INTELLIGENT WIRELESS MULTI-BEAM DIRECTIONAL ROUTING WITH SOFTWARE-DEFINED NETWORK IMPLEMENTATION

by

KE BAO

FEI HU, COMMITTEE CHAIR
SUNIL KUMAR
XIAOYAN HONG
AIJUN SONG
TODD FREEBORN

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of The University of Alabama

TUSCALOOSA, ALABAMA

2017
ABSTRACT

Wireless mesh networks (WMNs) have drawn lots of attention in the past decades due to its scalability, robustness and flexibility. However, the performance of WMNs is still limited by the wireless bandwidth, radio frequency interference, etc. To deal with the limitation, our research first focuses on a particular radio frequency technology, called Multi-beam directional antennas (MBDAs). Our goal in this study is to come up with a series of Artificial Intelligent (AI) strategies to explore MBDAs during routing implementation. Then, the AI method is expanded to a more general WMNs with heterogeneous wireless devices, in which we use centralized network management structure for the purpose of traffic engineering optimization.

First, we present a novel routing scheme for WMNs with MBDAs. It has the following 3 features: (1) Ripple-Diamond-Chain (RDC) shaped routing is proposed to explore the multi-direction transmission capability of the MBDAs. (2) Systematic link quality modeling is employed to measure the link quality in dynamical network conditions. (3) AI-Augmented path link selection enables an intelligent path determination in dynamic WMN environment.

Second, we expand the routing/TE problem from conventional WMNs to Software-Defined Networking based WMNs (SD-WMNs) and propose a novel TE structure on SD-WMN called “Prediction-based Link Uncertainty Solution in SD-WMNs” (PLUS-SW). The PLUS-SW possesses a centralized traffic engineering and wireless channel scheduling on WMNs according to the paradigm of SDN in order to efficiently arrange the network traffic and omit wireless interference in a global manner. Moreover, PLUS-SW employs double-layer supervised learning model to predict unexpected wireless link failure in the sense that the central controller can notice the potential link failure threat and send back the back-up solution to affected routers ahead the link failure.
Finally, a wireless network platform is also introduced. This platform is built with Software Defined Radio hardware, called USRP. In this platform, we achieved some preliminary functions of WMNs, such as real-time video transmission, cross-layer design and etc. The platform can work as a test bed to estimate the performance of proposed design of traffic engineering.
DEDICATION

This dissertation is dedicated to everyone that helped me in my pursuing of the doctoral degree. In particular, my parents and close friends who stood by me throughout the time taken to complete the Ph.D program.
LIST OF ABBREVIATIONS AND SYMBOLS

**DTW** Dynamic Time Warping.

**GEI** Gait Energy Image.

**HMM** Hidden Markov Model.

**IMU** Inertial Measurement Unit.

**LDPC** Low-Density Parity-Check.

**PIR** Pyroelectric Infrared.

**TDE** Time-Delay Embedding.

**ZVC** Zero-Velocity Crossings.
ACKNOWLEDGMENTS

I am glad to have the opportunities to express my gratitude to the faculty, colleagues and friends who helped complete the dissertation.

Foremost, I would like to give my sincere gratitude to Dr. Fei Hu, my advisor who, with extraordinary patience and consistent encouragement, gave me great help by providing me the innovative spot of the area, guided me into the fascinating projects for my Ph.D program. Without his guidance and constant feedback this PhD would not have been achievable.

Besides my advisor, many thanks also to Dr. Sunil Kumar. It is his suggestions that draw my attention to a number of deficiencies and make many things clearer. I am thankful for his patience, motivation, enthusiasm, and immense knowledge.

I would like to thank the rest of my committee members: Dr. Xiaoyan Hong, Dr. Aijun Song, Dr. Todd Freeborn for their encouragement, insightful comments, and challenging questions.

Last but not the least, I am indebted to my gratitude to my beloved parents who have always been helping me out of difficulties and supporting me without a word of complaint. I also owe my sincere gratitude to my friends and my fellow classmates who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis.
CONTENTS

ABSTRACT ................................................................. ii

DEDICATION ............................................................ iv

LIST OF ABBREVIATIONS AND SYMBOLS ............................... v

ACKNOWLEDGMENTS ................................................... vi

LIST OF TABLES ......................................................... xi

LIST OF FIGURES ....................................................... xii

1 INTRODUCTION ....................................................... 1
  1.1 Background and Motivation ...................................... 1
  1.2 Multi-beam network protocols .................................. 4
  1.3 Framework of SDN ................................................. 6

2 AI-AUGMENTED, RDC SHAPED ROUTING IN WMN WITH MBDAS .... 9
  2.1 Background ......................................................... 9
  2.2 Related Works .................................................... 13
  2.3 FL-Based Link Quality Dynamics Modeling ...................... 15
    2.3.1 Rateless Codes CDF ........................................ 16
    2.3.2 Capture Effect .............................................. 19
    2.3.3 Link Bandwidth ............................................. 20
    2.3.4 Diamond Transmission Potential .......................... 21
LIST OF TABLES

2.1 DCF Beam Table .............................................................. 35
## LIST OF FIGURES

1.1 WMN architecture ........................................ 1
1.2 Multi-beam Direction Antenna (MBDA) .......................... 4
1.3 Multi-beam Directional Antenna modes .......................... 5
1.4 SDN three-layer structure ................................... 6
2.1 Concept of Ripple-Diamond-Chain Routing ....................... 9
2.2 Proposed Solution: function modules .......................... 11
2.3 Multi-path routing .......................................... 15
2.4 Rateless Codes for multi-beam transmission .................... 17
2.5 Estimated CDF for different SNRs .............................. 17
2.6 Decoding CDF ............................................... 19
2.7 Capture effect ............................................... 19
2.8 Reinforcement learning for Main Path Search .................. 27
2.9 Diamond chain: Add side paths to the main path ............... 30
2.10 The cases of Path generation and collision .................... 30
2.11 Different situations of RDC Routing: (left) bottleneck link RD; (right) our solution. 32
2.12 Multi-beam MAC frame architecture .......................... 34
2.13 Enhanced DCF Phase ....................................... 35
2.14 Ripple-to-ripple propagation pattern .......................... 36
2.15 Timeline of the ripple schedule control scheme ............... 37
2.16 WMN Simulation Topology ................................... 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.17 Performance comparison between our multi-beam single-ripple</td>
<td>45</td>
</tr>
<tr>
<td>communications with enhanced PCF/DCF, without PCF enhancement (DCF</td>
<td></td>
</tr>
<tr>
<td>only), and conventional 802.11 protocol</td>
<td></td>
</tr>
<tr>
<td>2.18 QoS Performance in Ripple Communications</td>
<td>46</td>
</tr>
<tr>
<td>2.19 Performance comparison between our diamond-based routing scheme,</td>
<td>46</td>
</tr>
<tr>
<td>conventional PCF-based routing, and ripple-only routing protocols</td>
<td></td>
</tr>
<tr>
<td>2.20 Performance comparison: routing with and without Q-learning</td>
<td>47</td>
</tr>
<tr>
<td>2.21 Comparisons of three video transmission schemes</td>
<td>47</td>
</tr>
<tr>
<td>2.22 Effects of using AI algorithms</td>
<td>48</td>
</tr>
<tr>
<td>3.1 SD-WMN Architecture</td>
<td>54</td>
</tr>
<tr>
<td>3.2 Two-layer link failure prediction</td>
<td>56</td>
</tr>
<tr>
<td>3.3 Proposed SD-WMN MAC layer superframe structure</td>
<td>58</td>
</tr>
<tr>
<td>3.4 Model structure of Kalman Filtering</td>
<td>71</td>
</tr>
<tr>
<td>3.5 Filtered SNR (Kalman Filtering)</td>
<td>72</td>
</tr>
<tr>
<td>3.6 Alternative paths for link failure</td>
<td>77</td>
</tr>
<tr>
<td>4.1 ns-3 models</td>
<td>82</td>
</tr>
<tr>
<td>4.2 SDN based WMN platform in ns-3</td>
<td>83</td>
</tr>
<tr>
<td>4.3 SDN-switch structure SDN library</td>
<td>84</td>
</tr>
<tr>
<td>4.4 SDN-controller application interface</td>
<td>85</td>
</tr>
<tr>
<td>4.5 Performance evaluation of two-layer prediction</td>
<td>85</td>
</tr>
<tr>
<td>4.6 Performance comparison between PLUS-SW and conventional routing</td>
<td>86</td>
</tr>
<tr>
<td>4.7 Performance comparisons under different moving speeds</td>
<td>86</td>
</tr>
<tr>
<td>4.8 Jitter Comparison</td>
<td>88</td>
</tr>
<tr>
<td>4.9 Performance for different number of hops</td>
<td>90</td>
</tr>
<tr>
<td>5.1 GNU Radio GRC audio example</td>
<td>93</td>
</tr>
<tr>
<td>5.2 USRP board and host PC structure</td>
<td>94</td>
</tr>
<tr>
<td>5.3 Test bed hardware setup</td>
<td>95</td>
</tr>
<tr>
<td>Section</td>
<td>Content</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5.4</td>
<td>Real-time video transmission system at RF frequency</td>
</tr>
<tr>
<td>5.5</td>
<td>Spectrum sensing flowchart</td>
</tr>
<tr>
<td>5.6</td>
<td>Spectrum sensing result</td>
</tr>
<tr>
<td>5.7</td>
<td>FFT result of the frequency hopping experiment</td>
</tr>
<tr>
<td>5.8</td>
<td>A video frame with two different resolutions (left: 196×90 pixels; right: 800×600 pixels)</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 Background and Motivation

Wireless mesh networks (WMNs) have gained popular applications due to its fast wireless deployment, flexible network topology, low-cost maintenance, and large-scale radio coverage [4]. Unlike Wi-Fi, it does not need any cables among its wireless backbone nodes that are called mesh routers (MRs). Each MR serves as a tree root since it can communicate with many nodes in its tree. Each of these tree nodes is called mesh client (MC). The MC typically has less resources (memory, CPU speed, link rate, etc.) than the MR. A MC can send packets through its tree and reach a MR, which can reach another MR or use a gateway to talk with Internet, as shown in Figure 1.1.

![WMN architecture](image)

Figure 1.1: WMN architecture

Most of today’s WMN products use omni-directional antennas, which can cause interference to neighboring nodes even though it intends to talk with a specific node. The directional
antenna can limit all energy within a small angle (called beam), and does not leak energy to neighboring nodes as long as they are not in the beam coverage. Due to its well-focused directionality, it can propagate the radio for a longer distance than the omni-directional antenna. However, general directional antennas can only target one direction each time and are thus blind to other directions. This limits its throughput performance. Multi-beam directional antennas (MBDAs) overcome such a limit by allowing the simultaneous packet transmissions in multiple beams (directions) [16]. A MBDA can also simultaneously receive the packets from multiple beams. This is extremely helpful to large-scale ad hoc network where multiple sender-receiver pairs may establish the communication paths at the same time. If the intersection nodes (located in multiple routing paths) could simultaneously relay the traffic for different communication pairs through its multi-beam capability, we can greatly improve the network throughput and shorten the end-to-end routing delay.

In order to improve the performance of WMNs, especially for MBDAs, this thesis contains three parts on the study of WMNs. We first target the use of MBDA for routing performance enhancement. The goal of this part is to build a new routing protocol that can fully explore the benefit of MBDAs. We especially target the WMN applications like airborne networks, where each node (aircraft) has sufficient memory space and computation capability to handle the traffic in multiple beams. Although MBDA can significantly improve the throughput via multi-direction simultaneous transmission, conventional multi-hop wireless routing protocols cannot realize the benefits of MBDAs. For example, if we use conventional routing protocols (e.g. DSR or AODV), they typically only provide a single path between the source and destination. Thus we may not be able to fully use the potentials of MBDAs in the path since only one single direction of data forwarding is needed. We cannot use existing multi-path routing schemes either, since they do not have tight control of the convergence property of the paths: if the paths are widely disjointed, there will be very few nodes located in the intersections of multiple paths. Only those intersection nodes have the potential of using multiple beams. Thus most nodes still use single-beam communications since they only need
to talk with one node in the next hop. To overcome the problems of conventional routing
schemes, we propose diamond-shaped routing scheme, which uses regular traffic divergence
and convergence to relay the data. Each time a node aggregates or separates the traffic, it
can explore the benefits of MBDAs.

Second, we expand the idea of MBDAs-equipped WMNs on the Software Defined Net-
working (SDN), where a centralized control system in SDN-based WMNs replaces the dis-
tributed structure in traditional WMNs. In this fashion, we expect to significantly improve
the performance of WMNs in terms of traffic engineering by the network-wide observation
and management of SDN structure. Based on the centralized control feature of SDN, the
traffic balancing and radio frequency allocation are calculated by central controller period-
ically according to the traffic demand. Moreover, we employ supervised learning to predict
unexpected link failures. Then, a SDN switch can request back-up solution once a link failure
is detected. Finally, according to the potential problems predicted by SDN switch, a central
controller could calculate a back-up solution for the link failure and inform the switches.

Third, we introduce a wireless transmission testbed based on software-defined radio
(SDR). The goal of the platform is to develop a WMN research platform with respect to
the capabilities of cognitive radio network (CRN). In this test bed, we use GNU Radio as
the SDR software component, while the universal software radio peripheral (USRP) boards
serve as the hardware components. GNU Radio is a free and open-source software develop-
ment toolkit that provides signal processing blocks to implement software radios [4]. The
USRP board is a hardware support for the GNU Radio software, and is used for wireless
communication applications from DC to almost 6 GHz. Various wireless communication
research projects have used this SDR platform, such as public safety, spectrum monitoring,
radio networking, cognitive radio satellite navigation, and amateur radio [8]. Other func-
tions related to CRN, such as frequency hand-off and energy-based spectrum sensing, have
also been implemented in this test bed. Furthermore, the SDR platform are equipped with
multiple directional antennas to simply imitate the working patterns of MBDAs, in which a
series of higher layer protocols can be studied based on the WMNs test bed with MBDAs features.

1.2 Multi-beam network protocols

The Multi-beam directional antennas (MBDAs) module used in this thesis is based on the antenna module from [16] [31] [57]. An antenna system consists of a set of non-overlapping beams and each beam possesses a narrow main lobe covering an azimuth plane in pre-fixed direction. Suppose a multi-beam antenna system consists of $M$ beams, and these $M$ beams cover all 360 degree. Then each beam $b_i$ covers $\frac{360^\circ}{M}$ (Figure 1.2). Each beam $b_i$ has an on-off switch, and signals from each of the beams can be sent via either a combined transceiver or beam-specific transceiver. Thus a MBDA is capable of sending or receiving data in different beams, and can also turn on/off different subset of beams.

![Figure 1.2: Multi-beam Direction Antenna (MBDA)](image)

Here we further explain its principle in terms of concurrent packet receiving (CPR) and concurrent packet transmission (CPT). If a multi-beam antenna system turns on all the beams as shown in Figure 1.3a, it covers almost 360 degrees but with good signal separation between beams. This is different from an omni-directional antenna that just simply radiates the same signals to all directions without beam separations (thus with high RF interference between neighbors).
As a half-duplex antenna system, the MBDAs need to follow CPT/CPR principle. That is, it cannot send signals in part of beams and receive signals in other beams. At any time, a MBDA makes all beams operate in Tx (transmission) or Rx (receiving) status. If some beams are sending data and others are receiving data, the side lobes of the Rx beams will leak significant energy into the main lobes of Tx beams. This makes the node not able to interpret the envelopes of the modulated signals. Thus it will fail in packet decoding. This is the main reason of following CPT/CPR principle. A MBDA system however is able to selectively turn off some subnets of beams, and the other beams have to work under the same working mode, which is receiving mode or sending mode (see the figure 1.3b 1.3c).

Compare with the conventional omni-directional antennas, the biggest advantage of the MBDAs is spatial reuse, in which multiple packets can be forwarded or received simultaneously via different beams. This fact will greatly enhance the total throughput of WMNs. Moreover, we can take the advantage of the benefits of long communication range provided by the MBDAs. Assume the maximum power of an antenna system is denoted as $P_{\text{max}}$. Although the same power level can be used in the entire MBDA system, the transmission range of some particular beams (directions) can be further increased if this power is concentrated on some subnets of beams pointing to some particular directions.

A MBDA (see Figure 1.3) has three main features: (1) it can easily detect the incoming signals in any beam by using Direction of Arrival (DoA) estimation; (2) If it switches from transmission (Tx) to receiving (Rx) mode, or from Rx to Tx node, all beams must be
switched together into the same mode. This is because all antenna hardware elements are synchronized into the same antenna weight vector control; (3) Once the multi-beam antenna switches to Tx mode, the destination nodes in all beams should get ready for reception at the same time. Otherwise, the beam bandwidth is wasted.

1.3 Framework of SDN

A SDN possesses a three-layer structure as shown in figure 1.4. Data layer is the basic infrastructure of SDN. It consists of network devices, such as routers and switches. The control layer stands for one or multiple centralized controllers. At the top, the Application layer represents a number of software and applications, in which network engineers can use them to manage SDN. Control layer communicates with the data plane through a well-defined programming interface, while there are also some Application programming interfaces between Application Layer and Controller layer.

![Figure 1.4: SDN three-layer structure](image)

In this framework, the heterogeneous data plane is hidden behind the control layer from the network administrators who are working on the Application layer. As mentioned before, the control layer can provide a straightforward and abstracted view of the entire network to the administrator. Network administrators can easily manipulate the entire network via
writing programs upon the control layer. Then, the control layer further changes the rules of the network devices in data layer according the commands from the Application layer. In this process, some mechanisms are required to assure the commands from Application layer applicable for both controller layer and data layer, as well as a proper interaction between control layer and data layer.

The first mechanism is network observation. Back in the first section, we mentioned that SDN takes the advantage of global view of entire network. Specifically, SDN switches and routers are able to count the indicated category of packets. The counting results are forwarded to the centralized controller periodically. This process could be complicated and tedious due to the huge size of the network and various counting tasks according to different purposes. As a result, some works specialized address on the monitoring process of SDN, such as Frenetic [59]. The general goals of Frenetic including: (1) network traffic observation; (2) Network management; (3) Network consistency in network updating process [9]. With the help of the Frenetic platform, the network engineers working in the application layer are able to manage SDN without considering the lower level issues. Frenetic improves the efficiency of the network monitor process with multiple ideals, including data flow selecting, refining, splitting, combining and aggregating.

Second, the network updating mechanism is also a critical task in SDN. A regular network is extremely dynamic due to the network topology alteration, link collapse, etc. Ideally, all the network devices in the data plane are successfully updated by the central controller at the same time. However, due to the huge number of network devices and unstable connections between controller and switches, a flawless network update is hardly achieved. One solution is separating the upgrade process in two steps. Particularly, a controller first chooses a number of network devices who can be upgraded easily and stably. Then, these devices broadcast the update commands to the switches and routers around them. Finally, the upgrading process will cover the entire network. This idea lowers the burden of central controller by distributing the update process upon multiple switches. However, this idea brings a problem
that the switches and routers in the network may work in the different versions of network policies. In another word, in a particular moment, some nodes may have been upgraded into the newest policies, while others are still using the old version. This inconsistency would bring some problems, such as routing loop. Hence, the per-packet consistency is introduced to deal with this problem. The basic idea is ensuring a packet transmitted in the network should be processed by the same network policy.
2 AI-AUGMENTED, RDC SHAPED ROUTING IN WMN WITH MBDAS

2.1 Background

Considering the WMN multi-beam routing under highly diversified radio conditions, each region can have very different link qualities, and some links suffer fast channel fading. Moreover, the use of directional antennas further complicates the communication conditions since there could be node capture phenomenon. It means that one or multiple beams are stuck into carrier listening state due to other nodes data transmissions in those directions. The node keeps listening and performs window backoff (in MAC layer). Thus it wastes much time detecting unproductive traffic, i.e., packets not for itself.

![Concept of Ripple-Diamond-Chain Routing](image)

Figure 2.1: Concept of Ripple-Diamond-Chain Routing

Obviously, a one-shot link selection in each path without considering the comprehensive, long-term link quality state change could result in the sub-optimal routing. A Markov-like state transition model may be used to capture such link state change (in terms of fading, capture, interference, etc.) in different times. The path establishment should select each link based on link state transition model.
To solve the above issues, we propose an innovative WMN routing strategy that can fully utilize the multi-beam advantages through a special ripple-diamond-chain (RDC) transmission style. Its basic idea is illustrated in Figure 2.1. The nodes belonging to the same ripple have the same number of hops to the MR. Those ripple IDs can be easily determined through the popular ad hoc network routing protocols such as AODV or DSR. To form a RDC topology, we will first search for the main path, which consists of a series of links with the best quality statistically (such as the minimum of antenna capture probability) among all the links in the same ripple range. To utilize the node multi-beam communication capability, we also establish some side paths by asking the nodes in the same ripple to help forward data. Those main path nodes become the multi-beam divergence/convergence points (for traffic diffusion and aggregation). All the nodes in the main path and side paths form a chain of diamonds. The ripples look like water waves. Specifically, when one ripple is sending data, the next ripple can only receive data. At the same time, 2-ripple-away nodes can also send data for pipelined transmissions; and so on. Therefore, besides the diamond chain formation, we also propose a ripple-to-ripple transmission schedule control scheme in the RDC-based routing.

