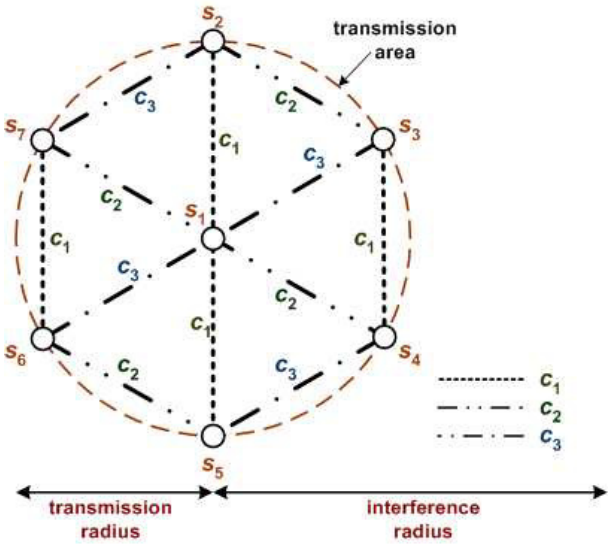
The problems associated with this design structure can be further demonstrated by duplicating the hexagonal structural unit. By doing so, it is easy to create a larger scaled wireless mesh network. In Fig. 2.2(b), a resulting 3-channel mesh network is now shown to encompass the entire interference range. Certainly, these meshing nodes can introduce interference among each other. The transmission range and two different interference ranges are shown to further demonstrate the problematic behaviour of such a system. In the worst case scenario for $R_{ir1} = 2.2 \times R_{tx}$, there are 18 other meshing nodes that may introduce interference

to a station. As each node is using all three available channels, each node within the interference range is capable of creating co-channel interference. On the other hand, if $R_{ir2} = 3 \times R_{tx}$, then 36 corresponding nodes can introduce undesirable levels of interference. In the network using R_{ir2} as the interference radius, there are a total of 37 nodes that are capable of using all 3 wireless channels simultaneously if $r = 3$.

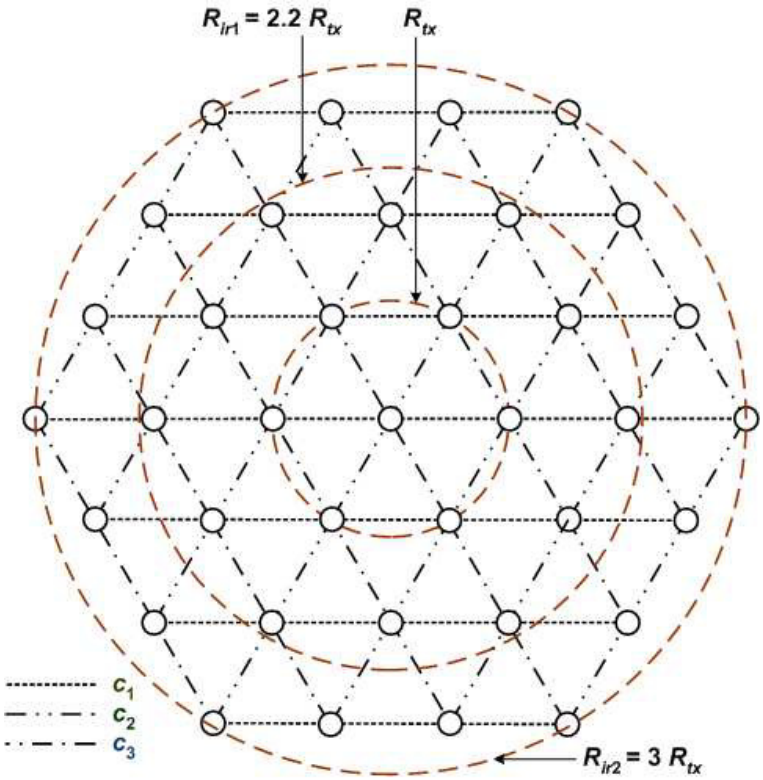
The previous *3-channel 7-station* basic structural unit needs to be re-evaluated to overcome the evident problems. Upon close investigation, the center station s_1 may possibly be removed while still maintaining the same level of wireless coverage and network access. This leads to the construction of an improved topology base-unit. This adapted structural unit, as shown in Fig. 2.3(a), is referred to as the *3-channel 6-station* unit. At this point, it can already be seen that no two adjacent links are operating over the same channel, which is an improvement over the previous scenario. This infers that all links can be active simultaneously, without having to worry about the transmission deferals. But, this begs the question of whether this advantageous property carries over for larger scaled mesh networks.

In order to construct larger scaled wireless mesh networks, this basic structural unit can be once again duplicated and interconnected in a similar fashion as the previous case. The resulting *3-channel 6-station* based wireless mesh network can create a mesh network as shown in Fig. 2.3(b). This mesh network offers similar coverage area as its 6-station predecessor. The transmission and interference areas are similarly shown to demonstrate the improved behaviour of this system. With this base unit design for R_{ir1}, there are 12 other meshing nodes when excluding the center node. Upon careful attention, not all nodes in the range use the same channel; only six stations may introduce co-channel interference at a station for a given wireless channel. Similarly for R_{ir2}, there are 34 other meshing nodes within the interference range, but only 15 of those stations may introduce co-channel interference for a given channel. In the network shown using R_{ir2} as the interference radius, there are in total 35 mesh nodes, and all nodes can use all 3 wireless channels simultaneously for $r = 3$. This resultant wireless mesh network can thus offer higher throughput performance as co-channel interference is reduced. The comparisons of the two wireless mesh network topologies are summarized in Table 2.1 and Table 2.2.

The importance of the control channel, for co-ordinated channel management, becomes essential when the number of radios is less than the number of available channels ($r < K$). For communication to occur between two neighbouring nodes, they must each have a radio tuned to the same data communication channel, such that the channel belongs to the set κ. As each node is required to operate over all of the K channels, the control interface is used to co-ordinate the switching between the channels as necessary. When designing a

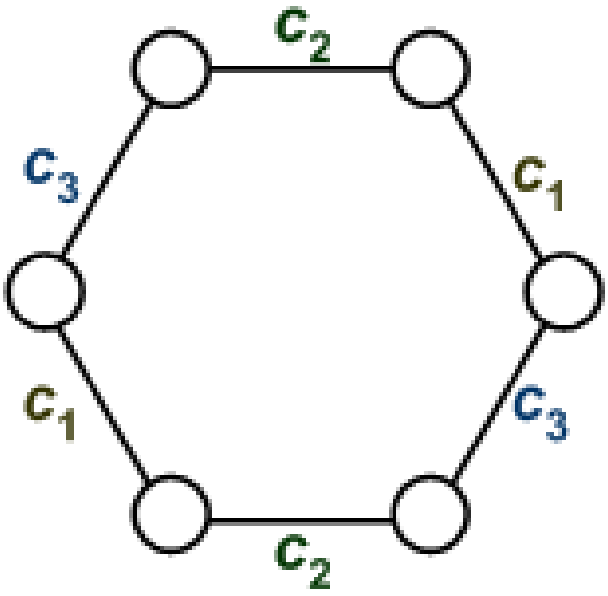


(a) Base structural unit

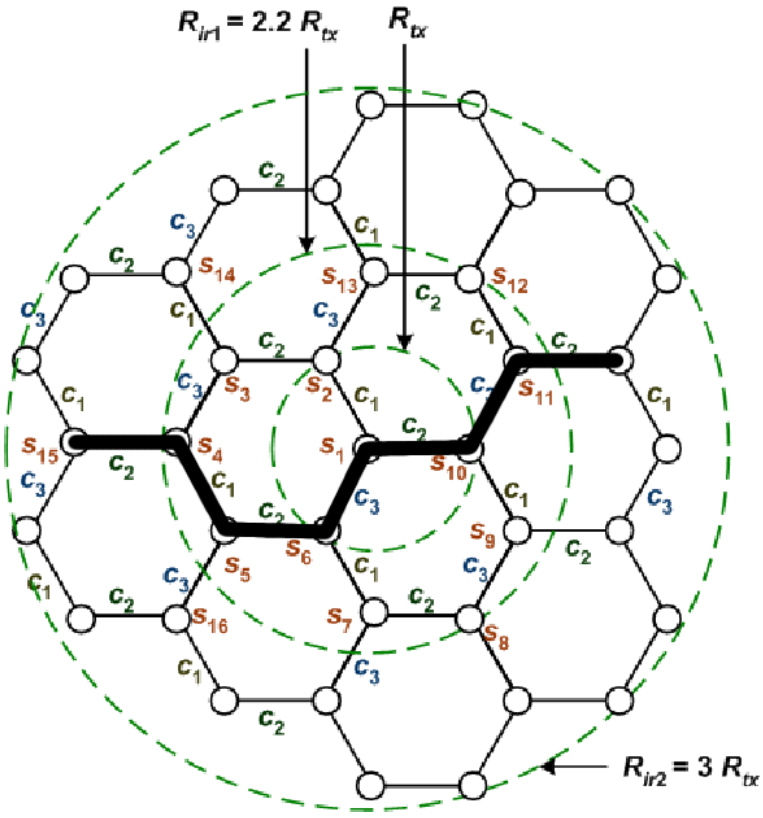


(b) Wireless mesh network

Figure 2.2: 3-channel 7-station topology



(a) Base structural unit



(b) Wireless mesh network

Figure 2.3: 3-channel 6-station topology

suitable protocol, it will be important to take this property into consideration.

Upon scaling the base structural unit to create Fig. 2.3(b), the improved interference property can be maintained. It is important to observe that no two adjacent links operate over the same channel for any multi-hop path within the wireless mesh network. It no longer suffers from the *no-activities* constraint if an adjacent link is being used, as stated in the *3-channel 7-station* network. In this design scenario, all links within any chain of nodes can be operated concurrently. Therefore, a chained network of multiple channels can be used, as long as the adjacent links are operating over different frequencies. Therefore, the chained network can serve as a suitable baseline test when analyzing mesh networks under these conditions.

Table 2.1: Number of co-channel interfering nodes excluding itself for *3-channel 7-station* model

Case	Total num. of nodes	Num. of co-channel nodes
$R_{ir1} = 2.2 \times R_{tx}$	18	18
$R_{ir2} = 3 \times R_{tx}$	36	36

Table 2.2: Number of co-channel interfering nodes excluding itself for *3-channel 6-station* model

Case	Total num. of nodes	Num. of co-channel nodes
$R_{ir1} = 2.2 \times R_{tx}$	12	6
$R_{ir2} = 3 \times R_{tx}$	34	15

2.3 Summary

When designing a suitable wireless mesh network, it is important to properly control the wireless resources which limits the inherent interference effects. It is difficult to completely remove the various causes of interference, but relatively simple techniques can be used to limit their effects. With single channel mesh networks being prone to heavy levels of co-channel interference, the usage of multiple channels can provide the necessary scalability and capacity requirements for wireless infrastructures. A multi-channel multi-radio design along with a corresponding control channel can be used to manage and coordinate the limited number of available channels. With the limited mobility associated with infrastructure networking nodes, the multi-radio system can be further combined with an interference controlled topology design. Using scalable topology structure units, the system can grow to larger networks as needed, while operating under the constraints imposed by the 802.11 underlying technology.

The advantage created by the *3-channel 6-station* model is that all links can be active for any chain of nodes established. Based on this idea, the subsequent chapters will attempt to further understand the behaviour caused by this intra-path interference associated with a mutli-channel system. It is important to first build a system that can suitably handle intra-path interference before trying to handle the inter-path interference effects.

Chapter 3

Analysis of Multi-Channel WMNs

From the topology-based investigation provided in Chapter 2, we saw how a proper arrangement of nodes can help control the interference levels of the network, without the need for complex channel assignment algorithms. Using the *3-channel 6-station* topology model, it is apparent that concurrent transmissions along all links is vital in improving the network's performance. The introduction of multiple channels to the system avoids excessive delays caused by deferred transmissions, and allows for the desired simultaneous transmission activity. Even along a chain of nodes operating over multiple channels, it is difficult to completely remove the effects of co-channel interference. An example of this chain of nodes is shown as the darker line originating from node s_{15} of Fig. 2.3(b). Despite overcoming the *no-activities* constraint, it is still susceptible to the intra-path, co-channel interference effects. Based on this issue, it is important to first understand the behaviour of a chain of nodes. Albeit simple, this particular topology can provide a baseline analysis and set of results describing the interactions between nodes at the MAC level and subsequently the end-to-end transport level. As previously mentioned, it is necessary to understand the effects of intra-path interference before inter-path interference effects can be considered. We will use this chain topology, as also shown in Fig. 1.2, in the further analyses and experiments.

In this section, an analysis of throughputs and latencies is provided as a packet traverses a multi-hop chain of nodes. It attempts to understand the cross-layer interactions between the transport and MAC layers, which allows us to see the problems associated with multi-hop communication. Upon noticing these problems, an improvement to the common transport layer and TCP is suggested. Table 3.1 provides a summary of all necessary symbols used in this analysis. The analysis provided here expands on my work

performed in [21].

3.1 Multiple Channels

It was briefly shown in Fig. 1.4 how multiple channels are utilized to counteract the spectral inefficiency in the network. When using multiple channels, especially overlapping channels, it is important to understand the effect that these adjacent channels have on the desired signals. The carrier sensing mechanism of the network interface cards (as part of the physical layer) is incapable of sensing these channels, which reduces the co-channel interference. Unfortunately, adjacent channel interference is consequently introduced, as we will see in this section. On a similar note, the analysis here will show why orthogonal channels are commonly used instead of overlapping channels. Signals that are heard, if at all, are interpreted merely as noise. If this noise level exceeds a certain threshold such that the node in question is unable to correctly read a signal, the packet is consequently discarded. Thus the transmitting node interprets the dropped packet as a collision or corruption, since no ACK or CTS message is received in return. It is this potential signal corruption, that needs to be understood before proceeding.

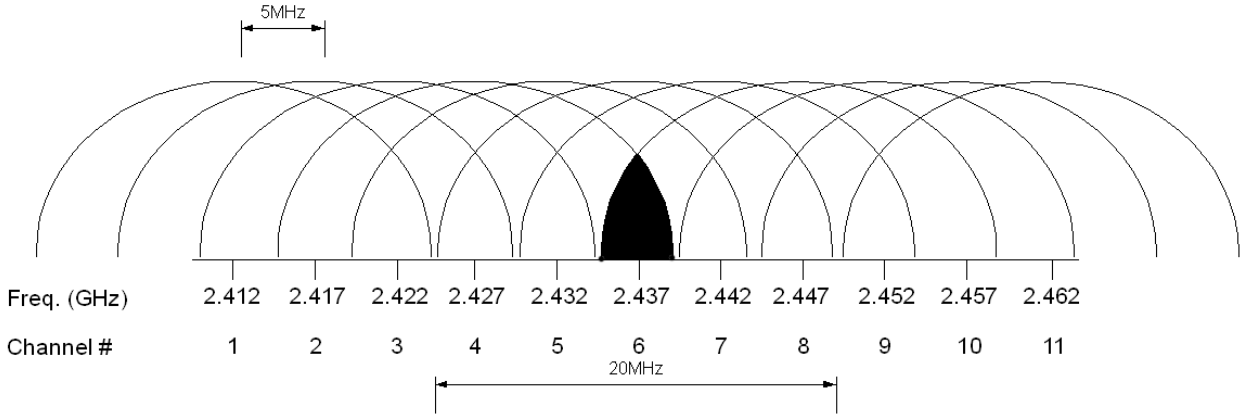


Figure 3.1: Calculating the overlapping region

To determine the effect of multiple channels, the extent of signal overlap needs to be found. For example, if channels 4 and 8 were being used, we would want to determine the amount of overlap, as shown by the shaded region of Fig. 3.1. Based on the specifications in [1], the spectral mask of the transmitter and receiver must at least have a 30 dB attenuation at 11 MHz and 50 dB attenuation at 22 MHz from the centre frequency

Table 3.1: Index of notations

r_s	Number of radios attached to station s
K	Number of available channels, $K = \|\kappa\| > 1$
R_{ir}	Interference range
R_{tx}	Transmission range
θ	Estimated channel overlap
f_c	Channel centre frequency
Pwr	Power spectrum density
W_{min}	Minimum MAC contention window size
p_i	Probability of collision on channel i for single link
τ_i	Probability of a node transmitting on channel i
I_i	Number of nodes in interference range
h	Number of hidden nodes
$f(I_j)$	Probability of no corruption from noise and adjacent channel interference
max	Maximum MAC retries
z	MAC layer transmission attempt
q	Arrival rate of packets
P_I	Probability of a node being idle
P_{tr}	Probability of any node in range transmitting
P_S	Probability of node successfully gaining access to channel
B_{link}	Single hop throughput
$E[L]$	Expected packet length
σ	Slot time
D_s	Expected delay of single-hop successful transmission
D_c	Expected delay of single-hop collision
t_r	Time for RTS transmission
t_{sifs}	Time for SIFS transmission
t_c	Time for CTS transmission
t_d	Time for data packet transmission
t_a	Time for ACK transmission
t_{difs}	Time for DIFS transmission
p	End-to-end probability of collision
$E[D_{mac}]$	Expected single-hop delay
$E[D_{trans}]$	Expected end-to-end delay
B_{UDP}	UDP throughput
MSS	Maximum segment size
RTT	Round-trip time
n	Proxy hop length
m	Source-destination hop length
$E[D_q]$	Expected queueing delay
λ	Packet arrival rate
μ	Packet service rate
$E[D_{proxy}]$	Expected end-to-end delay with proxies
B_{proxy}	End-to-end throughput with proxies

f_c. A simplified technique is to assume an ideal filter which is capable of meeting these requirements. As the name suggests, ideal filters are unattainable, but they are reasonable enough for our analyses. Thus, to analyze the adjacent channel effects, the power densities of a signal on channel x (such that $x \in \kappa$) and the possibly interfering channels y (such that $y \in \kappa$ and $y \neq x$) must be determined. Assuming a random signal with a power density function as a *sinc* function occupying the 20 MHz bandwidth, the percentage of channel y on channel x is found as follows:

$$\theta = \frac{\int_{-\infty}^{\infty} Pwr_x(f) \cdot Pwr_y(f)df}{\int_{-\infty}^{\infty} Pwr_x(f)^2 df} \tag{3.1}$$

where Pwr_x is the power spectrum density of a signal on channel x and Pwr_y is the power spectrum density of a signal on channel y.

This fraction of signal overlap represents the contributed levels of noise from adjacent channels (y). It can then be used to calculate a normalized signal to noise ratio SNR of the interfering stations. If Pwr_x denotes the received power and Pwr_y represents the same signal at the same location on channel y, then the SNR parameter is

$$SNR = \frac{signal}{noise} \tag{3.2}$$

$$= \frac{Pwr_x}{Pwr_y} \tag{3.3}$$

$$= \frac{1}{\theta}. \tag{3.4}$$

As we are only discussing the effects of multiple channels here, the noise parameter in Eqn. (3.4) does not consider the noise from external transmission sources outside of the mesh network.

With the estimated SNR values, we can further estimate the bit error rate of the system as a result of the interfering adjacent channels. The bit error rate is dependent upon the particular modulation used for the wireless communication. Assume that the data rate is operating at 11 Mbps, which is the maximum theoretical data rate achievable for an 802.11b system. At this particular rate, the quadrature phase shift keying (QPSK) technique is used for modulation. The bit error rate BER associated with QPSK is

$$BER = Q(\sqrt{2\frac{E_b}{N_o}}, \tag{3.5}$$

where,

$$\frac{E_b}{N_o} = (SNR)(\frac{BW}{rate}). \tag{3.6}$$

This bit error rate calculation uses the Q probability function for standard normal distributions. The per bit energy to noise ratio $\frac{E_b}{N_o}$ is commonly used to calculate bit error rates, which is a function of the SNR, the channel's bandwidth BW and the data's $rate$. Before this adjacent channel analysis can be useful, it needs to be translated into the packet error rate PER as follows:

$$PER = 1 - (1 - BER)^l, \tag{3.7}$$

where l represents the length of the packet in bits. We should expect channels that are further separated in channel number to provide lower packet error rates as the overlap between channels is also reduced.

