ON APPLYING NETWORK CODING TO WIRELESS AD HOC NETWORKS: A STUDY OF ROBUSTNESS, EFFICIENCY AND CROSS-LAYER SYNERGY

WANG JIN

(B.Eng., Harbin Institute of Technology, China)

Supervised by

Professor WONG Wai-Choong, Lawrence

Dr. CHAI Teck Yoong

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2015

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

WANG Jin

February 18, 2015

Acknowledgments

Having completed my PhD thesis, I would like to take this opportunity to thank the following people: Professor Wai-Choong Wong, Lawrence and Dr. Teck Yoong Chai, for their continuous guidance and support during my course of the PhD study. In the past five years, they had trained me not only on how to conduct research, but much more than that, including technical writing, communication, and social interaction skills. I will carry over the spirit of self-motivation and independence, which they taught me during my study. Without their help, I would not be able to finish my PhD study and this thesis would have been getting nowhere.

Pursuing a PhD degree is not merely an academic pursue. There are also many administrative matters that need to be handled. Without the assistance from Miss Jie Guo, it is impossible for me to handle all the paper works on time. As such, I wish to express my gratitude to her!

The China Scholarship Council offered me a scholarship and the School of Engineering, National University of Singapore waived the tuition fee for me and offered me a good place to study. This opportunity changed my life so much that I will always be thankful during the rest of my life.

Last but not the least; I would like to thank my parents, for their support and encouragement throughout my years of studies. They have guided me both in study and in life, I hope what I have done and what I will be doing make them proud of me. At the same time, my fiancé Mr. Cenzhe Zhu is one of the

most important persons, if not the most important one, in helping me complete my PhD study. Although he is not specialised in the area of my research, his perspectives always inspires me. In addition, the happiness and wisdom he brought into my life were very important in overcoming the difficulties I faced. The accomplishment of the dissertation undoubtedly reflects his contributions.

February 18, 2015

Table of Contents

Summary x

List of Tables xi

List of Figures xi

List of Symbols xv

List of Abbreviations xviii

1 Introduction 1

1.1 Background 1

1.2 Research Challenges 4

1.3 Thesis Contribution 8

1.4 Thesis Organization 11

2 Related Works 13

2.1 Overview of Network Coding 13

2.1.1 Robustness in wireless ad hoc networks 18

2.2 Network Coding for Scheduling Problem 19

2.2.1 Overheads in Network Coding 19
2.2.2 MAC-Layer and Physical-Layer Scheduling 22
2.3 Content Distribution in Wireless Ad Hoc Networks with Network Coding 24
2.3.1 Overview of Cross-layer Designs 24
2.3.2 Content distribution using BitTorrent-like protocols . . . 27

3 Improving the Robustness of Coding-aware Routing Protocols to Flow Arrivals 29

3.1 Introduction 29
3.2 Protocol Overview 35
3.2.1 Digest of the DCAR protocol 35
3.2.2 Two tables to store flow information 40
3.2.3 Self recommendations 41
3.2.4 Handling Self Recommendations 42
3.2.5 Decision Making 42
3.2.6 Controlling the Frequency of Self Recommendations . . . 43
3.3 Route-Change Procedure 44
3.3.1 Procedure Timeline 44
3.3.2 The Unbiased CRM Metric 46
3.4 Topology Analysis 48
3.4.1 Theoretical Induction of Indicators 49
3.4.2 Simulation Results 58
3.5 Evaluation 62

3.5.1 Simulation 1. Simple Topology 64
3.5.2 Simulation 2. "Wheel" Topology 66
3.5.3 Simulation 3. Grid Topology 69
3.6 Chapter Summary 72

4 Improving Coding Efficiency and Fairness by Network-layer Packet Scheduling Algorithm 75

4.1 Introduction 75
4.2 The Static Form of the Problem 79
4.2.1 Weighted Clique Cover Problem 79
4.2.2 Solution to WCCP 84
4.2.3 Scalability and Error Analysis 92
4.3 The Dynamic Form of the Problem 98
4.3.1 Heuristic Scheduling 100
4.3.2 Performance with Poisson Arrivals 105
4.4 Evaluation 109
4.4.1 Simulation 1 112
4.4.2 Simulation 2 116
4.5 Chapter Summary 118

5 Content Distribution in Wireless Ad Hoc Networks with Network Coding 119

5.1 Introduction 119
5.2 Routing Metric 122

5.3 Application-layer Strategies 126
5.4 Implementation 130
5.4.1 Initial Setup 130
5.4.2 Cross-layer Dynamics 133
5.4.3 Simulation Settings 134
5.4.4 Results 135
5.5 Chapter Summary 139

6 Conclusion **141**
6.1 Thesis Contribution 141
6.2 Future Work 145

Bibliography **147**

List of Publications **163**

Summary

Due to its fast deployment, error tolerance, and sophisticated protocol stack, the technique of wireless ad hoc networks is utilized in more and more applications. Network coding, typically as a method to improve throughput, is quite suitable for wireless networks. The broadcast nature of wireless networks is analogous to multi-casting for which network coding was originally devised.

In this thesis, we analyse several issues on applying network coding techniques to wireless ad hoc networks. Several aspects, including the robustness, fairness, the coding efficiency and the possible synergy between network layer and application layer are studied.

Specifically, we first propose a coding-aware routing protocol that is able to detect coding opportunities even if the order of arrival of packet flows is unfavourable to such detection. This robustness is essential because wireless ad hoc networks are prone to node failures, node joining, and topology changes. These can significantly impact performance if the routing protocol is not robust enough.

Then we dig deeper into the inner structure of the coding scheme and prove that coding-aware packet scheduling is pivotal in shaping the distribution of throughput among flows. Compared to a trivial round-robin scheduling, our scheduling method decreases the span and variance of per-flow throughputs. In addition, a well planned scheduling can improve the coding efficiency. A good scheduling algorithm is not one that only schedules packets with higher coding

benefits. Instead, it should plan ahead and avoid cases when some packets that could have been coded are left behind with no coding counterpart. Thus, a practical coding-aware packet scheduling algorithm is devised, with the goal to improve fairness and coding efficiency.