The above concept seems straightforward. However, a few challenging issues need to be solved: (1) **How to find the main path?** The main path does not necessarily have the shortest distance to the MR. Conventional ad hoc routing schemes may not apply here since they do not aim to select a path with the best cumulative (long-term) channel quality across all individual links. (2) **How to overcome fast fading in multiple beams?** This research targets fast fading radio environment, which is typical in WMNs deployed in congested cities or other places with many moving obstacles between mesh nodes (but the network nodes are relatively static). It is not realistic to select an accurate sending rate in each link that matches with the current channel conditions since the link quality can change in even sub-packet level. Rateless codes may be used to overcome such a shortcoming since it does not need to adjust the sending rates explicitly and just simply sends encoded packet
pieces (called symbols). And the receiver just needs partial symbols to construct the original packet. However, how do we integrate the rateless codes with the multi-beam transmission?

(3) Multi-beam transmission protocol in each ripple: The RDC routing consists of ripple-to-ripple localized communications, which typically belong to MAC layer issue. Our routing must be integrated with an efficient multi-beam MAC in each ripple. Especially, we need a multi-beam-oriented transmission control scheme to synchronize the beam communications since a MBDA cannot allow a part of beams to be in transmission status and the rest to be in reception status. This is mainly because all antenna beam elements must be involved in the same communication mode. Otherwise, one beams signal will leak out of its side lobs and then cause significant interference to the main lobe of other beams. Then the problem is: how do we design an integrated RDC routing (across ripples) and multi-beam MAC scheme (in each ripple)?

(4) The ripple-to-ripple schedule control: how do we control the alternative sending and receiving operations across all the ripples in the main path and side paths?

(5) QoS-oriented multi-beam transmission: How do we select proper side paths for different priorities of data? And so on.

Our solution to the above issues consists of a series of tightly coupled function modules, as illustrated in Figure 2.2. The 3 most important modules are 1 main path search (see Figure 2.1 the green, solid line), 2 diamond chain formation (it consists of the main path and
side paths), and ripple schedule control (for ripple-to-ripple, pipelined transmissions).

To achieve task 1, we will first define the weighted sum of different link quality metrics, including (1) the decoding cumulative distribution function (CDF), which reflects the statistical distribution of the ACK feedback delays during the transmission of rateless coded packets; (2) capture effects, which reflects the possibility of a nodes directional antennas being captured by unproductive traffic; (3) diamond transmission probability, which measures the possibility of a link becoming part of the main path of a diamond-chain routing. Based on different QoS requirements, we adjust the weights of those 3 items, and use fuzzy logic (FL) to obtain the fused metric that measures the dynamic link quality. Such a link quality will be used for the reward calculation in the reinforcement learning (RL)-based path searching. A low-complexity RL algorithm generates the main path.

To achieve task 2, we add side paths to the main path to form a diamond chain routing architecture. We then use rateless codes to encode the packets into different pieces (i.e., symbols), and dispatch the symbols to multiple beams. In each ripple (i.e., one-hop neighborhood), we propose a special multi-beam MAC protocol to achieve collision-free neighborhood communication.

Task 3 will be solved through a pipelined ripple-to-ripple schedule control scheme.

In summary, the main contributions of RDC routing scheme includes 3 aspects:

(1) Ripple-Diamond-Chain (RDC) shaped routing: to explore the multi-direction transmission capability of the MBDAs, first, we propose to use rateless codes to obtain loss-resilient symbols for original packets. Those symbols can go through each beam in the same time. Then we propose to use ripples to differentiate each hop of nodes in the tree topology of the WMN, which consists of mesh routers (tree roots) and a large number of mesh clients (tree nodes). Finally, we require that each node to send most symbols in the main path with the best overall link quality, and the node simultaneously dispatches the symbols to the neighboring nodes of the main path node. The symbols are divergent into multiple paths but converge into the main path node in the next hop. The entire routing topology looks
like a diamond chain. By using such RDC style routing, we can fully explore the MBDA benefits.

(2) Systematic link quality modeling: Our research targets the highly dynamic radio conditions in the WMN. The directional antennas can cause node capture issue. The link could have deep fading in each hop. The rateless codes need to adjust the transmission pause time. We propose to integrate all these factors together to determine the link quality in dynamical network conditions.

(3) AI-Augmented path link selection: Our routing scheme is augmented via Artificial Intelligence (AI) algorithms. Especially, we use two AI techniques to enhance the routing performance: Fuzzy Logic (FL): To adapt different QoS requirements, we propose to use FL to define the weighted link quality. Thus we know which link should be selected for different QoS flows. Reinforcement Learning (RL): Since the dynamic radio conditions need a long-term consideration of the throughput performance within multiple phases of routing path control, we propose to use RL to select the main path based on the cumulative throughput rewards in all links.

The rest of the chapter is organized as follows: In section 2, we will briefly summarize the related works, then we introduce our FL-based link quality modeling in section 3. Section 4 moves to the RL-based main path establishment process. The diamond routing process is further explained in section 5. It also provides the ripple schedule control principle. The detailed performance analysis and simulation results are given in section 6 and section 7, respectively. Finally, we conclude this paper in section 8.

2.2 Related Works

There is very little research on the special network protocols designed for MBDAs. Most of those works assume single-beam directional antennas. There are a few studies on how to optimize the MAC protocols to explore MBDA benefits. For example, an enhanced 802.11 distributed coordination function (DCF) framework is proposed in [31] in order to achieve
concurrent multi-beam transmissions. In [15] the point coordination function (PCF) enhancement is described in order to adapt to multi-beam QoS requirements. A well-controlled multi-beam scheduling protocol is discussed in [5].

However, to the best of our knowledge, we have not seen any systematic work on the establishment of a high-throughput routing scheme in fast fading, MBDA-equipped WMN. We are the first team to propose the RDC routing idea to fully explore the benefits of MBDA. We are also the first time to propose the use of AI (including RL and FL) to enhance the WMN routing performance. Compared to other existing work, our RDC routing fills out the blanks of multi-beam routing design since no throughput-efficient multi-beam routing was proposed before. And the traditional WMN routing schemes are mostly based on simple extensions of AODV or DSR-like ad hoc routing protocols. Most of them find only a single path. Even some use multi-path, they do not have our special ripple-diamond architecture for the use of multi-beam delivery capabilities.

Moreover, we are the first team to seamlessly integrate Rateless codes (called QoS-aware Fountain codes), as well as its CDF properties, with this proposed diamond routing topology. Rateless codes and diamond routing are a match made in the heaven because they work together to maximize WMN throughput: rateless codes use self-adaptive rate control to adapt to any complex radio conditions, and diamond routing uses multi-branch routing paths to distribute those rateless code symbols to different beams. Both aim to maximize the network throughput (i.e., let more packets go through the mesh network).

**Multi-path Routing:** Multi-path routing can overcome the impacts of fast fading channels by distributing the packets to multiple paths [53]. It can put more traffic load in better paths. Some methods aim to find totally disjoint paths in order to avoid any single-point failure impact [17] (Figure 2.3 (1)). Most methods allow the existence of some intersection nodes among multiple paths [58] (Figure 2.3 (2)). In our particular situation, we need to fully utilize the capability of multi-beam transmissions, and we assume each node (red color nodes) has enough buffer size to hold the aggregated traffic from other nodes.
beams. All the aggregation nodes are in the main path, which has the best overall link quality (statistically). Therefore, the diamond chain architecture (see Figure 2.3 (3)) is more suitable to our application. It tightly controls each hop (ripple) of traffic divergence and convergence schedules.

![Figure 2.3: Multi-path routing](image)

**AI-Enhanced Routing:** AI algorithms can add high intelligence to wireless networking. In our previous works [63], we have successfully used RL to solve the cognitive radio spectrum handoff issues. We have also used Bayesian learning to detect idle spectrum in cognitive radios [27] [28]. However, we are not aware of any work on AI-based multi-beam routing in WMNs. In this work, we will use both FL and RL to search the main path under fast fading channel conditions.

### 2.3 FL-Based Link Quality Dynamics Modeling

In this section, we will model the link quality dynamics. Especially, we will define a metric that indicates how good a link is, in terms of becoming a part of the main path. Such a metric will be an integrated value since we will need to consider multiple factors that contribute to the link quality. In this paper, we will consider the following 4 factors:

1. **Rateless codes CDF:** We will explain how rateless codes perfectly fit the multi-beam architecture, and show how CDF indirectly reflects the link dynamics under fast fading;
2. **Capture effect:** When a directional antenna is engaged into the continuous listening to an unproductive traffic flow and thus cannot respond to the traffic for it, the capture effect occurs. Obviously, the main path should avoid those links that can be easily captured.
3. **Bandwidth:** this refers to the link rate between the beams of the sender and receiver nodes.
(4) Diamond transmission probability: This measures how likely a node can join the main path. A node with more high-quality beams (links) can more likely participate in the side paths for a main path. Thus it better qualifies for the diamond chain routing process.

Besides the above 4 factors, this section will also explain how FL can be used to perform weighted integration of those factors. Our FL scheme can automatically assign different weights to each factor based on the QoS requirements of 5 typical traffic classes.

2.3.1 Rateless Codes CDF

Rateless codes have been proved to be able to automatically adapt to fast fading channel conditions without the need of sending rate adjustments [12]. While conventional wireless networks need to select the proper sending rate among a few pre-fixed ones, rateless codes just continuously send out the encoded packet pieces (called symbols). If the ACK is received, that means the receiver can successfully reconstruct a packet from the symbols, then the sender starts to send the symbols of the next packet. Otherwise, it keeps sending the symbols of the last packet until ACK arrives. Note that the receiver does not need to receive all pieces to recover the original packet. A good coded packet just needs a small redundancy to well recover the original packet. Rateless codes perfectly fit the multi-beam transmission architecture. As shown in Figure 2.4, by assigning the symbols to different beams, we can distribute those redundant packet pieces to redundant links. We can allocate more symbols (i.e., a longer queue size) to better quality beams (see Figure 2.4 left part). Our previously invented priority Fountain codes [64] can encode the packets based on their priority levels (Figure 2.4 right part). Higher priority packets have higher redundancy (thus can be more easily recovered).

Since rateless codes is rateless, it avoids direct observation of the link quality. However, we need to deduce the link quality for main path establishment. In this work, we propose to use the decoding CDF concept to measure the link quality. The decoding CDF is defined as the probability with which the encoded packet can be recovered successfully without errors after a certain number of symbols (n) have been received [29]. Obviously, such a
cumulative distribution distribution (i.e., CDF curves) increases monotonically with the number of symbols received. The CDF curve is sensitive to encoding parameters, channel conditions, and code block length. We just need to collect a small amount of records on the relationship between \( n \) (number of symbols sent between two consecutive pauses) and \( T \) (ACK feedback delay), in order to obtain the CDF curve.

To prove the decoding CDF is able to reflect the channel quality (SNR), Figure 2.5 is given.
to show the plot of decoding CDF for different SNR values, i.e., from -5dB to 25dB. These decoding CDF curves demonstrate that fewer symbols are required to decode a particular transmitted message for higher SNR, while more symbols are needed in a channel with lower SNR (Here the channel is assumed to be Gaussian Rayleigh fading channel). As one of metrics in our RL-based path selection, the decoding CDF enables our routing system select the best path with higher SNR than others.

The advantage of decoding CDF (probability of successful recovery vs. # of symbols) is to know the proper pause time for the sender. Without using CDF, the sender has to pause each time after sending the minimum number of symbols for each packet. If the receiver still cannot re-assemble packet #1 (Figure 2.6, left part), then no ACK will be sent back. But at least a few milliseconds of time has been wasted due to the pause time between the finishing of the minimum number of symbols and the transmission of more symbols (all for packet #1).

In CDF case (Figure 2.6, right part), since the sender already has the statistical distribution of how many symbols it should send before each pause, it can choose to pause in the right time after sending the minimum packet #1 symbols plus some extra symbols for packet #1 (just in case the channel condition is poor). Because the ACK takes some time to get back to the sender side, the sender can just simply send part of packet #2 symbols before its first pause. As we can see from Figure 2.6 (right), while the ACK for packet #1 is in transmission, the sender has sent out additional symbols for both packets #1 and #2, and then pauses in the right time to wait for ACK of packet #1 (the CDF curve tells us that the ACK should arrive at that time).

A good thing is that the decoding CDF can be easily obtained through on-line learning algorithm, either Gaussian approximation or Maximum-likelihood (ML) estimation. In this work, we will take the average ACK waiting delay within a unit time (such as one second) as the link quality metric: \( \overline{\Delta} = \frac{\sum_{i}(t_{i+1} - t_{i})}{M} \), here \( t_{i} \) is the pause time instant, \( M \) is total pause times in the unit time.


### 2.3.2 Capture Effect

As long as the directional antenna is used, the capture problem is unavoidable [14]. As shown in Figure 2.7, assume node A has data for node D (in beam ①). Before A sends data to D, it detects that there is signal in beam ②. This is because B is sending data to C. A may get stuck in carrier listening status (i.e., captured by Bs traffic) based on CSMA-based MAC protocols, since it may believe that beam ① has traffic for itself. Thus A cannot send data to D since the multi-beam antenna requires that all beams are in sending or receiving status simultaneously. If A prepares receiving in beam ①, it will not allow beam ② to be in sending status. The best way is to turn off beam ① completely if A knows there will be no data for itself in the next communication phase. But typically the multi-beam antenna keeps all beams in their on-state in case there is incoming data from any direction. To build a high-quality main path in RDC-based routing, we should avoid the links with high probability of being captured by other nodes.

![Capture Effect Diagram](image)

**Figure 2.7: Capture effect**

---

19
To define the capture seriousness level for a particular node, we first define an indicator, $I_{ij}$. If it is 1, it means that node $i$ is currently participating in active communications in beam $j$. Otherwise, it is 0. Even node $i$ itself is not actively communicating with any node (i.e., $I_i = 0$), it can still be captured by other neighboring nodes. We define $U_{ij}$ as the capture probability for node $i$ captured by nodes in beam $j$. Such a value can be obtained through the history capture event statistics in different beams of node $i$. Assume there are total $M$ beams, we can define the total capture seriousness level of node $i$, $C_i$, as follows:

$$C_i = \sum_{j=0}^{M} \{ W_1 \times I_{ij} + W_2 \times U_{ij} \},$$

where $W_1$ and $W_2$ depend on the importance of productive and unproductive traffic in node $i$, and $W_1 + W_2 = 1$. Since unproductive traffic contributes most to the capture effect, in this work, we set up $W_1 = 0.3$, and $W_2 = 0.7$.

### 2.3.3 Link Bandwidth

Link bandwidth measurement [62] is straightforward since it directly reflects the capacity of a link in any beams (directions). Generally, the bandwidth measurement can be obtained by the effective working time in a certain period, which is defined as channel idle time ratio (CITR):

$$CITR = \frac{\text{idle time period}}{\text{monitoring time period}}$$ (2.1)

Then the CITR can be updated in a weighted format, depending on how we evaluate the importance of history long-term average and a recent CITR value (this is similar to Internet RTT estimation):

$$CITR \leftarrow (1 - a) \times CITR_{i-1} + a \times CITR_i$$ (2.2)

Where $CITR_{i-1}$ and $CITR_i$ denote the previous $CITR$ value and the current $CITR$ value, respectively. The coefficient $a$ is set to 0.7. A link between two beams of the sender and receiver nodes can be represented as bandwidth factor (BWF):
Here \( c \) and \( x \) illustrate two connected nodes with MBDAs.

### 2.3.4 Diamond Transmission Potential

Here we introduce another metric to measure if a path is capable of diamond transmissions, and find out how many links in this path can participate in the diamond transmissions. Recall that RDC transmission distributes the data into different beams in each ripple, and all traffic converges at the second ripple. Therefore, the cycle of a diamond transmission consists of two ripples (hops). In practice, some nodes in a chosen route probably cannot utilize diamond transmission to forward data in next two ripples due to the lack of available links (for building side paths). Therefore, only a single link is going to be used to forward data to the next ripple. We call such a link (with side paths) as the bottleneck link, which can drag down the performance of a diamond transmissions due to the use of a single beam.

As one of the link quality metrics of diamond transmissions, the number of bottle links, \( B_i \) belonging to node \( i \), is maintained.

In order to determine if a specific node is capable of launching a diamond transmission in the next two ripples, two factors should be considered. First, a node has to perceive all available routes to the destination node. This routing knowledge could be achieved in routing discover protocol such as AODV. Second, every node has to possess a neighbor table that contains the information about all the accessible two-ripple (two-hop) neighbor nodes and the beam IDs through which it can reach the two-hop neighbors. Moreover, the number of available branches is also critical since more branches mean more side paths.

Here we introduce an indicator variable, \( D_{ix} \), to measure the potential of serving diamond transmission for each node. Suppose node \( i \) and node \( x \) are a pair of two-ripple neighbors, and node \( i \) can forward data to node \( x \) via \((n + 1)\) different beams. Then \( D_{ix} = n \). If node \( i \) merely arrives at node \( x \) through a single path, \( D_{ix} = n = 0 \), which means that the diamond transmission potential for node \( i \) to node \( x \) is zero.
transmission is not available for this two-ripple diamond unit. Otherwise, the value of \( D_{ix} \) contains the number of branches between node \( i \) and node \( x \). This value can be simply read from the two-ripple neighbor table in each node.

### 2.3.5 Put everything together: FL-based Metric Integration

Now we will use FL to integrate the above 4 factors together into a single variable that models the link quality dynamics in each node. FL can be achieve real-time, low-complexity variable fusion. Particularly, the Simple Additive Weighting Method (SAW) [20], a widely used FL method, will be used to integrate the above 4 factors.

To exemplify the concept of SAW, a decision matrix is given below, where the first column stands for the factor of link bandwidth (BW). Its range is \((0, 1)\). The value approaching to 1 means that the link possesses a larger bandwidth.

\[
D = \begin{pmatrix}
B_1 & 0.8 & 237 & 2.3 & 2 \\
B_2 & 0.5 & 1104 & 1.4 & 0 \\
B_3 & 0.3 & 300 & 0.2 & 1 \\
B_4 & 0.7 & 400 & 3.8 & 3
\end{pmatrix}
\]

The second column denotes a parameter related to the decoding CDF value. This parameter \( N_{CDF} \) represents the number of symbols have been sent when the receiver has 90\% propability to successfully recover the original packet. The smaller value of \( N_{CDF} \) represents a better channel quality of a link since it means a receiver has a better chance to recover the original packet with fewer symbols sent by sender.

The third column is the metric of antenna capture possibility (CP). We set up its range as \([0, 4]\). 4 is the worst case, i.e., most likely being captured.

The fourth column represents the diamond transmission potential (DT). Higher value in this column indicates more extra links available to serve as side paths.

We can use the following two formulas to normalize the above matrix. In both (2.5) and
(2.6), each $x_{ij}$ is the entry of $B_i$ with respect to a specific metric. If the case is the larger, the better, equation (2.5) can be applied for normalization. Otherwise, equation (2.6) should be used.

$$r_{ij} = \frac{x_{ij}}{x_{ij}^{max}} \quad i = 1, \ldots, 4 \quad j = 1, \ldots, 4$$ \hspace{1cm} (2.5)$$

$$r_{ij} = \frac{x_{ij}^{min}}{x_{ij}} \quad i = 1, \ldots, 4 \quad j = 1, \ldots, 4$$ \hspace{1cm} (2.6)$$

After applying equation (2.5) or (2.6), we can obtain the normalized matrix as:

$$D' = \begin{pmatrix}
B_1 & 0.8 & 0.84 & 0.1 & 0.5 \\
B_2 & 0.5 & 0.18 & 0.14 & 0 \\
B_3 & 0.3 & 0.67 & 1 & 0.25 \\
B_4 & 0.7 & 0.5 & 0.05 & 0.75 \\
\end{pmatrix}$$ \hspace{1cm} (2.7)$$

In the following we will discuss the settings of SAW weights for the following 5 types of traffic with various QoS requirements:

- Speech and audio streaming - 100kbps, 10 audio frames per second, up to 300ms delay, this type has the highest priority;

- Interactive video (e.g., video conferencing such as skype) - 480p, 10 frames per second, 512 kbps, up to 300ms delay, this type has the second highest priority;

- Live streaming (e.g., live sports events) - 720p, 30 frames per second, 2Mbps, up to 2 s delay, this has the third highest priority;

- Video on demand (streaming of pre-encoded video such as Netflix) - 720p, 2Mbps, 5-10 s delay, the 4th highest priority (we do not consider retransmission of lost packets here)
File downloads - 50MB file in 5 minutes, it is the lowest priority, no packet drops allowed (requires retransmission of lost/dropped packets)

These data flows have distinct preferences with respect to the setup of the metrics. For instance, the HD video data requires large bandwidth and high potential of diamond transmissions, while audio and file data do not have high requirements on those two metrics. However, they may prefer a better channel quality (thus less packet loss) and a less capture probability (again, less loss) on the route. Therefore, we should assign different SAW weights to the above mentioned 4 factors for each of the 5 types of data flows.