3.2 MAC Analytical Model

Before trying to evaluate the end-to-end throughput across multiple hops, we need to understand the per-hop performance. As our goal is to employ a pre-existing MAC protocol associated with WLANs, the analytical model needs to obey the 802.11 protocol as described in the previous section. Different analytical models [22, 23, 24, 25] have been developed for the 802.11 protocol. The one developed in [22] uses a Markov model representing the backoff stages. Using this methodology, a normalized throughput across a single hop for WLAN was formulated. The models used in [22, 23] only consider a saturated traffic model, which assumes that the buffers in a node always have packets ready to transmit. The operating model should be slightly different in a multi-hop environment. That is, a frame transmission at a node may depend on the transmissions from the previous node(s). As a result, the transmission buffers for each of the nodes do not necessarily have a packet to send at all times, especially if there are excessive delays at the previous node(s). In the case that a large amount of data such as a file is being transmitted, it can be assumed that the source node of an end-to-end connection operates under saturated conditions. On the other hand, subsequent nodes proceeding the source node within the path will likely experience conditions consistent with a non-saturated model. Hence, in [24, 25], a similar Markov-based analysis for the non-saturated scenario has been developed also for WLAN single hop communications. These models neglect to consider the effect of multiple channels. Hence in this section, we extend and devise an analytical model combining these saturated and non-saturated

WLAN models for the purpose of multi-channel wireless mesh networks.

Our initial goal is to model the probability of a packet colliding or being corrupted. A collision occurs when two or more nodes attempt to transmit a packet at the same time, or more particularly during the same time slot. Based on this simplified explanation, the probability of collision when transmitting on channel i is given as follows

$$p_i = 1 - (1 - \tau_i)^{I_i - 1 + h} \cdot f(I_j) \tag{3.8}$$

where τ_i represents the probability of a node transmitting on channel i, $I_i - 1$ represents the number of nodes in R_{ir} using channel i minus itself, h represents the number of hidden nodes, and $f(I_j)$ is the probability that the packet on channel i is not corrupted by noise from adjacent channel interference. It should also be noted that this collision probability is based upon the first person view, as seen by the node itself.

Transmissions from the I_j nodes on adjacent channel j, where $j \in \kappa$, may not be heard or interpreted by MAC layers of neighbouring nodes. These adjacent channels are interpreted merely as noise. If this noise exceeds a certain threshold in which the node in question is unable to correctly read a signal, the frame is discarded. On the other hand, the transmitting node interprets the dropped frame as a collision since no acknowledgement (ACK) or CTS message is received in return. Thus, the transmitting node interprets the dropped packet as a collision, since no ACK or CTS message is received in return. The $f(I_j)$ probability is required to handle this noise and adjacent channel interference.

If the mesh network consists of randomly placed nodes in the topology, determining the values for I_i (such that $i \in \kappa$) and parameter h requires finding the density of the nodes. Thanks to the topology-based considerations of Chapter 2, the difficult task of finding the node density is not necessary. If we consider a single multi-hop communication, it is safe to consider a chain of nodes. Suppose once again we consider a single multi-hop communication session. Using the simplified chain topology shown in Fig. 1.2, we can see that the number of interfering nodes is relatively less for nodes closer to the source or destination. For example, assume a transmission is being attempted from nodes s_1 to s_8; therefore, the number of interfering nodes for s_1 is two, while the number of interfering nodes for s_4 is four. The number of hidden nodes, which cause many of the observed collisions, also varies: node s_1 would be exposed to one hidden terminal, while s_4 would be exposed to two hidden terminals. With a suitable mesh protocol, the number of interfering nodes and the number of hidden nodes can be controlled. When using multiple channels, these numbers decrease since nodes are not obligated to share the medium with as many neighbouring nodes. The orthogonal

channel case should theoretically provide the best scenario in terms of the number of interfering nodes and probability of collision.

Without including $f(I_j)$ in Eqn. (3.8), the model would only take into consideration co-channel interference with known nodes in the mesh network. The probability of not having any corruption can incorporate the effects of external noise sources and/or adjacent channel interference. It is assumed that this probability of having no corruption from channel j is mutually exclusive to the probability of a successful transmission on channel i, which allows us to multiply the two terms. Since each channel has a bandwidth of approximately 20 MHz, it is safe to assume that a channel separation of more than 5 channels does not produce a significant amount of interference; therefore the $f(I_j)$ term approaches a value of one. It is apparent that there will be a tradeoff between the number channels available and the number of interfering nodes.

In Eqn. (3.8), the probability of transmission was introduced. A Markov model, which represents the backoff counter, is used to describe this probability of a node transmitting in a given time slot τ. using the saturated model described in [22], the two-dimensional Markov chain is depicted in Fig. 3.2. The theory behind this relates to 802.11's binary exponential backoff mechanism. As mentioned in the earlier description of the 802.11 protocol, a node will defer its transmission for a random time uniformly distributed between 0 and $2^z W_{min}$, upon experiencing a collision. Solving this Markov chain leads to the determination of the probability of a node transmitting on a channel i in a given time slot:

$$\tau_i = \frac{2(1-2p_i)}{(1-2p_i)(W_{min}+1) + p_i W_{min}(1-(2p_i)^{max})} \tag{3.9}$$

where W_{min} represents the minimum contention window, max represents the maximum retries upon encountering a collision, and p_i represents the probability of collision described earlier.

Generally, this saturated model would only be applicable to the source node. Subsequent nodes in the multi-hop chain are dependent upon the previous hops' transmissions. These nodes may not always have a packet to transmit, and should be analyzed as being in non-saturated conditions. In this scenario, the transitions between states is not as simple as the state transitions in Fig. 3.2. In this case, we need to take the arrival rate of packets to the particular node into careful consideration. Arrival rates of packets allow us to estimate whether or not the node has a packet to transmit in a given time slot. Using the non-saturated model developed in [24], the modified Markov chain is depicted in Fig. 3.3. The additional states, indicated by the dotted box outline, depict the states after transmitting a packet successfully. As shown in Fig. 1.3,

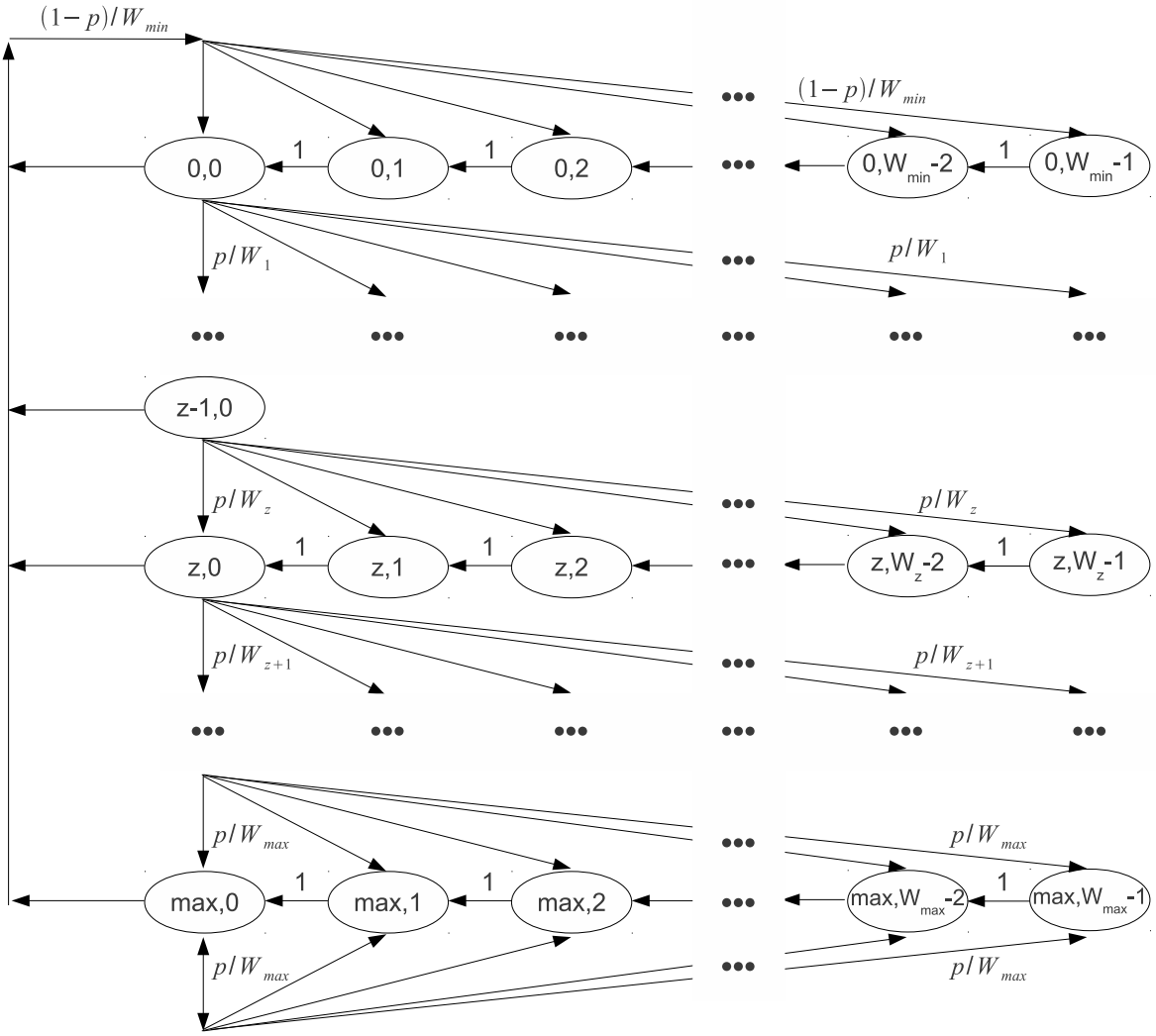


Figure 3.2: Markov chain for the 802.11 backoff mechanism under saturated network conditions

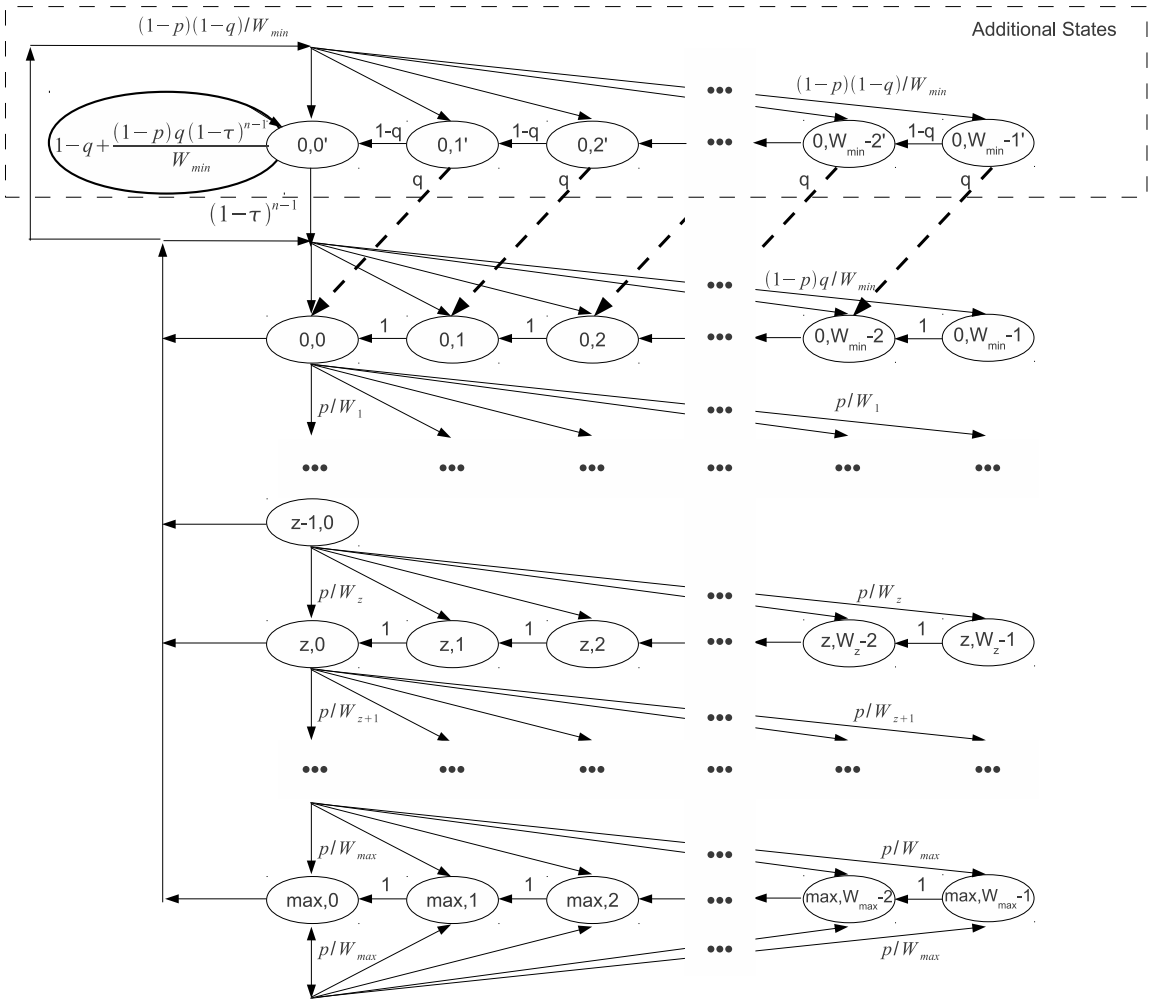


Figure 3.3: Markov chain for the 802.11 backoff mechanism under non-saturated network conditions

the MAC protocol undergoes an additional backoff after successfully transmitting a packet. Traditionally in the saturated framework, another packet is ready to transmitted after this additional contention window. In a non-saturated network, there may not be a packet available to transmit. Once again, solving this Markov chain leads to the determination and re-evaluation of the probability of a node transmitting on a channel i in a given time slot:

$$\tau_i = (\frac{1}{b})(\frac{q^2}{1-q})(\frac{W_{min}}{(1-p_i)(1-(1-q)^{W_{min}})} - (1-p_i)) \tag{3.10}$$

where,

$$\begin{aligned} b = {} & (1-q) + \frac{q^2 W_{min}(W_{min}+1)}{2(1-(1-q)^{W_{min}})} \\ & + \frac{q(W_{min}+1)}{2(1-q)} \left(\frac{q^2 W_{min}}{(1-(1-q)^{W_{min}})} + p_i(1-q) + q(1-p_i)^2 \right) \\ & + \frac{p_i q^2}{2(1-q)(1-p_i)} \left(\frac{W_{min}}{1-(1-q)^{W_{min}}} - (1-p_i)^2 \right) \\ & \cdot \left(2W_{min} \frac{1-p_i-p_i(2p_i)^{max-1}}{1-2p_i} + 1 \right). \end{aligned}$$

When dealing with non-saturated networks, the probability of transmission is dependent upon the probability of a packet being available in the transmission buffer q; this parameter is closely related to the arrival rate of packets. In a multi-hop network, the probability of transmission is now dependent upon the arrival rate. When operating with multi-hop networks, the arrival rate of packets to a node s_x is dependent upon the transmission rate at the previous node s_{x-1} and the appropriate propagation delays. The values for p_i and τ_i can be calculated by solving the system of non-linear equations. Then, if the number of wireless channels is large in a mesh network, then the complexities of these calculations increase, since the number of equations and number of unknowns increases for each channel.

The backoff mechanism of the 802.11 protocol requires the contention window to count down. As per the protocol, it will only countdown when no node in its carrier sense range transmits. As the previous equations were from the first person perspective of the node, the following equations take on a broader picture through a third party view. As such, the probability evaluations should not be confused. When a node senses another transmission on the channel in use, the backoff counter is required to freeze. The amount of time that the counter is frozen relates to the network allocation vector (NAV), if it is used. When the node is counting

down, we can consider this as the node being idle for a slot time σ. Suppose that P_I is the probability that a channel is deemed as being idle. Based on this understanding of the 802.11 protocol, the probability of a node being idle depends on the probability that no node in carrier sense range is transmitting P_{tr} in that given time slot. Considering I_i nodes, without any hidden terminal problems, this probability can be modelled by

$$P_I = 1 - P_{tr} = (1 - \tau_i)^{I_i}. \tag{3.11}$$

Since the analysis needs to model when to freeze the backoff counter, we also need to know when any node in carrier sense range has successfully acquired access to the medium. Successfully acquiring access to the channel infers that the node can successfully transmit a packet to its neighbour. From [24], the probability of a node successfully gaining access to the a particular channel, given that a transmission does occur, is as follows

$$P_S = \frac{I\tau(1-\tau)^{I-1}}{P_{tr}}. \tag{3.12}$$

This means that of the I nodes in interference range, only one can be granted access at a particular time slot given that a transmission occurs. Hence, all other nodes $I - 1$ are restricted from transmitting courtesy of the CSMA/CA mechanism.

It is our goal to evaluate the end-to-end, or transport layer's, performance; therefore, the per-hop MAC layer transmission delays and rates need to be analyzed first. The single-hop throughput B_{link} of a particular link as modelled in [22] is as follows:

$$B_{link} = \frac{P_S P_{tr} E[L]}{(1 - P_{tr})\sigma + P_{tr} P_S D_s + P_{tr}(1 - P_S) D_c}, \tag{3.13}$$

where $E[L]$ represents the expected packet length, D_s represents the expected delay for successful transmissions, and D_c represents the expected delay for collided packets. As we can see, the expected delay can be essentially decomposed into idle and busy delay times. The idle time refers to the slot time when the backoff timer is counting down. The system is considered idle, and counting down, when no node in a specific R_{ir} is transmitting. Conversely, the busy period can be decomposed once again to the amount of time that that the node is in the process of transmitting or trying to transmit. If a node successfully accesses the wireless medium, it transmits with a probability $P_{tr}P_s$. On the other hand, a node may encounter a collision upon transmitting with a probability $P_{tr}(1 - P_s)$.

When the RTS/CTS mechanisms are invoked, a successful transmission requires sending the appropriate packets and waiting for suitable periods to ensure that the channel is truly idle. The time for a successful transmission is as follows

$$D_s = t_r + 3t_{sifs} + t_c + t_d + t_a + t_{difs}, \tag{3.14}$$

where t_r is the delay associated with an RTS packet transmission, t_{sifs} is the short inter-frame spacing (SIFS), t_c is the delay associated with an CTS packet, t_d is the time to transmit a packet, t_a is the time to transmit an acknowledgement packet, and t_{difs} is the DCF inter-frame spacing (DIFS). These inter-frame spaces are the necessary delays to ensure the channel is idle. Since RTS packets are sent to ensure that it is safe to transmit and not impeded by hidden terminals, an unsuccessful transmission will find a collision after transmitting an RTS packet but not receiving an CTS packet in return. Hence, the delay associated with a collisions is found by

$$D_c = t_r + t_{difs}. \tag{3.15}$$

3.3 Transport Layer

Existing TCP algorithms assume that link layer errors and delays are attributed to the presence of congestion in the network. Consequently, congestion windows prematurely resize, resulting in an inefficient use of network resources. Compared to its wired counterpart, wireless links are more prone to errors and packet loss, which makes existing TCP solutions unacceptable for wireless mesh networks. Therefore, packet loss has a significant effect on the performance of a network. The usage of multiple channels, as previously mentioned, can significantly reduce the number of collisions and packet losses. The most collisions can be witnessed by a network with a single channel, while the network with orthogonal channels should eliminate many of the collisions.