Finally, the above-mentioned improvement in the network layer is incorporated into a Peer-to-Peer (P2P) content distribution protocol, capitalizing on the synergy between network layer and application layer. The availability of cross-layer information can greatly help both in network layer and in application layer. By taking the offered load information from the application layer as a substitute for the queue length, the routing protocol can now measure the traffic load of a flow more accurately. On the other hand, the routing metric obtained from the network layer can help the application layer to determine which neighbour peers to connect to. The peer with a better route would be favoured, thus enlarging the coding benefits. In addition, the network layer can back-propagate rate adjustment packets to actively adjust offered load according to the coding graph and the solution to the Weighted Clique Cover Problem (WCCP), where the information is only available in network layer but not in application layer. Results show that such cross-layer solution can greatly improve performance.

List of Tables

3.1 Definition of a Few Terminologies 49

4.1 Default Parameters for Performance Evaluation of the Static-Form Problem 95

4.2 Simulation 1 Parameter Settings 113

5.1 Simulation Parameters 135

List of Figures

1.1 A Typical Wireless Ad Hoc Network 2

3.1 A Simple Test Scenario for SCAR 32

3.2 Route-Change Procedure Timeline (SR: Self Recommendation. MRQ: Modified RREQ Packet. MRP: Modified RREP Packet.) . 45

3.3 Modified RREQ/RREP Packet Formats 47

3.4 Possible Topologies for 2 Coding-possible Route Pairs 53

3.5 Possible Topologies for 3 Coding-possible Route Triples 55

3.6 An Artificially Designed Topology to Measure the Effectiveness of CN Indicators 59

3.7 Scatter Plot of Throughput Gain with Different $C2$ Values 59

3.8 Average Throughput Gain versus Pre-adjusted $C2$ Values 60

3.9 Average Throughput Gain versus Adjusted $C2$ Values 60

3.10 Scatter Plot of Throughput Gain with Different $C3$ Values 62

3.11 Average Throughput Gain versus Pre-adjusted $C3$ Values 63

3.12 Average Throughput Gain versus Adjusted $C3$ Values 63

3.13 Simulation Results for the Simple Test Topology 65

3.14 Simulation Results for "Wheel" Topology 67

3.15 Simulation Results for Grid Topology 71

4.1 There Exists Polynomial Solutions to WCCP for Claw-free Perfect Graphs 83

4.2 One Example of Coding Graph 86

4.3 The Overall Block Diagram of the Algorithm for WCCP 88

4.4 Processing Time Sensitivity to N 96

4.5 Processing Time Sensitivity to p 96

4.6 Processing Time Sensitivity to mean(w) 96

4.7 The Error Rate and Average Error with Different Q Values . . . 98

4.8 Average Throughput Performance with Poisson Arrivals 107

4.9 Minimum Throughput Performance with Poisson Arrivals 108

4.10 A Topology where Source and Destination Nodes are on the Same Circle 113

4.11 The Generated Topology for Simulation 1 113

4.12 Coding Graph at Node O in Simulation 1 114

4.13 Per-flow Throughput in Simulation 1 115

4.14 Random Topology 117

4.15 Per-flow Throughput of Simulation 2 117

5.1 The flow chart depicting the generation of INC/DEC packets . . 130

5.2 Special RREQ/RREP Packet Formats 132

5.3 Cumulative Percentage Finished over Time 136

5.4 Number of Coded Packets Sent 137

5.5 Upload/Download Amounts for Selected Pairs 139

List of Symbols

$C2_{adj}$ The adjusted $C2$ indicator.

$C2, C3, \ldots, CN$ The CN series of topology indicators.

$\mathbf{E}$ An edge set.

$\lfloor \cdot \rfloor$ The floor of a floating-point number, i.e., the maximum integer that is smaller than the given number.

f_n A new coming flow in a network.

f_p A potential flow in a network.

$\mathbf{G}$ A graph.

$\binom{k}{2}$ The number of combinations choosing 2 from k items.

$K_{m,n}$ A bipartite where m and n are the cardinalities of its two disjoint vertex set.

$G(n, p)$ A random graph generator that generates a graph with n vertices and a probability parameter p.

MQ_d The dynamic version of MQ (Modified Queue Length).

MQ_s The static version of MQ (Modified Queue Length).

$N(v)$ The set of neighbouring vertices of a vertex v.

Φ The empty set.

P_l The packet loss rate of link l.

Q The quantization level of the approximation algorithm.

r_{in-use} Previously-in-use route.

$r_i \sim r_j$ A flow that takes route r_i can be coded with another flow that takes route r_j.

$R(S - D)$ The set of possible routes from node S to node D.

r_{sr} The recommended route from source to destination.

Σ^*, Γ The optimal allocation method.

$U(i, j)$ The amount of packets uploaded by node i to node j.

$\mathbf{V}$ A vertex set.

$|\mathbf{V}|$ The cardinality, or the number of vertices, of a vertex set.

V_{q_i}, Q_i The i-th clique of graph $\mathbf{G}$, represented by its vertex set. In some context, it is simply denoted by Q_i.

$w(\cdot)$ A function that maps a vertex to an integer weight.

w_{q_i} The uniform weight of the i-th clique of graph $\mathbf{G}$.

$\mathbb{Z}^+$ The set of positive integers.

List of Abbreviations

ACK Acknowledgement packet.

AODV Ad-Hoc On-demand Distance Vector.

ARQ Automatic Repeat-reQuest.

BT BitTorrent.

CCP Clique Cover Problem.

CRM Coding-aware Routing Metric.

DCAR Distributed Coding-aware Routing.

DMDP Deterministic Markov Decision Process.

DPSA Dynamic Packet Scheduling Algorithm.

DSR Dynamic Source Routing.

ECT Expected Coded Transmission Time.

EDGE Enhanced Data rates for GSM Evolution.

ETX Expected Transmission Count.

FIFO First-In-First-Out.