The first type of data does not require high data transmission rate since audio traffic does not have high amount (compared to video). Thus the diamond transmission is not so urgent (less side paths are acceptable), that is,

\[ W_{\text{audio}} = [0.3, 0.3, 0.2, 0.2] \]

The second type of data (interactive video) needs more bandwidth than the first type (audio data), and more side paths are preferred to deliver more data. Thus, we can set up the weights as:

\[ W_{\text{video}} = [0.7, 0.8, 0.3, 0.5] \]

For the HD live video streaming and HD video on demand, both types of data require high bandwidth because of the huge video data amount in unit time. More side paths are needed to deliver these HD video packets to the destination on time. Therefore, the diamond transmission is important. We use:

\[ W_{\text{HDvideo}} = [0.9, 0.9, 0.4, 0.8] \]

For the fifth type of data that possesses the lowest priority, we can use:

\[ W_{\text{data}} = [0.2, 0.4, 0.7, 0.1] \]

By multiplying the above weight vector by the normalized matrix \( D \), we can obtain the Q-value of the data flow. Such Q-value will be used in the Q-learning based main path establishment (see next section).
2.4 RL-Based Main Path Establishment

In this section, we will explain the detailed process of using the above link quality metric (a FL-fused value) to seek the best path (main path) based on RL scheme. Such a main path will become the central pipe of our proposed RDC routing trajectory (Figure 2.1). Before we describe RL-based path seeking algorithm, we will first briefly review how we can simply extend general WMN routing scheme (that aims to search the shortest path) to multi-path routing scheme. Those available multiple paths will be used to search the main path via the RL algorithm.

2.4.1 Multi-path WMN Routing

Dynamic source routing (DSR) [34] is a typical multi-hop path search protocol that aims to find the shortest path. As mentioned before, the shortest path does not mean that its links have the best quality under fast fading channel conditions. Therefore, we need to extend DSR to multi-path case in order to obtain all the candidate links/paths, which will be used to search for the main path.

If we ask a MR in the WMN backbone to issue the route request (RREQ) message to all its MCs, general DSR can easily find the hop ID for each node. We call those different hop IDs as ripples (Figure 2.1). Each MC belongs to certain ripple. In general DSR scheme, all intermediate nodes discard the RREQ messages they have received before. Through some minor modifications of route discovery process in DSR we can find all the available multiple paths between the source and destination: Each node broadcasts the received RREQ messages; and each RREQ message contains the node IDs that it has traversed by. To prevent count-to-infinity problem, an intermediate node discards the RREQ messages once their addresses have already existed in the path list of the RREQ messages. After the routing discovery process, each node would receive multiple RREQ messages containing the routing path information (a list of nodes) in the WMN.

To possess sufficient information for the routing selection, after the above multi-path
DSR, each node maintains a one-hop neighbor table and two-hop neighbor table. Note that
the tables should also have the information on which beam (direction) the neighbor is located
in. The above discussed link quality metric (after applying FL) should also be put in the
table for each beam (link).

In a nutshell, after a minor revision of DSR protocol, we will get to know all the available
candidate paths from a source to a destination. Those paths and their links have various link
quality levels. We need to use a cumulative method to search the best path (i.e., with the
best overall quality for all the links of the path). Obviously, such a path does not necessarily
have the shortest hop count. Otherwise, DSR would solve the problem. Such a path will
become the main path in our RDC routing scheme (to be discussed in section 5). Next we
will introduce RL (Q-learning) based algorithm to efficiently find out the main path. RL is
a typical cumulative optimization algorithm that considers the overall reward performance
after performing link-to-link state/action update.

2.4.2 RL-Based Main Path Establishment

To achieve real-time main path search, we cannot use a model-based RL scheme due
to its complex state space search. A model-free RL approach, called Q-learning [51], just
uses simple Q-table to represent the state space. Thus it can greatly speed up the path
search. Q-learning process can be realized by modeling the routing selection as a Markov
decision process (MDP) process, in which we can find the best route to the destination node
by maximizing the expected rewards of all links. Figure 2.8 shows how we apply MDP for
WMN main path search.

Briefly to say, a MDP consists of 4-tuple \((S, A, T, R)\), in which \(S\) represents a finite set of
states; \(A\) denotes a set of actions; \(T = P_{s,s'}(a)\) stands for the transition probability from state
\(s\) to state \(s'\) when taking action \(a\) in state \(s\); and \(R_a(s, s')\) is the reward function received
when switching to state \(s'\) after applying action \(a\). Normally, a MDP problem can be solved
through 5 steps of iterations: 1) an agent (node) gets aware of the entire MDP environment
(WMN network conditions) and which state it is in. 2) Based on the current state, the agent
adopts an available action $a \in A$. 3) In the next phase, the agent transfers to the next state $s'$ and obtains a reward value from the system. 4) The agent updates the path searching policy based on the reward it just received. 5) Repeat the above 4 steps at the current state.

To adapt to the dynamic channel conditions as well as the decentralized WMN architecture, we adopt the Q-learning to efficiently find the optimal routing path (i.e., main path). Q-learning has the following 5 elements:

- **States:** Here the state of a node includes the parameters in each beam (link) such as its throughput, packet drop rate, bit error rate (BER), etc.

- **Actions:** The set of the candidate actions $a_t \in A_t$ at each state $S_t$ denote the available transmissions to the one-hop neighbors of the node. Such an action should point out which beam to use, and which neighbor to communicate next.

- **Rewards:** Here the reward is the FL-based integration of the 4 link quality metrics discussed in section 2.3.5 (i.e. Decoding CDF, link bandwidth, capture possibility and diamond transmission potential).

- **State transition:** The state transition matrix $T = P_{s,s'}(a)$ is a fixed matrix, and each state transition probability can be determined beforehand based on the empirical data obtained before.
Online learning: We will adopt Bellman principle (discussion next) to perform on-line policy search based on the cumulative rewards.

The Bellman optimality equation [63] is a mathematical optimization method to optimize the Markov decisions based on the long-term reward calculation. It is used as a utility function in this thesis:

\[
V^*(s) = \max_{a \in A} E_s \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\}
\] (2.8)

Here \(0 < \gamma < 1\) is a discount factor that confines the impact from the long-term decisions. When \(\gamma = 0\), the node makes the decision in a myopic manner, i.e., without considering the long-term optimization. In this case the node just simply forwards the data to the next node which can generate the maximum immediate reward in this hop, but not considering the total cumulative reward. On the other hand, while is approaching to 1, the agent would more emphasize the future reward, in which the RL system is more farsighted. In the above equation, \(r_t, s_t\) and \(a_t\) denote reward, state and action, respectively. \(V^*(s)\) is the utility value for taking action \(a = a_t\) at state \(s = s_t\), and then executing the optimal policy \(\pi^*\) thereafter.

The Bellman optimality equation can be denoted as action-value equation \(Q^*(s, a)\):

\[
Q^*(s_t, a_t) = E \{ r_{t+1} + \gamma V^*(s') | s_t = s, a_t = a \}
= E \left\{ r_{t+1} + \gamma \max_{a' \in A} Q^*(s', a') | s_t = s, a_t = a \right\}
= E(r_{t+1}) + \gamma \sum_{s',a'} P_{ss'}(a) \max_{a' \in A} Q^*(s', a')
\]

We can utilize the model-free Q-learning to optimize the main path selection by iteratively updating the Q-values for given links between nodes:

\[
Q(s, a) = (1 - \alpha) Q(s, a) + \alpha \left\{ r_{t+1} + \gamma \max_{a' \in A} Q(s', a') \right\}
\] (2.9)
Besides states, action $a$ and discount factor $\gamma$ explained previously, the other two parameters used in Q-learning are: (1) $\alpha$: The learning rate, set between 0 and 1. Setting it to 0 means that the Q-values are never updated, hence nothing is learned. Setting to a high value such as 0.9, means that the learning can occur quickly. (2) $\max_{a' \in A} A$: The maximum reward that is attainable in the state following the current one, i.e., the reward for taking the optimal action thereafter.

At the beginning, every node of a WMN maintains a Q-table. Once a node is about to send out a packet, it may have multiple candidate one-hop neighbors according to the available routes found during the DSR-based multi-path routing discover process (section 4.1). The node sends the packet to one of these candidates as an action $a$. After the packet is sent to the next node, the reward $r$ is collected and piggybacked by the ACK message, and the node updates the Q-value in the Q-table with the reward $r$. Then, the next node holding the packet becomes the current node of Q-learning. This process is being repeated until the packet arrives at the final destination.

### 2.5 RDC-Shaped Routing

After the main path is established via the above RL scheme, we can then add side paths to it, in order to form a diamond chain topology that can fully utilize the multi-beam capacity. In this section, we will explain how we can form a RDC routing architecture as well as the ripple schedule control scheme.

#### 2.5.1 Diamond Chain

As described in section 4.1, through the use of multi-path DSR, we are able to find all candidate paths between a source and a destination. By 'hitting two birds with one stone' in the DSR routing, each node can maintain its 1-hop and 2-hop neighbor table for side path establishment purpose. As shown in Figure 2.9, after the main path (red arrows) is formed, we can easily add side paths (red solid lines) around the main path by selecting the neighbors that belong to the same ripple (i.e., the same hop ID) as the next-hop relay nodes.
Figure 2.9: Diamond chain: Add side paths to the main path

Figure 2.10: The cases of Path generation and collision

Figure 2.10a shows the basic idea of side path generation. Suppose the main path has been decided, where S, F, and C are the main path nodes. Through DSR protocol RREQ propagation, we know that B, E, and G are in the same ripple as F. Here S could maintain those nodes profiles (such as their ripple IDs, beam directions, link quality to that node, etc.) in its 1-hop and 2-hop neighbor tables. Here suppose node S chooses nodes B and E as the relaying nodes of the side paths. As a result, node S distributes data (rateless coded symbols) to nodes B, F and E in three different beams in the same superframe time duration (to be discussed in section 5.2). During the next superframe time duration, nodes B, F and E forward packets to node C. This process is repeated every two ripples (hops). They thus form a diamond chain routing architecture. Each node in the main path serves as either the source of the divergence links (such as node S) or the destination of the convergence links (such as node C).
Since we adopt the rateless codes in this diamond chain (see section 3.1), the packet pieces (called symbols) go through the main path and side paths. Based on the 1-hop neighbor table, we know the link quality in each beam (direction). While we should allocate the most number of symbols to the main path direction (here it is S→F), we can assign certain number of symbols that is proportional to the link quality in each side path link (for example, S→B and S→E). The link quality here is an integrated FL-based metric, as discussed in section 3.

**Path collision issue:** During the above diamond chain formation, some particular issues have to be considered here. One of them is path collision problem. As shown in Figure 2.10b, suppose the main path is known, S is the divergence node, and D is the convergence node. S needs to find side paths among all of its 1-hop neighbors. Suppose S establishes two side paths to B and C. Here S sends data to both B and C. And in the next phase (superframe time) both B and C are supposed to send data to D.

If B and C are in the same beam coverage of the convergence node D, path collision issue comes up. This is because D cannot simultaneously talk with two (or more) nodes in the same beam. To avoid this issue, we ask any two main path nodes that are two hops away (here it is S and D) to exchange their 1-hop neighbor tables. In such a table, the main path node should be aware of the beam ID of each neighbor. Here the main path node D can simply use multi-beam antenna DOA (direction of arrival) detection schemes [31] [5] to find out that B and C are located in the same beam ID. By exchanging its neighbor table with S, S can avoid the selection of both B and C as side path relays (just select one of them).

**Handle bottleneck links:** Each diamond ‘chain’ of RDC route consists of nodes in a 2-hop range, including the first hop for data divergence and the second hop for data convergence. However, a bottleneck link can exist when the number of hops from the sender to receiver is an odd number, as shown on Figure 2.11 (left part). Since node R receives more data than what it sends out in any time, a bottleneck of the data flow pipeline is generated, which is the link R→D.
If there are multiple links available to the destination node, a solution is to simply use disjoint paths in the last 3 hops, as shown in Figure 2.11 (right). After the data is diverged in first hop, the relay nodes may choose not to converge. Here S chooses A and B as relay nodes of side paths as usual. A and B know that D is two hops away. They will try to find other relay nodes (instead of R) in order to make the data converge to D. Through the one-hop and two-hop neighbor tables, they could easily find the proper relay nodes to achieve disjoint path transmissions.

Another solution is change the allocation of ripple-to-ripple airtime (packet transmission duration). For example, as shown in Figure 2.11 (left), while A and B send the data to R during one time interval, R can be allocated a longer time interval after it gets the "token" to forward the data to the next ripple (Section 5.3 will discuss ripple-to-ripple pipelined transmission control).

![Figure 2.11: Different situations of RDC Routing: (left) bottleneck link RD; (right) our solution.](image)

Algorithm 1 and 2 shows the essential process of our RDC-based routing scheme.

### 2.5.2 In Each Ripple: Multi-Beam MAC

In each ripple we need to deal with the localized, one-hop neighborhood communications. Although it is typically controlled by MAC protocols, we need special transmission control strategy since conventional 802.11 protocols cannot be applied in multi-beam antennas (they typically assume omni-directional antennas). For completeness, we provide our MBDA-oriented MAC scheme here. Note that all nodes in each ripple can either all be in receiving or in sending status due to the properties of MBDA and our RDC routing scheme. We
**Algorithm 1** RDC-based routing scheme

1: Input: Unique ID of the node "i", current time slot "t"
2: Output: Chosen star node and $Tx/Rx$ mode of the node /* chose the star node I, select mode and launch ripple process if necessary*/
3: $i \leftarrow patlen$
4: **while** $k \in$ Maximum number of nodes in WMN **do**
5: node $i = hash(k \oplus t)$;
6: **if** node $i$ possesses high priority data flow to forward **then**
7: Transmission mode of node $i = Tx$
8: **if** RDC is required **then**
9: Node $i$ launches RDC process and forwards tokens
10: **else**
11: Node $i$ transfers to Rx mode
12: **end if**
13: **end if**
14: **end while**

**Algorithm 2** RDC processing

1: Input: The basic information of the data flow from application layer
2: **if** RDC is required **then**
3: Find all available paths to Destination
4: Choose the best path according to the Q-value
5: Determine the length of the time-slot and other information
6: Send out the tokens to decided beams and broadcast SCH frames to the other neighbors
7: According to these control frames, corresponding nodes start forward data
8: **else**
9: Proceed the regular two-layer MAC protocol
10: **end if**
thus define a single round of sending or receiving time duration in a particular ripple as a superframe. As shown in Figure 2.12, to adapt to the multi-beam QoS-aware communication requirements, we propose to use enhanced point coordination function (PCF) and enhanced distributed coordination function in each superframe. General PCF and DCF have been defined in 802.11 standards.

Figure 2.12: Multi-beam MAC frame architecture

**Enhanced PCF operations:** The node in the main path can serve as a point coordinator (PC) in PCF mode. Each PCF operation includes 3 phases (Figure 2.12): The PC uses QoS query phase to ask each node (in side paths) in the next ripple to feedback its flow QoS parameters. The collision resolution phase is then used to solve the collisions during QoS response phase since each beam of the node may have multiple nodes sending back QoS responses. The polled data phase is used for official data transmission from the PC to each node of each beam.

**Enhanced DCF operations:** Figure 2.13 shows an example of multi-beam DCF operations. Note that C and A are in the same beam. The node (in the main path) can only talk with one of them (based on the traffic priority; for example, VBR has higher priority than ABR). Here the node uses (RTS-CTS-DATA-ACK) to communicate with A first. C needs to wait for DNAV (directional network allocation vector) based on the time specified in A’s CTS. The enhancement of conventional DCF is that we adjust the backoff timer to guarantee concurrent multi-beam transmissions since this is a basic requirement for MBDAs. Specifically, we do not use random backoff time after each DIFS as what conventional 802.11 DCF does. Instead, we remove CW-based backoff after DIFS for beam synchronized communications. But we require each node to wait for the CW-based random backoff before (not after) DIFS duration. And
Table 2.1: DCF Beam Table

<table>
<thead>
<tr>
<th>Beam ID</th>
<th>Active nodes</th>
<th>Traffic type</th>
<th>Airtime</th>
<th>bi-directional data</th>
<th>Antenna capture probability</th>
<th>Rateless CDF factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D</td>
<td>CBR</td>
<td>2ms</td>
<td>ACKs</td>
<td>0.7</td>
<td>101μs</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>VBR</td>
<td>2.3ms</td>
<td>Reverse video flow</td>
<td>0.2</td>
<td>60μs</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>ABR</td>
<td>1ms</td>
<td>No reverse data</td>
<td>0.8</td>
<td>91μs</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

all beams should wait for the same time durations if not receiving CTS.

Each main path node, no matter it is the sender of divergence paths or the receiver of the convergence paths, it should maintain a table, called DCF Beam Table (see Table 2.1), and updates such a table after sending RTS in any beam. The main path node needs to know what nodes are actively communicating with itself in each beam.

2.5.3 Ripple Schedule Control

From the above discussions, we can see each ripple is in either diffusion (sending) or aggregation (receiving) status. Obviously, there is a need to control ripple-to-ripple data propagation since two neighboring ripples cannot both be in the same status. Inspired by the real water ripple propagation pattern (with alternate crests and troughs), (see Figure 2.14), we will describe below our ripple transmission schedule control scheme [14]. Since we are looking into local communication behaviors, we again discuss about how to enhance the general MAC scheme.

Besides traditional IEEE 802.11 control messages (DATA, NULL, RTS, CTS and ACK),
we introduce another two new messages, called RTS with Intelligent Feedback (RIF) and CTS with Intelligent Feedback (CIF). They are employed as ”tokens” in the ripple protocol. Moreover, another control frame SCH is also introduced to arrange beam transmissions other than the ones negotiated via RIF/CIF.

![Ripple-to-ripple propagation pattern](image)

Figure 2.14: Ripple-to-ripple propagation pattern

Figure 2.15 exemplifies the schedule control of the ripple transmissions. The use of Rateless code and decoding CDF enable the use of Block Acknowledgement scheme (specified in 802.11e) in this thesis. In this example, Node S holds the token in the beginning. It forwards the data flow to next-hop nodes R1 and R2 through RDC path. Meanwhile, node S is also capable of other data forwarding operations via other beams. As shown in the Figure 2.15, node S is transmitting the RTS message to node R3, while it is passing token message RIF to R1 and R2. All the behaviors of data exchanges through the beam to R3 are controlled by the schedule of data exchange in RDC scheme. Beside the beams to R1, R2 and R3, if other beams do not have any data to forward to other neighbors, node S sends out a SCH message to inform these nodes to defer from accessing the medium.

Each time cycle of the ripple process is from the beginning of a DIFS to the next DIFS. The above figure shows an ideal situation, in which nodes R1 and R2 successfully receive all the data from node S. If R1 and R2 cannot successfully recover all the data, S is going to forward the rateless symbols in the next round. This process will repeat until the R1 and R2 received all the data from S successfully. Then nodes R1 and R2 have the token. In the
next cycle, both of them will forward data to the node in the next hop.

![Timeline of the ripple schedule control scheme](image)

**Figure 2.15:** Timeline of the ripple schedule control scheme

### 2.6 Performance Analysis

#### 2.6.1 DSR routing overhead

Our RDC routing is based on the multi-beam, multi-path extension of the basic DSR routing. Let’s analyze the protocol overhead of the multi-path DSR scheme. Later on we will analyze the extra overhead brought by our multi-beam extensions.

In the multi-path DSR scheme, suppose node S intends to discover multiple paths to node D. To find multiple paths, an intermediate node (say node M) does not discard all the duplicated RREQ messages from source node S to destination node D as long as these RREQ messages have passed through different intermediate nodes. If node M has the path information to node D, it directly sends back the duplicated RREP messages through all the paths with the path information to node D. If node M does not know paths to the node D, it will broadcast a RREQ to find all the available paths from itself to node D and then feedbacks the path information to node S.

Note that the basic single-path DSR makes an intermediate node discard all the repeated
RREQ messages between the source node and destination node. The destination node only
sends back one RREP message through the path with the shortest distance.