When using any transport layer protocols (i.e UDP, TCP, etc.), packets are often required to hop across many links. Since there are no returning ACK messages at the transport layer for UDP, we can attempt to find an analytical model for UDP based on the developed MAC model, which characterize the packet losses from the link layer perspective. If orthogonal channels are used, then we may make the assumption that each link is independent and mutually exclusive. As a result, we can take the product of each of the links to determine the total probability of packet los across the multi-hop connection. The total probability of

packet loss p of a chain of m hops is

$$p = 1 - \prod_{l=1}^{m}(1 - p_l)^2, \tag{3.16}$$

where p_l represents the frame loss associated with the l^{th} link. It is apparent that as the number of hops of a communication flow increases, the probability of packet loss also increases. The power of two is required to account for the reverse transmission of the MAC layer ACK. The value of p_l is equal to the probability of collision as calculated in Eqn. (3.16). It should be noted that the use of multiple channels may reduce the number of collisions, but it cannot completely remove all packet loss from the system. Even the orthogonal channel case is still be susceptible to packet loss from uncontrollable external sources.

Each packet that collides consequently contributes to the transport layer delay, as shown in Fig. 3.4. From the MAC layer analysis from above, we can determine the approximate time required by a packet to traverse across the m multiple hops from the source to its intended destination. Based on the theoretical throughput, it is easy to derive the estimated delay of a packet across one hop $E[D_{mac}]$, as the two are inversely proportional. Using this, end-to-end delay of a packet $E[D_{trans}]$ is

$$E[D_{trans}] = \sum_{l=1}^{m} E[D_{mac,l}] \tag{3.17}$$

where $E[D_{mac,l}$ is the estimated delay crossing the link l.

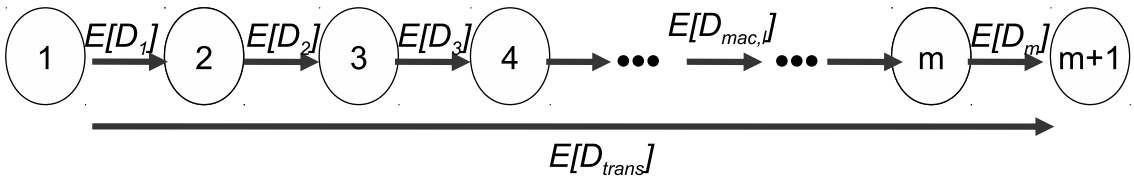


Figure 3.4: End-to-end multi-hop, transport layer delay

As we can see, the transport layer delay of a packet is associated to the sum of all the delays of all links across the multi-hop connection. Each link may suffer from different probabilities of collision; hence, each link will also suffer from different packet delays. Since UDP operates in a connectionless fashion, this

particular delay analysis can be used to calculate the throughput of a UDP session, as shown in Eqn. (3.18).

$$B_{UDP} = \frac{MSS}{E[D_{trans}]}, \tag{3.18}$$

where MSS refers to the maximum segment size. The connectionless nature of UDP infers that data is not guaranteed to be received at the intended multi-hop destination. UDP-based traffic sources will continue to push packets into the network at a constant rate regardless of whether packets have been received correctly.

It is easy to observe that the UDP throughput depends upon the MAC layer performance, but the TCP throughput is more complicated to analyze. Unlike UDP, TCP provides a connection oriented approach that ensures reliability of properly ordered packets. Therefore, this relates directly to the designs of congestion control algorithms that are embedded within the TCP protocol. But, the devised probability of packet loss shown in Eqn. (3.16) can help comprehend and model the end-to-end TCP throughput. Modeling of TCP throughput in wired networks has been thoroughly investigated in [26], which has focused on congestion avoidance design. In general, the throughput is inversely proportional to RTT and also inversely proportional to p. Therefore, when RTT and/or p is decreased the throughput is improved. The RTT of a packet should be estimated regarding sending through all hops with a returning transport layer ACK message. The RTT of a perfect forwarding and returning scenario can be estimated as

$$RTT = 2 * E[D_{trans}]. \tag{3.19}$$

Due to the retransmissions associated TCP protocols, this RTT has to be modified in order to properly model multi-hop TCP sessions. It is our outright goal to use proxies to improve this overall throughput, by attempting to reduce these parameters. These modelling parameters can depict a suitable relationship between throughput, round-trip time, and packet loss probability. Using our calculations for probability of packet loss and RTT, we can further observe the relationship between transport layer and MAC layer characteristics. Therefore, we shall carry out experiments to measure the overall system performance for TCP connections.

Furthermore, upon transmitting across multiple hops, a queueing delay is incurred at each hop. The queueing delay experienced on a per-hop basis effects the end-to-end throughput. But for TCP proxy design, discussed in the following section, it may not change due to the storing time at each TCP proxy.

Hence, it is safe to exclude this particular type of queueing delay from our analysis.

3.4 TCP Proxy

Apart from multi-channel designs, TCP proxies can be introduced into the wireless mesh networks in attempt to further improve the overall performance of TCP connections. The proxy concept can be found in Split TCP [27] and indirect TCP [28] designs. They were originally designed to separate wired and wireless links, by introducing TCP proxies to separate the different mediums. In light of poor TCP performance in multi-hop single channel 802.11 networks, a combination of n-hop TCP proxies and multi-channel design can be introduced along the path between any pair of communicating devices.

The proxy concept breaks long, multi-hop connections into relatively shorter connections with fewer hops. As shown in Fig. 3.5, an extra transport layer is added between the source and destination, which intercepts the packets traversing across the multiple hops. In general, as shown in Eqn. (3.16), the probability of packet loss p grows with the number of hops. Therefore, these shorter connections inevitably provide the TCP connection with a tighter control. Shorter connections infer that fewer retransmissions are required to traverse across the entire end-to-end connection; retransmissions now only need to be sent from the previous proxy. The n-hop TCP proxy design splits a long multi-hop TCP connection into multiple TCP connections. Each of these connections has fewer number of wireless hops, which may be able to improve the TCP performance. For example, consider a path with seven wireless hops as shown in Fig. 3.6. A typical TCP connection would be established between the first and eighth node, in this scenario. If a 3-hop TCP proxy design is used, then there are $\lceil 8/3 \rceil = 3$ shorter TCP connections. Now, three TCP connections are established: the first connection is established between the first and fourth nodes; the second between the fourth and seventh nodes; and the last connection for the remaining nodes. The proxy is set up simply by observing the time-to-live (TTL) field in the packet's Internet Protocol (IP) header. Whenever it has decremented by a multiple of n, a TCP proxy should be set up for this connection at the station.

Although TCP proxy design requires almost no changes to the TCP specification, the cost associated with using proxies is the excessive buffering and queueing delays at each of the proxies. The additional transport layer used to intercept packets must buffer these packets in order to properly operate the separate TCP connection, which adds the delay in question. In order to determine this queueing delay, we need to determine the arrival rate λ and the service rates μ at the transport layer, as also shown in Fig. 3.5. The

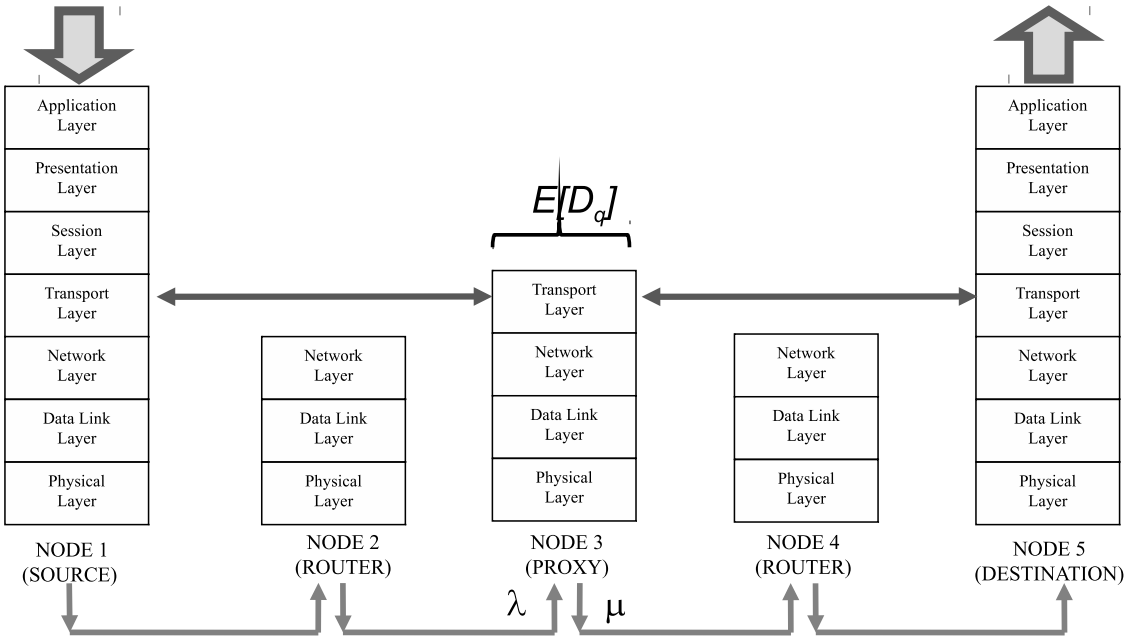


Figure 3.5: TCP proxy layers and added queuing delay

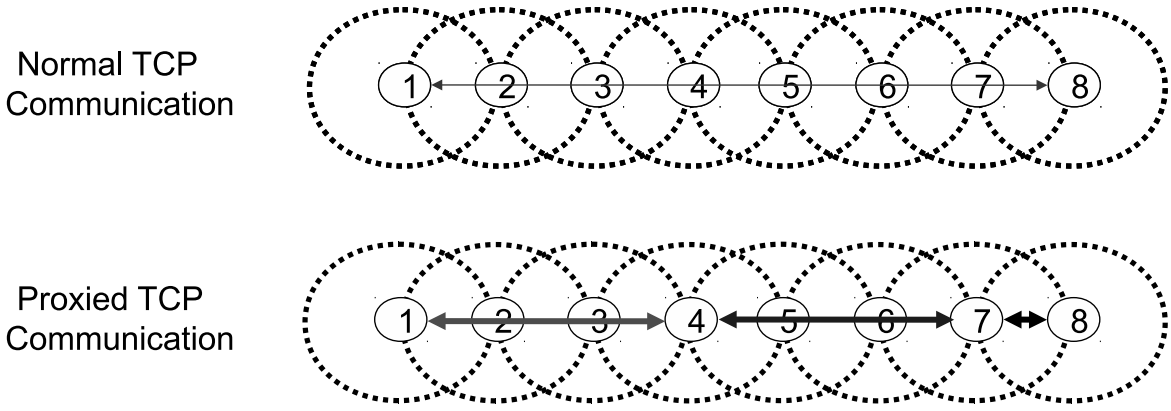


Figure 3.6: TCP proxy example using 3-hop proxies ($n = 3$)

arrival rate is dependent of the TCP goodput, as we are now only concerned with useful application layer data and not underlying protocols' overheads. Goodput is considered for the arrival, instead of throughput, since only in order data packets are forwarded to the next proxy. On the other hand, the service rate is dependent upon the speed at which the MAC layer can handle packet transmission, as shown by Eqn. (3.13). This is why goodput is a more valid means of evaluation then throughput. Assuming links are never permanently broken, all data packets transmitted will eventually be received through proper TCP retransmission. However, through our experience in setting up the testbed, links may fail if too many collisions have occurred, as a result of TCP permanently closing the connection. Assuming that the nodes operate as an $M/M/1$ queueing system, the expected queueing delay across one proxy D_q is as follows:

$$E[D_q] = \frac{1}{\mu - \lambda}. \tag{3.20}$$

For an n-hop proxy to run on an m-hop source-destination pair, there are $\lceil m/n \rceil$ TCP connections with fewer numbers of wireless hops. This performance can be summarized by the delays incurred as a packet traverses across all m hops, as shown in Fig. 3.7. For simplicity, we will also assume that each proxy incurs the same delay. Therefore, the total end-to-end delay using proxies is

$$E[D_{proxy}] = \sum_{i=1}^{\lceil \frac{m}{n} \rceil} (E[D_{trans,i}] + [E[D_q]) \tag{3.21}$$

where $E[D_{trans,i}]$ is the travelling delay to reach the i-th proxy from the $(i-1)$-th proxy. When the value of n is relatively small, many proxies are required to establish the end-to-end connection. If there are too many proxies, these delays can accumulate to the point where using proxies is counterproductive. On the other hand, when n is relatively large, we run into the problems associated with long connections as described in Eqn. (3.17). As a result of these properties, there is an apparent trade-off between the number of proxies and the length of the proxies within a single end-to-end connection. Thus, the goodput can be approximated by associating the delay found in Eqn. (3.21) and the maximum segment size of a packet (MSS),

$$B_{proxy} = \frac{MSS}{E[D_{proxy}]} \tag{3.22}$$

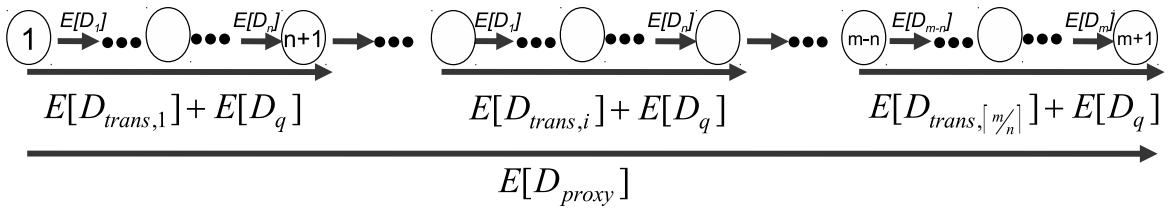


Figure 3.7: End-to-end multi-hop, TCP proxy delay

3.5 Summary

From the physical layer to the transport layer, nodes are susceptible to the effects of interference from neighbouring nodes. Whether operating over a single channel or multiple channels, latencies are created throughout the network as packets traverse across a series of nodes. Even along a chain of nodes, it was shown in this section that co-channel interference effects can be quite detrimental to the operation of the network. It is important to understand, and attempt to overcome, this intra-path behaviour first. When using a single channel, the underlying 802.11 protocol forces nodes to defer their transmissions creating heavy delays in the system that add up across the multiple hops. The usage of multiple channels should limit the number of interfering nodes, and consequently allow for all links across the multiple hops to be active simultaneously. Thus, performance measures and delays should be improved upon using multiple channels. The following section will help verify this analysis to demonstrate the gains from using multiple channels and the need for adopting TCP proxies.

Chapter 4

Performance Measurements

Various algorithms and protocols have been researched and developed within the research community, but much of these designs have only been tested via a simulator. Simulators such as NS-2 [29] provide an inexpensive, fast and convenient means of evaluating a design. However, the use of simulators only provides a mere estimate, and likely only an upper-bound on the achievable performance. Real wireless networks would inevitably be affected by uncontrollable factors. Unfortunately, these simulators do not operate under the same set of parameters as testbed experiments. Actual wireless networks would inevitably be exposed to uncontrollable, and random, interferences. Therefore, the other option is to implement a testing protocol in a wireless mesh network testbed. Previously unknown hardware and interference delays can only be found when implemented on a testbed environment. What may work nicely in the controlled environment of a simulator may not work as nicely in a real-time multi-radio multi-channel wireless mesh network testbed.

Therefore in this section, we evaluate and compare the performance of transport layer traffic along single-channel and multi-channel 802.11 wireless mesh networks by means of a wireless testbed. Additionally for TCP traffic, n-hop TCP proxies are also added in the experiments. There are numerous difficulties in setting up a multi-hop wireless testbed, which are susceptible to uncontrollable interference from sources outside of the testbed. It is difficult to mitigate these noises, as they can be found easily everywhere. The testbed topology follows the setup shown in Fig. 1.2. Each station operating under Linux v2.6 is equipped with two Atheros-based 802.11b compliant wireless cards, such that there is no requirement for channel switching. As we are dealing with a chain topology, the required number of radios ($r = 3$) for the *3-channel 6-radio* scenario is no longer necessary. For experiments, all data packets are transmitted with an equal size of 1500

bytes (the MSS), and are transmitted over 11 Mbps capacity links. An unmodified 802.11 MAC protocol, which implements CSMA/CA, is used. It should be noted that the RTS and CTS messages are always transmitted at 1 Mbps as per the specification. On a similar note, the fragmentation capability of the MAC layer is disabled to ensure that each packet is preceded with its corresponding RTS and CTS messages. The reasoning for the necessary RTS and CTS messages will be further emphasized in Chapter 5. With regards to TCP, the NewReno congestion algorithm is implemented for both the proxy case and the scenario without proxies. A summary of the testbed parameter settings is provided in Table 4.1.

Table 4.1: Testbed parameter settings

Attributes	Values
Network size, m	2 to 8 nodes
Channel set 1	$\kappa = \{1\}$, $K = 1$
Channel set 2	$\kappa = \{1, 4, 8, 11\}$, $K = 4$
Channel set 3	$\kappa = \{1, 6, 11\}$, $K = 3$
Data payload size	1500 bytes
Basic rate	1 Mbps
Data rate	11 Mbps
Min. contention window, W_{min}	32
Max retries, max	7

4.1 Channel Switching

We have demonstrated that when attempting to set up a testbed under the *K-channel N-station* topology for wireless mesh networks, each node may be equipped with varying number of radios. If the number of transceivers in a node is identical to the number of operating wireless channels, then each transceiver can simply be locked to each operating channel frequency. However, if the number of transceivers at a node is fewer than the number of packet relaying channels (i.e. $r < K$), then the operating frequency channel of a wireless card has to be switched upon being needed. This dynamic channel switching allows the wireless mesh network to function properly in the multi-channel setting.

Experiments have been carried out on setting up channels for path establishments among source and destination pairs. Portions of the Linux kernel module code developed to support this functionality, and the mesh network framework, is supplied in Appendix B. The source and destinations are separated by m hops, where $1 \leq m \leq 5$. Each station contains three radios: one for control message passing and coordination;

and two radios for packet relaying ($r = 2$) meant to accommodate for ingress and egress traffic. It should be noted that the control channel is shared amongst all meshing nodes. Hence, the channel behaves similarly to a single channel multi-hop communication. What differentiates this control channel from the typical data relay channel is the size of these control packets. Channel setup packets can be much less than the maximum segment size. Smaller packets tend to occupy the channel for a relatively shorter amount of time. The experiments are aimed at finding the delay incurred by switching channels, and the nature of these delays as hop length increases. The tests were run 100 times and the averages, standard deviations, and variances were determined. The results of these findings are graphicallyh shown in Fig. 4.1 with a 95% confidence interval. In this setup, the channels were set to 4, 8, and 11 repetitively along the wireless hops. It should be noted that besides control information passed, no data from the application layer is transmitted to prevent any extra overhead.