GPRS General Packet Ratio Services.

GSM Global System for Mobile Communication.

HEU Heuristic Scheduling Scheme.

LTE Long Term Evolution.

MAC Media Access Control.

MDP Markov Decision Process.

MIMO Multiple-Input-Multiple-Output.

MIQ Modified Interference Queue Length.

MIT Massachusetts Institute of Technology.

MQ Modified Queue-length.

MRP Modified RREP Packet.

MRQ Modified RREQ Packet.

NA Non-Approximation Algorithm.

OLSR Optimal Link State Routing.

OSI Open System Interconnect.

P2P Peer-to-Peer.

PHY Physical layer.

pps Packet Per Second.

QoS Quality of Service.

RC Route-Change.

RNC Random Network Coding.

RR Round-Robin.

RREP Route REsPonse Packet.

RREQ Route REQuest Packet.

SCAR Self recommendation Coding-Aware Routing Protocol.

SR Self Recommendation.

TCP Transmission Control Protocol.

TCP/IP Transmission Control Protocol / Internet Protocol suite.

UMTS Universal Mobile Telecommunications System.

VANET Vehicular Ad Hoc Network.

WCCP Weighted Clique Cover Problem.

WCG Weighted Coding Graph.

Wi-Fi Wireless Fidelity.

XLiNC Cross-layer Solution with Network Coding.

Chapter 1

Introduction

1.1 Background

Recent years have seen the proliferation of wireless solutions for communication purposes. Industry-level mobile networks have advanced from analogue signals to digital signals, from the 2G Global System for Mobile Communication (GSM) technology, through 2.5G General Packet Ratio Services (GPRS)/Enhanced Data rates for GSM Evolution (EDGE), 3G Universal Mobile Telecommunications System (UMTS) to the now rising 4G Long Term Evolution (LTE) networks. In fact, a survey in 2013 [1] has already pointed out that the pace of growth of wireless subscriptions has slowed to the extent of reaching saturation. New growth areas include multi-device plans and greater customization of service plans. Concurrently, home Wireless Fidelity (Wi-Fi) coverage is growing steadily, tapping the power of 802.11 series of protocols. Its bandwidth, reliability, and manageability have persuaded more and more users to shift from wired

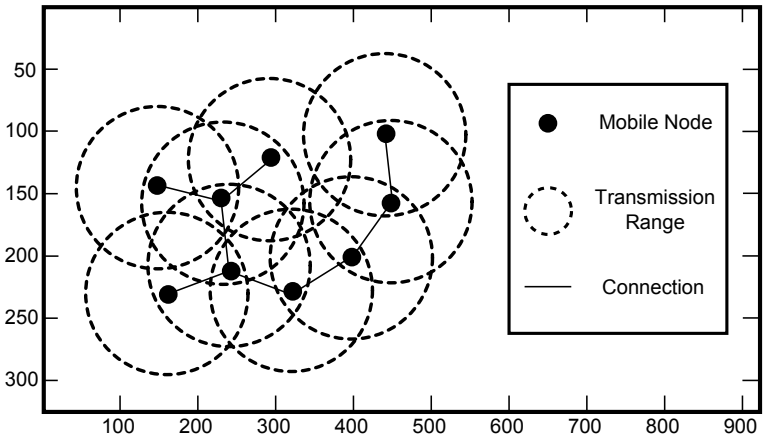


Figure 1.1: A Typical Wireless Ad Hoc Network

Internet connections to wireless ones.

Yet these advances are based on infrastructural topologies where all wireless terminals connect to a single gateway. A wireless ad hoc network is a collection of wireless mobile nodes that dynamically form a temporary network without an infrastructure. Fig 1.1 demonstrates a typical wireless ad hoc network. There are nine nodes in the network. Two nodes, which cannot reach each other directly, form a path consisting of several forwarding nodes. The forwarding nodes that are in between then work cooperatively to establish a multi-hop route from the source node to the destination. Increasing deployment of ad hoc networks for different applications is seen due to its inherent advantageous characteristics: such as, self-configuration, mobility, absence of single point of failure, autonomous behaviour, infrastructure-less operation, ease of deployment, and low cost.

The concept of network coding has begun to rapidly alter the way we think about and implement communication networks [2]. It can be used to improve performance and increase capacity in both wired and wireless communication

networks. The origin of this concept is from Ahlswede [3] in 2000, in which network coding is proposed as a theoretical formulation as a max-flow min-cut problem. With network coding enabled, intermediate nodes in networks can encode several received packets into a single coded packet before forwarding. Sounded like a small improvement, network coding nevertheless has a great impact on the research field of information theory. The early developments of network coding were based on theoretical formulation of the information flows corresponding to the packet transmissions. The research objective is generally to reach the upper bound of throughput as promised in the information flow formulation.

The first breakthrough on network coding is the discovery of random linear network coding by Li [4]. In this paper, an explicit construction of a multicast coding scheme that achieves the max-flow bound on the information transmission rate is given. With this coding scheme, each intermediate node sends through its outgoing channels a linear combination of the packets received on its incoming interface. The random linear combination is proved to perform well enough for decoding. This type of network coding is usually referred to as intra-flow network coding because all coded packets are from a single flow, either unicast or multicast.

The scope of network coding is later extended to inter-flow coding where COPE [5] is the pioneering work. In this scheme, usually packets from different sources and to different destinations are coded together at intermediate nodes. Inter-flow network coding is gaining ground rapidly as a result of the

profusion of its applications. A network with multiple unicast flows is usually more commonly seen than a network with only one flow. COPE is a practical protocol for inter-flow network coding where each node promiscuously listens to all packet transmissions in its proximity. Intermediate nodes encode packets in an opportunistic manner. Downstream nodes can decode the coded packet with previously overheard packets.

The technique of inter-flow network coding is a good fit for applications in wireless ad hoc networks, due to their common concepts of *multi-hop transmissions* and *promiscuous listening*. Packet forwarding is required in network coding because coding can only happen at intermediate nodes. It is required in wireless ad hoc networks in order to deliver packets to far-away nodes. On the other hand, the broadcast nature of wireless transmissions enables nodes in a network to overhear other packets. Overheard information is mandatory in determining whether packets can be coded and how can the coded packet be decoded.