We introduce a model here to compare the overhead between two DSR protocols. Suppose
we have a network consisting of $N$ nodes, and each node possesses $L$ links to its one-hop
neighbors. In each second, each node launches $\beta$ times of route discover requests in average.
For each route discovery request, the total number of forwarded RREQ and RREP messages
is $M$. Therefore, the overhead of route establishment for the network is: $O(N) = \beta \times N \times M$.
Moreover, the parameter $M$ can be divided into two parts: the number of RREQ messages
$M_{RREQ}$, and the number of RREP messages $M_{RREP}$. For each route discovery request, both
basic single-path DSR and modified multi-path DSR broadcast the RREQ messages to each
link once. The $M_{RREQ}$ can be expressed as: $M_{RREQ} = \frac{N \times L}{2}$. On the other hand, the number
of $M_{RREP}$ can be estimated as: $M_{RREP} = \frac{N \times L}{2} \times \delta$, where $\delta$ stands for a ratio between
$M_{RREQ}$ and total number of links of the network $(\frac{N \times L}{2})$. Hence, the overhead of DSR has
the following property:

$$O(N^2)_{single-path DSR} = \beta \times N \times \frac{N \times L}{2} \times (1 + \delta)$$

(2.10)

According to [42], most of the overhead in a traditional reactive routing protocol is
generated by RREQ messages. The value of $\delta$ can be estimated as tiny value (close to 0).
The multi-path DSR sends back RREQ messages through a number of selected paths from
the destination to the source. Therefore, the overhead of multi-path DSR can be estimated
as:

$$O(N^2)_{multi-path DSR} = \beta \times N \times \frac{N \times L}{2} \times (1 + C\delta)$$

(2.11)

where $C$ stands for the number of paths that the destination uses to send the RREP messages
back to the source. Since $\delta$ is a very small number that is close to 0, and $C$ is normally a one-
digit constant, the overhead of multi-path DSR is very close to the overhead of single-path
2.6.2 Overhead of RL-based path optimization

In our RL-based path searching scheme, the overhead between nodes are incurred by updating the Q-table. As shown in equation (2.9), the current node requires a reward $r$ and $\max_{a' \in A} Q(s', a')$ for Q-table updating. The reward $r$ is obtained through four metrics, while the term $\max_{a' \in A} Q(s', a')$ is attained in the state of the node in the next hop. As a result, the update of Q-table incurs some overhead due to the demand of some necessary information from next state.

To obtain the reward $r$, a node first has to collect four metrics and then combines them through Fuzzy Logic algorithm locally. In those four metrics, the decoding CDF is based on the recent history of number of symbols required for successful decoding at the receiver, in which the node is able to learn this value from the received ACKs. Similarly, the Diamond Transmission Potential also can be read locally. Alternatively, the node has to get the state information from the next-hop node to figure out the value of Capture Effect and Link Bandwidth. The Capture Effect measures the captured possibility of the next-hop node and Link Bandwidth requires the CITR value of next-hop node as well.

Based on the above descriptions, totally three values have to be delivered from the next-hop node to the current node in order to update the Q-table of the current node. These three values of next-hop node are Capture possibility, CITR, and Q-value of next-hop node.

We proposed to transmit such information through ACK piggybacks in order to minimize the overhead generated by Q-table update. According to Q-learning scheme, the Q-table can be updated once before a node makes an action and forwards a packet. The node updating the Q-table via ACK sent from receivers, can conform the process of Q-learning. Suppose each value is represented in half-precision floating format. Then total 6 bytes of overhead are incurred when the Q-table needs to be updated. Since we employ Block ACK and aggregate packets strategy, the 6 bytes of overhead can be further reduced.
2.6.3 Data link Layer overhead analysis

Since our RDC routing protocol is integrated with ripple-to-ripple MAC protocol, we need to estimate the total control overhead in Data Link layer. Unlike CTS and RTS messages used in 802.11, RTF and CTF messages are used as a token in the RDC scheme of this paper. The size of CTS and RTS is 20 Bytes each, while RTF and CTF are in the size of 22 bytes [32].

As shown in figure 2.15, we use $T_{cycle}$ to represent the time consumed by one cycle of ripple process. $T_{cycle} = t_{pkts} + t_{overhead}$, in which $t_{pkts}$ stands for the time used to forward payloads and $t_{overhead}$ means the time used for control messages. Suppose the data rate is $K$ Mbps, $t_{pkts} = \frac{(l \cdot n)}{K}$, where $l$ and $n$ are the packet length and quantity, receptively. On the other hand, $t_{overhead}$ is a sum of the time overhead for DIFS, SIFS and other control messages, such as CIF, ACK and Preamble & Header:

$$t_{overhead} = DIFS + t_{RIF/RTS} + 3 \times SIFS + t_{CIF/CTS} + t_{ACKs} + t_{pqh}$$

(2.12)

Therefore, the valid throughput fora single beam in one RDC cycle is:

$$\text{Throughput}_{RDC} = \frac{l \cdot n}{T_{cycle}} = \frac{(l \cdot n)}{(DIFS + t_{RIF/SCH} + 3 \times SIFS + t_{CIF/SCH} + t_{ACKs} + t_{pqh} + t_{pkts})}$$

(2.13)

Suppose the block ACK scheme is also used in 802.11 DCF, the throughput of 802.11 DCF is:
Throughput_{DCF} = \frac{l \times n}{T_{cycle}} = (l \times n)/(DIFS + t_{RTS} + 3 \times SIFS + t_{CTS} + t_{ACKs} + t_{p&h} + t_{pkts}) \quad (2.14)

Here the difference of $T_{cycle}$ between RDC ripple and DCF is that the length of RIF, CIF and ACK is larger than that of 802.11 DCF, and RDC ripple sends out 10 bytes more data than 802.11 DCF due to the overhead of ripple-to-ripple coordination in each cycle. Use a concrete MAC protocol as an example: for 802.11b protocol with 11Mbps bandwidth, suppose one frame with the size of 1500 bytes is being transmitted. The total transmission time of DCF is [26]:

$$T_{cycle-DCF} = t_{overhead-DCF} + t_{pkts-DCF} = 1534\mu s \quad (2.15)$$

Without considering the benefits of MBDAs and aggregated ACKs, the RDC ripple MAC merely costs $0.91\mu s$ more time than 802.11 DCF due to the 10 bytes message overhead. Since more than $1500\mu s$ is used for the transmission of each 1500-byte MAC frame, $0.91\mu s$ of extra communication overhead from our RDC (in MAC layer) can be ignored.

Note that the use of Block Acknowledgement (aggregated ACKs) can further decrease 10 bytes of overhead of RDC ripple process. As mentioned before, since multiple packets or ACKs are aggregated together in each cycle, the 10 bytes overhead of RDC does not happen in every packet transmission.

In this section, we can see that the overhead of RDC ripple routing scheme is negligible. Without such little extra protocol overhead, RDC ripple scheme brings us the benefits of MBDAs and enables QoS. RDC ripple and MBDAs enable multiple branches to forward data flows simultaneously. Assuming one node is able to forward a data flow with $n$ beams, the throughput of the data flow is $n$ times (ideally) more than traditional single-path DSR.
routing scheme.

### 2.6.4 Computational Complexity Analysis

The main computation load comes from the Reinforcement Learning (RL) algorithm used in main path search. Other protocol components do not involve much calculation. Among RL algorithms, Q-learning is widely used in online system due to its simple computational complexity [2]. Our algorithm includes the reward collection of four metrics, fuzzy logic and Q-value update.

Algorithm 3 presents the framework of Q-value updating process in RL-based main path establishment. Once a node forwards data to the next-hop neighbors and receives corresponding ACKs, the 4 metrics (Decoding CDF, Capture effect, Link bandwidth and Diamond transmission potential) are observed and combined via Fuzzy logic. Then the corresponding Q-value is updated.

**Algorithm 3** Q-value updating process in RDC

1: **Input:** Q-learning algorithm, length of each path $L$, discount factor $\gamma$, learning rate $\alpha$, Exploration probability $\epsilon$

2: Initialize $Q(s, a)$ arbitrary

3: **while** True **do**

4: Choose action $a$ between Explore and Exploit via $\epsilon$-greedy policy

5: Forward data and collect 4 metrics

6: Combine 4 metrics via Fuzzy logic and figure out the reward $r$

7: Update Q-value:

8: $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha \left\{ r_{t+1} + \gamma \max_{a' \in A} Q(s', a') \right\}$

9: $s \leftarrow s'$

10: **end while**

The RL-based main path establishment process is based on discrete action space. Between each pair of source node and destination node, the action space $A$ can be demonstrated as $A_{SD} = \{a_1, \ldots, a_n\}$. The maximization of Q-value update can be obtained via enumeration over the discrete action space $A$, which has a linear complexity $O(n)$ [2]. Note that before the RL-based main path establishment process, the DSR has already discovered candidate paths from Source node to Destination node. This significantly minimizes the size of action
space $A$.

In each step of the Q-value update, 4 metrics have to be collected and combined via Fuzzy logic (see Line 5 and 6 in Algorithm 3). With respect to the 4 metrics, the computational complexity of decoding CDF is bounded by $O(1)$ (constant) according to the algorithm in [29]. This result is same as both metrics of Link Bandwidth and Diamond transmission potential, while the complexity of Capture Effect is related to the number of beams $M$ of each node, $O(M)$.

Regarding the metrics combination via Fuzzy logic, the computational complexity is decided by the number of actions (beams) that a node can use to reach the destination. Suppose the available beams to destination is $B$, the complexity of Fuzzy logic process should be $O(B)$. Due to the limited number of beams equipped on the wireless node in reality, $M$ is a small integer and $B < M$.

In the Q-value update process of enumeration over the discrete action space $A$, the total complexity is $O(n(B+M))$. Note that $B$ is smaller than $M$, and $M$ is a small integer due to the limited number of beams equipped in the wireless node, $B+M$ can be seen as a constant. Therefore, to search the path between a source and a destination via RL and Fuzzy logic, the computational complexity can be estimated as a linear value $O(n)$.

The Q-tables are maintained in each wireless node. To minimize the delay in each relay node, it is critical to lower the computational complexity generated by Q-learning process in each node. According to the Algorithm 3, the computational complexity is $O(B+M) \approx O(1)$ for each Q-value update step. Thus, the complexity of RL process for each node in each update is very low. It involves merely a low, linear computation complexity.

2.7 Simulation Results

The performance of our RDC-based multi-beam routing protocols will be evaluated in this section, including the proposed DSR the multi-beam ripple-to-ripple MAC protocol, diamond transmission, and the RL-based paths selection. In the simulations, we assume
that every node in the WMN is equipped with MBDAs with 6 beams for data receiving and forwarding. Each beam has 2.5 Mbps of capacity. Thus 6 beams gives total 15 Mbps maximum node capacity. Each beam covers 55 degrees of angle, and there is 5 degrees of gap between any two neighboring beams.

![Figure 2.16: WMN Simulation Topology](image)

Our WMN simulation topology is shown in Figure 2.16. A node acts as a MR connecting to the wireless backbone, when the others are all MCs. They form a tree topology with the MR as the tree root. The tree levels are like ripples since all nodes in the same ripple have the same number of hops to the MR. In the following experiments, the time slot duration of 10ms is used, as recommended by IEEE 802.22 standard [6].

2.7.1 The Ripple-to-Ripple Transmission Performance

As discussed in section 5, our RDC routing consists of a series of localized diamond chain transmissions. Especially, in each ripple we run a special multi-beam MAC protocol with enhanced PCF and DCF modes. Such a localized transmission control is important since the whole routing path consists of those pipelined diamond chain transmissions. We thus evaluate the communication performance in each ripple first, especially the packet loss rate,
Figure 2.17: Performance comparison between our multi-beam single-ripple communications with enhanced PCF/DCF, without PCF enhancement (DCF only), and conventional 802.11 protocol delay, and throughput metrics.

We implemented and compared 3 types of localized transmission protocols in each ripple: (1) MB-PCF+DCF: our proposed multi-beam MAC with enhanced PCF and DCF; (2) MB-DCF: multi-beam MAC without enhanced PCF (DCF only); and (3) OM-802.11: conventional 802.11 protocol (use omni-directional antennas; without multi-beam enhancement) in each ripple. In Figure 2.17a, the x-axis is the average data generation rate of each node. The y-axis stands for the average throughput of each node in the WMN. As we can see, our multi-beam MAC has a higher throughput than other two schemes. The conventional 802.11 protocol has the lowest throughput. This is mainly because that 802.11 protocols were designed for a wireless network with omni-directional antennas. Thus it is not suitable to WMNs with MBDAs. From Figure 2.17b we can see that our scheme has the lowest packet loss rate. This is because the multi-beam links can select the proper number of symbols to send based on the link quality in each beam. A higher quality beam can transmit more symbols. Thus we can reduce packet drop events by pushing more data to better beams. Figure 2.17c shows the delay performance. Again, our scheme has the lowest ripple transmission delay due to the multi-beam concurrent communication capability.

2.7.2 QoS performance

In this set of experiments, we evaluate the localized ripple communication performance in terms of data flows with different QoS priorities. Here we assume all those 3 priorities
Figure 2.18: QoS Performance in Ripple Communications

Figure 2.19: Performance comparison between our diamond-based routing scheme, conventional PCF-based routing, and ripple-only routing protocols

of flows are all video data. The packet size is 1500 bytes. But they have different QoS requirements: Priority #1 can only tolerate maximum 200ms of end-to-end delay; Priority #2 tolerates up to 300ms; Priority #3 is 550ms. Figure 2.18 illustrates the performance of delivering packets with different priorities.

Again, we compare the above mentioned 3 protocols. It is apparent that our protocol delivers more packets than other two protocols. Furthermore, while the traffic load is increasing, our protocol apparently still guarantees that the data flow with higher priority has better performance. Figures 2.18b and 2.18c also give similar results in terms of delay and loss rate. We have used circles to mark out the results from our scheme.

2.7.3 Diamond Transmission Performance

After evaluating the localized, single-ripple, multi-beam communications, here we investigate the entire diamond chain based routing performance. Especially, we compare our
RDC-based scheme with the following two routing variants: (1) PCF scheme: it refers to the conventional DSR routing scheme (no diamond chain formation) based on original 802.11 PCF mode (thus it does not have enhanced PCF with multi-beam transmissions in each ripple). (2) Ripple scheme: here it refers to the use of multi-beam transmission control in each ripple. However, it does not form the diamond chain in the whole routing path. Instead, it just randomly pick up multiple, disjoint paths to deliver the data (via revised, multi-path DSR routing, described in section 4.1). Thus it has less nodes using multi-beam capabilities.

As shown in Figure 2.19, first, Figure 2.19a shows that our diamond-based routing has higher successfully arrived packet amount (per second). Especially when the node sending rate is higher than 4 Mbps, our scheme is obviously superior to PCF. When the node traffic is more than 6 Mbps, our scheme is much better than PCF, and also higher than simple ripple routing scheme. This is because that multi-beam transmissions have more advantage in higher traffic load since more links in each node are especially helpful for heavy load.
Figure 2.22: Effects of using AI algorithms

Figure 2.19b shows that our diamond routing has lower end-to-end routing delay than both schemes. When the node traffic load is higher, our scheme has more advantage.

Figure 2.19c) shows that packet loss rate has better performance than PCF when the node traffic is higher than 4 Mbps. After 6 Mbps, our scheme is better than both of them. This is not surprising since our diamond-chain routing fully explores all beams for transmission, and the amount of traffic in each beam is proportional to its link quality in that beam direction. Thus, we can minimize the packet loss events.

### 2.7.4 Learning-Enhanced Routing Performance

Then we focus on the performance enhancement after using learning-based algorithms, especially RL-based main path routing establishment scheme. Here we still assume the tree-like WMN topology as shown in Figure 2.16. As discussed in section 4, with the use of Q-learning we are able to find a main path with the best overall link quality under fast fading environment. In this experiment the discount rate ($\gamma$) of our Q-learning system is set to be 0.5. Each MBDA has 6 beams.

The comparison of the ripple-based routing performance with and without Q-learning is shown in Figure 2.20. (a) **Routing delay**: the end-to-end delay performance is shown in (a). When the total traffic load in the node is high (>6Mbps), the learning-based routing shows more advantage and has obviously shorter delay. This is because Q-learning can accurately find the best path as the main path, which is especially suitable to heavy load
communications. Here again 15 Mbps refers to the peak traffic amount in a node that may aggregate all traffic from its 6 beams. Thus each beam has average traffic load of 2.5 Mbps, which is typical in multimedia applications. (b) **Packet loss rate:** We can see that Q-learning selects the best quality links to construct the whole main path, thus it has the lower packet loss rate (i.e., higher reliability) than non-learning case. We can see that under all traffic loads our RL-based scheme has lower loss rate. (c) **Throughput:** here we again use the resident packet amount (i.e., queue length) to measure the node throughput performance. As we can see, Q-learning has shorter queue length (especially when the sending data rate is high). This indicates that each node can more quickly clear out its queued packets. This is obvious since Q-learning finds the best path that can avoid frequent node capture, and thus can better dispatch all buffered packets to multiple beams.

### 2.7.5 Video Transmission Effects

We use H.264 encoded video transmission to test the efficiency of our RDC-based routing scheme. To achieve this experiment, we use empty video packets in the simulations. In other words, in our simulations we use some packets without real video data in the payload. In the sender side, those empty packets are sent in different beams of the MBDA. If any of those packets is lost due to long delay (passed the video replay deadline) or due to network congestion, in the receiver side we record those lost packets. Then finally we get back to original sent video packets and take those lost packets away. The video is then recovered via the remaining, successfully received video packets. We can then take some frames of the video and check its image quality (see Figures 2.21 and 2.22).

The video frame resolution is 800x600 pixels; bit rate is 850kbps; frame per second = 60. Figure 2.21 shows the received video effect (taking one frame as an example) based on the following 3 WMN multi-hop transmission schemes: (a) conventional DSR-based routing protocol, without using the multi-beam routing (only selects the shortest path), also without using multi-beam oriented MAC in each hop (instead, general 802.11 protocol is used). (b) still using common DSR-based routing, but using our proposed multi-beam MAC protocol
in each ripple-to-ripple communication. (c) our proposed RDC routing plus the use of multi-beam MAC in each ripple.

As we can see, Figure 2.21c has the best video resolution because it uses RDC routing to maximize the throughput of the multi-hop routing. Each beam helps to forward part of the rateless coded video packets (symbols). In each ripple, our multi-beam MAC protocol uses enhanced PCF and DCF to explore multi-beam delivery capability and also support the video QoS metrics. Figure 2.21b used RDC routing. However, each ripple still uses conventional 802.11 protocol, which assumes omni-directional antennas, and thus only explores one beam for communications. Figure 2.21a has the worst quality due to its ignorance of multi-beam antennas in both routing and MAC layers.

In order to show the advantage of our learning-based main path selection, a HD video with a higher resolution (1024x576) is transmitted in each ripple. The bit rate is 1.57Mbps, fps = 60. Figure 2.22 is the comparison result of 3 cases. Here case (a) is DSR-based routing, no diamond chain formation, no learning algorithms used during the routing path search. Case (b) adopts diamond architecture to build multiple paths. However, it does not have learning-based main path establishment strategy (only uses simple shortest-path routing protocol, like DSR). Case (c) uses our Q-learning based algorithm to search the main path, and form diamond chain routing topology.

Figure 2.22 clearly shows the advantage of using learning-based scheme to build the main path. Such a path guarantees the best overall link quality in the whole diamond chain routing pipe. The worst case, i.e. (a), has many packet loss events and causes the poorest video quality. This is because it does not have learning-based routing search strategy and also diamond-chain architecture is not used. Thus it uses single-beam for communications in most nodes. Such a strategy cannot meet the HD video delivery demands.
2.8 Concluding Marks

In this chapter, we have presented an innovative multi-beam routing protocol based on RDC formation. It can fully explore the high capacity of multi-beam antennas through the diamond chain routing architecture. The RDC-based routing consists of a main path and multiple side paths. We have applied rateless codes to such a multi-beam transmission in order to adapt to the fast fading links with dynamic channel conditions. We further used AI algorithms (FL and RL) to build the main path that consists of the best quality links from a statistical distribution perspective. The ripple-to-ripple schedule control is designed to achieve pipelined transmissions in different ripples. Our simulations with real-time video transmissions have validated our RDC routing efficiency.
3 A TRAFFIC ENGINEERING WITH PREDICTION-BASED LINK UNCERTAINTY SOLUTION FOR WMNS IN SDN STRUCTURE

3.1 Introduction

Wireless Mesh networks (WMNs) have drawn many attentions in the past decade [35]. A WMN has some powerful wireless nodes, i.e., Mesh routers (MRs), which aim to build the backbone of a WMN and provide peer-to-peer connectivity to end users in an ad-hoc fashion. Although MRs are able to freely form various network topologies to adapt to varying radio conditions, the WMNs still have many performance limits such as imbalanced bandwidth allocations, difficulties of performing centralized network management for many distributed nodes, etc.

On the other hand, to enable an easy network management and monitoring, the concept of Software-Defined Networking (SDN) has been proposed. A network-wide central controller in SDNs is a firm support for the large, complex network management. The SDN-based network structure has been widely studied in wired networks, in which SDN enables a centralized control by separating the control plane from the data plane. Due to its advantages a few elementary studies have expanded the platforms of SDNs from wired networks to wireless scenarios. The SDN is a promising solution to overcome the drawbacks of current WMNs through its agile protocol reconfiguration and optimized resource allocation.

As such, we propose a SDN-based mechanism to improve the performance of WMNs in terms of dynamic routing management under mobility-caused link failures and its associated traffic allocation issues. To adapt to the dynamic WMN topology, we perform a centralized machine learning in the control plane for the prediction of unexpected link failure in WMNs.
The proposed work is called PLUS-SW (Prediction-based Link Uncertainty Solution in SD-WMN). Its contribution includes three aspects as follows:

(1) *Dynamic traffic management*: We propose a network-wide traffic management based on the centralized control of SDNs. The traffic balancing and radio resource allocation are periodically re-evaluated by the SDN controller according to the real-time traffic demands and the network topology dynamics. By using this approach, the performance of the WMN can be improved by a global, optimal traffic monitoring and management.