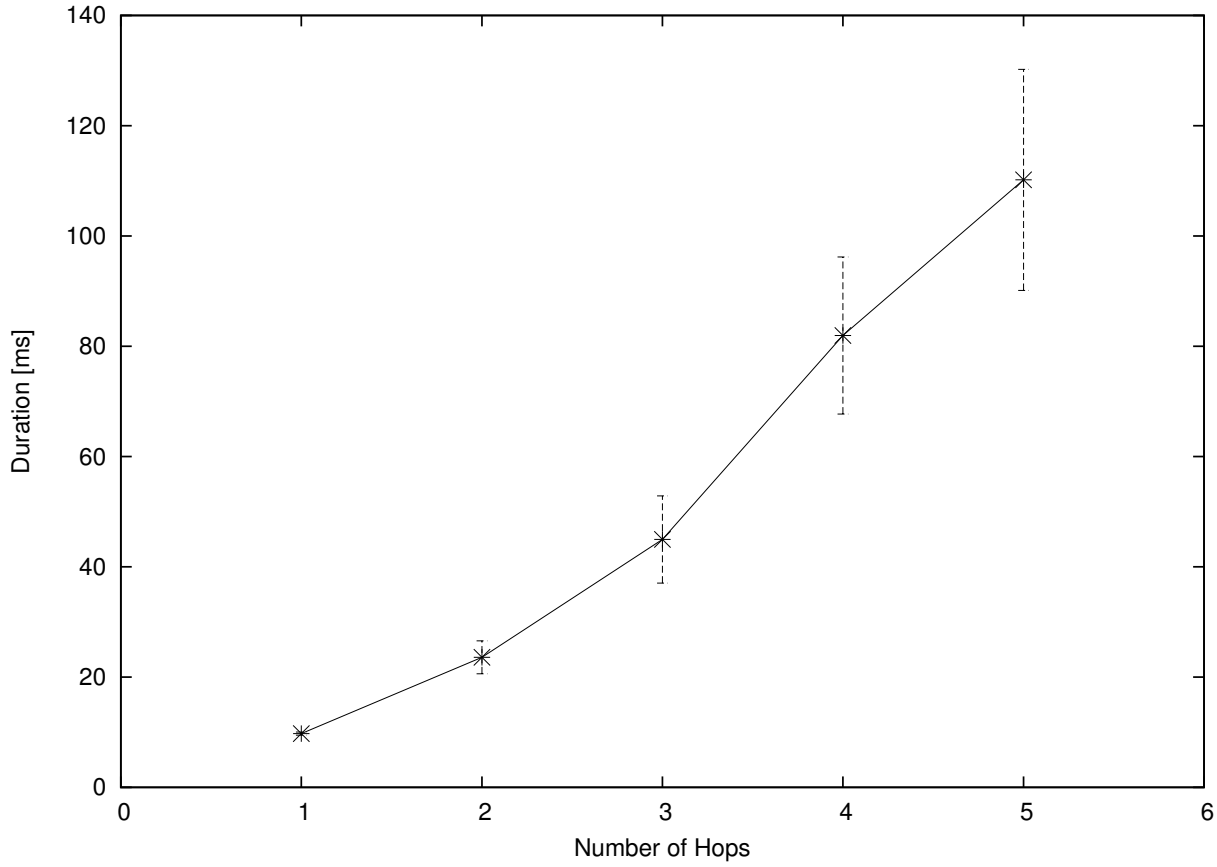


Figure 4.1: Mean setup times for dynamic channel switching

From these measurements, it was found that the average time duration required to set a transceiver radio to a specific channel takes about 3.97 msec. This result is specific to the Atheros wireless cards used; different cards can achieve different channel switching times. Looking at how a typical network interface card switches channels provides some light as to why the duration is quite large. When the operating system requests to change the channel, the device must be completely reset. The channel switching time is considerably large, with respect to the time required to transmit a single packet. As dynamic channel switching can provide an improved level to wireless networks, it is safe to assume that in the future these channel switching times will decrease as new technologies are introduced. As of now, this latency cannot be ignored, and must be taken into consideration when designing the corresponding protocol.

As we can see from the results in Fig. 4.1, the measured setup durations are about linear, especially when the number of hops is small. Indeed, when the number of hops are four or five, the mean setup times get longer and have a greater variance, but still maintains a linear growth on average. It is important to note that the behaviour of these setup times does not follow an exponential curve, as expected. This exponential trend, commonly associated with data communication over multiple hops will be demonstrated in the following section. Although seemingly small, this delay can hinder the performance of a network. As each node is required to perform channel switching, the incurred delays add up as the network scales to larger hop counts. If properly handled, this form of channel switching can be reasonable for establishing multi-hop multi-channel wireless mesh networks.

4.2 UDP Measurements

Before implementing the TCP proxies, it is important to next evaluate and confirm the performance gains from using multiple channels. To do so, we temporarily omit the control channel and channel switching operations described in Section 4.1. Three sets of experiments have been carried out. The first experiment is to compare the throughput performances of an m-hop source-destination pairing, for $1 \leq m \leq 7$, such that each link in the network operates on a single available channel ($K = 1$). The second test case operates under the same topology using four overlapping channels ($K = 4$). This setup may suffer side-lobe interference, but it has four times as many available channels to communicate than its single channel counterpart. Lastly, three orthogonal channels are used in the testbed. No channel assignment is needed in this scenario because they are each orthogonal.

Using UDP, we are able to investigate the direct relationship between the number of hops across a multi-hop connection, while not being restricted by the congestion window of a TCP connection. To observe the effect of multiple hops prior to the introduction of multiple, a single channel network was initially tested. Observing the effects of a multiple hop communication provides us with a point of comparison for the proceeding discussions on the effects of multiple channels. Tests have been performed by transmitting a large number of packets over an m-hop source-destination pair, for $1 \leq m \leq 7$. Sending a large number of packets is consistent with the idea that the source node is always saturated. For the testbed experiments, 5000 packets were transmitted to give the effect of a saturated source node. Based on these testing scenarios, the mesh network testbed UDP results using a single channel (*Channel Set 1*) is shown in Fig. 4.2. The derived analysis from Eqn. (3.18) for UDP performance is then compared against the measured testbed results. The model was evaluated based on MATLAB simulations (see Appendix A). For the MATLAB simulations, it should be noted that no packets are transmitted as the different models attempt to accomodate for the saturated and non-saturated conditions. The results from the MATLAB evaluations are plotted against the testbed results of Fig. 4.2 to emphasize the validity of the model. Exact data for the single channel UDP case is summarized in Table 4.2.

From this test, it can be seen that the mathematical analysis provided in Chapter 3 matches closely to the tested results; thus, helping prove the validity of the analytical method. It is apparent that as the number of nodes increases the throughput decreases. This is consistent with our analysis since increased hops translates to increased probability of packet loss, which in turn translates into increased delay. An increased delay leads to the observed decrease in throughput. For an 11 Mbps backbone single-channel network, the goodput of UDP traffic goes from around 5 Mbps for 1-hop down to 2 Mbps for 3-hop transmissions, eventually dropping to below 1 Mbps for the 7-hop transmissions. A single-hop communication as shown by the first point in the graph expectedly performs at the highest level, as it does not need to share and contend for the medium with other nodes in the network. As more hops m are added to the system, more co-channel interference is introduced. Therefore more nodes have to share and contend for the same wireless medium, which results in the seemingly exponential drop of the curve.

Although, the analytical model does over-estimate the testbed results for the first point by 0.88 Mbps; this translates to an approximate 5.7% drop based on the analytical model's set of data. As previously mentioned, the real-life factors affecting the testbed's transmissions is difficult to match to all parameters of a simulation; various assumptions are necessary to adequately model the system. Although relatively small,

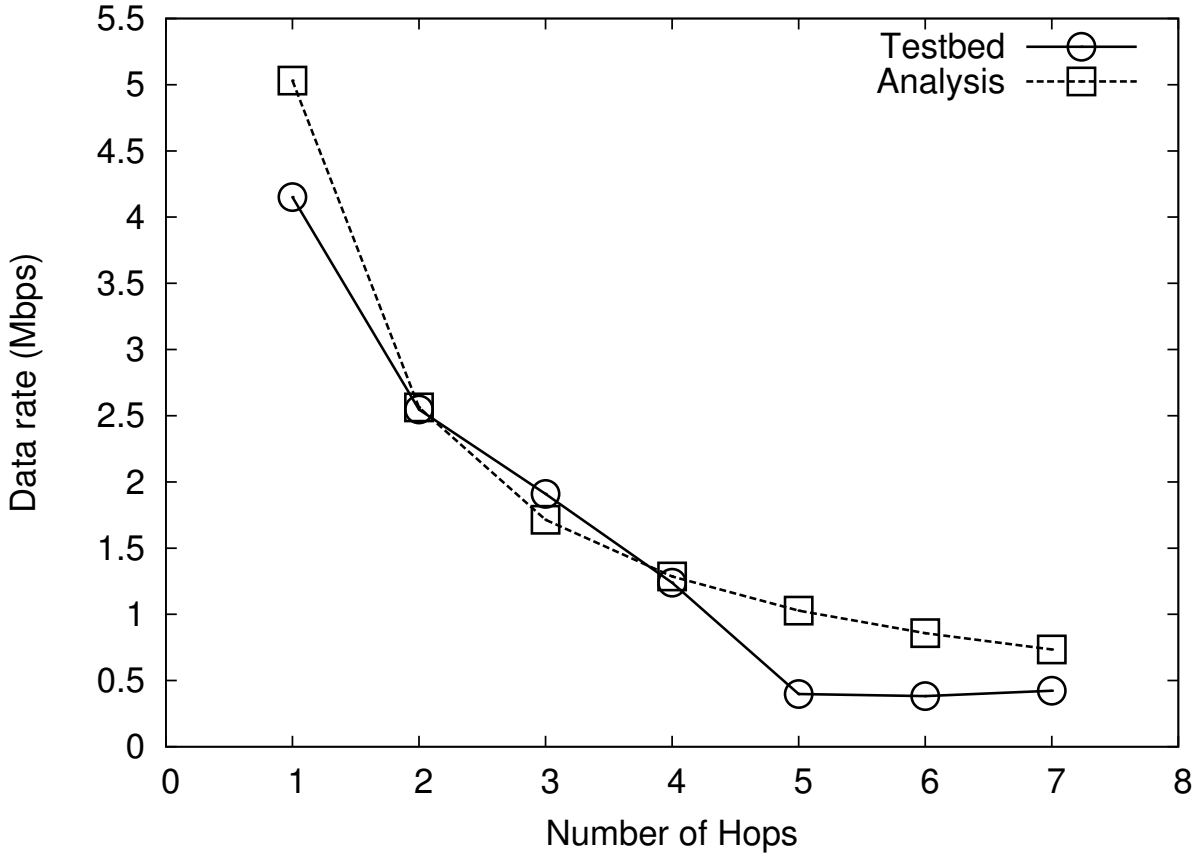


Figure 4.2: Single-channel 802.11 wireless mesh networks based on UDP session

the main factor that can possibly cause this difference is the noise from surrounding wireless transmission sources. As tests were performed in an academic environment of a university, the school's pre-existing wireless network operating on the same frequency bands can potentially interfere with the proper transmission and reception of packets. Another discrepancy between the analytical results and testbed results is noticeable at the longer hop counts. For example, at the longest hop length of 7 hops, a discrepancy of 0.31 Mbps. This discrepancy is much more significant than that noticed for the first hop, as the difference is approximately 42% of the analytical data. As multi-hop communication is being implemented, discrepancies from each hop of the connection adds up; thus, it makes the discrepancies at longer hop counts relatively more significant. In addition to the noise (if any), the additional discrepancy can possibly be attributed to the assumption of interference range. The MATLAB simulations were based on the assumption that the interference range was based on $R_{ir} = 2 \times R_{tx}$. For the smaller values of m, the number of nodes in the R_{ir} and the number of hidden nodes h are relatively fewer; this infers that an underestimation of these parameters has a less significant effect for smaller values of m. Despite these discrepancies, the model still matches quite closely to the actual testbed results.

To mitigate the impact of co-channel interference, as observed in the plotted single channel UDP graph of Fig. 4.2, multiple channels are additionally used in the chained topology. Under similar topology and conditions as the single channel scenario, the overlapping channel (*Channel Set 2*) and orthogonal channel scenarios (*Channel Set 3*) are shown in Fig. 4.3. Once again, tests were performed. Tests have been performed by transmitting a large number of packets over an m-hop source-destination pair, for $1 \leq m \leq 7$. To reduce the effect of random anomalies in the network, the tests were run 100 times and the average was found. Based on these averages, deviations, and variances the 95% confidence interval was determined.

When using the partially overlapping channels, i.e., channels 1-4-8-11, there are slight improvements over the single-channel network testbed. A similar trend of exponential decrease is apparent upon comparison to the single channel case. In this specific case, co-channel interference is reduced by increasing the value of K from one to four. Consequently, the introduction of overlapping channels adds a significant level of side-lobe interference, or more commonly known as adjacent channel interference. Recalling back to the description of the frequency spectrum shown in Fig. 1.4, nodes s_1 and s_2 will undergo the effects of adjacent channel interference, despite overcoming co-channel interference Recalling that distance is an important factor in interference and carrier sensing ranges, the distance between the these two nodes is not large enough to prevent the system from overcoming adjacent channel interference. One important point to observe from

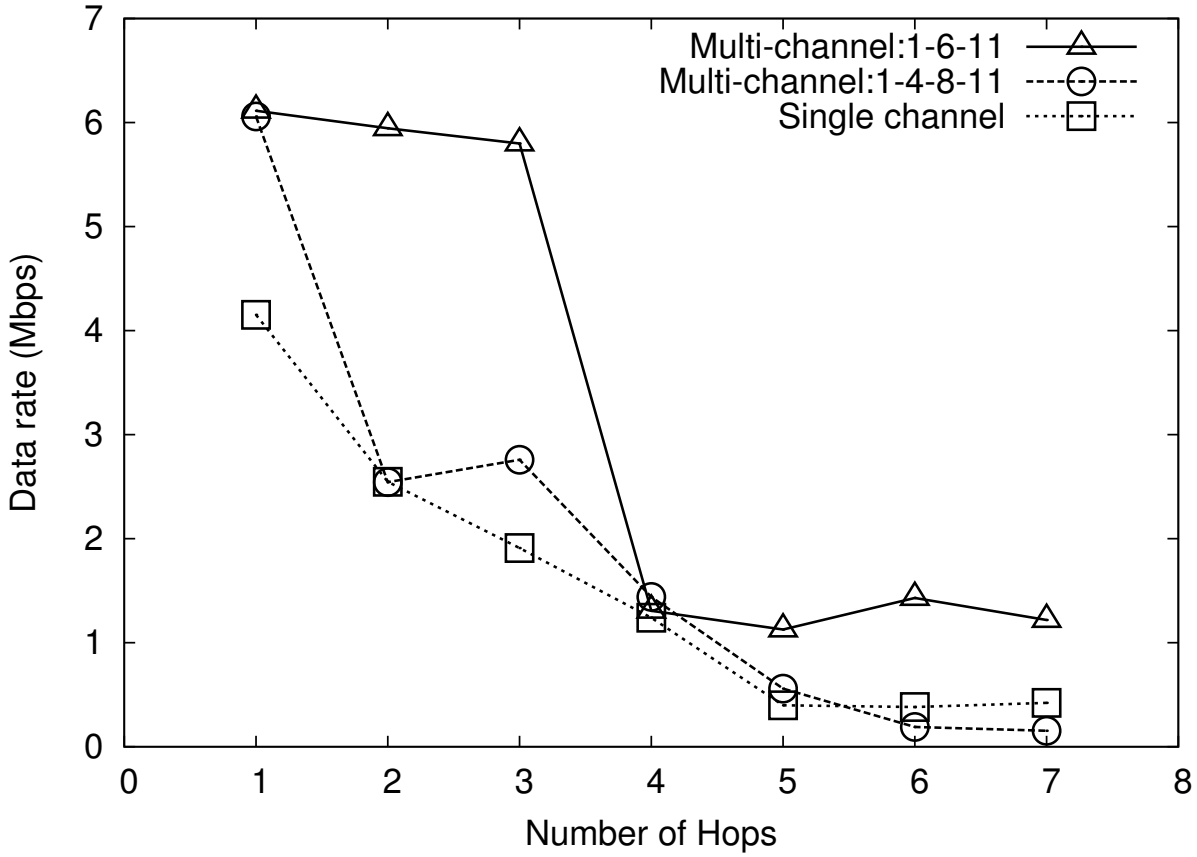


Figure 4.3: Single-channel and Multi-channel UDP performance

the usage of *Channel Set 2* relates to the increase in throughput from $m = 2$ to $m = 3$. Referring back to the frequency spectrum once again, the third hop which operates over the eighth channel can safely be considered orthogonal to the first channel; therefore, this is allows for both links (the link between s_1, s_2 and the link between s_3, s_4) to be active and undergo transmission simultaneously. It is this simultaneous transmission capability that provokes this effect. The additional levels of interference result in only a slight improvement. With the extent of this improvement, it may not warrent the possible usage within a wireless mesh network. Further testing regarding overlapping channels is required, as performed by the following sections, to deem this channel set as a possibility. Despite the gains from having multiple active links, it seems that the increased adjacent channel interference cancels most of the gains by reducing co-channel interference.

Upon using orthogonal channels, i.e., 1-6-11, significant improvements are observable. Noticeable gain is observable, especially if the distance between source and destination nodes are within three hops ($m = 1$, $m = 2$, and $m = 3$). For a three orthogonal multi-channel network, measured results report that the goodput sustains around 6 Mbps from 1-hop to 3-hop UDP transmissions, dropping by only 0.31 Mbps. Once again referring to the frequency spectrum of Fig. 1.4, it is understandable as to why the rates are sustained at these points. In this case, the first hop (between s_1 and s_2) operate over channel 1, the second hop (between s_2 and s_3) operate over channel 6, and the third hop (between s_3 and s_4) operates over channel 11. Up to this point, all links can be activated simultaneously, and undergo transmissions simultaneously. It is this fact that allows us to sustain the rates at the high levels. But, a noticeable drop occurs between $m = 3$ and $m = 4$. The main cause of this effect is that channels are required to begin being re-used at the fourth hop (between s_4 and s_5). In this case, the first channel is re-used, as we are limited to only three available channels. The impact of the large interference range once again impacts performance even in these simple cases. As a result of this drop, it is safe to assume again that the interference range encompasses more than the assumed $2 \times R_{tx}$. If this is true, each channel thus has to share and contend for the wireless medium. Although having to contend for the channel with nodes that are multiple hops away, the number of interfering nodes still remains significantly lower than the *Channel Set 1* and *Channel Set 2* counterparts. After and including the $m = 4$ point, the throughput once again flattens out, and sustains a relatively consistent level of UDP performance. This follows a similar reasoning as the first 3 hops. It is apparent here that even when using orthogonal channels in a mere chain topology mesh network we cannot ignore the effects of hidden terminals.

Table 4.2: Average UDP Throughput

Hop Count	Analytical Model	Channel Set 1	Channel Set 2	Channel Set 3
1	5.03	4.15	6.06	6.11
2	2.56	2.55	2.54	5.94
3	1.71	1.91	2.76	5.80
4	1.28	1.23	1.44	1.31
5	1.03	0.40	0.56	1.13
6	0.86	0.38	0.19	1.43
7	0.73	0.42	0.15	1.22

4.3 Channel Overlap Effects

As demonstrated with the UDP results of Section 4.2, the overlapping channel usage from *Channel Set 2* has a slight improvement while the orthogonal channel case of *Channel Set 3* has a relatively more significant impact. To further understand these effects, the analysis provided in Section 3.1 can be further investigated. Using MATLAB, the various parameters influencing this analysis were mathematically estimated (see Appendix A). Assuming an ideal filter and a random signal with a power density function as a *sinc* function occupying the 20 MHz bandwidth, the fraction of channel overlap θ, the estimated bit error rate BER, the estimated PER are calculated. The results of these calculations are summarized in Table 4.3. The noise factor only takes into consideration the noise contributed by one channel on channel 1 (i.e. the fifth row discusses the interference effects of channel 5 on channel 1). It is important to note that two calculations of packet error rate are performed. The first calculation, PER_1, pertains to the packet error rate associated with a typical data packet with an MSS of 1500 bytes, which is equivalent to 12000 bits. On the other hand, the second packet error rate PER_2 corresponds to a typical 802.11 DCF control message. For example, the RTS message is approximately 112 bytes operating at the basic data rate of 1 Mbps.