In this thesis, our key topic is on how to strategically adapt network coding techniques to the characteristics of wireless ad hoc networks. Many research challenges are encountered and corresponding solutions are proposed.

1.2 Research Challenges

Binding network coding techniques to the existing protocol stack for wireless ad hoc networks is not as easy as it may seem.

Network coding is usually applied in a sub-layer between Media Access Control (MAC) layer and network layer (as is done in COPE [5]). Network layer

protocols encapsulate application-related data into packets, determine the route to be taken, and then send them to output queues. Before contending for media access as the queue packets usually do, they are screened for coding opportunities in the sub-layer. If coding opportunities were found, multiple packets can be coded and the combined/coded packets are inserted into the output queue to replace the previously uncoded packets (native packets). After that, the legacy network transmission process resumes and MAC layer protocols take their usual course of action.

This approach is straight-forward and easy to implement. However, it is subject to many problems:

1. The way a node processes its packet backlog affects the network performance significantly. It is non-trivial to find an optimal coding-aware scheduling method to maximize throughput.

2. Network coding is applied only for two-hop flows. For coding structures that involve more than two hops, the disorganized opportunistic coding scheme is unable to discover additional coding opportunities.

3. There lacks a usable congestion control mechanism. In the typical 802.11 plus Transmission Control Protocol (TCP) protocol stack, congestion control is enforced by TCP's sliding window mechanism. However, this mechanism is no longer effective because network coding usually breaks the sequential packet order, which is nevertheless pivotal for TCP's congestion control.

To solve these problems, we aim to:

1. Find a packet scheduling algorithm that jointly considers per-flow fairness, overall throughput, and the configurability that balances fairness and efficiency.

2. Devise a coding-aware routing protocol that enables multi-hop inter-flow network coding. With a source-routing-based coding-aware routing protocol, the route with highest coding benefits can be chosen, and the two-hop restriction imposed on the opportunistic coding scheme can be overcome.

3. Design a cross-layer solution that jointly considers network conditions in the network layer and in the application layer. With the cross-layer approach, the application layer can be notified of the underlying coding opportunities as well as traffic congestion. Conversely, the network layer can be informed about future offered load, leading to better routing and scheduling decisions.

Though network coding has its strength in boosting throughput by reducing the total number of transmissions, we should evaluate its appropriateness when applied to wireless networks with a more comprehensive performance metric. The following questions should be properly addressed before we can confidently employ a coding-aware network protocol stack in wireless networks:

1. Is the protocol backward compatible? It is well understood that coding benefits are highly dependent on coding structures and traffic patterns.

Network coding is possible only when there *are* packets to encode, and when the encoded packet can be decoded at recipient nodes. In cases where the coding structure or traffic pattern does not encourage network coding, can we still properly handle the packets without severely degrading network performance?

2. When coding is possible, can the protocol promptly detect such possibilities and accurately evaluate the coding benefits against the coding costs? Wireless ad hoc networks are usually self-organizing and failure-prone. A coding-aware protocol should be able to detect coding opportunities regardless of the order of arrivals of the traffic flows and the topology changes in the network.

3. Despite the throughput gain, how is the protocol performing with respect to other performance metrics? Some Quality of Service (QoS) metrics should be evaluated, including but not limited to packet delay, fairness, etc. This is an area that is often overlooked. Nevertheless, we believe factors other than throughput gain are just as important in practical networks.

4. Is a traffic pattern suitable for network coding? If the traffic pattern inherently prohibits network coding, the improvement brought about by network coding would be minimal. On the other hand, if the traffic pattern encourages network coding, or if the traffic pattern can be adapted for coding schemes, the overall coding benefits can be magnified.

There are even more challenges that await solutions, like the Acknowledge-

ment packet (ACK) problem for broadcast packets, the hidden terminal problem, the problem of channel fading that results in decoding failure, etc. All of these challenges put pressure on designing a practical protocol that implements network coding in wireless networks. Yet we are gaining steady improvements as more and more researchers contribute to this course.

1.3 Thesis Contribution

The first problem solved in this thesis is the discovery of coding opportunities. It is observed that current solutions usually assume a certain order of flow arrivals. Only when certain flows start earlier than the others can the protocol detect coding opportunities. If the order is reversed, it is very likely those coding opportunities will be overlooked. If routes are selected based merely on information collected from a single flow, the routing protocol essentially achieves only a local optimum. If we are able to alter the routing decisions for other flows, we might end up getting a higher overall throughput.

Our work provides us with a means for intermediate nodes to sense the arrival of a new flow and to participate in the route-maintenance phase. In the route-maintenance phase, existing routes are reviewed with the aim of finding better routes. In Self recommendation Coding-Aware Routing Protocol (SCAR) we propose a series of procedures to conduct such maintenance. In addition to the procedures to conduct route maintenance, the routing metric is further revised to remove a bias. This metric essentially removes the bias that would have been incorporated in the routing metric favouring the currently-in-use route. With

the introduction of the route-maintenance dynamics, our coding-aware routing protocol is much more robust against different order of flow arrival.

The second problem we tackle is the problem of coding efficiency and fairness among flows. Existing protocols that utilize the power of network coding usually take the most straightforward means to do network-packet scheduling, namely round-robin scheduling. Whenever the MAC layer of one node succeeds in contending the media, the network layer sequentially polls each flow for packets to send. Before sending out the packet, the node searches the packet queue, looking for counterpart packets to be coded with. However, it is observed that such scheduling is far from optimum. On one hand, simple linear or ad hoc searching for one packet's coding counterpart is inefficient. There can be a more organized way of choosing a coding counterpart. On the other hand, the respective throughputs of the individual flows are unevenly distributed and severely biased as observed in simulations. Flows with more coding counterparts will be transmitted more frequently, while those "less popular" flows are starved.