(2) *Intelligent link failure prediction*: SD-WMN needs to handle the unexpected link failure due to node mobility. Conventional distributed WMN management has difficulties to accurately predict the link failure. We employ the supervised learning methods to predict unexpected link failures. This prediction system has a two-layer structure to fully explore the advantage of the SDN: In the lower layer, a supervised learning based classification model is used in the SDN switches; In the higher layer, the central controller uses a global learning model to enhance the accuracy of the link failure prediction in the entire WMN.

(3) *Optimal back-up path determination*: Once a link failure is confirmed by the two-layer prediction model, the SDN controller immediately calculates the alternative path and notifies the related nodes to bypass the link that will fail soon. The calculation of the back-up path is based on the pattern of current traffic. The control overhead and traffic congestion caused by the path switching are minimized.

### 3.2 Background and Motivation

In SDN the control plane comprises of one or multiple centralized controllers to manage the entire network. The data plane contains underlying routers and switches, which serve as the data forwarding devices. The control plane manages the entire network by writing or updating the forwarding rules in the flow tables of each switch. Correspondingly, a SDN-based switch handles all the ingress packets according to the flow tables stored in its memory. Such a structure allows a network-wide observation and control of the network traffic, as well
as a simplified hardware/software structure for the switches in the data plane.

A WMN consists of Mesh Routers (MRs), Mesh Clients (MCs), and Gateways. The MRs constitute the backbone of WMNs, while a gateway enables the Internet connectivity. MCs stand for end users of a WMN. A SDN-based WMN (SD-WMN) [31] also has the separation of control plane and data plane. The basic structure of SD-WMN is shown in Figure 3.1. In the SD-WMN, a dedicated central controller is used for the data forwarding control by writing forwarding rules in the flow tables residing in each switch. This central controller could be a gateway or a MR of the WMN. Each MC is thus able to get rid of the complicated self-operating module, and turns into a flow-table-based switch for packet forwarding. The main advantages of SD-WMN include the simplified network management, low-cost MC architecture, and flexibility of vendor control, etc. [5] [15].

The motivation of this study is to overcome the shortcomings of a typical SDN when it is applied to WMN traffic management. First, in current SDN structure, it does not have a robust wireless link control among SDN switches when handling the imbalanced traffic flows in WMNs. Second, it does not have a flexible rerouting function that can minimize the rerouting impact on the active data flows by using optimal traffic allocations. Third, while we target the accurate fine-grained network configuration under uncertain radio conditions of the WMN, it is also important to minimize the communication overhead generated by the control traffic.
Due to the round trip time between the controller and switches, a SD-WMN may not be able to deal with a random wireless link failure or node malfunction rapidly. To solve this issue, the most straightforward solution is to enable a localized path selection in a SDN-based switch and handle an uncertain link failure in a distributed manner. Nevertheless, an unforeseen change of the local rerouting behaviors tends to incur many packet loss events, since a SDN switch cannot timely and accurately find a proper back-up path. Therefore, we introduce a supervised learning mechanism to predict the uncertain link behaviors of a SD-WMN. After a prediction of the link failure or the node malfunction is made, a SDN controller can generate a globally optimal rerouting plan and sends it to the relevant switches to establish a back-up routing path.

In order to establish a fine-grained link failure prediction (LFP) for the SD-WMN, we introduce a two-layer prediction paradigm. The basic structure of two-layer LFP system is shown in Figure 3.2. Since SDN assumes dummy forwarding devices in the data plane, it is improper to add complicated modules in a SDN router. Thus, a pre-trained link prediction model (via supervised learning) is used in the SDN switches to accommodate for the simple structure of switches. Such a model can be calculated by a central controller in advance and pre-installed in the switches. Due to the myopic operations of SDN switches, such a model will only be responsible for the local link failure prediction with one-hop neighboring nodes. Once a potential link failure is detected, the switch will immediately feedback a warning request to the central controller, which will make global traffic allocations.

Besides the above switch-triggered lower-layer LFP, there is an upper-layer LFP located at the central controller, which is able to determine if any node is going to fail by sampling the parameters of the outgoing links of this node. For example, it will check whether a moving node $M$ tends to incur abnormal amplitude fluctuations for its outgoing links. Once a neighbor of $M$ predicts this uncommon situation and forwards a warning message to the central controller, the controller will monitor all the links of $M$. Based on the link quality pattern analysis, the controller can determine if node $M$ is going to fail in the near future.
If a failure is confirmed by the controller, a corresponding back-up routing path will be established.

### 3.3 Related Works

As the preliminary studies of SD-WMNs, only a few basic prototypes have been built so far. Those SD-WMN prototypes can be classified into two main categories (based on the pattern of transmissions in control / data traffic [53]), i.e. in-band and out-of-band. The out-of-band structure is consistent with the policy of conventional SDNs used for wired network, in which the packets of control traffic are transmitted separately with the data traffic through an independent channel. Dely et al. [15] proposed the SD-WMN prototype with out-of-band style. The separation of control / data traffic is realized in [50] by assigning different Service Set IDentiﬁers (SSIDs) to the control / data traffic. Due to the limited resources in WMNs, an in-band based SD-WMN structure, i.e., control / data traffic is transmitted in the same channel, have been realized as well. Both Detti et al. [15] and Yang et al. [31] have designed
a SD-WMN platform with in-band fashion. Particularly, Detti et al. [15] employed OLSR as the routing protocol for both control and data traffic, while Yang et al. [31] established an in-band Openflow-based WMN platform for traffic balancing.

All these works aim to establish a platform for SDN-based WMN rather than propose a novel control policy to overcome the shortcomings of SD-WMNs. We believe that the centralized control offered by SDN provides an opportunity for a more intelligent networking management, by utilizing machine learning schemes. Especially, a SDN can be very helpful to deal with the management difficulties caused by the dynamic WMN topology.

Regarding the wireless link quality estimation (LQE), some mechanisms have been proposed, such as PRR [13], ETX [18], filtering (WMEWMA [61], KLE [54], fout-bit [23]), Regression (WRE [65]) and Supervised learning [56]. Two schemes have focused on the link failure prediction issues: in [41] it achieved the failure prediction via data mining algorithm, and in [36] a cross-layer design is proposed. For those works, LQE is achieved by the estimation of the link qualities instead of link failure events. Namely, they are not capable of accurately predicting the unexpected link failures. Their accuracy limit is due to a distributed, localized network management. The estimation of the link status depends on the performance metrics of the link (such as signal-to-noise ratio). Unnecessary network operations (such as rerouting, channel switching, etc.) can be triggered by a mistaken observation of the network. In this work, the central controller will be used to improve the precision of LFP in a global management fashion.

### 3.4 Proposed Construction Model of SD-WMNs

#### 3.4.1 Physical layer model

In the physical layer, various access/antenna technologies have been proposed to improve the bandwidth utilization, such as CDMA, OFDMA, MIMO, multi-beam directional antennas (MBDAs), etc. As in many WMN studies, we assume a multi-channel multi-radio (MC-MR) feature in the links. As an example, in 802.11a standard, there are 12 non-overlapping
channels in the 5G Hz range.

### 3.4.2 Data link layer model

In a *wired* SDN, the control plane mainly performs the traffic engineering, and the traffic control commands are distributed to different routers/switches based on a global coordination plan of the control plane. Other than the wired SDN, a SD-WMN not only requires a global control of traffic engineering, a link access scheduling scheme to overcome the wireless link interference is also critical to minimize the link bit error rate (BER). As a result, a corresponding SD-WMN MAC layer is required.

![Figure 3.3: Proposed SD-WMN MAC layer superframe structure](image)

Figure 3.3 shows the structure of the MAC layer for the proposed SD-WMN. It uses a hybrid scheduling. The time domain is divided into multiple time periods called superframes. Since a centralized link scheduling requires the synchronization of all the SDN switches, a short time slot is reserved to orchestrate the clocks of all the SDN switches. After a DCF Interframe Space (DIFS), the switches start to transmit data according to the link scheduling (calculated by the central controller in advance). Note that our link scheduling policy is adaptive to the link interference conditions in the dynamic network topology, and the scheduling calculation is based on the traffic demands and routing status, as well as the priorities of different data flows. More details about the link scheduling will be explained in Section V.

A centralized scheduling efficiently provides a global management to reduce link interference. Considering the link uncertainty and potential access conflicts, a CSMA-based schedule control is also used in the MAC protocol for link collision avoidance. Further-
more, the CSMA backoff policy is used for unexpected traffic collisions which may not be
timely captured by the SDN controller. Thus CSMA-based distributed coordination in the
local range can compensate for the difficulty of performing timely regulations by the remote
controller.

3.4.3 Routing Protocol and Traffic Demand

Although the SDN is promising in terms of traffic balancing, the existing studies on
SD-WMN still employ traditional routing protocols (e.g., AODV, DSR, OSLR, etc.) for the
management of the network topology and the route discovery [31] [15]. OLSR has strong
capability of overcoming the impacts of node mobility. Once a SDN controller fetches the in-
formation of the SD-WMN topology and the traffic demands, it is able to distribute the data
flows of each node to different links for optimized traffic engineering. While OLSR is capable
of performing path discovery, a controller still needs to optimize the traffic management in
a global manner.

3.4.4 Mathematical model of the SD-WMN

In a network $G = (V, E)$, $V$ represents the set of all vertices of a WMN, while $E$ stands
for all links. Let $TD_v$ denote the traffic demand of flows generated in node $v$ (note that
the flows relayed by node $v$ are not counted in $TD_v$). Since the transmissions via different
channels are not influenced by each other, we first consider the data flows in single channel.
Let $TD_v = \{f_{P}^{D:1}, f_{P}^{D:2}, f_{P}^{D:3}, ..., f_{P}^{D:n}\}$ denote traffic demands of the flows from node $v$
with different priorities. Given a traffic distribution $\Delta_v$, we can obtain the outgoing flows’
demands in different paths of node $v$ by using the following model:

$$\{f_{A_1}^{\rho_1}, f_{A_2}^{\rho_2}, ..., f_{A_m}^{\rho_m}\} = \Delta_v \times TD_v \quad \forall v \in V \quad (3.1)$$

in which $f_{A}^{\rho}$ implies an aggregated flow $A$ that is sent out from node $v$ along with path $\rho$.
It is a sum of multiple outgoing flows through the path $\rho$. Hence, path $\rho$ has a set of edges.
$f_{A}^{\rho}$ can be represented as:
\[ f_A^\rho = \left\{ \alpha_A^{f(ID;1)} f_{P=\text{in}}^{ID;1}, \alpha_A^{f(ID;2)} f_{P=\text{in}}^{ID;2}, \ldots, \alpha_A^{f(ID;n)} f_{P=\text{in}}^{ID;n} \right\} \]  

(3.2)

\( \alpha_A^{f(ID;i)} \) stands for the ratio of outgoing flow \( f_{P=\text{in}}^{ID;i} \) of node \( v \) assigned on the aggregated flow \( f_A^\rho \) along with path \( \rho \). It is obvious that the aggregated flows on each link must be less than the capacity of the link. Let \( c(e) \) represent the capacity of link \( e \) in time slot \( \tau \), we have:

\[
\sum_{\rho \in f(e)} f^\rho \leq c(e) \quad \forall e \in E
\]

(3.3)

The sum of all aggregated flows that go through edge \( e \) in time slot \( \tau \) has to be smaller than the capacity of edge \( e \).

3.4.5 Transmission Collision Model

The conflict model of wireless links denotes the wireless interference among a neighborhood. It can be categorized into protocol interference model (PrIM) and physical interference model (PhIM) [10]. The conflict model in this paper is based on PrIM [25], which is defined as follows: assuming node \( n_i \) and node \( n_j \) are one-hop neighbors with each other, and a data transmission is in progress between \( n_i \) and \( n_j \) in period via the radio channel \( ch_i \). We say that the transmission is successful as long as all the neighbors of \( n_i \) and \( n_j \) that may interfere with link \( e \), will keep silent in \( ch_i \) during the entire time slot. This concept is compatible with 802.11 MAC layer protocol (CSMA/CA). In the case that the nodes are equipped with directional antennas, the interfered nodes should be in the coverage of the transmission beam of \( n_i \) and reception beam of \( n_j \). If an interfered node is equipped with MBDAs, then only those beams that can reach \( n_i \) or \( n_j \) must keep silent.

The network model with radio conflict is defined as follows [48]: Given a network model \( G = (V, E) \), in which \( V \) denotes the set of all switches and \( E \) stands for all the edges. For each \( e \in E \), there will be a set of edges which can disturb the wireless communications on the edge \( e \), when they are transmitting data on the same channel. Here this set of edges is represented as \( I(e) \). It is obvious that all the edges in \( I(e) \) must keep silent in channel \( i \).
while edge $e$ is active on channel $i$. Given $f_{e}^{ch(i)}$ as the expected throughput in edge $e$ during time slot $\tau$, recall that $c_{ch(i)}(e)$ is the capacity of edge $e$ in channel $I$ during time slot $\tau$. Then, the constraint can be represented by the equation below:

$$\frac{f_{e}^{ch(i)}}{c_{ch(i)}(e)} + \sum_{e' \in I(e)} \frac{f_{e'}^{ch(i)}}{c_{ch(i)}(e')} \leq 1 \quad \forall e \in E$$

(3.4)

While edge $e$ is idle, multiple edges in $I(e)$ could be activated. Hence we have:

$$\frac{f_{e}^{ch(i)}}{c_{ch(i)}(e)} + \sum_{e' \in I(e)} \frac{f_{e'}^{ch(i)}}{c_{ch(i)}(e')} \leq C_{I(e)} \quad \forall e \in E, i \in K$$

(3.5)

The value of constant $C_{I(e)}$ is determined by the network topology and the link interference with each other. Thus we can calculate it when we know the link interference matrix of network $G$.

3.4.6 Flow smoothness

In order to prevent each node from queue overflow (i.e., traffic congestion), the input and output data amount for a relay node has to be equal in period $T$. When a flow $f$ goes through a relay node $v$, node $v$ may use different channels to receive and send flow $f$. Hence, we expand the traffic analysis from a single channel $k$ to all the channels. Let $F_{v;in}^{ch;k}$ and $F_{v;out}^{ch;k}$ denote the amount of ingress and egress flows of node $v$, respectively. We have:

$$F_{v;in}^{ch;k} = \sum_{e \in E_{in}(v)} f_{e}^{ch;k}, \quad F_{v;out}^{ch;k} = \sum_{e \in E_{out}(v)} f_{e}^{ch;k}$$

(3.6)

Assuming node $v$ possesses $K$ channels for data transmissions. Thus we have:

$$\sum_{k \in K} F_{v;out}^{ch;k} = \sum_{k \in K} F_{v;in}^{ch;k}$$

$$\forall v \in V, \{F_{v;out} \cup F_{v;in}\} \nsubseteq \{f(v, \sim), f(\sim, v)\}$$

(3.7)

Because a node $v$ may be a source or destination of data flows, the amount of total
ingress and egress flows should not contain the flows whose source or destination is node $v$. Here $f(v, \sim)$ and $f(\sim, v)$ represent those two kinds of flows, respectively. Thus, the amount of ingress and egress data of each relay node has to be equal when we ignore $f(v, \sim)$ and $f(\sim, v)$.

After listing all the constraints above, we can now maximize the throughput of the entire network. The most straightforward method is to maximize the amount of flows transmitted in the network as long as they obey the constraints defined above. Furthermore, we also add weight factors $\omega$ to each of flows according to their priorities in order to meet their QoS requirements.

$$\begin{align*}
\text{Max} : & \sum \omega_P \times \alpha_A^{f(ID;i)} f^{ID;i} \\
\text{Subject to:} & \\
& \{f^{p_1}_{A_1}, f^{p_2}_{A_2}, ..., f^{p_n}_{A_n}\} = \Delta_v \times TD_v \quad \forall v \in V \\
& \sum_{p \in f(e)} f^p \leq c(e) \quad \forall e \in E \\
& \sum_{k \in K} F^{ch:k}_{w:out} = \sum_{k \in K} F^{ch:k}_{v:in} \quad \forall v \in V, \{F_{w:out} \cup F_{v:in}\} \not\subset \{f(v, \sim), f(\sim, v)\} \\
& \frac{f^{ch(i)}_{e:ch:i}(e)}{c^{ch(i)}_{e:ch:i}(e)} + \sum_{e' \in I(e)} \frac{f^{ch(i)}_{e':ch:i}(e')}{c^{ch(i)}_{e':ch:i}(e')} \leq C_{I(e)} \quad \forall e \in E, i \in K
\end{align*}$$

### 3.5 Link scheduling for collision prevention

#### 3.5.1 Out-of-band transmissions of control/data traffic

Control messages of SDNs play a significant role in the network management since the data plane becomes so dummy and only the control panel issues the transmission rules. To guarantee the transmission quality of the control traffic, we assign a secure path between central controller and each switch of the data plane. Thus we adopt a out-of-band style [53] for the SD-WMN structure, which means that the bandwidth of the control traffic cannot be used for the data traffic. In this way, an control channel is independent of the traffic in the data plane. Comparing with the link scheduling of the data traffic, the control traffic
forwarding system has a relatively stationary architecture. As long as the control traffic is transmitted efficiently, the path between controller and each switch does not need to change. This invariance is also applied to the channel assignment on the links of these paths. Note that only a single path is assigned to the link between a SDN controller and a switch. In other words, a switch exchanges data with a central controller through a stationary and independent path.

Except for the resources occupied by the control traffic, all the other available channels are dynamically allocated for data plane traffic. The following section explains the scheduling strategy with respect to the channels for data traffic.

3.5.2 Single Channel scheduling

Our scheduling scheme aims to reduce the occurrences of traffic transmission bottlenecks by periodically re-allocating the radio resources in the entire network. Assume that the central controller regularly manages the entire network in every period $T$. That is, in each $T$, a central controller arranges the transmission of wireless links in multiple super-frames. Each super-frame contains a series of time slots, and the packets are forwarded in the links with the pre-scheduled channels during each time slot without interference with each other.

Let $FD(e_i)$ denote the data traffic on edge $e_i$ including both types of flows, i.e., generated or relayed by the sender of edge $e_i$. Since the traffic assignment $\Delta$ can be performed according to traffic demand $TD$, it is straightforward to obtain $FD(e_i)$ over the entire WMN. The set of flows in node $v$ upon edge $e_i$ is denoted by $PRI(e_i)$. Since both QoS and fairness of data forwarding have to be considered in link scheduling policy, we introduce a weight factor $\omega_f(\tau)$ to measure the comprehensive urgency level of forwarding a flow in time slot $\tau$. Specifically, the weight factor $\omega_f$ is related to both the priority of flows and waiting time $\gamma_f$ in the queue of a node. For each flow, we have:

$$\omega_f = pri_f + k \times \gamma_f$$

$$pri_f \in PRI$$

(3.8)
Where, \( pri_f \) is the priority of flow \( f \), and \( k \) is a constant coefficient to measure how much the waiting time \( \gamma_f \) contributes to the urgency of the flow. Let \( X_\tau \) denote the expected transmission rate of the flows scheduled in time slot \( \tau \) on edge \( e_i \), and \( W_{X_\tau(e_i)} \) represents the set of corresponding weight factors \( \omega_f \) for the flows in \( X_\tau \). Thus, the objective of the link scheduling in each time slot \( \tau \) can be formulated as follows:

\[
Maximize : \sum_{e_i \in E} W_{X_\tau(e_i)} \cdot X_\tau(e_i) \quad (3.9)
\]

In order to reflect the dynamic channel conditions in link scheduling process, we add a constraint with respect to the link quality. We assume that the channel conditions can be predicted and collected by a central controller. To simplify the model of channel conditions in link scheduling, we employ the signal-to-noise ratio (SNR) as the metric. Assume a vector \( G = \{g(e_1), g(e_2), ..., g(e_i), ...\} \) represents the gain of each link. The data transmission power is denoted by a vector \( P = \{p(e_1), p(e_2), ..., p(e_i), ...\} \). Then, the SNR can be calculated as:

\[
SNR_v(P(e_i)) = \frac{g_v(e_i)p_v(e_i)}{n_v} \quad (3.10)
\]

in which \( n_v \) stands for both background noise and the signal strength from other nodes at \( v \). Let the vector \( R = \{r^1, r^2, ..., r^m, ..., r^M\} \) denote a series of transmission rates. If node \( v \) expects to achieve a transmission rate \( r^m \) through link \( e_i \) \( (X_\tau(e_i) = r^m) \) for any wireless modulation scheme, the \( SNR_v(P(e_i)) \) must be higher than a certain level, such as \( SNR_v(P(e_i)) \geq H(r^m) \), where \( H \) is a function to calculate the minimum required SNR in terms of transmission rate \( r^m \). Then, we can deduce the minimum power \( p_{\min}(e_i) \) to achieve the transmission rate \( r^m \) while keeping the packet loss rate (PLR) in a low level.

\[
p_{\min}(e_i) = SNR^{-1}(H(r^m)) = SNR^{-1}(H(X_\tau(e_i))) \quad (3.11)
\]

Then, a constraint (via the linear programming model) in link scheduling is given by:
Here $P_{\text{max}}$ stands for the maximum transmission power threshold. This constraint is used to restrict the flow amount for a low link quality.