Table 4.3: Adjacent and orthogonal channel overlap simulations

Channel	θ	BER	PER_1	PER_2
1	1	0.023	1	3.08×10^{-7}
2	0.808	0.013	1	2.37×10^{-9}
3	0.546	0.003	1	0
4	0.279	7.64×10^{-5}	0.645	0
5	0.064	1.10×10^{-15}	1.50×10^{-11}	0
6	0	0	0	0

The three channel sets tested can be examined based on the calculated estimations of Table 4.3. The

single channel case of *Channel Set 1* relates to the first row, which shows the effect of co-channel interference on the system. Since a single channel is inferred for this scenario, the channel overlap has a value of one ($\theta = 1$), which is consistent with our concept of co-channel interference. Based on this, all packets are likely to become corrupted. But, the low bit error rate of RTS control messages infers that corresponding nodes in the network should be able to properly read the packets in order to appropriately run the protocol's distributed coordination function.

The second scenario using *Channel Set 2* tested the effects of overlapping channels, which incorporated the effects of three and four channel separations. The three channel separation refers to adjacent links operating on channel 1 and channel 4, or adjacent links operating on channel 8 and channel 11. Although lower than the single channel case, the relatively lower power spectrum overlap causes the adjacent channel interference. The low bit error rate can be misleading, as errors will consequently add up to provide a large packet error rate of 64.5%. The corresponding packet error rate for an RTS control message approximates to a value of 0; this means that a node operating on channel 1 cannot properly detect a signal occurring on channel 4 via the RTS-CTS message handshake mechanism. The four channel separation refers to the case where adjacent links are operating on channel 4 and channel 8 (which is equivalent to the effects of channel 5 on channel 1). The adjacent channel only contributes approximately 6.4%, causing an estimated packet error rate 1.50×10^{-11}. This is a relatively small packet error rate, which should not significantly affect transmission occurring on the adjacent channels of 4 and 8.

Lastly, the orthogonal channel case of *Channel Set 3* corresponds to the effects of channel 6 on channel 1, which is the last line of Table 4.3. Channels 7 through 11 have been omitted from the table as they follow the same trend as the 5 channel separation shown. Based on ideal filters no channel overlap occurs for channel separations exceeding 4 channels, there is no significant adjacent channel interference causing packet errors. Hence, these non-overlapping channels can safely be used together to overcome some of the co-channel interference effects. Although, it is difficult to completely remove the effects of co-channel interference as seen from the previous section, since it is difficult to attain an ideal filter. Filters that are close to ideal can be expensive and may not be suitable as cost is one main design goal for wireless mesh networks.

4.4 TCP Measurements

As TCP remains the prevalent means of accessing the Internet, it would be prudent to ignore its effects. Under similar topology and conditions as the previous UDP-based experiments, the transport layer protocol is changed from UDP to TCP. Under these scenarios, the resulting transmission duration and throughput from transmission of a 3-MB file are illustrated in Fig. 4.4 and Fig. 4.5, respectively. At each point for m for all three channel sets, the average TCP throughput was found by taking the average of 100 TCP sessions. Based on these averages, deviations, and variances the 95% confidence interval was determined and included in the graph. A successfully designed wireless mesh network should at least guarantee high successful rate in transmitting files, whether small or large in sizes, from a source to a destination that may be a few or many hops away. Sending a large file is also consistent with the idea that the source node is always saturated. The exact throughput results from the testbed experiments of the three channel set scenarios is summarized in Table 4.4. The drivers for the atheros-based wireless cards are accompanied by an application layer program that can be used to read the network interface's wireless transmission and reception statistics, including the various collisions that occur. To further understand and discuss these results, the number of collisions between data packets measured in each of the three cases are shown in Table 4.5.

Table 4.4: Average TCP Throughput without proxies

Hop Count	Channel Set 1	Channel Set 2	Channel Set 3
1	3.01	3.64	4.43
2	1.69	2.55	4.24
3	1.04	1.89	2.00
4	0.72	0.84	1.33
5	0.60	0.77	1.34
6	0.46	0.68	1.06
7	0.27	0.43	0.58

As shown in both graphs, multi-channel networks always perform better than single-channel networks. As with the UDP results, it becomes apparent that as the number of nodes increases the throughput decreases. This is consistent with our analysis since increased hops translates to increased probability of packet loss, which in turn translates into increased delay. An increased delay leads to the observed decrease in throughput. If congestion is to occur at any of the stations in the multi-hop network, the shape of throughput curve can be significantly different than the throughput curves of its UDP counterpart. It can been seen that the plotted curves for the single channel and multiple channel cases follow a similar shape to the previously plotted UDP

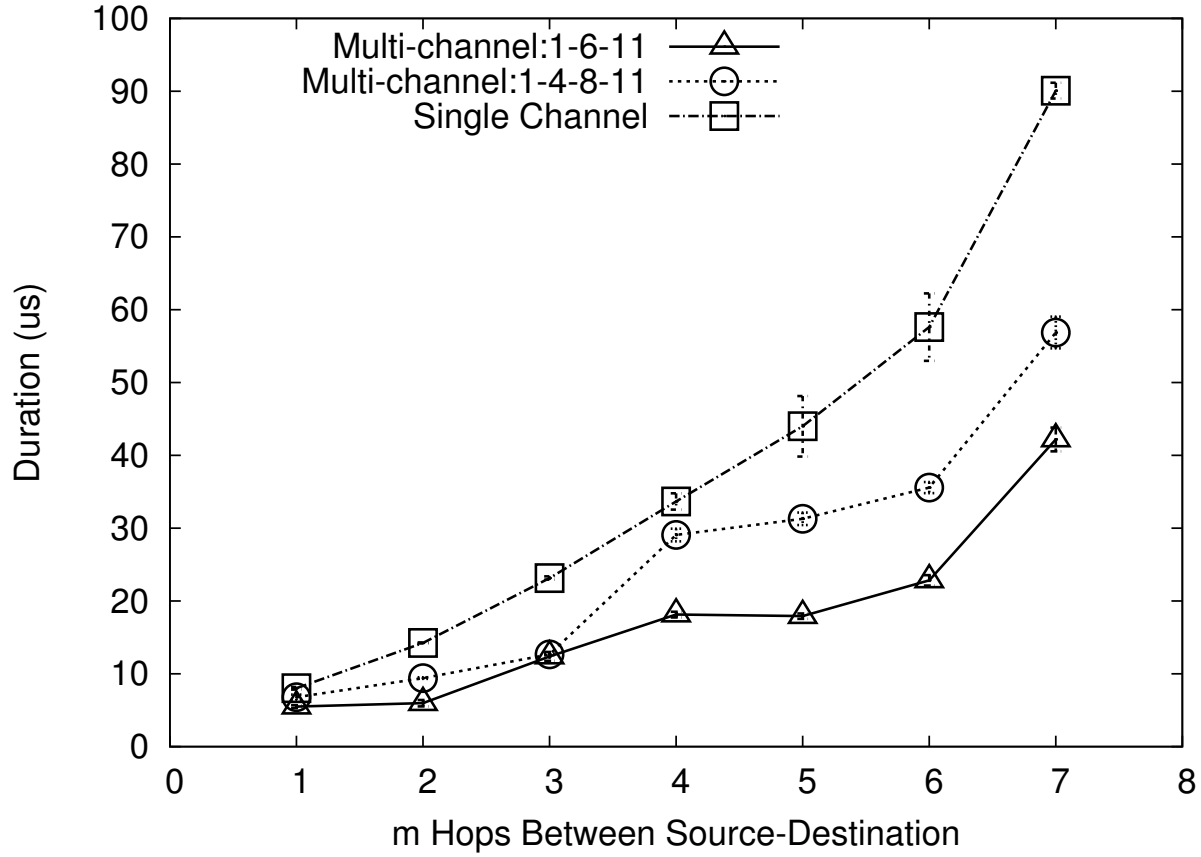


Figure 4.4: Single-channel and Multi-channel 802.11 wireless mesh networks duration based on TCP transmission of 3 MB file without Proxies

Table 4.5: Average number of collisions

Hop Count	Channel Set 1	Channel Set 2	Channel Set 3
1	381.07	266.22	2.05
2	1039.93	4615.11	87.45
3	1818.69	6229.45	1597.3
4	2487.45	7486.86	2426.53
5	3853.12	6515.71	2357.29
6	5068.74	8697.17	2190.52
7	5120.14	9516.53	3535.13

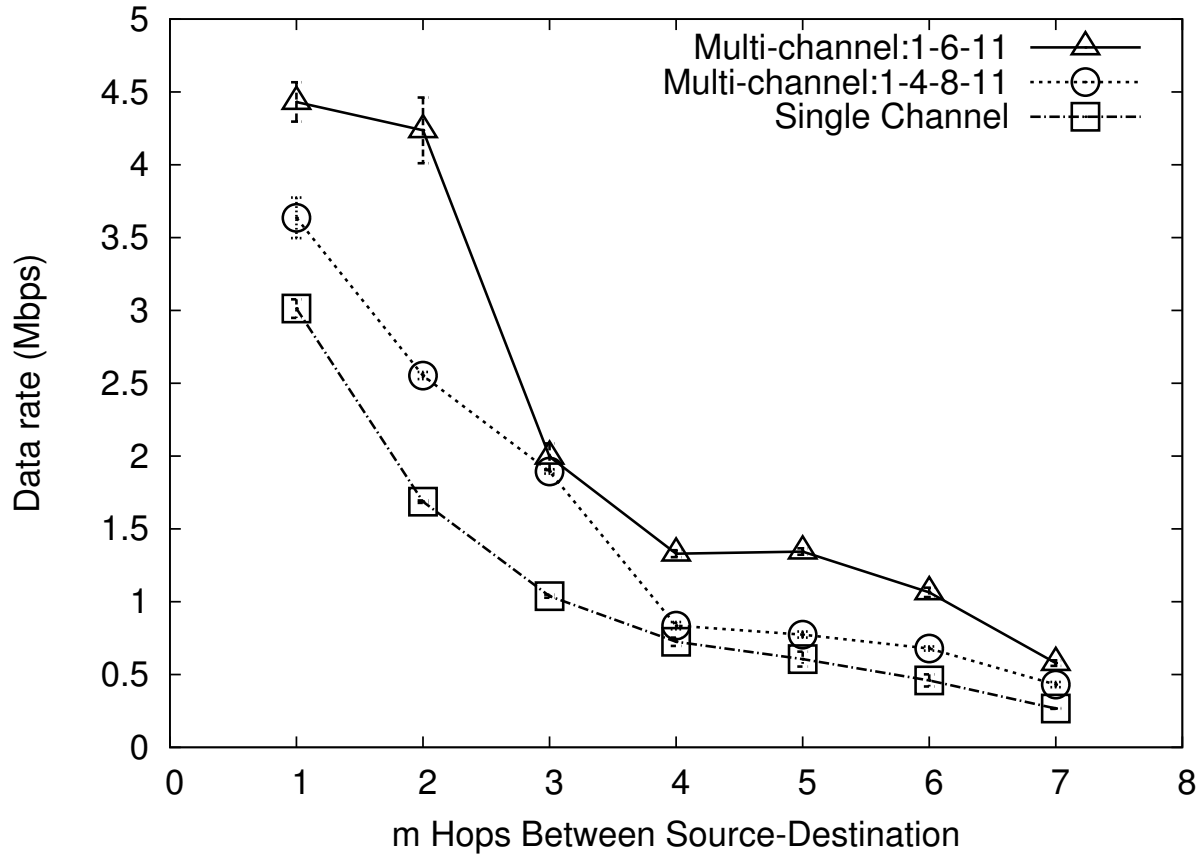


Figure 4.5: Single-channel and Multi-channel 802.11 wireless mesh networks throughput based on TCP transmission of 3 MB file without Proxies

graphs of Fig. 4.3. The higher throughputs noticed in some of the UDP results are understandable. With TCP being connection-oriented and UDP having a connectionless design. There is an overhead of extra TCP packets required to establish a connection, which reduces the achievable throughput of TCP. Also as predicted, the number of lost packets increases with number of hops for each scenario, which contributes to the throughput trends.

In the single channel case, excessive backoffs from the carrier sensing mechanism occur, which results in the relatively poor duration and throughput measurements. Referring to the number of collisions experienced, the RTS and CTS messages are seen to be incapable of preventing all of the collisions between data packets. As RTS messages operate at a lower data rate causing a significantly lower packet error rate than corresponding data error rates, they will tend to better overcome the effects of external noise factors. Therefore, the RTS-CTS mechanism may grant access to the noisy channel, but upon transmitting the data packet a packet error will likely occur in an excessively noisy environment. This consequently results in the transmitter failing to receive an ACK for the lost packet, which inevitably causes a packet timeout and retransmission of the packet. These timeouts and retransmissions lead to additional delays.

Even though there are four channels available in the second scenario to increase capacity, the RTS and CTS messages are incapable of properly deterring packet collisions, as shown by the greater number of collisions. Based on the data available in Table 4.3, the RTS and CTS control messages are seemingly incapable of detecting the transmissions on adjacent channels. With the RTS-CTS handshake not properly functioning, the MAC protocol will undergo fewer contention window backoffs as it cannot detect that the channel is busy. Therefore, packets are pushed into the network at a higher rate causing some the reduction in performance delays. As a result of this, the transmissions from adjacent channels will interfere with one another causing packets to become corrupted. This is in addition to the corruptions caused by the already noisy wireless channel. This again results in the transmitter failing to receive an ACK for the lost packet, which is inevitably another cause of packet timeout and retransmission. These timeouts and retransmissions lead to additional delays, and counteract the improvements of overcoming co-channel interference. Despite the excessive number of collisions, the use of overlapping channels still outperforms the single channel case.

Once again, the use of orthogonal channels provides the best performance gains, as expected, since the probability of collision is significantly reduced. The throughput curve for orthogonal channels demonstrates that the lower probability of collision results in a greater throughput. Similar to the overlapping channel scenario, RTS-CTS messages will more often detect that the channel is not busy. Although in this case,

the channel is not busy, since the orthogonal channels do not produce a significant interference effect (as seen in Table 4.3. Therefore, packets are pushed onto the network at a higher rate without succumbing to as much interference. Based on the collision numbers, the system is still exposed to excessive noise levels, but still producing less collisions than *Channel Set 1*. A jump in the number of collisions is noticeable when channels begin to be re-used; this infers that the larger interference range makes RTS-CTS messages incapable of listening to all the hidden terminals of the system. A real life wireless mesh network system cannot completely remove these noise factors and must be taken into consideration. These results confirm that the multi-channel case, either *Channel Set 2* or *Channel Set 3*, is a better option.

4.5 TCP Proxies

Wireless networks are always prone to packet losses due to noise and interference within medium. As measured and reported in Table 4.5, using multiple channels may not be able to completely eliminate all collisions and packet losses. The usage of TCP proxies can be implemented to help counteract the effects caused by the uncontrollable packet losses. The n-hop TCP proxy is designed to offer higher reliability for the wireless networks in heavy packet loss environments. Experiments have been performed using a 7-hop source-destination pair with proxies of length n for $1 \leq n \leq 6$. The goal of the TCP proxy scheme is take advantage of the demonstrated higher performance levels, for longer multi-hop connections. As shown in Fig. 4.5, the worst performance was found for the measured 7-hop connections: 0.27 Mbps, 0.43 Mbps, and 0.58 for *Channel Set 1*, *Channel Set 2*, and *Channel Set 3* respectively. By employing a variable number of TCP proxies in between the end-to-end connection, it is predicted that the performance levels will exceed these achievable data rates without using proxies. Upon adding the TCP proxy design, measured results are plotted in Fig. 4.6. At each point for n and for all three channel sets, the average TCP throughput was found again taking the average of 100 TCP proxy sessions. A 95% confidence interval is included. A summary of the TCP proxy results is provided in Table 4.6.

When using a single channel, no noticeable performance gains are achieved from the introduction of proxies to the network. This implies that the gains from shorter connections are completely counterbalanced by the excessive queueing and buffering at the proxies. On the other hand, the relationship between proxy length and number of proxies is clearly observable in both of the multi-channel scenarios. There is an apparent trade-off between proxy length and number of proxies. When the value of m is small, a larger number of

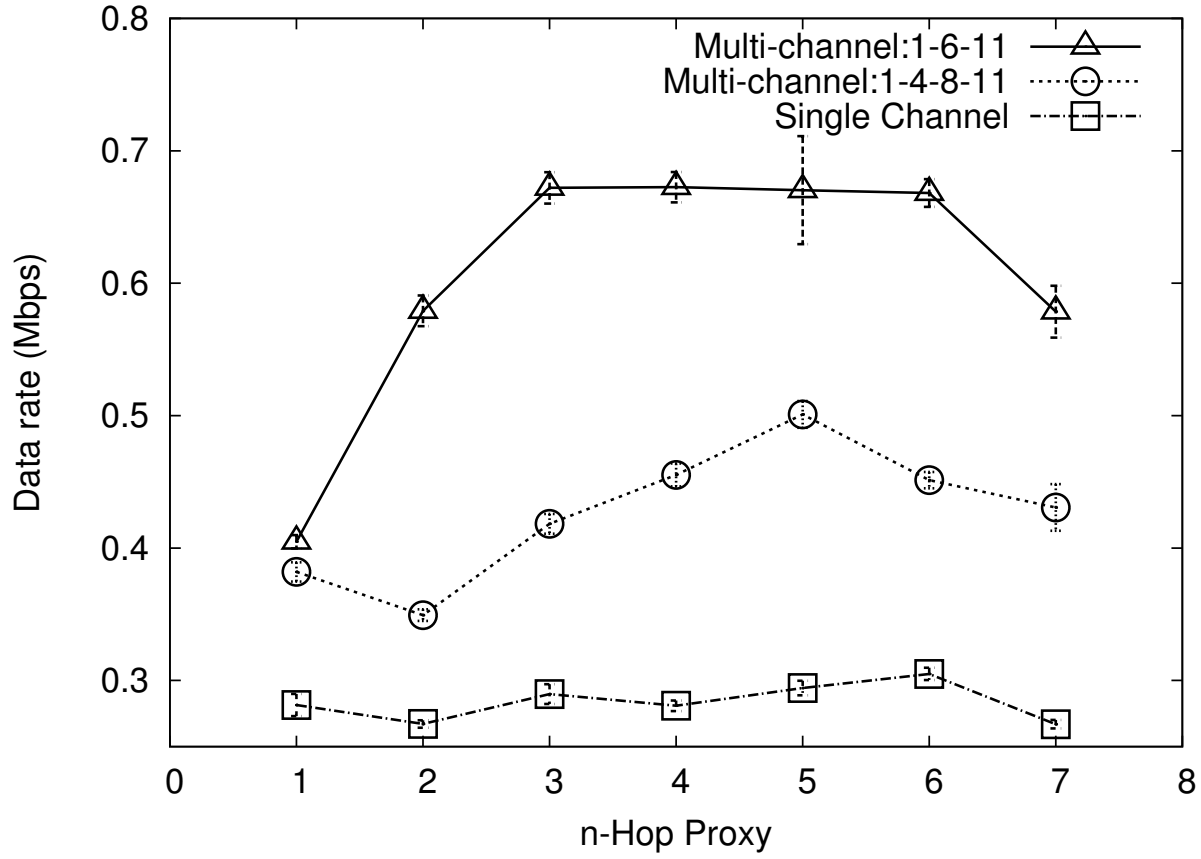


Figure 4.6: Single-channel and Multi-channel 802.11 wireless mesh network throughput based on TCP transmission of 3 MB file with Proxies

Table 4.6: Average TCP Throughput with proxies

Hop Count	Channel Set 1	Channel Set 2	Channel Set 3
1	0.28	0.38	0.40
2	0.27	0.35	0.58
3	0.29	0.42	0.67
4	0.28	0.46	0.67
5	0.29	0.50	0.67
6	0.30	0.45	0.67
7	0.27	0.43	0.58

proxies should be set up which may cause excessive queueing delays in the proxies. These queueing delays are shown to exceed the gains introduced by a shorter connections. Similarly, the larger connections associated with relatively larger values of m may have lower queueing delays but are prone to higher packet losses. Therefore, we can see that there is an optimal point that maximizes throughput as a factor of packet loss and channel share of the wireless capacity. In our testbed, the 3-hop to 6-hop TCP proxy designs in a three orthogonal channel wireless mesh networks seem to give the best goodput results. For example, the goodput of a 4-hop TCP proxy system on multi-channel network for the 7-hop TCP connection reaches close 0.7 Mbps, which is almost three times better upon comparing to the measured results for single-channel network.