In this thesis, our target is to maximize the minimum per-flow throughput first. When this target is fulfilled, there may be multiple choices. The solution that can maximize the average per-flow throughput is then selected. The reason we take this two-step maximization target is that we can demonstrate the importance of packet scheduling along the way. Simply maximizing the overall throughput would be undesirable as it suffers from starvation. In the meantime, maximizing overall throughput is comparably trivial in the context of packet scheduling, and one can easily adapt our derivation to achieve that.

The derivation of our solution is done in a step-by-step manner. First, by formulating the scheduling problem in a static setting, i.e., where flows are already predetermined, we transform the packet scheduling problem to a mathematical optimization problem. Though this problem is proved to be NP-hard, we propose a search-based algorithm that turns out to perform quite well. Our algorithm adopts a series of pruning rules to remove unnecessary search branches to reduce computational complexity. When no pruning rule is applicable, an approximation method is used to keep the complexity under control. The solution to this static form of scheduling problem is then extended to dynamic cases where we assume Poisson arrivals. A heuristic algorithm is adopted to adapt the static solution to the new settings. After this, we further relax our assumption of the Poisson arrival, and test an adjusted version of the heuristic algorithm in wireless network simulations. Results show that our method can significantly reduce the variance of throughput distribution among flows. At the same time, the overall throughput is mostly retained. In fact, with the more organized scheduling algorithm that we proposed, we demonstrate a network can perform better before reaching saturation.

Moving beyond applying network coding in the network layer alone, we demonstrate its use in wireless P2P content distribution. A cross-layer approach is taken, in the hope of exploiting a synergistic relationship between the network layer and the application layer. In fact, the network layer can make better routing decisions if it is given the offered load information from the application layer. It can organize packet scheduling in a more informed way as well. Likewise, the

application layer can choose one node's peer neighbours more wisely and control the offered load to avoid congestion based on the knowledge of the network state.

The contribution of this thesis is mainly three-fold.

Firstly, we overcome coding-aware routing protocols' susceptibility to traffic patterns. Our routing protocol is equipped with the tool to discover potential coding opportunities even after routes have been established. Secondly, we offer a new powerful tool, coding-aware network-layer packet scheduling, for better control over per-flow performances. Lastly, a full-fledged coding-aware APP-cum-NETWORK layer protocol is devised, creating a synergy that can benefit the overall performance.

1.4 Thesis Organization

In Chapter 2, we give a thorough review of literatures in the domain of wireless ad hoc networks, network coding, and P2P content distribution systems. The problem of coding-aware routing protocols' susceptibility to the flow arrival dynamics is addressed in Chapter 3. We then further analyse the problem of packet scheduling in Chapter 4. A new cross-layer solution is proposed in Chapter 5 where all our efforts culminate at a full-fledge protocol set. Finally we conclude and envision possible future directions in Chapter 6.

Chapter 2

Related Works

2.1 Overview of Network Coding

Network coding is a research area that may have interesting applications in practical networks. It can improve throughput when two wireless nodes communicate via a common intermediate relay node. Works on network coding started with a pioneering paper by Ahlswede et al. [3], who showed that with network coding, as symbol size approaches infinity, a source can multicast information at a theoretically maximum rate. This rate can be calculated by modelling the network as a max-flow min-cut problem in graph theory. The the max-flow min-cut optimization theorem states that in a flow network, the maximum amount of flow from the source to the sink is equal to the minimum capacity that, when removed in a specific way from the network results in no flow from the source to the sink. Network coding can be seen as a generalized routing scheme in which the intermediate nodes are allowed to mix data from multiple incoming links be-

fore sending the mixed data onto the outgoing links. This is an obvious contrast to the popular store-and-forward routing scheme implemented in most current networks where the intermediate nodes just simply forward the received data. This new transmission paradigm offers several benefits, including improvements in throughput, network reliability and robustness. It is significant that it has been shown that multicast capacity can be achieved for many networks by using network coding. Li et al. [4] and Chou et al. [6] mainly focus on addressing the efficiency of network coding. Lun et al. [7] and Ramamoorthy et al. [8] mainly study on the efficiency of network coding in multi-hop wireless networks. In recent years, there have been works [5, 9, 10, 11, 12, 13, 14, 15] on multiple unicast sessions by applying wireless network coding. These approaches are categorized as inter-flow network coding, which encodes multiple packets destined to different next hops and broadcasts them together. In addition, in [16], an algebraic framework for studying capacity in arbitrary networks and robust networking using linear codes are presented. The authors have provided necessary and sufficient conditions for the feasibility of any set of connections over a given network. Specifically, a connection between the solutions to network problems and the underlying mathematical theory has been made.

Network coding techniques have also been widely used to improve throughput, energy efficiency, and robustness of wireless networks. In particular, Nguyen et al. [17] have proposed XOR-based network coding for wireless broadcast networks. They have shown that the source can reduce retransmissions by combining lost packets together before broadcasting them to all receivers. Also, in

[18], it is shown that network throughput efficiency can be improved by up to 3.5 times over the traditional Automatic Repeat-reQuest (ARQ) approach with joint network and channel coding. Additionally, the authors have shown that network coding can be used to improve network throughput and media quality by using the Markov Decision Process (MDP) [19] and Deterministic Markov Decision Process (DMDP) [20]. Furthermore, the network throughput region can be substantially enlarged in prioritized transmission scenarios consisting of an oracle source and multiple wireless users [21]. In [7], it has been shown that with a linear optimization function with a distributed solution, the problem of minimum-energy multicast in lossless wireless networks with omnidirectional antennas can be approximated. Furthermore, it has been shown in [22, 23, 24, 25, 26] that network coding can be utilized to reduce energy consumption in wireless ad hoc networks. In particular, it is shown that by utilizing linear network coding for multi-hop wireless networks, minimum energy-per-bit can be achieved. Specifically, a polynomial time algorithm exists, in contrast to the NP-hard problem of constructing minimum-energy multicast tree when store-and-forward routing method is used.