Furthermore, all the activated links in time slot $\tau$ must satisfy the link interference constraint as well:

$$\frac{f_{e}^{ch(i)}}{c_{ch}(e)} + \sum_{e' \in I(e)} \frac{f_{e'}^{ch(i)}}{c_{ch}(e')} \leq 1 \quad \forall e \in E_{\tau}, e' \in E_{\tau}, e \neq e'$$  

(3.13)

Where $E_{\tau}$ is the set of edges activated in time slot $\tau$. This constraint is the sufficient condition of wireless link interference. Hence, the linear programming model of link scheduling in single channel $i$ is as below:

Max: $\sum_{e_i \in E} W_{X_\tau(e_i)} \cdot X_\tau(e_i)$

Subject to:

$$SNR^{-1}(H(X_\tau(e_i))) \leq P_{\text{max}} \quad \forall e_i \in E$$

$$\frac{f_{e}^{ch(i)}}{c_{ch}(e)} + \sum_{e' \in I(e)} \frac{f_{e'}^{ch(i)}}{c_{ch}(e')} \leq 1 \quad \forall e \in E_{\tau}, e' \in E_{\tau}, e \neq e'$$

(3.13)

Algorithm 4 shows the data forwarding scheduling policy for a single channel case.

**Algorithm 4** Single channel scheduling

1: Input: Traffic Demand of edge $e_i$ with Priority label: $FD(e_i)$
2: Initialize: Schedule $S_T(e_i^\tau)$, Time slot $\tau = 0$,
3: while $T \geq \tau$ do
4: Solve the Linear programming problem of link scheduling during time slot $\tau$ on a single channel $i$
5: Set the $S_T(e_i^\tau)$ with the link scheduling results
6: Upgrade $FD(e_i)$
7: $\tau = \tau + \tau'$
8: end while
3.5.3 Multi-channel scheduling

Algorithm 4 can be easily extended to multi-channel case. First of all, the link scheduling policy must fully utilize all the channels. Given \( K \) available channels in the transmissions, we have:

\[
\text{Max:} \sum_{\sigma=1}^{K} \sum_{e_i \in E} W_{X_{\tau}}^{ch(\sigma)} \cdot X_{\tau}^{ch(\sigma)}(e_i)
\]  

(3.14)

Regarding the transmission power constraint, each of the active links (no matter which channel is used) in time slot \( \tau \) must be smaller than the maximum allowable power level. That is,

\[
\beta_{\tau}^{ch(\sigma)}(e_i)X_{\tau}^{ch(\sigma)}(e_i) \leq P_{\text{max}} \quad \forall e_i \in E, \sigma = 1, ..., K
\]  

(3.15)

Similarly, the constraint of link interference can be represented as:

\[
\frac{f_{e_i}^{\text{ch}(\sigma)}}{c_{\text{ch};i}(e)} + \sum_{e' \in I(e)} \frac{f_{e_i'}^{\text{ch}(\sigma)}}{c_{\text{ch};i}(e')} \leq 1
\]  

\[
\forall e \in E_\tau, e' \in E_\tau, e \neq e', \sigma = 1, ..., K
\]  

(3.16)

The edge scheduling strategy can vary based on the features of the physical layer. Particularly, OFDMA cannot receive and send data through different channels at the same time, while MR-MC and MBDAs are able to receive or send data simultaneously via different channels (or beams). We simplify the data plane operations into two categories: first, in each time slot, the node only operates in receiving or sending mode, which is suitable for OFDMA-like techniques. Second, a node is capable of receiving and sending data at the same time (due to MR-MC features). In the case of MBDAs, it increases the throughput via multiple directional beams for space reuse purpose. Assuming the link interference with various physical layer techniques (e.g. MBDAs, MC-MR, OFDMA, ETC.) can be reflected
in the link interference matrix \( I(e_i) \), we propose to use Algorithm 2 to achieve the link scheduling in multi-channel circumstances.

### Algorithm 5 Multi-channel scheduling

1: Input: Traffic Demand of edge \( e_i \) with Priority label: \( FD(e_i) \) \( \forall e_i \in E \)
2: Initialize: Schedule \( S_T(e_i^T) \), Time slot \( \tau = 0 \), Available channels \( K_{e_i} \) of edge \( e_i \) \( (e_i \in E) \)
3: while \( T \geq \tau \) do
4: Calculate the multi-channel link scheduling problem by using equations (3.14)(3.15)(3.16)
5: Traversal entire network, find the traffic demand \( FD(e_i) \) with the highest priority and corresponding edge \( e \). If the edges (channels) of the traffic demand \( FD(e_i) \) are all occupied, move to next top-priority traffic demand.
6: while \( K_{e_i} \) is not empty do
7: Assign the Traffic Demand \( FD(e_i) \) to available channels (High priority first)
8: Set the \( S_T(e_i^T) \) with the link scheduling results
9: Upgrade \( FD(e_i) \)
10: end while
11: \( \tau = \tau + \tau' \)
12: end while

### 3.6 Training Model for link failure prediction

Once a link failure occurs, a SDN router sends a request message to a SDN controller to update the flow tables. This is because the SDN model assumes that the dummy routers are not capable of solving an unexpected traffic collision issues by themselves. When a controller receivers a request from the data plane, it calculates the solution from a global viewpoint and sends new flow table rules to the relevant routers.

When a controller intends to reroute a data flow caused by an access collision or link failure, it will update the flow tables in the routers/switches belonging to the path of the data flow. A controller can thus indirectly forward control messages to all the routers on the path via the flow table rule changes. Note that all the flow tables of routers belonging to the same path have to be updated simultaneously to prevent inconsistency among the flow table rules.

However, a SD-WMN can still suffer from unexpected link failures because a SDN-based
control may be not able to handle the link failure problem fast enough due to the flow table update delay. Hence, we propose a link failure prediction (LFP) strategy, in the sense that a back-up routing solution can be made by the controller according to the LFP result, and the back-up link IDs can be saved in the relevant routers in advance. In this approach, a router is able to immediately respond to a link failure event. Thus the waiting time of receiving commands from the controller, could be significantly reduced.

In WMNs, a number of issues could cause packet loss or data corruptions, such as signal blocking from a metal object, link noise, node capture issue from directional antennas, traffic congestion, node mobility, etc. A WMN may be able to tolerate packet loss temporarily, but a permanent link failure has to be fixed as soon as possible. The IEEE 802.11 DCF assures a link failure when the RTS-CTS handshake cannot be established after a sender sends a neighbor RTS attempts for 7 times [7]. Nevertheless, such a simple link failure detection has two main drawbacks. First, a link failure decision highly depends on the frequency of RTS attempts. Even a receiver is unavailable in terms of one of its neighboring senders in a short time due to some reasons mentioned above, a false link failure may still happen due to the failure of a certain number of RTS attempts. Second, there would be certain response delay when a router starts to make decisions after 7 times of RTS attempts. Hence, a link failure prediction is necessary in a WMN, especially under the node mobility.

At present, some works have studied the link failure detection issues in WMNs. Most of them mainly concentrate on the node mobility problem. A number of metrics and mechanisms are used to deal with the problem, including packet reception ratio (PRR), ETX [18], Receiving Signal Strength Indicator (RSSI) [55], SNR [3], etc.

Since SDN assumes that the routers in data plane have a dummy structure, the link failure prediction (LFP) strategy in a SDN-based wireless router cannot adopt a complicated strategy. And regarding the feasibility and performance of a LFP in SD-WMN, three aspects have to be considered:

- **Simplicity**: The LFP strategy cannot use complex algorithms in the data plane.
• **Accuracy**: The LFP has to be accurate. Otherwise the control plane will generate much unnecessary control traffic due to incorrect routing control.

• **Timeliness**: A link failure must be confirmed in a fast manner to leave enough time for back-up routing establishment before the link failure happens.

In traditional WMNs, the link failure detection is part of the routing discovery process, such as link-state routing protocol [44]. The routing protocol actively broadcasts HELLO messages to neighboring nodes from time to time. The neighboring nodes can then establish the connectivity with each other after receiving these HELLO messages. On the contrary, if a node cannot receive the HELLO messages, then the link between them is determined as being broken. In WMNs the packet loss could happen frequently because many factors are able to significantly interfere with a wireless transmission. There are some drawbacks in the link-state routing protocol in WMNs. First, the active neighboring detection causes much overhead. Second, a detection delay can be very long since the link failure detection highly depends on the forwarding frequency of HELLO messages.

Unlike the link-state routing protocol, some cross-layer detection schemes are proposed [33] [40] [52] [33]. Specifically, these approaches utilize the acknowledgement in MAC layer to detect the failure. The overhead caused by HELLO messages thus can be waived due to the passive link failure detection. Nevertheless, these approaches require cross-layer design, where a link failure in MAC-layer will have impacts on the behaviors in the routing layer. The inconsistency between the routing and MAC in terms of link detection results may also decrease the transmission reliability of the network. Moreover, using acknowledgement alone in MAC layer is inappropriate in certain cases, since some other factors may cause the consecutive loss of ACKs. In [40] [52] [33] they assume a fixed data rate in physical layer, which is not applicable to general WMNs. In [41] it gives a data mining based LFP scheme. Nevertheless, it is merely based on traditional distributed WMN system (thus involving much control overhead). And it does not consider node malfunction/mobility.
The proposed LFP particularly utilizes the advantages of SD-WMNs, and is based on
the packet loss or corruption statistics during data transmissions. A data transmission error
can be categorized into two types: transient errors or permanent link failure. A transient
error stands for a temporary fluctuation of the link quality. It does not need the change
of the routing rules in switches. On the contrary, a permanent failure means that the link
does not work anymore. As a result, this issue requires an update of flow tables to reroute
the corresponding data flows. Such an update has to be operated by a SDN controller. The
objective of the proposed LFP is to rule out the cases of transient errors and detect the
permanent link failure in advance. This process relies on the observation that a permanent
link failure can cause transmission errors with a particular pattern. In order to improve the
accuracy and reliability of the prediction, a two-layer LFP structure is proposed in this work.

3.6.1 Link quality measurement

The SNR is employed as the measure of link quality. According to 802.11 standard [30],
the measurements of receiving signal strength (RSSI) and background noise are typically
observed by the firmware and drivers. Regarding the monitoring period $T_m$, the setup of its
value is a trade-off between the prediction accuracy and delay. A longer SNR monitoring
period could collect more data and thus improve the accuracy of the prediction. However,
the node can take more time to generate the prediction result.

The measurement of SNR fluctuates, and hardly reflects the distance between the sender
and receiver due to the noise from the air. To overcome this issue, Kalman Filtering [60] can
be used to smooth out the fluctuations of the measured SNR. The goal of Kalman Filtering
is to obtain a relatively precise state through a number of inaccurate measurements observed
over noises. Let $x_k$ stand for the state in time $k$, we have:

$$x_k = Ax_{k-1} + Buk - 1 + w_{k-1} \quad (3.17)$$

Then, the observed value $Z_k$ could be obtained by the following equation:
\[ z_k = H x_k + v_k \] (3.18)

In the above equations, \( w_k \) and \( v_k \) stand for two independent noises. \( A \) is a \( n \times n \) matrix containing the correlation information between consecutive data points. The control term \( u_k \) is determined by the matrix \( B \). While \( H \) is the coefficient matrix to define the relations between the observed value \( z_k \) and the state \( x_k \).

Since the state \( x_k \) cannot be directly observed, Kalman Filtering evaluates the state \( x_k \) by using a priori state estimate, observed value and posteriori state estimate. The iteration of state estimation of Kalman Filtering includes two steps (Figure 3.4). First, given state estimate \( \hat{x}_{k-1} \) and priori estimate error covariance \( P_{k-1} \) as the inputs of the system, priori state estimate \( \hat{x}_k \) and priori estimate error co-variance \( P_k^- \) at time \( k \) can be obtained as:

\[ \hat{x}_k = A \hat{x}_{k-1} + B u_{k-1} \] (3.19)

\[ P_k^- = AP_{k-1}A + Q \] (3.20)
In Step 2, the matrix $K$, a gain factor, can be obtained as:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$  \hspace{1cm} (3.21)

Meanwhile, the state estimation $\hat{x}_k$ and estimate error covariance $P_k^-$ are updated by the following equations:

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - \hat{x}_k^-)$$  \hspace{1cm} (3.22)

$$p_k = (I - K_k H) P_k^-$$  \hspace{1cm} (3.23)

Figure 3.5 shows the result obtained via Kalman Filtering. The blue spots denote the observed SNRs, while the red curve shows the filtered SNRs via Kalman Filtering. Due to the noise the filtered SNR curve is still oscillating to a certain extent. However, the filtered SNR has a linear relationship with sender-receiver distance (see the green dash line in Figure 3.5). Thus, according to the distance (that is related to SNR), we can detect the moving
tendency of a node. It is further feasible that we can predict the link failure caused by node mobility.

3.6.2 Prediction process in the data plane

After the noise is filtered from the observed SNR via Kalman Filtering, the next step is to extract the observed SNRs in the link history and make a comparison between the observed data and the training data in each step. If the pattern of observed data is similar to the training data labeled as link failure, a failure will likely happen in this link. In order to complete this task, a normalized cross-correlation [38] is employed to find the pattern similarity between the observed and training data. Assuming the length of observed SNR is \( l \) and the delay is \( d \), the normalized cross-correlation \( \gamma(d) \) can be computed as:

\[
\gamma(d) = \frac{\sum_{i=1}^{l} [q(i) - \bar{q}] [t_{a,b}(d + i) - \bar{t}_{a,b}]}{\sqrt{\sum_{i=1}^{l} [q(i) - \bar{q}]^2} \sqrt{\sum_{i=1}^{l} [t_{a,b}(d + i) - \bar{t}_{a,b}]^2}} \tag{3.24}
\]

The idea in this equation is straightforward: we can predict the link quality by calculating the normalized cross-correlation between the observed SNR and training data [21]. Specifically, \( q(i) \) stands for the observed SNR value, and \( t_{a,b}(d + i) \) represents the corresponding SNR in the training data for the link between the sender and receiver.

3.6.3 Link failure prediction in the control plane

The above is low-layer (in data plane) prediction, which only considers local node mobility. To improve the accuracy and exploit the global control of the SDNs, the control plane possesses a link prediction model as well. As long as a node mobility is detected in the data plane, a warning message will be sent and received by the SDN controller. Then the controller will pull out the SNR information from the neighbors of the suspect node to confirm its mobility status. Unlike the LFP in the data plane, the controller has multiple sets of SNR data and can thus make an accurate, comprehensive decision. Considering that this is a classification problem with multiple high-dimensional inputs, the Support Vector Machine (SVM) method [24] is employed. SVM is a supervised learning based classification method.
that has been used in many pattern recognition applications. In the following discussions
the procedure of the SVM-based LFP training in the controller is described.

Given a set of training data points \( T = \{x_i, y_i\}_{i=1}^{\text{len}} \), \( x_i \in \mathbb{R}^N \) represents the \( i \)-th input
feature vector, and \( y_i \in \{-1, 1\} \) is the vector label of \( x_i \). The SVM aims to classify the
set of observed data into two classes via the optimal hyperplane \( \omega \phi(x) + b = 0 \), where \( \omega \)
is a weight vector, \( b \) is a bias and \( \phi(x) \) stands for the mapping from input \( x_i \) space to a
high-dimensional space. In order to find the maximum margin hyperplane to separate the
data points with positive and negative labels, we have to decide the weight vector \( \omega \) and
bias \( b \). This problem can be solved by minimizing the following function:

\[
R = \frac{1}{2} \|\omega\|^2 + C \frac{1}{\text{len}} \sum_{i}^{\text{len}} |y_i - f(x_i)|_\varepsilon
\]  

(3.25)

Here the norm of \( \omega \), denoted as \( \|\omega\| \), is used to constrain the model complexity for better
efficiency. \( C \) is a factor for the trade-off between training errors and flatness. Equation (3.25)
can be transformed to Equation (3.6.3) by setting \( \varepsilon \)-intensive zone and further introducing
the positive slack variables \( \xi_i \) and \( \xi_i^* \):

Minimize:

\[
R = \frac{1}{2} \|\omega\|^2 + C \frac{1}{\text{len}} \sum_{i}^{\text{len}} |\xi_i - \xi_i^*|
\]

Subject to:

\[
\begin{align*}
\omega \phi(x_i) + b - y_i & \leq \varepsilon + \xi_i^* \quad i = 1, 2, ..., \text{len} \\
y_i - \omega \phi(x_i) - b & \leq \varepsilon + \xi_i \quad i = 1, 2, ..., \text{len} \\
\xi_i, \xi_i^* & \geq 0 \quad i = 1, 2, ..., \text{len}
\end{align*}
\]

Here \( \xi_i \) and \( \xi_i^* \) are respectively the upper and lower training errors from the \( \varepsilon \)-intensive tube.
After introducing Lagrange multipliers \( \alpha_i \) and kernel function \( k \), Equation (3.6.3) can be
expressed as:

\[
f(x, \alpha_i, \alpha_i^*) = \sum_{i=1}^{\text{len}} (\alpha_i - \alpha_i^*)K(x, x_i) + b
\]  

(3.26)
Then, the linear programming problem can be transformed to a dual QP problem that can be solved by using the Gaussian radial basis function kernel (see equation (3.27) below).

\[ K(x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right) \]  

(3.27)

Maximize:

\[ R(\alpha_i, \alpha_i^*) = -\frac{1}{2} \sum_{i,j=1}^{\text{len}} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) \]

\[ -\varepsilon \sum_{i=1}^{\text{len}} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{\text{len}} y_i(\alpha_i - \alpha_i^*) \]

Subject to:

\[ \sum_{i=1}^{\text{len}} (\alpha_i - \alpha_i^*) = 0 \]

\[ 0 \leq \alpha_i \leq C, i = 1, 2, ..., \text{len} \]

\[ 0 \leq \alpha_i^* \leq C, i = 1, 2, ..., \text{len} \]

In the above QP problem, the constraint can be adjusted by changing the factors \( C \) and \( \varepsilon \). As a necessary and sufficient condition, Karush-Kuhn-Tucker (KKT) theorem is applied to the QP problem [37]. As a result, we can get rid of most of the coefficients when \((\alpha_i - \alpha_i^*)\) has to differ from zero in equation 3.26. In this way, support vector can be determined in the corresponding training data [11]. Meanwhile, the bias factor \( b \) can be determined as well.

After the SVM module finishes training, the controller will trigger the SVM classification algorithm once the link failure warning message is received from the data plane. Specifically, the controller will pull out the very recent SNR data related to the target node with failed links. This information can be either directly collected from the target node or its neighbors. Since the locations of the neighbors are distinct, the SVM of central controller aims to detect the moving status of the target node. More concisely, the SVM is used to confirm if the target node is keeping moving away from this area in a fixed direction. Besides the SNR data from the node that sent the warning message, we can collect more sets of SNR data from different neighbors of the target node.

As long as a link failure is confirmed, the controller will immediately assign a new back-
up path (to be discussed in the next section) to the affected nodes in order to bypass the moving-away node. On the other hand, there is a chance that the controller cannot pull out enough sets of SNR data due to the random network topology and traffic. As a result, the controller will directly send a back-up route to the affected nodes. Then, the affected nodes will launch the back-up paths when they confirm the link failure by using the traditional method such as IEEE 802.11.

3.7 Back-up route determination after a link failure

The proposed traffic management (section 3.4) and link scheduling (section 3.5) methods are based on periodic network behavior control, and the central controller calculates the traffic allocation and scheduling in each cycle. By following the same periodic control style, a back-up route determination will be made to handle a predicted link failure.

The data forwarding on each link is scheduled by the SDN controller in each control cycle. But a link failure can also trigger a set of other control behaviors, such as link re-scheduling and flow re-routing. According to the global view of the central controller, the alternative paths for the link failure can be easily obtained. Thus, the back-up route seeking is actually a task of finding the best path among the alternative paths. Basically, the best alternative path has to satisfy the following conditions: (1) the shortest path distance; (2) the lowest control overhead; and (3) the least impact on other regularly scheduled data traffic.

Due to the link error accumulations in multi-hop transmissions (i.e., each time the data passes through one more hop, the probability of losing packets gets higher), the distance (the number of hops) of an alternative path has to be as small as possible. The SDN controller has to send the notification message to the nodes on the alternative path, when the information of the rerouted data flow is not cached in the flow tables of those nodes. In other words, when a SDN router receives a new type of data flow that it did not handle before, it has to request the corresponding forwarding rules in its flow tables, and such rules should be sent from the SDN controller. This will generate certain control traffic. Hence, an entirely
new alternative path (i.e., without any overlap links with the previous path) will incur more control overhead. Similarly, an alternative path also tends to affect other originally scheduled data flows.