4.6 Summary

Performing experiments over a scaled down multi-channel wireless mesh network testbed are necessary to help quantify the effects of the multi-hop communication design. The introduction of multiple channels to the system can improve the performance of the network for UDP and TCP sessions. But, the main problem associated with TCP-based communications is the misinterpretation of congestion in the network. If multiple paths and multiple sources were initially tested to find the inter-path interference effects, this issue would not have been as apparent. With the long connections being a possiblity for scaled mesh networks, it is important to try and take advantage of shorter connections. Adopting TCP proxies allows the system to benefit from these shorter connections, while still accommodating for the presence of long multi-hop connections. Comprehension of the system's end-to-end communication behaviour is important for the development of the mesh network protocol. The delays caused by co-channel interference and adjacent channel interference need to be minimized with the forthcoming protocol's algorithms.

Chapter 5

Future Work

To properly operate as a wireless mesh network, the underlying protocol is required to be self-forming and self-healing. This has the intention of simplifying the deployment and maintenance of such a system. As a result of these properties, it should inherently provide a sense of reliability to clients connecting seamlessly to the mesh. In fact, these types of networks are similar to ad-hoc networks, except that the nodes in wireless mesh networks remain stationary and are not constrained by limited power. This allows developers to further exploit the resources to overcome the shortcomings of a multi-hop system. Dynamic approaches may be more suitable allowing the mesh network to adapt to the changing network conditions. With regards to using commodity 802.11 wireless cards, dynamic channel assignment has been avoided since channel switching is relatively slow and requires precise synchronization to avoid permanently partitioning the network. Faster, less computationally intensive, and often less optimal techniques are required for algorithms that adjust on the fly. These are often characterized as being centralized or distributed. Centralized networks are often incapable of adapting to network changes quickly as time-sensitive information does not propagate across the network as often as required. The problem with distributed networks is the excessive overhead of distributing control information between nodes. To accommodate for this, a dedicated control channel and control interface is introduced, as described in Section 2.1. This has the intention of separating data packets from mesh management control packets.

Using the analysis and understanding of multi-radio, multi-channel wireless mesh networks thoroughly discussed in the previous sections, it is the future goal to design a distributed multi-channel wireless mesh network testbed using commodity 802.11 wireless cards. As we have already witnessed, design of a dynamic

multi-channel wireless mesh network will encompass multiple layers from the standard OSI networking model shown in Fig. 5.1(a). As the MAC layer name suggests, the layer is responsible for granting a node with controlled channel access. With respect to the aforementioned interference issues, it is the MAC layer's responsibility of minimizing these interference levels and minimizing the number of collisions the transmitted packets encounter. It is logical that the MAC layer should be responsible for channel management and channel related operations. Although these aspects improve performance and control of the wireless medium, they do not necessarily contribute to the goal of self-configuration. As packets are required to traverse across multiple hops, nodes need to be aware of how they should reach corresponding nodes within the mesh; conversely, this is the responsibility of the networking IP layer and the routing protocol. With respect to dynamic multi-channel wireless mesh networks, the new mesh protocol is required to perform the following operations at minimum:

- neighbour node detection
- efficient channel switching
- optimized channel assignment
- efficient notification and reporting of channel information.
- minimum cost routing
- minimized end-to-end performance.

It is important for each of the network interfaces to work together for the mesh to efficiently use its limited available resources. The results found from the thorough analysis of this thesis should be used to design the necessary optimization algorithms. The knowledge of channel interference, hidden nodes, interference range, etc. must be used as metrics and weights to determine suitable mesh networking operation. As already used in some of the experiments in this thesis, the framework for the multi-radio multi-channel system was created (see Appendix B for snippets of the kernel module code).

5.1 System Architecture

Fig. 5.1(a) shows the lower layers of the unmodified OSI network layer. When multiple network interface cards (NICs) are employed, each card has their own MAC layer; hence, each NIC will operate independently

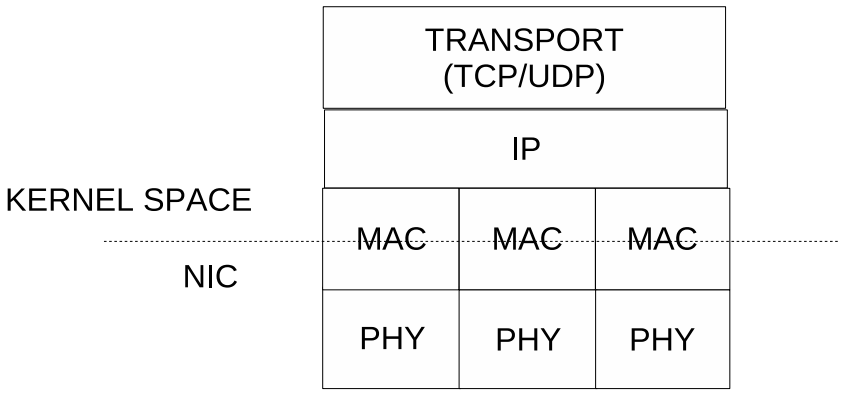
and unaware of one another. It is possible for each NIC to operate independently, but this will inevitably result in an unnecessarily large number collisions between a node's own interfaces in addition to collisions with surrounding nodes. For suitable level of control, a new MAC protocol is required.

There are two methods of achieving an appropriate MAC layer. The first method requires a wireless radio frontend, for the physical layer, and an attached FPGA to test the novel MAC layer. This method can often be quite difficult and expensive in terms of implementing and testing protocols. The second method requires using commercially available wireless cards. With most of these cards, the MAC layer is not completely available to reprogram within the kernel level's driver, as shown in Fig. 5.1(a). Much of the required MAC layer operations are programmed onto the network card's firmware. Therefore, an additional control layer is required to operate the mesh network, as shown in Fig. 5.1(b). The introduction of a new Mesh Control layer can provide the control operations necessary, as the new control layer is now aware of each of the radios attached to the node.

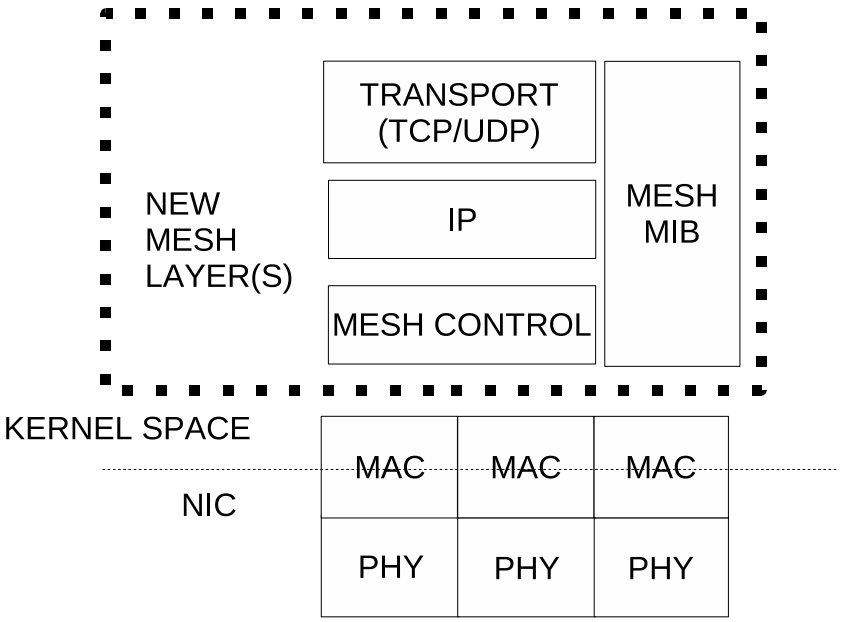
The open-source Linux operating system provides an adequate means of attaching a new layer between the original MAC and the IP layers using loadable kernel modules and/or extending the functionalities of the pre-existing drivers. Linux provides a simple mechanism known as Netfilter, which provides a series of hooks at different points in the Linux network stack. These hooks allow new protocols to seamlessly intercept packets as they move up, or move down, the protocol stack. Upon intercepting the packets, headers can be added, packets can be dropped, packets can be mangled, new packets can be created, etc. Using appropriate adjustments, new mesh control packets can be provided that perform the necessary operations stated above.

5.2 Channel Assignment

As previously mentioned, dynamic channel assignment is a key aspect in these types of protocols. To incorporate these operations, dynamic channel switching is required. One of the main reasons that dynamic channel assignment is avoided is the slow switching times associated with commodity hardware. Upon switching the channel of a wireless card, the card needs to be completely reset to accommodate for the new channel. Most wireless cards provide this option, but the latency of this operation will differ. For example, Atheros based cards can take roughly 3 msec to switch between channels based on our tests (see Section 4.1). This operation can be performed at the application layer, but another delay incurs from context switching between the kernel and the application layers. Therefore, channel switching operations need to be performed



(a) Original OSI network layers



(b) Mesh network layers

Figure 5.1: Network stack

as low as possible in the kernel, to avoid any excessive latencies. In such a system where switching delays are relatively large, their delays need to be factored into optimization algorithms as a metric when choosing channel assignments and other resource-related decisions.

5.3 Neighbour Detection, Reporting and Routing

When operating in a dynamic and distributed system, it is important that nodes are up to date on the state of the network. Report messaging is a key functionality of any dynamic multi-channel wireless mesh network design. Report messages need to be occasionally sent to neighbouring nodes to keep them up to date on resource usage. Nodes need to be informed when certain links go down, links are too congested, links are experiencing high packet loss, etc. For instance, in order to minimize the interference and prevent the hidden/exposed node problem, neighbours need to be up to date on channel usage of neighbouring nodes. This channel usage, which can be stored in a certain management information base, can then be used to estimate a channel for its network interfaces upon attempting to establish a new connection. As shown in Fig. 5.1(b), a separate module labelled MESH MIB is attached to the mesh control layer to store pertinent information. This additional messaging system allows nodes within the mesh to work together to provide the best performance.

The downside of such a system is that excessive information is propagated throughout the network, which adds overhead that takes up valuable bandwidth resources from data communication. When there are a sufficient number of available channels and radios on each node, a separate control channel interface can be beneficial. A separate channel dedicated to this control information allows the network to maintain up-to-date information. In turn, this up-to-date information allows nodes to make relatively better decisions regarding the allocation of resources.

As previously mentioned, each of the nodes act as a router, which relays the multi-hop packets between the various sources and destinations. As shown in Fig. 5.1(b), the MESH MIB is attached to the network layer. The information maintained in the management information base allows the routing mechanism to establish suitable routes. Any routing algorithm for ad hoc networks can technically be used.

Chapter 6

Conclusion

In this thesis, I have studied several aspects of designing wireless mesh networks in future. With multiple channels and multiple radios per node, network topology with small number of nodes can be designed with small overlapping coverage area. Then I have investigated end-to-end performance of transport layer traffic in the mesh networks. By initially studying the interference impact in a simple chain network model, we can devise an analytical equation which can fits the measured results of UDP traffic in a wireless chained network. The analytical results are generated through saturated and unsaturated MAC models. The measured results in experiments have validated the negative impact of deploying 802.11 technology for wireless mesh networks. In order to improve the system performance, I have been able to show through experiments that combining multiple channels with proxies is capable of improving the performance of TCP traffic in wireless mesh networks. The experimentation provides a baseline result which enables us to understand how the multiple hop wireless network interacts in response to even this simple scenario. The measured results for the TCP proxy concept indicates that there are queueing delays in proxies while they provide better reliability of traffic flow. There is a trade-off between number of proxies and the length of these proxies. The wireless medium is volatile and susceptible to time variant packet losses. On the other hand, I have shown and discussed different factors that may influence the decision of a suitable usage of proxies. These factors and parameters need to be used to establish a suitable cross-layer protocol that manages and controls channel assignment as well as proxy size, in order to provide the self-forming and self-configuring necessities of a wireless mesh network.

Appendix A

MATLAB Simulations

```
%------------CONSTANTS--------------

tx_rate = 11*10^6;  % wireless tranmsission rate/capacity [bits/sec]
tx_rate2 = 1*10^6;  % transmission rate of rts, cts, acks

CW_min = 31;        % minimum contention window
time_slot = 20*10^(-6); % slot time according to 802.11 [seconds]

size_data = 1500;   % maximum transmission unit of data [bytes]
size_data_bits = size_data*8; % size of data [bits]

time_propogation = 1/tx_rate;   % propogation time of data packet
time_propogation2 = 1/tx_rate2; % propogation time of RTS/CTS/ACK packet

% minimum total number of bits to successfully transmit a packet of data
size_total = size_data_bits + size_ack + size_rts + size_cts;

max_retries = 11;   % maximum number of MAC layer retries

size_file = 3*(2^20)*8;

total_pkts = size_file/size_data_bits;
%---------------------------------

% number of nodes in interference range including itself
nodes_int_range = zeros(7,8);

% for R_ir = 2*R_tx
nodes_int_range2= [ 2 2 0 0 0 0 0 0;
                    3 3 3 0 0 0 0 0;
```

```
                    3 4 4 3 0 0 0 0;
                    3 4 5 4 3 0 0 0;
                    3 4 5 5 4 3 0 0;
                    3 4 5 5 5 4 3 0;
                    3 4 5 5 5 5 4 3; ]

% for R_ir = 3*R_tx
nodes_int_range = [ 2 2 0 0 0 0 0 0;
                    3 3 3 0 0 0 0 0;
                    4 4 4 4 0 0 0 0;
                    4 5 5 5 4 0 0 0;
                    4 5 6 6 5 4 0 0;
                    4 5 6 7 6 5 4 0;
                    4 5 6 7 7 6 5 4; ]

nodes_sense = nodes_int_range - 1;

num_hops = 7;
hops = 1:num_hops;

for i = 1:7
    for j = 1:7
        if (nodes_sense(i,j)<=0)
            continue;
        end
        if (j==1)
            tau_test = @(p1) prob_transmission(p1,CW_min,max_retries,nodes_sense(i,j),1,0,1)
- prob_transmission(p1,CW_min,max_retries,nodes_sense(i,j),1,0,2);
        else
            q = 1/mac_pkt_delay(i,j-1);
            tau_test = @(p1) prob_transmission(p1,CW_min,max_retries,nodes_sense(i,j),q,0,4)
- prob_transmission(p1,CW_min,max_retries,nodes_sense(i,j),q,0,2);
        end
        p1 = fzero(tau_test,0.3);
        p(i,j) = p1;
        tau(i,j) = prob_transmission(p(i,j),CW_min,max_retries,nodes_sense(i,j),
q,bit_error,1);
        % calculate the single node throughput
        mac_tp(i,j) = calc_mac_tp(tau(i,j), p(i,j), nodes_int_range(i,j), 2);
        mac_pkt_delay(i,j) = (1/mac_tp(i,j))*size_data_bits;
    end
end

% calculate the multihop thorughput
Nhop_pkt_delay = zeros(1,num_hops);
for j = hops
    for i=1:j
        Nhop_pkt_delay(j)=Nhop_pkt_delay(j) + mac_pkt_delay(j,i);
    end
```

```
    Nhop_udp_tp(j)=size_data_bits/Nhop_pkt_delay(j);
end

figure;
plot(hops, Nhop_udp_tp,hops, udp_data_1ch);
```

```
% Calculate the probability of transmitting in a given time slot
%   Parameter:  p - probability of collision
%               CW_min - MAC layer's minimum contention window
%               max_retries - MAC layer's maximum retries
%               nodes - number of nodes in interference range (excluding
%                       self)
%               q - probability of packet arrival
%               b - bit error rate
%               test - test type
function [tau] = prob_transmission(p, CW_min, max_retries, nodes, q, b, test)

if (test==1)
    tau = (2.*(1-2.*p))./((1-2.*p).*(CW_min+1)+p.*CW_min.*(1-(2.*p).^max_retries));
elseif (test==2)
    %p_error = calc_prob_error(b);
    p_error =0;

    tau = 1-(1-(p+p_error)).^(1/nodes);

elseif (test==3)
    b00=((q*CW_min)/(1-(1-q)^CW_min))+
((q*CW_min*(q*CW_min+3*q-2))/(2*(1-q)*(1-(1-q)^CW_min)))+
(1-q)+((q*(CW_min+1).*(p.*(1-q)-q.*(1-p).^2))/(2*(1-q)))+
((p.*q^2)./(2*(1-q).*(1-p))).*(((CW_min)/(1-(1-q)^CW_min))-
(1-p).^2).*(((2*CW_min.*(1-p-p.*(2*p).^(max_retries-1)))/(1-2.*p))+1);
    tau = (1./b00).*(((q^2)*CW_min)/((1-q)*(1-p)*(1-(1-q)^CW_min))-((q^2)*(1-p))/(1-q));
elseif (test==4)
    time_slot = 20*10^(-6);
    size_data = 1500;   % maximum transmission unit of data [bytes]
    size_data_bits = size_data*8; % size of data [bits]
    size_mac = 272; % size of 802.11 MAC header


size_phy_h = 192;   % physical header

size_ack = 112+size_phy_h;     % size of MAC's ACK packet [bits]
size_rts = 160+size_phy_h;     % size of MAC's RTS packet [bits]
size_cts = 112+size_phy_h;     % size of MAC's CTS packet [bits]
tx_rate = 11*10^6;  % wireless tranmsission rate/capacity [bits/sec]
tx_rate2 = 1*10^6;  % transmission rate of rts, cts, acks

    % time to transmit one data packet [seconds]
    time_data = (size_data_bits+size_phy_h+size_mac)/tx_rate;
    time_rts = size_rts/tx_rate2;   % time to transmit RTS packet [seconds]
    time_cts = size_cts/tx_rate2;   % time to transmit CTS packet [seconds]
    time_ack = size_ack/tx_rate2;   % time to transmit ACK packet [seconds]
    time_difs = 50*10^(-6);         % interframe space [seconds]
    time_sifs = 10*10^(-6);         % short interframe space [seconds]
    time_propogation = 1/tx_rate;   % propogation time of data packet
```

```
time_propogation2 = 1/tx_rate2; % propogation time of RTS/CTS/ACK packet

    T_s = time_rts + time_sifs + time_propogation2 + time_cts +
time_sifs +time_propogation2 + time_data + time_sifs +
time_propogation + time_ack + time_difs + time_propogation2;
    T_s = T_s/time_slot;

    T_c1 = time_rts + time_difs + time_propogation2;
    T_c1 = T_c1/time_slot;

    temp = p/(2*(1-p));

    tau = (q + q*(T_s +T_c1*temp)*((1-2*p)/(1-p-p*((2*p)^max_retries)))
*(2/CW_min))/(1-q*nodes*(T_s +T_c1*temp));
else
    disp('Unacceptable parameter: test ');
end
```

```
% Calculate the per-hop MAC layer delay
%   Parameter:  tau - probability of a given node transmitting in a given
%                       time slot
%               p - probability of collision
%               nodes - number of nodes in an interference range including
%                       itself
%               test - version number of test
function [throughput] = calc_mac_tp (tau, p, nodes, test)

tx_rate = 11*10^6;  % wireless tranmsission rate/capacity [bits/sec]
tx_rate2 = 1*10^6;  % transmission rate of rts, cts, acks

CW_min = 31;        % minimum contention window
time_slot = 20*10^(-6); % slot time according to 802.11 [seconds]

size_data = 1500;   % maximum transmission unit of data [bytes]
size_data_bits = size_data*8; % size of data [bits]

size_phy_h = 192;   % physical header

size_ack = 112+size_phy_h;     % size of MAC's ACK packet [bits]
size_rts = 160+size_phy_h;     % size of MAC's RTS packet [bits]
size_cts = 112+size_phy_h;     % size of MAC's CTS packet [bits]

size_mac = 272; % size of 802.11 MAC header

% time to transmit one data packet [seconds]
time_data = (size_data_bits+size_phy_h+size_mac)/tx_rate;
time_rts = size_rts/tx_rate2;   % time to transmit RTS packet [seconds]
time_cts = size_cts/tx_rate2;   % time to transmit CTS packet [seconds]
time_ack = size_ack/tx_rate2;   % time to transmit ACK packet [seconds]
time_difs = 50*10^(-6);         % interframe space [seconds]
time_sifs = 10*10^(-6);         % short interframe space [seconds]

time_propogation = 1/tx_rate;   % propogation time of data packet
time_propogation2 = 1/tx_rate2; % propogation time of RTS/CTS/ACK packet

% minimum total number of bits to successfully transmit a packet of data
size_total = size_data_bits + size_ack + size_rts + size_cts;

max_retries = 11;   % maximum number of MAC layer retries

h=1;    % number of hidden nodes
k = 1;

% First calculate probability that atleast one node is transmitting in a
% given time slot
p_tr = 1 - (1 - tau)^nodes;
```

```
% Calculate the probility that a transmission occuring on the channel is
% succesful
if (test == 1)
    p_succ = (nodes*tau*(1-tau)^(nodes-1))/(p_tr);
elseif (test==2)
    p_succ = (nodes*tau*((1-tau)^(nodes-1))*((1-tau)^(h*k)))/(p_tr);
end


% calculate the average time that the channel is sensed busy
% Re: T_s = RTS + SIFS + prop_time + CTS + SIFS + prop_time + phy_hdr +
%               MAC_hdr + exp_payload + SIFS + prop_time + ACK + DIFS +
%               prop_time
T_s = time_rts + time_sifs + time_propogation2 + time_cts + time_sifs
+ time_propogation2 + time_data + time_sifs + time_propogation
+ time_ack + time_difs + time_propogation2;

% calculate the average time channel sensed busy by collision
T_c1 = time_rts + time_difs + time_propogation2;
T_c2 = time_rts + time_sifs + time_propogation2
+ time_cts + time_difs + time_propogation2;
T_c3 = time_rts + time_sifs + time_propogation2 + time_cts
+ time_sifs +time_propogation2 + time_data + time_difs + time_propogation;
T_c4 = time_rts + time_sifs + time_propogation2 + time_cts
+ time_sifs +time_propogation2 + time_data + time_sifs
+ time_propogation + time_ack + time_difs + time_propogation2;

% calculate expected length of slot time
exp_slot_time = ((1-p_tr)*time_slot)+(p_tr*p_succ*T_s)+(p_tr*(1-p_succ)*T_c1);

% calculate expected payload size
exp_payload = p_succ*p_tr*size_data_bits;

exp_payload2 = p_succ*p_tr*time_data;

% calculate the trhoughput
throughput = exp_payload/exp_slot_time;
```

```
% Calculate the overlapping channel interference. Assume that each channel
% has the same properties.
%   Parameters: seperation - sepearation of channels (i.e. if using
%                            channels 1 and 4; therefore seperation = 3)
%               chan_bw - bandwidth of each channel
function [interference] = overlapping_chan_interference(separation, chan_bw)
% assuming ideal filters
x = -100:0.01:100;
p = power(sin(x/(chan_bw/2))./(x/(chan_bw/2)), 2);
p(10001)=1;

%create ideal filter
t = -100:0.01:100;
filt = zeros(1,size(t,2));

for R = 1:size(t,2)
    if (t(R) >=-(chan_bw/2) && t(R) <= (chan_bw/2))
        filt(R) = 1;
    else
        filt(R) = 0;
    end
end

p3 = p.*filt;

%signal 2
ch_separation = 11;
freq_separation = 5*separation;

y = freq_separation-100:0.01:freq_separation+100;
%p2 = power(sin((x-freq_separation)/11)./((x-freq_separation)/11), 2);

p2 = zeros(1,size(t,2));
factor = freq_separation/0.01;
for R = 1:(size(t,2)-factor)
    p2(R+factor)= p3(R);
end

channel_overlap = (p3.*p2);

overlap_power = trapz(channel_overlap);
cochannel_overlap = (p3.*p3);
cochannel_power = trapz(cochannel_overlap);

interference = overlap_power/cochannel_power;
```