One-hop coding opportunities were first studied and exploited by the COPE protocol in [5]. The simple one-hop nature and the empirical success of COPE have since motivated numerous subsequent works. Some examples include: [27], which finds the energy-efficient scheduling with opportunistic coding, [13], which calculates the maximum number of overhearing opportunities under practical wireless settings. [28, 29, 30, 31] develop techniques to select routes that create

more coding opportunities, [32, 33, 34, 35] jointly optimize network coding and scheduling, [36] picks the modulation rate that takes into account both coding gain and data rate, and [37] proposes a technique to XOR packets that use different modulation schemes. Some recent efforts considered cross-layer approaches in the context of coding-aware routing [29].

The benefits of network coding in multi-hop wireless networks have been further studied. The main idea in this research theme is to exploit the natural propagation of wireless signals; thus, in a single transmission, several neighbouring nodes in the vicinity of a sender can receive the transmitted data. In [38], a wireless data exchange example has been demonstrated in which the relay node utilizes network coding to reduce the number of transmissions. Moreover, in [5], a practical network coding system, namely, COPE, has been demonstrated in a wireless ad hoc network to improve network throughput. It has been shown that network coding can offer several-fold gain in throughput, by allowing the nodes to snoop on the medium, learn the neighbours' status, and detect coding opportunities. Additionally, the transmission delay with network coding has also been investigated [39].

Topologies are highly dynamic in wireless ad hoc networks. As a result, a network protocol works very well if only it operates in a distributed manner. Protocol design is difficult for traditional store-and-forward routing in a distributed manner. In [40], experimental results have shown that the network throughput gain depends on the network traffic patterns. Hence, the network protocol utilizing network coding needs to have a mechanism to control the information

flows to maximize the coding opportunities and to work efficiently. To realize network coding in practice, we must always pay attention to the overhead of data exchange and the complexity of scheduling protocols for coordinating communication between the intermediate nodes in a network. In 2006, a seminal work, namely, Random Network Coding (RNC) is proposed in [41]. The main idea of random network coding is to allow the intermediate nodes to select independently and randomly linear mappings from their incoming links onto outgoing links over some large finite field. It is shown that RNC can help achieve the theoretical network capacity with high probability.

Chaporkar and Proutiere [32] also studied the issue of joint scheduling and COPE-like coding, focusing on characterizing the capacity region of a simplified version of COPE combined with scheduling according to back pressure. In [28], Le et al. proposed an on-demand Distributed Coding-aware Routing (DCAR) for selecting a high throughput path with more potential coding opportunities by introducing a coding-aware routing metric called Coding-aware Routing Metric (CRM), which jointly considers coding opportunities, congestion levels, and related factors for comparing coding-possible and coding-impossible routes when multiple routes are available. In addition, DCAR can detect coding opportunities on an entire path, thus eliminating the two-hop coding limitation in COPE. However, both of these existing coding-aware routing protocols do not consider the dynamic nature of data flows in a network.

2.1.1 Robustness in wireless ad hoc networks

In recent years, a number of routing protocols have been designed for wireless ad hoc networks. Many routing issues from different aspects are addressed, for example, energy efficiency, reliability, robustness, security and data fusion. In this subsection, we will mainly discuss these aspects on existing routing protocols. Some of the routing protocols designed for general multi-hop wireless networks are also very useful for wireless ad hoc networks, so we will also include them in the investigation.

2.1.1.1 Routing Structure

Generally, routing structure for wireless ad hoc networks can be classified as flat, hierarchical or location-based. For the flat routing structure, all the nodes have equal status in routing functionality. For example, the routing protocol Directed Diffusion proposed in [42] is a flat routing protocol. For hierarchical-based routing protocols, different nodes may have different roles, where some of them are used for local routing while others are used for global routing. For example, the routing protocols proposed in [43, 44, 45] are hierarchical-based. In the case of location-based routing, geographical information is always needed in forming the routing structure such as the one proposed in [46].

2.1.1.2 Path Selection

Path selection is an important issue for all routing protocols. There exist many path selection criteria, for example, the hop count, transmission costs and signal

strength. Hop count is the most widely used path selection criterion both in wired and wireless networks. Protocols such as Ad-Hoc On-demand Distance Vector (AODV) [47], Dynamic Source Routing (DSR) [48], and Optimal Link State Routing (OLSR) [49] all use it as the path selection criterion. This criterion is also very popular in wireless ad hoc networks. However, De Couto et al. [50] proposed that using hop count as the only criterion in path selection is not good enough. This is mainly because wireless links are not just of perfect or bad quality. Many links have intermediate qualities [51] instead. The link quality mentioned here is referred to the capability of a link, how fast it can deliver a packet to the next node in one transmission. For a random loss channel, this quality can be valued by a number, such as 0.6, which means that when a packet is transmitted over the link, the probability of successfully receiving this packet is 0.6. The links with intermediate quality have significant influence on the reliability of data transmission, especially when the number of retransmissions allowed per hop is limited. Since link quality information has not been included in the hop count, end-to-end routing reliability degrades significantly when poor quality links are chosen for data forwarding.

2.2 Network Coding for Scheduling Problem

2.2.1 Overheads in Network Coding

In network coding, in order to carry out successful encoding and decoding of packets, network nodes need to exchange information with each other. However,

this can lead to high network overhead which can significantly degrade network performance. This is due to the fact that overhead introduces additional congestion, delay, inefficient bandwidth utilization and energy consumption. Research studies show that there are ways by which this overhead can be minimized.

A buffer model is proposed in [6], which employs traditional generation-based network coding. It involves grouping packets according to generations and linear combination of packets that belong to the same generation. The idea is to minimize or limit packet overhead generated by the coefficient vector at the expense of coding opportunities. This approach is susceptible to packet loss which is partially mitigated by flushing the old generations and this leads to some performance improvements. However, discarding packets at the receiver is inefficient as the bandwidth consumed is wasted unproductively.