Figure 3.6 gives an example of alternative paths. Assuming path A is the original path with an unexpected link failure, there are two alternative paths available (path B and path C). Path B merely needs to update the flow tables for two nodes, and most nodes of the original path A remain in path B. On the contrary, path C needs to update the flow tables of all its nodes, which will generate more control overhead than path B. In addition, path C has higher probability of affecting other existing routing paths as well due to its larger scope of link update.

Therefore, it is preferable to keep the relay nodes of the original path untouched (as many nodes as possible) in the alternative path. Based on the linear programming equations in Section 3.4.4, it is necessary to meet several necessary conditions for the alternative path. Given \( \rho_{\text{alt}} \) and \( f_{\text{rrt}} \) as an alternative path and rerouted flow, respectively, we have:

\[
\sum_{\rho \in f(e)} f^\rho + f_{\text{rrt}} \leq c(e) \quad \forall e \in \rho_{\text{alt}}
\]

\[
\sum_{k \in K} F_{e;\text{out}}^{\text{check}} = \sum_{k \in K} F_{e;\text{in}}^{\text{check}}
\]

\[
\forall v \in \rho_{\text{alt}}, \{ F_{v;\text{out}} \cup F_{v;\text{in}} \} \not\subseteq \{ f(v, \sim), f(\sim, v) \}
\]

\[
\frac{f_{\text{check}(i)}}{c_{\text{check}(e)}} + \sum_{e' \in f(e)} \frac{f_{\text{check}(i)}}{c_{\text{check}(e')}} \leq C_i(e) \quad \forall e \in \rho_{\text{alt}}, i \in K
\]

The above three equations describe the features for the edges and nodes of the alternative path when dealing with a predicted link failure. They denote the constraints for available
capacity of each edge, node throughput consistency, and radio frequency collision avoidance, respectively.

The back-up routing calculation is processed in the central controller. This process is triggered by the proposed two-layer LFP model. Specifically, nodes of data plane keep sensing the mobility status of their neighbors. Once a suspect node is detected, it will send a warning message to the controller, which will double check the mobility status of this suspect node via SVM training model from a global perspective. As long as the link failure is confirmed by the controller, it immediately seeks the alternative path and sends the necessary re-routing commands to the relevant nodes. Based on the traffic engineering model described in section 3.4, the alternative route is determined by three necessary conditions for the purpose of minimizing the network overhead and impacted network area caused by the link failure.
4 SDN-WMN SIMULATION PLATFORM IN NS3

4.1 Simulation overview

To validate the proposed network paradigm, we utilize a widely-used network simulation tool, called ns-3. The ns-3 simulator is a discrete-event network simulator targeted primarily for research and educational use. Its library covers the most of network structures, such as Ethernet, mesh network, Wi-fi and WiMAX, etc. In this thesis, we employ a set of wireless network models to achieve a WMN platform. These ns-3 models include wireless channel model, mobility model, UDP model, MAC layer model and wireless channel lost model, etc.

The SDN structure is a relative new concept in network standard, so Ns-3 doesn’t provide a complete SDN model. As such, an external SDN library is introduced to integrate with Ns-3 platform. The SDN library is originally based on the Ericsson TrafficLab 1.1 softswitch implementation and then modified in the forwarding plane to support OpenFlow 1.3 standard. In this thesis, the interface of the SDN library is further revised to establish the interconnection between the SDN library and Ns-3 platform. Furthermore, the SDN concept is mainly based on wired network paradigm. To expand the SDN features on wireless network environment, we also modified some models of Ns-3 and SDN library.

4.2 Background of Simulation

The communication networks can be very huge and complex, in which traditional analytical methods are not enough to estimate the performance and the behavior accurately. Thus, network simulators have been introduced to improve the network evaluation. Normally, a network simulator contains various models to achieve the functionality of computer
networks, such as channel model, mobility model, routing model and application model etc. Since there are a huge number of functionalities in different layers (i.e. physical layer, data link layer, routing layer, application layer etc.) of a computer network, a network simulator could be a large and complex software system with the coverage of all these functionalities. Besides various network functionalities, a network simulator also needs to logically integrate the models together and make them work together as a one whole system. This procedure involves some critical aspects, such as network synchronization, parameter statistics, thread management, model integration, and internal simulator clock, etc. In a network simulator, all the API of the models need to be well designed to cooperate with other relative models. Simulating a new protocol in a simulator would incur lots of works, since a huge number of models must be modified to adapt the new model. Based on this consideration, it is preferred to select a professional network simulator that cover the functionalities we need as many as possible. Nevertheless, SDN paradigm hasn’t been well built in the most network simulators. Especially, the current research of SDN is more focusing on wired network and none of the simulation tools have combined the SDN framework with a wireless network environment. On the other hand, there is also a specialized network simulator for SDN structure, called Mini-net. MINI-net is mainly focused on wired network structures. It is short of functionalities of WMN, in which the Ad-hoc protocol and mobility functions cannot be achieved. Furthermore, as a light-weighting SDN simulator, MINI-net is not able to simulate any traditional distributed networking structures either. Users cannot make comparisons between SDN structure and traditional distributed network structure by Mini-net.

In brief, we decided to employ ns-3 simulator due to the following advantages:

- **Open source simulator.** As an open and free network simulation tool, ns-3 has a better flexibility than other simulators. Thus, it is relatively easier to add new components to ns-3 for the performance evaluation of proposed network protocol.

- **Reliability.** ns-3 has been developed and widely used in more than 10 years. Because it has been widely used and maintained in a long time, it provides a more reliable
network simulation system.

- Optimized simulation structure. As the inheritor of ns-2, ns-3 has been optimized to obtain more efficient accurate simulation.

- closer to Realism. The protocol models in ns-3 are designed to be closer to real computers, where some hardware can be integrated with ns-3 to further achieve emulation.

- Complete Tracing system. Multiple tracing methods are developed in ns-3. These tracing methods enable an easier way to trace the network behaviors and parameter statistics.

The ns-3 simulator is a discrete-event network simulator targeted primarily for research and educational use. It is written in C++ and mainly supported by Unix Operation system. Like its ancestor ns-2, ns-3 is developed to provide an open, extensible network simulation platform. However, it is not backwards-compatible with ns-2. Basically, the ns-3 is a set of libraries of networking models to simulate how the packet data is operated in a various protocol. These models can be integrated together by a simulation engine to perform various network simulation tasks. Thanks to the open structure, ns-3 can work with other external software libraries.

Figure 4.1 shows a set of typical models of ns-3. The models of ns-3 cover the most network standard in both wired and wireless networks. The models of ns-3 can be classified into three categories in figure 4.1: devices (left), utilities (middle) and protocol (right). Device models of ns-3 contain a series of network devices in different standard. Particularly, the CSMA device models a simple bus network in the spirit of Ethernet. LTE model is designed to evaluate the LTE system, such as radio resource management, QoS-aware Packet scheduling, Inter-cell interference Coordination and dynamic Spectrum access. WiMAX aims to create models of 802.16-based networks. The Wi-Fi models a wireless network interface based on the IEEE 802.11 standard. The mesh module extends the Wi-Fi module to provide mesh networking capabilities based on the 802.11s standards.
The models located on the right-hand of the figure 4.1 are some protocol models. aodv, olsr, dsdv and olsr are routing protocol for ad-hoc wireless networks or wired networks. Click is a protocol of pipeline based routing system. The Openflow is a popular standard of SDN. This OpenFlow model of ns-3 only provides simplified features on single OpenFlow node simulation, so it cannot accomplish the simulation task for SDN network.

Utility models are used to build a platform by integrating various models. This category contains events scheduler, time arithmetic, tracing, logging, random variables etc. The packet formats are defined by network module. Mobility component provides different mathematical mobility models for wireless network simulation.

4.2.1 Proposed SDN based WMN in NS-3

Since the SDN implementation in ns-3 is outdated and has limited functionalities of SDN, we introduce an external SDN library to enable a complete SDN protocol on NS-3 platform. The SDN library is based on the Openflow protocol version 1.3. It provides a SDN-based switch module and a controller application in Openflow standard. The main structure of the SDN library is shown in figure 4.2.

The SDN-based network includes data plane and control plane. The data plane consists
of SDN-based switches, while a control plane includes one or multiple central controllers. In the simulation, both controller and switches stand for network devices. As such, the basis of these two kinds of devices are existing models of ns-3, called "netdevice". In terms of C++, the "netdevice" is an important class defining a set of basic functionalities of network devices. All the other network devices in ns3 are actually the inheritance of the "netdevice" class. They inherit the interface and functionalities of the "netdevice" class and all more other functions to accomplish different network devices, such as Ethernet device, Wi-Fi device and mesh network device etc. In the simulation, the central controller also stands as a network device, so the external SDN library enables a set of SDN features based on the "netdevice" class to obtain a SDN controller. Similarly, a SDN-based switch is also based on the ns3 "netdevice" class, while the external SDN library is equipping corresponding SDN features on it.

4.2.2 SDN-switch structure of SDN library

The Openflow components provided by SDN library established upon the "netdevice" class of ns-3. There are proper interfaces between Openflow devices and ns-3 "netdevice". As such, the OpenFlow device can be used to interconnect ns-3 nodes using the existing network devices and channels. Figure 4.3 shows the internal switch device structure.

The SDN library provides OpenflowPort model replacing the port model from ns-3, each port associated with a ns-3 underlying NetDevice. The switch device acts as the intermediary
between the ports, receiving a packet from one port and forwarding it to another. The other
OpenFlow switch functionalities (flow tables, group table, and meter table) are also provided
by the SDN library. Basically, packets entering the switch are sent to the flow table pipeline
processing before being forwarded to the correct output port(s). Similarly, the control traffic
between OpenFlow switch and controller is also processed by flow table and then process the
corresponding configuration.

4.2.3 SDN Controller Application Interface

Regarding the SDN controller application interface, the SDN library provides the basic
features of OpenFlow based controller implementation. It can handle a collection of OpenFlow
switches, as illustrated in Figure 4.4. For constructing OpenFlow configuration messages and
sending them to the switches, the controller interface relies on the dpctl utility provided by
the SDN library. With a simple command-line syntax, this utility can be used to add flows
to the pipeline, query for switch features and status, and change other configurations.

For control traffic from SDN-based switches, the controller interface provides a collection
of internal handlers to deal with the different types of messages. Some handlers cannot be
modified by derived class, as they must behave as already implemented. Other handlers
can be overridden to implement the desired control logic. In addition, the SDN library
also equipped a table in central controller to cache the information of network topology.
The controller monitors the network by collection particular control messages from switches.
According to the collected information, the central controller updates the table in real-time. Meanwhile, the central controller will send the control messages to switches based on the data in the table.

### 4.3 Performance Validations

The section presents the performance evaluation for the proposed traffic engineering strategies in a SD-WMN. Specifically, we have implemented the proposed centralized traffic engineering, the two-layer LFP model and the SDN-based WMN structure, in a ns-3 based simulation system.

The simulation platform contains 50 nodes deployed in an $300m \times 1500m$ area. Regarding the transceiver capacity, we adopt the $DsssRate 11Mbps$ model from ns-3. This means that
Figure 4.6: Performance comparison between PLUS-SW and conventional routing

Figure 4.7: Performance comparisons under different moving speeds
the physical bandwidth of each node is 11Mbps. The transmitter’s power is set to 7.5dBm. All the wireless nodes in the data plane is managed by a SDN controller. The packet size is set to 1200 bytes.

Figure 4.5 demonstrates the performance of the proposed two-layer LFP strategy. In Figure 4.5a, the prediction error rate is displayed in terms of false alarm rate and failure detection missing rate. The X-axis is the measurement time interval, while the Y-axis is the error rate (in percentage). As shown in Figure 4.5a, while the individual data plane prediction can keep the error rates in a relative low level, the two-layer prediction scheme significantly improves the accuracy of failure prediction. Meanwhile, a longer SNR measurement time would also lower the prediction error rate. Figure 4.5b illustrates the benefits of using the proposed two-layer prediction, i.e., the PLUS-SW enables a timely link hand-off control (i.e., switching the link or the channel in use) once a link failure is predicted. As a result, the network actively avoids the packet loss caused by low SNR in data transmissions. Based on our experiment, the proposed two-layer LFP is able to precisely predict the link failure events for the general mobility cases, such as human walking, vehicle moving, etc.

To further estimate the performance of the proposed prediction model, we will consider multiple data flows with different traffic generating rates and mobility speeds. As such, the two-layer LFP can be comprehensively assessed in the entire SD-WMN. In this experiment, we aim to compare the performance of the proposed PLUS-SW with the traditional distributed network control. The throughput and delay are collected under different traffic generating rates (from 2Mbps to 10Mbps for each flow). we create 4 different data flows in the network (unlike the previous experiment with only one data flow).

Figure 4.6 illustrates the performance of the network with different traffic generating rates. As shown in Figure 4.6a, the proposed PLUS-SW traffic management outperforms the tradition routing protocol based on the shortest-path strategy (such as AODV). Since the traffic flows are scheduled via a centralized control, PLUS-SW achieves better traffic balancing level than the traditional distributed routing protocols. On the other hand, the
delay of PLUS-SW also outperforms AODV (see Figure 4.6b). This indicates that the data traffic transmission collisions can be effectively prevented through PLUS-SW.

Since we employ the LFP strategy to cope with mobility issue in the WMN environment, the performance evaluation for different mobility speeds is also conducted in ns-3 simulator. As in the previous experiments, the simulation contains 50 nodes with total 4 data flows. The traffic generating rate of each data flow is 2 Mbps.

As shown in the Figure 4.7, the proposed PLUS-SW has a better performance under mobility. For the same traffic generating rates, the PLUS-SW achieves a higher data delivery rate than the traditional routing scheme, under different mobility speeds (see Figure 4.7a). And the delay of PLUS-SW is lower (Figure 4.7b). Thus, the proposed LFP strategy is efficient in mobility environment.

The PLUS-SW supports the QoS by scheduling the data flows with different priorities. As one of the widely used metrics, jitter plays an important role for real-time multimedia transmissions. Figure 4.8 illustrates the jitter performance under various traffic generating rates and moving speeds of the nodes. As in the previous experiments, the traffic generating rates change from 2 Mbps to 10 Mbps for each data flow. As shown in Figure 4.8a, the jitter of the proposed PLUS-SW traffic engineering model has the lower jitter than the tradition
routing scheme. On the other hand, the faster moving speed can cause higher jitter, since the network has to discover a new path when the path is broken due to mobility. Nevertheless, the proposed PLUS-SW still exceeds the performance of conventional routing with respect to jitter.

To further evaluate the performance of PLUS-SW, we test the packet delivery rate (PDR), end-to-end delay, and the jitter of data flows with different number of hops. In this experiment, we create multiple data flows with different path lengths (i.e., number of hops). As in the previous simulations, 4 data flows are used in the entire network.

As shown in the Figure 4.9, more hops can cause higher packet loss rate. This is due to traffic conflicts, radio interference, etc. Through the proposed network-wide management in PLUS-SW, the packet delivery rate is higher than the traditional routing protocol. At the same time, the delay of PLUS-SW is smaller. This is because the optimal path selection through a global control tends to manage the traffic allocations with less conflicts (Figure 4.9b). In Figure 4.9c, the jitter of PLUS-SW is lower as well. The repeatedly forwarding times (shown in Figure 4.9) matches with the result of Figure 4.8, since the less repeated forwarding times in PLUS-SW protocol can lower the jitter in each data flow’s transmission.

4.4 Conclusions

In this paper, we have presented an innovative SDN-based WMN structure. This paradigm can fully explore the centralized SDN control features and overcome some challenging issues in SD-WMN such as the difficulty to handle mobility. First, we utilize the central controller to implement a globally optimized traffic engineering. Second, two supervised learning models are introduced to cope with the permanent link failure issues caused by WMN mobility. Third, a back-up route seeking scheme is proposed based on a new traffic management model under link failure. The entire system is validated on the ns-3 simulator, where the performance of the SD-WMN is improved in both static and mobility network environments.

Because SDN offers a great opportunity for intelligent network control, in the future work
Figure 4.9: Performance for different number of hops
we will use new intelligent models (for example, based on machine learning algorithms) for SD-WMN routing/congestion control. Meanwhile, the overhead of the control traffic will be reduced through low-complexity SDN control schemes in our future research.
This chapter is organized as follows. GNU Radio and USRP are introduced in Sections 1 and 2, respectively. The architecture of the real-time video transmission test bed is described in Section 3. Section 4 presents programming level details of the test bed. Other functions of the test bed related to CRN are presented in Section 5, including spectrum sensing, frequency hand-off, and cross-layer design. Finally, Section 6 provides the details of the test bed.

5.1 GNU Radio Introduction

GNU Radio provides a simple-to-use and rapid-application-development environment for the design of a real-time high-throughput radio [43]. Basically, GNU Radio code is written in both Python and C++ programming languages. C++ code is primarily in charge of implementing the critical signal processing function units, while the Python code groups them together as an entire system. GNU Radio performs most of the signal processing tasks in an SDR. It can be used to build applications to receive data out of digital streams or to push data into digital streams, which are then transmitted using the hardware components. The GNU Radio library provides various basic wireless communication tools, such as filters, channel codes, synchronization elements, equalizers, demodulators, encoders, decoders, and many other elements (called element blocks) that are typically found in radio systems. In addition, it includes a method of connecting these blocks and manages how data is passed from one block to another. GNU Radio also allows users to develop other blocks that are not already present in the GNU Radio library.
5.1.1 How GNU Radio Works

GNU Radio can be seen as a library of blocks for accomplishing signal processing tasks. Users are able to connect these blocks together to achieve some functions of traditional communication devices. Generally, GNU Radio can be operated in two ways: GNU Radio companions (GRC) and regular code programmed in Python and C++.

We explain the working principle of GNU Radio with the help of the following example (shown in Figure 5.1, which is based on the GRC graphic interface [47]. Users are able to construct a flow graph by picking up some blocks with specific functions from GRC of the GNU Radio library and connecting them together. This system starts from the source block in the left-hand side of Figure 5.1. The source block is mainly used to generate various signals, such as audio source, random source, and noise source. The sink block (right-hand side) can be seen as a consumer of data. We can either display the processed data on screen or store it in memory of the sink block. Between the source block and sink block, there are series of functional processing units (i.e., blocks). GNU Radio provides plenty of functional blocks such as filter, selector, modulation, demodulation, etc. Furthermore, a number of parameters of each block are decided. For example, the sample rate of the audio source and sink should be decided in advance. Because some blocks have multiple inputs or outputs, the system constructed by GRC can be quite complicated.

5.2 USRP Introduction

GNU Radio is a pure software library. To implement SDR, hardware support is also required. Several types of hardware have been used in SDR research, such as USRP, Comedi,
and Perseus. Currently, USRP is the most popular hardware among them. The family of USRP devices, produced by Ettus Research, includes different motherboards with USB or Gigabit Ethernet interfaces, with sampling rates up to 100 MSPS and a range of front-ends for reception and transmission from 0 Hz up to 5.8 GHz. They are available as PC-bound devices or as stand-alone embedded devices [49]. All USRP devices use the same universal hardware driver (UHD), which is natively supported by GNU Radio. UHD software is developed as a separate project from GNU Radio, but all UHD devices can be easily used within GNU Radio.

Figure 5.2: USRP board and host PC structure

Generally, a motherboard and a daughterboard are the two most critical components of a USRP hardware system. The motherboard generally includes an analog-to-digital converter (ADC), a digital-to-analog converter (DAC), and an FPGA. ETTUS Research provides a series of daughterboard products for different applications, depending on the radio frequency range and full- or half-duplex operation. Figure 5.2 shows the structure of an SDR composed of a USRP board and a host computer. First, the USRP board is connected to a host PC where GNU Radio software is installed, with a USB or Ethernet cable. Second, a proper daughterboard is plugged onto a USRP motherboard in order to send or receive radio frequency (RF) analog signals. Third, an antenna, the front-end of the system, is connected with the daughterboard. ETTUS provides several kinds of antennas for different applications.
5.3 Test Bed Architecture with Test Bed Architecture

Our test bed uses the USRP N210 motherboard and an SBX daughterboard as hardware support for real-time video transmission. An important feature of N210 is its MIMO capability with high bandwidth and dynamic range [45]. Two N210 units may be connected together to realize a complete 2×2 MIMO configuration using the optional MIMO cable. Moreover, the Gigabit Ethernet interface serves as the connection between the N210 and the host computer. This enables the user to realize 50 MS/s of real-time transmission between the sender/receiver sides simultaneously (i.e., full duplex). In addition, N210 also has a faster FPGA than other USRP boards. The SBX daughterboard is a wide-bandwidth transceiver that provides up to 100 mW of output power, and a typical noise figure of 5 dB [46]. It is also MIMO capable, and provides 40 MHz of bandwidth. Generally, the SBX daughterboard is able to handle most of the applications requiring access to a variety of bands in the 400 to 4400 MHz range.