Appendix B

Linux Kernel Module Code

```
/*********************************************************************
FILE: mesh_main.c
*********************************************************************\

#include "mesh.h"
#include "mesh_route.h"

/*******************************/

static struct proc_dir_entry *mesh_proc = NULL;

static int mesh_show_stats(char *buffer, char **buffer_location,
off_t offset,int count, int *eof, void *data)
{
int len=0;
struct mesh_nbr *nbr=getNbrHead();

len = sprintf(buffer, "Address\tState\tChannel\n");

while(nbr!=NULL){
len+=sprintf(buffer+len,"%x\t%d\t%d\n",
nbr->ip_addr, nbr->channel, nbr->state);
nbr=nbr->next;
}

return len;
}

static int mesh_show_routing(char *buffer, char **buffer_location,
off_t offset,int count, int *eof, void *data)
```

```
{
int len=0;
struct mesh_route *mesh_rt=getRtHead();

len = sprintf(buffer, "Address\tChannel\tState\n");

while(mesh_rt!=NULL){
len+=sprintf(buffer+len,"%x\t%x\t%d\t%d\t%s\n",
mesh_rt->dst_ip, mesh_rt->next_hop,
mesh_rt->metric,mesh_rt->channel,
mesh_rt->dev->name);
mesh_rt=mesh_rt->next;
}

return len;
}

static int mesh_show_topology(char *buffer, char **buffer_location,
off_t offset,int count, int *eof, void *data)
{
int len=0;
struct mesh_nodes *node=getNodeHead();
struct mesh_adj *adj;

len = sprintf(buffer, "Node\tAdjacencies\n");

while(node!=NULL){
len+=sprintf(buffer+len,"%x\t%u\t%d",
node->dst_ip, node->total_metric, node->seq);
if (node->valid_flag==RT_VALID && node->total_metric != 0 &&
   node->gateway != NULL){
len+=sprintf(buffer+len,"\t%x\t%d",
node->gateway->ip_addr,
node->gateway->channel);
}
len+=sprintf(buffer+len,"\n");
adj = node->adj_head;
while(adj != NULL){
len+=sprintf(buffer+len,"\t%x\t%u\t%u\n",
adj->adj_ip, adj->channel,
adj->metric);
adj=adj->next;
}
node=node->next;
}

return len;
}
```

```
void mesh_proc_init ()
{
struct proc_dir_entry *a, *b, *c;

mesh_proc = proc_mkdir("mesh", init_net.proc_net);

if (mesh_proc == NULL) {
printk("Unable to create the mesh proc directory\n");
return;
}

a = create_proc_entry("Mesh_Neighbours", 0, mesh_proc);
b = create_proc_entry("Mesh_routing_table", 0, mesh_proc);
c = create_proc_entry("Mesh_topology", 0, mesh_proc);

if (!a) {
remove_proc_entry("mesh", init_net.proc_net);
mesh_proc = NULL;
printk("Unable to create proc entry\n");
return;
}
a->read_proc = mesh_show_stats;

if (!b) {
remove_proc_entry("mesh", init_net.proc_net);
mesh_proc = NULL;
printk("Unable to create proc entry\n");
return;
}
b->read_proc = mesh_show_routing;

if (!c) {
remove_proc_entry("mesh", init_net.proc_net);
mesh_proc = NULL;
printk("Unable to create proc entry\n");
return;
}
c->read_proc = mesh_show_topology;
}

void mesh_proc_exit ()
{
/*unregister proc file(s)*/
if (mesh_proc) {
remove_proc_entry("Mesh_topology", mesh_proc);
remove_proc_entry("Mesh_routing_table", mesh_proc);
remove_proc_entry("Mesh_Neighbours", mesh_proc);
remove_proc_entry("mesh", init_net.proc_net);
mesh_proc = NULL;
```

```
}
}

static struct net_protocol mesh_protocol = {
.handler = mesh_rcv,
};

static struct packet_type mesh_packet_type = {
.type = __constant_htons(ETH_P_MESH),
.func = mesh_rcv,
};

/*protocol registration process
  registration process for layer 4 registration*/
void protocol_reg1 ()
{
/*register mesh protocol as subprotocol of IP*/
if (inet_add_protocol(&mesh_protocol, IPPROTO_MESH) < 0)
printk("inet_init: Cannot add MESH protocol\n");
}

void protocol_reg2 ()
{
dev_add_pack(&mesh_packet_type);
}

/*protocol unregistration process*/
void protocol_unreg1 ()
{
/*unregister IPPROTO_MESH*/
if (inet_del_protocol(&mesh_protocol, IPPROTO_MESH) < 0)
printk ("Cannot delete MESH protocol\n");
}

void protocol_unreg2()
{
dev_remove_pack(&mesh_packet_type);
}

static int __init mesh_init(void)
{
struct net_device *dev;

printk("Initializing Module\n");

/*set up proc file(s)*/
mesh_proc_init();

//flag_queue = 1;
```

```
//status = STATE_IDLE;

/*should set channel to 1 (control channel)*/
/*if ((dev=dev_get_by_name(&init_net, "ath0"))!=NULL)
mesh_switch_ch(dev, 1);
if ((dev=dev_get_by_name(&init_net, "ath1"))!=NULL)
mesh_switch_ch(dev, 1);
if ((dev=dev_get_by_name(&init_net, "ath2"))!=NULL)
mesh_switch_ch(dev, 1);*/

protocol_reg1();

return 0;
}

static void __exit mesh_exit(void)
{
printk("Module exitd\n");

mesh_proc_exit();

protocol_unreg1();

return;
}

module_init(mesh_init);
module_exit(mesh_exit);

MODULE_LICENSE("GPL");
```

```
/*****************************************************************************
FILE: mesh_mib.c

Description:
  - Mesh management information base
*****************************************************************************\

#include "mesh.h"

struct neigh_entry * add_neighbour (__be32);

/*neighbour list pointers*/
struct neigh_entry *neigh_head;

struct neigh_entry *getNbrHead(){
return neigh_head;
}

__be32 getNextHop(__be32 dst_addr){

return 1;
}

int setNextHop(){

return 1;
}

struct neigh_entry * getNeighbour(__be32 gw_addr){
/*Search for neighbour in table*/
struct neigh_entry *nbr=neigh_head;

while (nbr!=NULL){
if (nbr->ip_addr == gw_addr)
break;
else
nbr = nbr->next;
}

/*if it does not exist in the table, add it*/
if (nbr==NULL){
nbr = add_neighbour(gw_addr);
}

return nbr;
}

/*return the status from the destination table based on destination address*/
int getStatus (__be32 gw_addr){
```

```
/*Search for neighbour in table*/
struct neigh_entry *nbr=neigh_head;

while (nbr!=NULL){
if (nbr->ip_addr == gw_addr)
break;
else
nbr = nbr->next;
}

/*if it does not exist in the table, add it*/
if (nbr==NULL){
nbr = add_neighbour(gw_addr);
}

/*return it's status*/
return nbr->state;
}

int setStatus(){
return 1;
}

/*add neighbour to table*/
struct neigh_entry * add_neighbour(__be32 gw_addr){
struct neigh_entry *nbr_ptr=neigh_head;
struct neigh_entry *new_nbr=kmalloc(sizeof(struct neigh_entry),GFP_KERNEL);

/*if unable to assign memory for the new neighbour*/
if (new_nbr == NULL)
return NULL;

if (nbr_ptr==NULL){
neigh_head=new_nbr;
}
else{
while (nbr_ptr->next!=NULL){
nbr_ptr=nbr_ptr->next;
}

nbr_ptr->next=new_nbr;
}

new_nbr->next=NULL;
new_nbr->ip_addr=gw_addr;
new_nbr->channel=0;
new_nbr->state=STATE_IDLE;
skb_queue_head_init(&(new_nbr->list));
```

```
return new_nbr;
}

/*delete neighbour from table, part of garbage collector*/
int del_neighbour(){

return 1;
}


/*add destination to table*/
int add_destination(){

return 1;
}

/*delete destination from table, part of garbage collector*/
int del_destination(){

return 1;
}

void garbage_collector(){

}

static int __init mesh_MIB_init(void)
{
neigh_head = NULL;

return 0;
}

static void __exit mesh_MIB_exit(void)
{

}

module_init(mesh_MIB_init);
module_exit(mesh_MIB_exit);

MODULE_LICENSE("GPL");

EXPORT_SYMBOL(getNbrHead);
EXPORT_SYMBOL(getNeighbour);
EXPORT_SYMBOL(getStatus);
EXPORT_SYMBOL(setStatus);
EXPORT_SYMBOL(getNextHop);
EXPORT_SYMBOL(setNextHop);
```

```
/*****************************************************************************
FILE: mesh_ch.c

Description:
  - Channel switching and channel handling operations
*****************************************************************************\

#include "mesh.h"

int mesh_switch_ch(struct net_device *dev, int channel)
{
struct iwreq iwr;
iw_handler handler;
struct iw_request_info info;
int err;

// set m to channel and set e to 0 for channel change using ioctl
iwr.u.freq.m = channel;
iwr.u.freq.e = 0;

rtnl_lock();

if(!netif_device_present(dev)) {
printk("mesh_switch_ch: device not present\n");
return -ENODEV;
}

if(dev->wireless_handlers == NULL) {
printk("mesh_switch_ch: wireless handlers not present\n");
return -EINVAL;
  }

handler = dev->wireless_handlers->standard[SIOCSIWFREQ-SIOCIWFIRST];

  info.cmd = SIOCSIWFREQ;
    info.flags = 0;

if((err = handler(dev, &info, &(iwr.u), NULL)))
printk("mesh_switch_ch: Wireless handler returned
an error value of %d\n", err);

rtnl_unlock();

return err;
}

static int __init mesh_ch_init (void){

return 0;
```

```
}

static void __exit mesh_ch_exit(void){
//clear_neighbours();
}

module_init(mesh_ch_init);
module_exit(mesh_ch_exit);

EXPORT_SYMBOL(mesh_switch_ch);

MODULE_LICENSE("GPL");

/*****************************************************************************
FILE: mesh_output.c

Description:
  - Handling of outgoing messages
*****************************************************************************\
#include "mesh.h"
#include "mesh_route.h"

/* This is the structure we shall use to register our function */
static struct nf_hook_ops nf_hook_send;

extern int (*mesh_send)(struct sk_buff *);

/*used to find the gateway IP address
  But, might not be necessary if
only sending it to the neighbour
based on MAC address only, and NOT
the gateway's IP address*/
__be32 mesh_find_gw(__be32 dest_addr)
{
struct rtable *rt;

{
struct flowi fl = {
.nl_u = {
.ip4_u = {
.daddr = dest_addr
}
},
};
if (ip_route_output_key(&init_net, &rt, &fl))
printk("MESH : can't find route\n");
else
printk("MESH : route found\n");
}
```

```
return rt->rt_gateway;
}

int mesh_sendrts (struct net_device *dev, __be32 s_addr, __be32 d_addr, __be32 gw_addr)
{
struct sk_buff *skb;
struct iphdr *iph;
//struct net_device *dev = skb->dev;
struct meshhdr *mh;
struct rtable *rt;
__be32 gw_addr_ctrl = ((gw_addr) & 0x00FFFFFF)|0x01000000;

//printk("MESH : %x\n", (d_addr && 0xFFFF0000);
do_gettimeofday(&val1);

if ((skb = alloc_skb(MAC_LEN + IPMESH_LEN, GFP_KERNEL))== NULL) {
printk("mesh_xmit: Cannot allocate socket buffer\n");
return 1;
}

{
struct flowi fl = {
.nl_u = {
.ip4_u = {
.daddr = gw_addr_ctrl,
}
},
.proto = IPPROTO_MESH
};
if (ip_route_output_key(&init_net, &rt, &fl))
printk("mesh_reply: ip_route_output_key
is unable to find a route\n");
else
printk("mesh_reply: ip_route_output_key has found a route\n");
}

//ctrlskb->dst = dst_clone(skb->dst);
skb->dst = &rt->u.dst;
skb->dev = skb->dst->dev;

skb_reserve(skb, MAC_LEN);
skb_reset_network_header(skb);

iph = ip_hdr(skb);
skb_put(skb, IP_LEN);
iph->version = 4;
iph->ihl = (sizeof(struct iphdr))>>2;
iph->tos = 0x00;
iph->frag_off = htons(IP_DF);
```

```
iph->ttl = 10;
iph->daddr = rt->rt_dst;
iph->saddr = rt->rt_src;
iph->protocol = IPPROTO_MESH;
iph->tot_len = htons(IPMESH_LEN);
ip_select_ident(iph, &rt->u.dst, NULL);

ip_send_check(iph);

//skb->transport_header = skb->network_header + IP_LEN;
mh = (struct meshhdr *)skb_put(skb, MESH_LEN);
//mh = (struct meshhdr *)skb_transport_header(skb);
mh->type = TYPE_CTRL;
mh->control = RQST_CONN;
mh->channel = htons(CH5);
mh->srcip = s_addr;
mh->dstip = d_addr;

skb->protocol = htons(ETH_P_IP);

skb->dst->output(skb);

return 1;
}

unsigned int mesh_send2(unsigned int hooknum, struct sk_buff *skb,
const struct net_device *in, const struct net_device *out,
int (*okfn)(struct sk_buff *))
{
int status;
__be32 gw_addr;
struct mesh_nbr *nbr;
struct sk_buff *cpskb; /*buffer to hold copied data packet*/

printk("MESH SEND3\n");
/*IF it's not a mesh control packet*/
if (((ntohl(ip_hdr(skb)->daddr)) & 0xFFFF0000) == convert_addr(198,168,0,0)
   &&(ip_hdr(skb)->protocol != IPPROTO_MESH)){
/*get next hop address*/
gw_addr = mesh_find_gw(ip_hdr(skb)->daddr);

/*check status and destination/neighbour table*/
nbr = getNeighbour(gw_addr);

if (nbr->state == STATE_IDLE){
nbr->state == STATE_RQST_SENT;
if((cpskb = skb_copy(skb,GFP_KERNEL)) == NULL) {
printk("mesh_xmit: Cannot copy the socket buffer
to be transmitted\n");
```

```
return 1;
}
// add the socket buffer to the end of the queue
skb_queue_tail(&(nbr->list), cpskb);

/*send request to send message to set up path*/
if(!(mesh_sendrts(skb->dev, ip_hdr(skb)->saddr,
    ip_hdr(skb)->daddr, gw_addr))){
printk("MESH ERROR\n");
}

kfree_skb(skb);

return NF_STOLEN; /* accept ALL packets, and continue */
}
else {
/*add it to the queue or send it off immediately*/
/*therefore, pass it to scheduling thread*/
if((cpskb = skb_copy(skb,GFP_KERNEL)) == NULL) {
printk("mesh_xmit: Cannot copy the socket buffer
to be transmitted\n");
return 1;
}

// add the socket buffer to the end of the queue
skb_queue_tail(&(nbr->list), cpskb);

kfree_skb(skb);

return NF_STOLEN; /* accept ALL packets, and continue */
}
}
/*if it is a mesh packet*/
else{
return NF_ACCEPT;
}
}

int init_nf_hook_send()
{
/* Fill in our hook structure */
nf_hook_send.hook = mesh_send2; /* Handler function */
nf_hook_send.hooknum = 4; /* NF_IP_POST_ROUTING First hook for IPv4 */
nf_hook_send.pf = PF_INET;
nf_hook_send.priority = NF_IP_PRI_LAST; /* Make our function first */

nf_register_hook(&nf_hook_send);

return 0;
```

```
}

/*thread function to handle scheduling of queued packets*/
int mesh_scheduling(void *arg)
{
struct sk_buff *skb;
struct mesh_nbr *nbr;

struct dst_entry *dst;
struct net_device *dev;
unsigned int hh_len;

while(1) {
nbr=getNbrHead();
while(nbr!=NULL){
if(nbr->state == STATE_PATH_EST){
if((skb=skb_dequeue(&(nbr->list))) != NULL){
/*if(dev_queue_xmit(skb)){
printk("MESH scheduling:
dev_queue_xmit returned
an error or negative integer\n");
kfree_skb(skb);
}*/

/********************/

dst=skb->dst;
dev=dst->dev;
hh_len=LL_RESERVED_SPACE(dev);
/* Be paranoid, rather than too clever. */
if (unlikely(skb_headroom(skb) < hh_len
    && dev->header_ops)) {
struct sk_buff *skb2;

skb2 = skb_realloc_headroom(skb,
LL_RESERVED_SPACE(dev));
if (skb2 == NULL) {
kfree_skb(skb);
break;
}
if (skb->sk)
skb_set_owner_w(skb2, skb->sk);
kfree_skb(skb);
skb = skb2;
}

if (dst->hh)
neigh_hh_output(dst->hh, skb);
else if (dst->neighbour)
```

```
dst->neighbour->output(skb);

//kfree_skb(skb);
/**********************/

//if the queue is empty, increment counter, and
//if counter reaches certain amount,
//return state to IDLE
continue;
}
}
/*if state is IDLE or WAITING,
cannot transmit yet or has nothing to transmit*/
nbr=nbr->next;
}
msleep(100);
}

return 0;
}

static int __init mesh_out_init(void)
{
init_nf_hook_send();

/*start schuduling thread*/
kernel_thread(&mesh_scheduling, NULL, 0);

return 0;
}

static void __exit mesh_out_exit(void)
{
nf_unregister_hook(&nf_hook_send);
}

module_init(mesh_out_init);
module_exit(mesh_out_exit);

MODULE_LICENSE("GPL");
```

```
/*****************************************************************************
FILE: mesh_input.c

Description:
  - Handling of incoming messages
*****************************************************************************\
#include "mesh.h"