A better approach is proposed in [52], which employs Multi-Generation Mixing and eliminates the need to increase buffer size while improving performance. In this scheme, packets are grouped into generations and generations are grouped into mixing sets. Packets belonging to a new generation can be linearly combined with those from old generations. This ensures that the receivers which may not have received full-rank matrix to perform decoding on the previously received coded packets could use information in newly arriving coded packets to perform decoding and thus guaranteeing reliability and reducing overhead. Even though this approach is better than the former one, it may still experience performance deterioration due to increased overhead when the network traffic is so high that each generation contains quantities of packets to be sent out. Kim [53] presented

the concatenated random parity forwarding technique that enables network coding gain to increase with each additional hop in wireless networks. This scheme employs partial network coding in which a relay node combines and sends out parity bits (instead of entire codewords) received from neighbouring nodes. The destination node then mixes information bits from direct path from the source nodes, and parity bits from the relay nodes to improve reliability and lessen overhead. Nonetheless, the main difficulty for this scheme is that additional control information overhead may be required to coordinate relay encodings.

Jin et al. [54] studied an adaptive random network coding in WiMAX by taking into consideration the requirements of a number of upstream nodes and dynamically modifying block size in reaction to channel conditions. In their scheme, a downstream node prematurely sends feedback to a subset of upstream nodes to stop transmission in order to minimize unnecessary blocks and thus lessening protocol overhead. In order to effectively handle the overhead, a new encoding strategy, XOR-TOP, which employs local topology to effectively measure the available non-coded or native packets at the neighbouring nodes, is introduced in [55]. They claim that their scheme, which does not employ information exchange among neighbouring nodes, can always accurately identify coding opportunities according to the local topology. However, their claim on accurate identification of coding opportunities is questionable as wireless link conditions are unpredictable. Therefore, there is a chance of making an error in estimating available native packets at neighbouring nodes. This is especially true in a high-mobility environment in which the local network topology frequently

changes. Although reliability is crucial, the idea of acknowledging each and every packet sent (as in [55]) can degrade performance because of the additional overhead introduced especially in a network with high TCP traffic in which the MAC and the transport layer performs retransmissions.

Katti et al. [5] employed bit-map format in the packet's reception report to report packets which have been overheard by a node. Even though the representation for reception reports in their approach has two advantages of compactness and effectiveness, their approach of handling overheard packets has some shortcomings. Consider a network with high traffic. According to their approach which employs fixed packet waiting time, there is high probability of many packets being overheard by nodes. Therefore, packet overhead is more likely to be high due to long reception reports thereby degrading network performance. Furthermore, since more packets will be overheard, more memory will be required at each node. Also, high power consumption will be required to transmit packets with high overhead thereby increasing cost.

2.2.2 MAC-Layer and Physical-Layer Scheduling

Network coding is a great fit for wireless networks. Fragouli et al. [56] study the case of applying network coding in wireless networks in many aspects. These include throughput, reliability, mobility and management. Several key challenges are proposed as well.

There are many researchers working on the topic of scheduling in network coding settings. However, most of them focus on MAC-layer scheduling which

usually gives "higher priority" to coding nodes. [57, 58, 59, 60, 61, 62] are examples of this kind.

Yomo and Popovski [63] propose an opportunistic scheduling algorithm. The links between the intermediate nodes and receiver nodes are subject to channel fading. When this fading is time-varying, the authors propose an opportunistic network coding scheme to determine how many and which packets are coded. It is shown that such an opportunistic scheme can provide higher throughput than any coding scheme that involves a fixed number of packets.

Ni et al. [64] discuss the optimal physical-layer transmission rate in the Massachusetts Institute of Technology (MIT) *Roofnet* platform. The basic idea is that lower rate gives rise to longer transmission range or higher delivery ratio. Furthermore, the distances from the intermediate node to the two recipients may not be the same. So their delivery ratios would be different. In 802.11b, if 5.5Mbps or 11Mbps is used, the total Expected Coded Transmission Time (ECT) can be different.

Most network-coding schemes are agnostic about Physical layer (PHY) and MAC layer. They assume a greedy algorithm to enforce network coding. Chaporkar and Proutiere [32] advocate that exploiting every opportunity to enforce network coding may downgrade the overall throughput of the system. The hypothesis is demonstrated using a scenario where interference is incorporated into the analysis. Hence, a specially tailored MAC-layer scheduling algorithm is proposed in order to achieve the expected throughput gain.

Zhao and Yang [65] consider the joint scheduling problem for Multiple-Input–

Multiple-Output (MIMO) and network coding (MIMO-NC) in wireless networks. A network in which each node is equipped with two antennas is analysed. In order to achieve higher throughput, the authors propose a packet scheduling algorithm which in essence matches all source-destination flows into pairs. When a pair is discovered, MIMO-NC is applied to reduce the number of transmissions. However, this approach requires full knowledge of the network and traffic patterns. In addition, it is highly dependent on the traffic pattern. When the prescribed pair is not available, the benefits are gone leaving only cost of building a MIMO system.

2.3 Content Distribution in Wireless Ad Hoc Networks with Network Coding

2.3.1 Overview of Cross-layer Designs

In recent years, cross-layer design methods are widely used for performance optimization in wireless ad hoc and sensor networks ([66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77]). In [66], cross-layer design is defined as a simple violation of layered communication architecture. In many different ways, cross-layer interactions can occur, either by merging adjacent layers, or by creating new interfaces between adjacent layers or by a shared database between the layers.

In [78], a cross-layer adaptation GRACE is proposed for energy conservation and QoS improvement in mobile multimedia terminals. GRACE is an adaptation framework. It performs only local adaptation by interfacing system layers to a

central resource manager which acts as a coordinator.

Optimal results are achieved by the mediation between the layers of the manager to yield a proper combination of configurations for each layer. Although several cross-layer interactions are currently studied, there is limited work yet in exploring the cross-layer design in the realms of network coding.

Cross-layer design is based on the cooperation of different protocol layers. For example, Madan et al. [79] propose joint optimization over transmit powers, rates, and link schedules of wireless ad hoc networks to maximize lifetime. Constraints for the lifetime are flow conservation, maximum rate, energy conservation and transmission range. For a given incidence matrix of the network graph, link gain matrix and initial energy, convex optimization is used to solve for the optimal rates and powers. However, the computational complexity of the algorithm grows as a double exponential function of the size of the network. Meanwhile, in real-time applications, it is very difficult to obtain the inputs that are necessary for the algorithm.