![Test bed hardware setup](image)

Figure 5.3: Test bed hardware setup

The test bed uses two host computers as shown in Figure 5.3. The transmitter-side computer is connected with a web camera, which is used to capture live video. The captured
video is encoded into the H.264 bit stream in the host computer and processed by GNU Radio software. In order to transmit the video data, the host computer is connected to a USRP board with an Ethernet cable. Here, the USRP board acts as an RF transmitter. Similarly, the receiving side also consists of a computer and a USRP board. This USRP board receives the video signal data from the antenna and delivers them to the host computer, which then demodulates and disassembles the received packets. Finally, an H.264 decoder on the computer decodes the packets and displays the video.

Figure 5.4: Real-time video transmission system at RF frequency

Figure 5.4 shows the system flow diagram. GNU Radio and FFmpeg software are installed on both computers. On the transmitting side, FFmpeg performs the video capture and compresses the video streaming in H.264 format. GNU Radio obtains encoded packets from FFmpeg and then performs packet header assembly and modulation before sending the packets to the USRP board. Some fields (such as the preamble, access code, CRC, etc.)
are then appended to each packet. As the last step of the transmission side, the assembled packets are modulated with GMSK and sent to USRP board through the Ethernet interface. On the receiving side, the USRP board captures the analog signals from the air and passes them to GNU Radio. Through GMSK demodulation and packets disassembly by GNU Radio, the received link layer packets are disassembled into application layer packets with sequence numbers. The video packets are finally decoded and replayed by FFplay software. The communication between GNU Radio and FFmpeg/FFplay is through the UDP socket from Python.

The FFmpeg software is employed in the process of capturing real-time video by a webcam and encoding the video data in H.264. As one of the most popular media software in Linux system, FFmpeg is able to decode, transcode, mux, demux, stream, filter, and play most media files [22]. H.264 is a standard for video compression that is one of the most common formats for the recording, compression, and distribution of high-definition video.

5.4 Code Explanation of real-time video transmission on SDN test bed

5.4.1 Sender-Side Python code

After FFmpeg exports H264 packets in real time, GNU Radio is employed to implement some data processing modules for H264 packets, including packet header assembly and modulation functions on the sender side. From the standpoint of the flow graph, the system is generally built by gluing blocks of GNU Radio library together in Python code, as discussed in the following text.

class my_top_block(gr.top_block):
    def __init__(self, modulator, options):
        gr. top_block._init_(self)

        if (options.tx_freq is not None):
            # Work-around to get the modulation’s bits_per_symbol
args = modulator.extract_kwargs_from_options(options)

symbol_rate = options.bitrate / modulator(**args).bits_per_symbol()

self.sink = uhd_transmitter( options.args, symbol_rate,
                             options.samples_per_symbol,
                             options.tx_freq, options.tx_gain,
                             options.spec, options.antenna,
                             options.verbose)

options.samples_per_symbol = self.sink._sps

elif(options.to_file is not None):
    sys.stderr.write(("Saving samples to '%s'.\n\n" % (options.to_file)))
    self.sink = gr.file_sink(gr.sizeof_gr_complex, options.to_file)
else:
    sys.stderr.write("No sink defined, dumping samples to null sink.\n\n")
    self.sink = gr.null_sink(gr.sizeof_gr_complex)

# do this after for any adjustments to the options that may
# occur in the sinks (specifically the UHD sink)
self.txpath = transmit_path(modulator, options)
self.connect(self.txpath, self.sink)

First, the code defines a class named “my_top_block” that is a container for the flow graph; it is derived from another class “gr.top_block”. All blocks of the system are connected under this class. Generally, the sender side Python code consists of two main blocks, named ‘self.txpath’ and ‘self.sink’. Note that ‘self.’ refers to the instance attribute in Python. The ‘self.txpath’ block defines a series of parameters and functions (including modulation, signal amplitude sent to USRP, sample frequency, bit rate, etc.) that belong to GNU Radio software. On the other hand, ‘self.sink’ is derived from the “uhd_interface” class. The UHD
is the driver of USRP and works as an interface between GNU Radio and USRP. All the parameters and functions of the ‘self.sink’ block are related to the USRP board, such as the sampling frequency in the ADC, RF signal amplitude from the antenna, and signal frequency. After these two blocks are defined, the command self. connect(self.txpath, self.sink) is used to connect them together.

The “my_top_block” class is followed by a main function, which is where the program execution starts. All functions are executed in a certain sequence in this main function. In the first dozens of lines of codes, the necessary functions and variables are defined, such as modulation options, sampling rate of USRP, etc. Then, the class “my_top_block” is invoked to build the flow graph in the main function, as shown in the following text. “gr.enable_realtime_scheduling()” is a module from the GNU Radio library, and it is used to check if the code works in real-time scheduling.

```python
# create a socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
host = ''  # can leave this blank on the server side
try:
    s.bind((host, options.port))
except socket.error, err:
    print "Could not set up a UDP server on port %d: %s" % (options.port, err)
    raise SystemExit

while True:
    data = s.recv(4972)
    print "raw data length = %4d" % (len(data))
    if not data:
        break
```

Once GNU Radio code receives the H.264 data packets from FFmpeg, a data header assembly process is launched. In this step, we use a Python module named ‘structure.pack()’. 
Some necessary information is added in the header, such as packet number, packet length, etc. Each H264 packet is then sent to the USRP board. In future work, the routing information would also be added in the MAC and data link layer applications.

```python
while (len(data) > max_pkt_size): # (maximum allower packet size: 4096)
    partial_pl = data[0:max_pkt_size] # extract 3072 bytes
    # print "partial_pl 1 length = %d" % (len(partial_pl))
    partial_pl = struct.pack(’!H’, 0x5555) + partial_pl
    # print "partial_pl 2 length = %d" % (len(partial_pl))
    payload = struct.pack(’!H’, pktno & 0xffff) + partial_pl
    # print "partial_pl 3 length = %d" % (len(payload))
    send_pkt(payload)
    n += len(payload)
    pktno += 1
    sys.stderr.write(’*’) # "*" to denote split packet
    data = data[max_pkt_size:] # update "data"
data = struct.pack(’!H’, 0xaaaa) + data
payload = struct.pack(’!H’, pktno & 0xffff) + data
    send_pkt(payload)
    n += len(payload)
    sys.stderr.write(’.’)
pktno += 1
send_pkt(eof=True)
```

### 5.4.2 Receiver-Side Python Code

The receiver side executes the inverse process of the sender side. After a USRP board captures live video data from the antenna, it is converted from an analog to a digital signal,
and H.264 video packets are recovered. Then, the Python code based on GNU Radio software is used to separate the header from the packet and read necessary information from it. Next, the raw video data is sent to the video player and gets played in real time.

Similar to the sender side, the packets also need to go through the flow graph built by GNU Radio on the receiver side. The Python code of receiver side also begins with a `my_top_block` class, which consists of two main blocks in the receiver side, named ‘source’ and ‘xpath’. The “source” block derives from the UHD and provides necessary parameters and functions to the receiver-side USRP board, while ‘xpath’ integrates some blocks from the GNU Radio library. Generally, ‘xpath’ consists of a demodulation block, an ‘rx_callback’ function, and an ‘options’ module of Python. The ‘rx_callback’ is an unpacking function that will be explained later. The ‘options’ module enables the user to input some parameters when the Python code is executed. The code of ‘my_top_block’ class is shown in the following text.

```python
class my_top_block(gr.top_block):
    def __init__(self, demodulator, rx_callback, options):
        gr.top_block._init_(self)
        if(options.rx_freq is not None):
            args = demodulator.extract_kwargs_from_options(options)
            symbol_rate = options.bitrate / demodulator(**args).bits_per_symbol()

            self.source = uhd_receiver( options.args, symbol_rate,
                                        options.samples_per_Symbol, options.rx_freq, options.rx_gain,
                                        options.spec, options.antenna, options.verbose)
            options.samples_per_symbol = self.source._sps
        elif(options.from_file is not None):
            sys.stderr.write(( "Reading samples from '%s'.

            (options.from_file)))
            self.source = gr.file_source( gr.sizeof_gr_complex, options.from_file)
```

101
else:
    sys.stderr.write( "No source defined, pulling samples from null
                      source.\n\n"
)  
self.source = gr.null_source(gr.sizeof_gr_complex)
self.rxpath = receive_path(demodulator, rx_callback, options)
self.connect(self.source, self.rxpath)

The structure of the main function of the receiver-side code is very similar to the sender-
side code. The first dozens of lines of code define some parameters and simple functions.
Then, the Python module `socket` establishes a UDP protocol pipeline for sending the video
data out of the GNU Radio flow graph. Next, the `rx_callback` function, given in the
following text, is used to unpack the header from raw packets.

def rx_callback(ok, payload):
    global n_rcvd, n_right
    print "Received packet length = %4d" % (len(payload))
    (pktno,) = struct.unpack (’!H’, payload[0:2])
    n_rcvd += 1
    if ok:
        n_right += 1
    print "ok = %5s pktno = %4d n_rcvd = %4d n_right = %4d" % (ok, pktno, n_rcvd,
                                                             n_right)
    print "Sent packet length = %4d" % (len(payload[2:])))
    cs.send(payload[2:])

After all the parameters and functions are ready, the flow graph of GNU Radio is built
and executed using the following commands.

# build the graph

# build the graph

# build the graph

# build the graph

tb = my_top_block (demods[options.modulation], rx_callback, options)
5.5 Other Functions of Test Bed

Besides the real-time video transmission, the following functions are implemented in this test bed to realize the CRN:

- Spectrum sensing;
- Spectrum hand-off;
- Cross-layer design (such as routing/MAC cross-layer design).

5.5.1 Spectrum Sensing Function

Spectrum sensing is a critical function in a CRN and is used to detect if the radio channel is occupied by the primary user. Specifically, the spectrum sensing can be seen as a binary hypothesis [39]:

\[ H_0 : y[n] = w[n] \quad n = 1, ..., N \]
\[ H_1 : y[n] = x[n] + w[n] \quad n = 1, ..., N \]

Here, \( x[n] \) is the primary users signal, \( w[n] \) is the noise, and \( n \) presents time factor. If the received signal \( y[n] \) is in \( H_0 \) status, the channel is considered as idle, that is, available to other users. On the other hand, \( H_1 \) means the channel is occupied by the primary users. It is complicated to decide whether the channel is occupied or not in practical RF environment because \( x[n] \) may be too weak to be detected, or noise signals may be very similar to the
primary users and can confuse the spectrum sensing unit. Hence, several spectrum sensing techniques have been used, which can be divided in three categories [1]: energy detection, matched filter detection, and cyclostationary detection. Among them, energy detection is most widely used because it is very simple and does not require any a priori knowledge of the primary signals or complicated algorithms. In energy detection, the noise energy of the channel is sensed and compared with a predefined threshold. If the noise energy is higher than the threshold, the channel is considered as occupied; otherwise, the channel is idle.

Our test bed uses the energy detection for spectrum sensing. The Python code of spectrum sensing is explained in the following text.

5.5.2 Explanation of Code for Energy-detection based Spectrum Sensing

The Python code used to realize the spectrum sensing function comes from the GNU Radio library ‘usrp_spectrum_sense.py’. The entire system is divided into several blocks as shown in Figure 5.5, where each block represents a basic function of the system. These blocks are glued together by the class ‘gr.top_block’ from the GNU Radio library.

The USRP receiving module is represented by ‘self.u = uhd.usrp_source()’. This command is generally used to set necessary parameters of the USRP board and ensure that the RF signal is received properly. After streaming data are captured by the USRP board, they is sent to the fast Fourier transform (FFT) block. Between these two blocks, command
's2v = gr.stream_to_vector()' is introduced to take a sample of streaming data in vector format. For the next step, the output data of the FFT block goes through the 'c2mag' block (converting data from time domain to frequency domain), and it is finally accepted by a data statistics block called ‘stats = gr.bin_statistics_f()’. Generally, this module is used to control scanning and record the frequency domain statistics. The USRP board can only detect 8 MHz bandwidth of the RF frequency channel at a time. If we want to perform sensing in a wider frequency band, the USRP board should sense 8MHz each time and repeat this process in different frequency bands until the entire expected RF band is covered. The ‘stats’ block is used to record result when the system gets one noise sensing process done and sets the system to the next sensing process. Moreover, an invoking function named ‘tune()’ is predefined to set the USRP board to next frequency interval.

5.5.3 Spectrum Hand-Off

The spectrum hand-off is another important component of a CRN. When a primary user reoccupies the channel, the secondary users who are currently occupying this channel must vacate it immediately. In this case, the secondary user hops to another available frequency channel by using a spectrum hand-off function. A simple cross-layer application is also achieved based on frequency hand-off. The cross-layer approach allows parameter exchange between different protocol layers [19]. It improves the communication efficiency and speed through coordination, interaction, and joint optimization of protocols across different layers. The frequency hand-off is implemented in Python in our test bed. The basic idea is to realize the frequency hopping while the video is transmitting in real time. On the basis of GNU Radio and URSP hard drive software (UHD), the USRP working frequency can be simply changed by calling the frequency setting function ‘set.enterFreq()’ from the UHD library, which is programmed in C++. The frequency setting function is already defined in the USRP hardware driver module ‘uhd_interface_hopping.py’. In our test bed, we use two radio frequency channels, and USRP can switch transmissions between them. The code is shown in the following text.
def set_next_freq(self):
    freq_interval = 1000000

    target_freq = self.next_freq
    if self.next_freq > self._freq + freq_interval/2:
        self.next_freq = self._freq
    elif self.next_freq < self._freq + freq_interval/2:
        self.next_freq = self._freq + freq_interval

    rn = self.u.set_center_freq(target_freq, 0)

    if rn:
        print 'Succeed to set frequency to, target_freq'
    else:
        print 'Failed to set frequency to, target_freq'

    return target_freq

Another problem that should be considered is how to synchronize the spectrum handoff between the sender and the receiver sides. Currently, our USRP boards operate in the half-duplex mode. If the receiver side fails to synchronize with the sender side after the latter performs a hand-off, the packets will be lost. We use data packets to transmit a frequency hand-off command from the sender side to the receiver side. When the sender side decides to switch to another frequency channel, a command is written into the header of a control packet and sent to the receiver side. In response, the frequency-setting function is immediately executed at the receiver. The hand-off synchronization can be improved in the following two ways. A test bed with full-duplex communication between the sender and
receiver side can be used. Alternatively, a third USRP board can be used that works as a spectrum sensing station. When a primary user is detected on the current frequency channel, it will send commands to both the sender and the receiver, and both of them will carry out the spectrum hand-off at the same time.

5.5.4 Cross-Layer Design of Test Bed

We have also implemented a cross-layer design by adjusting the video resolution with different radio frequency channels during the real-time video transmission. The cross-layer design facilitates the information exchange between the application layer and physical layer. Based on the aforementioned spectrum hand-off design, we can achieve this cross-layer design by producing video output in two resolutions and transmitting each of them in different FR channels. Two different resolution video outputs are achieved by using FFmpeg, as discussed in the next section. With respect to the GNU Radio flow graph code, two pipelines are employed by the socket module. Each of the two video outputs is assigned a pipeline to transmit video data from FFmpeg to the GNU Radio flow graph. Based on the different frequency channels the test bed is operating on, the GNU Radio flow graph will switch to the corresponding pipeline and receive the video streaming data.

5.6 Test Bed Demo for Real-Time Video Transmission

The following four functions are implemented in our test bed:

- High-resolution real-time video transmission.
- Spectrum sensing.
- Frequency hopping for spectrum hand-off.
- A simple cross-layer design.

Because the GNU Radio flow graph needs to invoke other modules, all the Python files of GNU Radio are put in one directory: ‘gnuradio/example/digital/narrowband/’.
5.6.1 Signal Sensing and Video Transmission

The general demo steps are as follows:

- Sense the signals on the specified spectrum bands.

- Find out the channel with the lowest signal amplitude, and select it on both the sender and receiver sides.

- Transmit the video in real time.

For spectrum sensing in Step 1 above, the ‘hop_usrp_spectrum_sense.py’ file is executed by running a command in ‘hop_usrp_spectrum_sense.py 900M 1500M’. Specifically, 1500M and 900M are the maximum and minimum values of the frequency range, respectively. Then, a spectrum sensing result is generated in seconds. Figure 5.6 shows the result of the spectrum sensing process, in which the noise amplitude is presented.

In the next step, the test bed transmits the real-time video on the frequency channel that has the lowest noise amplitude.
On the sender side:

1. The FFmpeg software is executed in a new terminal window via the command `ffmpeg -f video4linux2 b : v 32k s 1080 * 960 -r 120 -i/dev/video0 -c:v libx264 -tune zerolatency -y -f h264 udp://127.0.0.1 : 12347`.

2. In a new terminal, go to the directory `usr/local/share/gnuradio/examples/digital/narrowband` and execute the sender side GNU Radio flow graph code with the command `./video.x.py -f 910M -port 12347`.

On the receiver side:

1. The video player ffplay is opened first in standby status using the command: `ffplay -f h264 udp://127.0.0.1 : 12345`

2. In a new terminal window, input the `./packet_tranceiver.py` command to run the Python file. This file is used to connect the GNU Radio flow graph Python code with the ffplay software.

3. Execute the GNU Radio flow graph code in a new terminal window by using the `video_rx.py -f 910M -port 12347` command.

5.6.2 Frequency Hopping

The general demo steps are as follows:

1. Set two frequency channels in advance.

2. The sender side hops from one channel to the other after transmitting a certain number of packets.

3. The receiver also hops such that it operates in the same frequency channel as the sender side in order to uninterruptedly receive the real-time video packets.
Figure 5.7: FFT result of the frequency hopping experiment.

The GNU Radio flow graph at the sender and receiver sides are called ‘freq_hopping_tx.py’ and ‘freq_hopping_rx.py’, respectively. Similar to the experiment in last section, the FFmpeg software is run first with the same command as the last section. Then, the GNU Radio flow graph Python code is executed with the command ‘./freq_hopping_tx.py’. On the receiver side, the process is also the same as the last section, except that ‘video_rx.py’ is replaced by ‘freq_hopping_rx.py’. Figure 5.7 shows the FFT of the result of the frequency hopping experiment.

In Figure 5.7, the blue curve presents the real-time video transmission signal amplitude, while the green curve represents the peak amplitude for the entire transmission process. Obviously, the video test bed is transmitting on a 911 MHz channel. A moment later, the working frequency shifts to the 910 MHz channel. Note that the user is able to set two frequency channels in any radio frequency bands in the test bed. The hardware used in this test bed can handle any frequency between 850 and 4400 MHz. Owing to fast frequency switching in USRP, only a few video frames may be lost in this process, while the delay between the sender side and the receiver side is still small.
5.6.3 Demo of Cross-Layer Design

On the basis of the frequency hopping in last section, a cross-layer function is added in this part that changes the video resolution with frequency hopping. Two video outputs with different resolutions are generated from the FFmpeg software. Because all the frequency channel values and the two ports of pipelines connected with FFmpeg have been written in the GNU Radio flow graph code, the procedure here is similar to the previous sections.

On the sender side:

1. Run the FFmpeg software with the command `ffmpeg -f video2linux2 b:v 32k s 320x240 r 64 i/dev/video c:v libx264 tune zerolatency y f h264 udp://127.0.0.1:12347 s 196x98 c:v libx264 tune zerolatency y f h264 udp://127.0.0.1:12346`.

2. Run the GNU Radio flow graph Python code `./crosslayer_tx.py` in a new terminal window.

On the receiver side:

1. Run the ffplay software by `ffplay f h264 udp://127.0.0.1:12345`.

2. `./packet_tranceiver.py` in a new terminal window.
3. `./crosslayer_rx.py` in a new terminal window.

Figure 5.8 shows a video frame in two different resolutions at the receiving side.
6 CONCLUSIONS AND FUTURE WORK

This thesis aims to improve the performance the WMN in both traditional distributed network and SDN-based centralized control network.

In the traditional distributed network environment, we have presented an innovative multi-beam routing protocol based on RDC formation. It can fully explore the high capacity of multi-beam antennas through the diamond chain routing architecture. The RDC-based routing consists of a main path and multiple side paths. We have applied rateless codes to such a multi-beam transmission in order to adapt to the fast fading links with dynamic channel conditions. We further used AI algorithms (FL and RL) to build the main path that consists of the best quality links from a statistical distribution perspective. The ripple-to-ripple schedule control is designed to achieve pipelined transmissions in different ripples. Our simulations with real-time video transmissions have validated our RDC routing efficiency.

In the SDN-based network, we have presented an innovative SDN-based WMN structure. This paradigm can fully explore the centralized control advantages and overcome some issues in SDN-WMN such as difficulty to handle mobility. On one hand, we utilize the central controller to realize a global optimized traffic engineering. On the other hand, two intelligent supervised learning models are introduced to cope with the permanent link failure caused by WMN mobility issue. The entire system is validated on the ns-3 networking simulator, where the performance of the SD-WMN is improved in both static and mobility network environments.

Because SDN offers a great opportunity for intelligent networking system, other AI methods could be adopted for wireless network optimization. Meanwhile, the overhead of control traffic also can be addressed in the next step.
REFERENCES


[9] Gurashish Brar, Douglas M Blough, and Paolo Santi. Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh


[29] Peter Anthony Iannucci, Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. No symbol left behind: a link-layer protocol for rateless codes. In Proceedings of the 18th an-