#include "mesh_route.h"

struct timeval val1;
struct timeval val2;
struct timeval val3;
struct timeval val4;

__be32 mesh_find_gw(__be32 dest_addr)
{
struct rtable *rt;

{
struct flowi fl = {
.nl_u = {
.ip4_u = {
.daddr = dest_addr
}
},
};
if (ip_route_output_key(&rt, &fl))
printk("MESH : can't find route\n");
else
printk("MESH : route found\n");
}

return rt->rt_gateway;
}

struct net_device * get_gw_dev(__be32 dest_addr){
struct rtable *rt;

{
struct flowi fl = {
.nl_u = {
.ip4_u = {
.daddr = dest_addr
}
},
};
if (ip_route_output_key(&rt, &fl))
printk("MESH : can't find route\n");
}
```

```
return (&rt->u.dst)->dev;
}

int mesh_reply(struct sk_buff *skb){
struct sk_buff *new_skb;
struct iphdr *iph;
struct meshhdr *mh;
struct rtable *rt;

//printk("MESH : %x\n", (d_addr && 0xFFFF0000);

if ((new_skb = alloc_skb(MAC_LEN + IPMESH_LEN, GFP_KERNEL))== NULL) {
printk("mesh_xmit: Cannot allocate socket buffer\n");
return 1;
}

{
struct flowi fl = {
.nl_u = {
.ip4_u = {
.daddr = ip_hdr(skb)->saddr,
}
},
.proto = IPPROTO_MESH
};
if (ip_route_output_key(&rt, &fl))
printk("mesh_reply: ip_route_output_key
is unable to find a route\n");
else
printk("mesh_reply: ip_route_output_key has found a route\n");
}

//ctrlskb->dst = dst_clone(skb->dst);
new_skb->dst = &rt->u.dst;
new_skb->dev = skb->dst->dev;

skb_reserve(new_skb, MAC_LEN);
skb_reset_network_header(new_skb);

iph = ip_hdr(new_skb);
skb_put(new_skb, IP_LEN);
iph->version = 4;
iph->ihl = (sizeof(struct iphdr))>>2;
iph->tos = 0x00;
iph->frag_off = htons(IP_DF);
iph->ttl = 10;
iph->daddr = rt->rt_dst;
iph->saddr = rt->rt_src;
```

```
iph->protocol = IPPROTO_MESH;
iph->tot_len = htons(IPMESH_LEN);
ip_select_ident(iph, &rt->u.dst, NULL);

ip_send_check(iph);

//skb->transport_header = skb->network_header + IP_LEN;
mh = (struct meshhdr *)skb_put(new_skb, MESH_LEN);
//mh = (struct meshhdr *)skb_transport_header(skb);
mh->type = TYPE_CTRL;
mh->control = RQST_ACCEPT;
mh->channel = htons(CH5);
mh->srcip = mesh_hdr(skb)->srcip;
mh->dstip = mesh_hdr(skb)->dstip;

new_skb->protocol = htons(ETH_P_IP);

new_skb->dst->output(new_skb);

return 0;
}

int mesh_forward_rqst(struct sk_buff *skb){
struct sk_buff *new_skb;
struct iphdr *iph;
struct meshhdr *mh;
struct rtable *rt;
__be32 gw_addr = mesh_find_gw(mesh_hdr(skb)->dstip);
__be32 gw_addr_ctrl = ((gw_addr) & 0x00FFFFFF)|0x01000000;

//printk("MESH : %x\n", (d_addr && 0xFFFF0000);

if ((new_skb = alloc_skb(MAC_LEN + IPMESH_LEN, GFP_KERNEL))== NULL) {
printk("mesh_xmit: Cannot allocate socket buffer\n");
return 1;
}

{
struct flowi fl = {
.nl_u = {
.ip4_u = {
.daddr = gw_addr_ctrl,
}
},
.proto = IPPROTO_MESH
};
if (ip_route_output_key(&rt, &fl))
printk("mesh_reply: ip_route_output_key
is unable to find a route\n");
```

```
else
printk("mesh_reply: ip_route_output_key has found a route\n");
}

//ctrlskb->dst = dst_clone(skb->dst);
new_skb->dst = &rt->u.dst;
new_skb->dev = new_skb->dst->dev;

skb_reserve(new_skb, MAC_LEN);
skb_reset_network_header(new_skb);

iph = ip_hdr(new_skb);
skb_put(new_skb, IP_LEN);
iph->version = 4;
iph->ihl = (sizeof(struct iphdr))>>2;
iph->tos = 0x00;
iph->frag_off = htons(IP_DF);
iph->ttl = 10;
iph->daddr = rt->rt_dst;
iph->saddr = rt->rt_src;
iph->protocol = IPPROTO_MESH;
iph->tot_len = htons(IPMESH_LEN);
ip_select_ident(iph, &rt->u.dst, NULL);

ip_send_check(iph);

//skb->transport_header = skb->network_header + IP_LEN;
mh = (struct meshhdr *)skb_put(new_skb, MESH_LEN);
//mh = (struct meshhdr *)skb_transport_header(skb);
mh->type = TYPE_CTRL;
mh->control = RQST_CONN;
mh->channel = htons(CH5);
mh->srcip = mesh_hdr(skb)->srcip;
mh->dstip = mesh_hdr(skb)->dstip;

new_skb->protocol = htons(ETH_P_IP);

new_skb->dst->output(new_skb);

return 0;
}

int mesh_forward_accept(struct sk_buff *skb){
struct sk_buff *new_skb;
struct iphdr *iph;
struct meshhdr *mh;
struct rtable *rt;
__be32 gw_addr = mesh_find_gw(mesh_hdr(skb)->srcip);
__be32 gw_addr_ctrl = ((gw_addr) & 0x00FFFFFF)|0x01000000;
```

```
//printk("MESH : %x\n", (d_addr && 0xFFFF0000);

if ((new_skb = alloc_skb(MAC_LEN + IPMESH_LEN, GFP_KERNEL))== NULL) {
printk("mesh_xmit: Cannot allocate sreturnocket buffer\n");
return 1;
}

{
struct flowi fl = {
.nl_u = {
.ip4_u = {
.daddr = gw_addr_ctrl,
}
},
.proto = IPPROTO_MESH
};
if (ip_route_output_key(&rt, &fl))
printk("mesh_reply: ip_route_output_key
is unable to find a route\n");
else
printk("mesh_reply: ip_route_output_key has found a route\n");
}

//ctrlskb->dst = dst_clone(skb->dst);
new_skb->dst = &rt->u.dst;
new_skb->dev = new_skb->dst->dev;

skb_reserve(new_skb, MAC_LEN);
skb_reset_network_header(new_skb);

iph = ip_hdr(new_skb);
skb_put(new_skb, IP_LEN);
iph->version = 4;
iph->ihl = (sizeof(struct iphdr))>>2;
iph->tos = 0x00;
iph->frag_off = htons(IP_DF);
iph->ttl = 10;
iph->daddr = rt->rt_dst;
iph->saddr = rt->rt_src;
iph->protocol = IPPROTO_MESH;
iph->tot_len = htons(IPMESH_LEN);
ip_select_ident(iph, &rt->u.dst, NULL);

ip_send_check(iph);

//skb->transport_header = skb->network_header + IP_LEN;
mh = (struct meshhdr *)skb_put(new_skb, MESH_LEN);
//mh = (struct meshhdr *)skb_transport_header(skb);
```

```
mh->type = TYPE_CTRL;
mh->control = RQST_ACCEPT;
mh->channel = htons(CH5);
mh->srcip = mesh_hdr(skb)->srcip;
mh->dstip = mesh_hdr(skb)->dstip;

new_skb->protocol = htons(ETH_P_IP);

new_skb->dst->output(new_skb);

return 0;
}

int mesh_rcv(struct sk_buff *skb)
{
struct meshhdr *mh;
struct iphdr *iph;
struct mesh_nbr *nbr;

printk("MESH Receive: \n");

mh = mesh_hdr(skb);
iph = ip_hdr(skb);

/*is the message a routing message*/
if (mh->type=TYPE_RT_INFO){
mesh_route_rcv(skb);
return 1;
}

/*received request connection packet*/
if(mh->control == RQST_CONN) {
//check if packet needs to be forwarded

nbr = getNeighbour(mesh_find_gw(mh->srcip));

if ((iph->daddr & 0x00FFFFFF) != (mh->dstip & 0x00FFFFFF)){
printk ("MESH : forwarding connection request message\n");
if (mesh_forward_rqst(skb)){
printk("mesh_rcv: mesh_sendcts
failed to send CTS message\n");
}

nbr->state = STATE_RQST_SENT;

kfree_skb(skb);
return 1;
}
mesh_switch_ch(get_gw_dev(mh->srcip), NUM5);
```

```
//if not forwarding, reply with the accept message
if(mesh_reply(skb)) {
printk("mesh_rcv: mesh_sendcts failed to send CTS message\n");
}

nbr->state=STATE_PATH_EST;

kfree_skb(skb);
return 1;
}
else if (mh->control == RQST_ACCEPT) {
printk ("MESH : received connection acceptance\n");

nbr = getNeighbour(mesh_find_gw(mh->dstip));

if ((iph->daddr & 0x00FFFFFF) != (mh->srcip & 0x00FFFFFF)){

mesh_switch_ch(get_gw_dev(mh->srcip), NUM5);
mesh_switch_ch(get_gw_dev(mh->dstip), NUM5);

nbr->state=STATE_PATH_EST;

printk("MESH : forwarding accept message\n");
if (mesh_forward_accept(skb)){
printk("mesh_rcv: mesh_sendcts failed
to send CTS message\n");
}

kfree_skb(skb);
return 1;
}

mesh_switch_ch(get_gw_dev(mh->dstip), NUM5);

nbr->state=STATE_PATH_EST;

kfree_skb(skb);
return 1;
}
else {
printk ("MESH : received invalid mesh control message type\n");
}

kfree_skb(skb);

return 1;
}

EXPORT_SYMBOL(mesh_rcv);
```

```
EXPORT_SYMBOL(val1);
EXPORT_SYMBOL(val2);

MODULE_LICENSE("GPL");
```

```
/*********************************************************************
FILE: mesh.h

Description:
  - mesh header file
  - mesh header frame structure
  - packet types
*********************************************************************\

#ifndef _MESH_H
#define _MESH_H

#define IPPROTO_MESH 70
#define ETH_P_MESH 0x5000
#define TIMEOUT 30
#define MAX_NEIGHBOURS 3
#define MAC_LEN ETH_HLEN
#define IP_LEN sizeof(struct iphdr)

struct meshhdr{
__u8 type;
__u8 control;
__be16 channel;
__be32 srcip;
__be32 dstip;
};

static inline struct meshhdr *mesh_hdr(const struct sk_buff *skb)
{
return (struct meshhdr *)skb_transport_header(skb);
}

/*message types*/
#define TYPE_CTRL 0x01
#define TYPE_RT_INFO 0x02

/*control message types*/
#define HELLO  0x01 /*hello message*/
#define HELLO_REPLY 0x02 /*reply to hello message*/
#define RQST_CONN 0x03 /*request connection - include requested channel*/
#define RQST_ACCEPT 0x04 /*accept requested connection and channel request*/
#define CH_BUSY 0x05 /*channel busy - include channel status of node*/
#define CONN_FULL 0x06 /*connections full*/
#define CONN_REFUSE 0x07 /*refuse connection*/
#define DISCONN 0x08 /*Disconnection notice*/
#define RQST_CHANGE 0x09 /*Request channel change*/
#define SWITCH 0x0A /*Channel switch notice*/
#define BACK_CTRL 0x0B /*back in control*/
#define MESH_RTS 0x0C /*request to send*/
```

```
#define MESH_CTS 0x0D /*clear to send*/
/**************/

#define MESH_LEN (sizeof(struct meshhdr))
#define MACIP_LEN (MAC_LEN+IP_LEN)
#define IPMESH_LEN (IP_LEN + MESH_LEN)

extern int mesh_rcv(struct sk_buff *skb);
extern int mesh_switch_ch(struct net_device *dev, int channel);

/* flag to signal a thread to transmit a new socket buffer through
dev_hard_start_xmit */
//extern int flag_queue;

/* flag to signal the completion of control message exchange */
//extern int status;

static inline __u32 convert_addr(unsigned char ip1,
unsigned char ip2, unsigned char ip3, unsigned char ip4)
{
__u32 addr = 0;
addr += ip1;
addr <<= 8;
addr += ip2;
addr <<= 8;
addr += ip3;
addr <<= 8;
addr += ip4;

   return addr;
}

/*this node's status*/
/*since it could be used for multiple communications,
needs to be set as a linked list*/
/*based on next hop address*/
struct node_status{
struct node_status *next;
struct node_status *previous;
int state;
__be32 dst_addr;
int count;
};

/* socket buffer list or queue (a link list) */
struct sk_buff_head list;

/*for simulation purposes only, simulated channel variable*/
int channel;
```

```
extern struct timeval val1;
extern struct timeval val2;

/* used to maintain list of wireless devices and their names*/
/* should attempt to find a method to search for wireless devices */
struct mesh_dev {
struct net_device *dev;
    //struct mesh_route *route_entry;
    int index;
    u_int32_t ip;
    u_int32_t netmask;
    char name[IFNAMSIZ];
    struct mesh_dev *next;
    struct socket *sock;
int num_active; //number of active communications using this mesh device
};

/*********channel information***********/
#define CHANNEL_CTRL 0x01

//get rid of all bits except for bit 10(channel 11), 7(channel 8), and 3(channel 4)
#define CH11 0x0400
#define CH8 0x0080
#define CH4 0x0010
#define NUM4 4
#define NUM8 8
#define NUM11 11

#endif
```

References

[1] *IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2007.

[2] K. Nahm, A. Helmy, and C.-C. Jay Kuo, "Tcp over multihop 802.11 networks: issues and performance enhancement," in *Proc. of MobiHoc '05*, 2005, pp. 277–287.

[3] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on tcp performance," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 209–221, 2005.

[4] J. Deng, B. Liang, and P. Varshney, "Tuning the carrier sensing range of ieee 802.11 mac," in *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, vol. 5, 29 2004, pp. 2987 – 2991.

[5] P. Kyasanur and N. H. Vaidya, "Capacity of multi-channel wireless networks: impact of number of channels and interfaces," in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005, pp. 43–57.

[6] *IEEE P802.11s/D3.0 Draft*, 2009.

[7] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network," in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005, pp. 31–42.

[8] *IEEE Std 802.11: Amendement 5: Enhancements for Higher Throughput*, 2009.

[9] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh, "Partially overlapped channels not considered harmful," in *Proceedings of the joint international conference on Measurement and modeling of computer systems*, 2006, pp. 63–74.

[10] A. Subramanian, H. Gupta, and S. Das, "Minimum interference channel assignment multi-radio wireless mesh networks," in *Proc. of the Sensor, Mesh, and Ad Hoc communications*, 2007, pp. 481–490.

[11] P. Kyasanur and N. H. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *SIGMOBILE Mobile Computing and Communication Review*, vol. 10, no. 1, pp. 31–43, 2006.

[12] A. Raniwala and T. Chiueh, "Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, March 2005, pp. 2223–2234.

[13] A. Das, H. Alazemi, R. Vijayakumar, and S. Roy, "Optimization models for fixed channel assignment in wireless mesh networks with multiple radios," in *Proc. of the Sensor, Mesh, and Ad Hoc Communications*, 2005, pp. 463–474.

[14] J. Postel, *Transmission Control Protocol*, 1981.

[15] S. Floyd and T. Henderson, *The NewReno Modification to TCP's Fast Recovery Algorithm*, 2004.

[16] J. Postel, *User Datagram Protocol*, 1980.

[17] K. L. E. Law and A. Kohn, "Topology designs with controlled interference for multi-radio wireless mesh networks," in *Mobility '08: Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, 2008, pp. 1–6.

[18] J. So and N. H. Vaidya, "Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, 2004.

[19] P. Bahl, R. Chandra, and J. Dunagan, "Ssch: slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, 2004.

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

[21] A. Kohn and K. E. Law, "Experiments of multi-channel 802.11 wireless mesh networks with tcp proxies," in *Biennial Symposium on Communications*, 2010.

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

[23] B.-J. Kwak, N.-O. Song, and L. E. Miller, "Performance analysis of exponential backoff," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, 2005.

[24] K. Duffy, D. Malone, and D. Leith, "Modeling the 802.11 distributed coordination function in non-saturated conditions," *Communications Letters, IEEE*, vol. 9, no. 8, pp. 715–717, Aug 2005.

[25] O. Tickoo and B. Sikdar, "Modeling queueing and channel access delay in unsaturated ieee 802.11 random access mac based wireless networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 878–891, 2008.

[26] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: a simple model and its empirical validation," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 303–314, 1998.

[27] S. Kopparty, S. Krishnamurthy, M. Faloutsos, and S. Tripathi, "Split tcp for mobile ad hoc networks," in *IEEE Global Telecommunications Conference, 2002*, vol. 1, Nov. 2002, pp. 138–142.

[28] A. Bakre and B. Badrinath, "I-tcp: indirect tcp for mobile hosts," *International Conference onDistributed Computing Systems*, p. 0136, 1995.

[29] "Network simulator. ns2. http://www.isi.edu/nsnam/ns."