The layering approach derived from the modelling, design and implementation of wired networks is very strict, so it is one of the major limitations in addressing the challenges mentioned in 2.1 in the wireless networking system context. Traditionally, the protocol stack is divided into layers similar to the Open System Interconnect (OSI) Architecture, for example in the well-known Transmission Control Protocol / Internet Protocol (TCP/IP) protocol suite. In the layered system architecture, each layer performs its own specific functions, such as routing or end-to-end flow control, strictly within the knowledge of the

same layer. Information exchange between adjacent layers is limited to the flow of data packets but nothing else. In the original context, it is an excellent approach because it enforces modular design and development of the operating systems and simplifies protocol design and implementation.

However, in the wireless ad hoc networks, relying merely on the information available within a particular layer is no longer sufficient to provide the desired QoS, or implement novel system or application services made possible by wireless communications. For example, the Data Link layer, whether it is link capacity or loss resiliency, cannot choose the appropriate data rate to improve the performance of a wireless link, without the feedback of wireless channel characteristics from the physical radio. Without such kind of information, transport and application layers cannot adapt (or react) to the additional physical-layer time-varying constraints such as highly variant jitter or loss, besides the traditional congestion control. For another example, without leveraging application service information such as localization service, the network layers and below cannot optimize the coordination of transmissions and traffic flows, either to reduce the traffic-induced interference or implement novel services based on location and other environmental information. As a third example, without the direct knowledge of application characteristics, the MAC and PHY layers cannot adjust their behaviour (such as transmission power, data rate and adaptive modulation schemes) according to the relative priority of individual traffic flows or even individual packets, to provide better service or simply to conserve energy.

2.3.2 Content distribution using BitTorrent-like protocols

BitTorrent (BT) [80] is a pioneering work and a prevalent protocol used in P2P content distribution. Many other P2P protocols are designed on top of it. [81] considers applying network coding techniques in multicast P2P networks. The authors base their analysis on a star network topology and conclude that there is no significant benefit applying network coding in that topology. However, the authors point out that non-multicast network models can lead to new results.

Avalanche [82] utilizes the power of network coding in an intra-flow manner. In this system, an original data file is divided into n pieces and before sending out these native pieces, each peer encodes them using random linear network coding. Receivers, upon receiving enough coded pieces, can decode to get the native pieces. Other related works on network coding-based wireless content distribution systems are described in [83, 84, 85, 86].

[87] is among the first to propose a cross-layer solution applying P2P techniques in wireless ad hoc networks. However, we will show that a straightforward implementation of P2P on an ad hoc network is not satisfactory in terms of overhead and overlay connectivity. [88] investigates the impact of practical resource constraints of mobile devices (namely disk access, computation overhead, memory constraints, and wireless bandwidth) on the performance of content distribution using network coding in Vehicular Ad Hoc Network (VANET)s. After validating its model against real experiments, the authors design a novel data pulling strategy to reduce the overall delay of content distribution.

Chapter 3

Improving the Robustness of Coding-aware Routing Protocols to Flow Arrivals

3.1 Introduction

Coding-aware routing protocols [29] are recently gaining popularity as a result of their seamless integration with existing protocol stacks and their superior throughput gain in multiple-unicast scenarios. Practitioners do not need to know the topology or to schedule nodes' transmission sequence prior to deployment. Yet still the coding-aware routing protocol is able to detect coding opportunities and route flows in a way such that coding benefits could be exploited.

Coding-aware routing protocols are generally on-demand source-routing-based. Coding opportunities are extremely dependent on network traffic patterns. Cod-

ing benefits can be realised only when coding flow pairs have sufficient and comparable amount of packets to send. In this sense, it is impractical to plan all possible routes before the arrival of flows. The *on-demand* character therefore suits coding-aware routing protocols well. From another perspective, coding opportunities can be voided if any participating flows took a different route. It is possible that the flow does not route through the coding node at all. It is also possible that the flow slightly detours and some destination nodes of the participating flows fail to overhear the packets from this flow. Those destination nodes are then not able to decode the coded packet and the bandwidth used to transmit the coded packet is wasted. In order to avoid such situation, routing decisions are put on the level of flow, instead of segments, nor per-hop basis. That is why source routing techniques are utilized where the whole route is determined even before the first hop of the packet is traversed.

Coding-aware routing protocols should be able to fairly evaluate the benefit that network coding brings as well as the drawbacks of the higher interference when routing flows closer to each other. DCAR [28] proposes a routing protocol that thoroughly weighs the factors that need to be considered. It proposes a CRM to quantify and compare the merits between coding-possible and coding-impossible routes. This metric is based on the queue length at the network layer. A longer queue length implies heavier traffic, higher channel occupation rate, and higher interference. Queue length is then modified with coding benefits considered. If packets from one flow can be coded with another one, DCAR considers the free-ride effect by adding the queue length only once for both flows. The

resultant metric can then by large reflect the expected number of transmission slots that one new arriving packet should wait before it gets transmitted.

Although current coding-aware routing protocols, like DCAR, are able to discover coding-possible routes, they lack a mechanism to conduct coordination among different flows. In this thesis, the term *flow* refers to the source-destination pair where packets are *flowing* from the source node towards the destination node. If the route is selected merely based on information collected from a single flow, this routing decision essentially achieves only a local optimum. If we alter routing decisions for other flows, we might end up with a higher overall throughput. Our work provides us with a means for intermediate nodes to sense the arrival of new flows and to participate in the route-maintenance phase. This is called Self Recommendation (SR) while our protocol is named as SCAR. SR enables a re-assessment of prior routing decisions and includes a procedure to update it. Consider the scenario shown in Fig 3.1 where each node can only reach its nearest neighbour. Suppose there is an existing flow 1→4→2. With the arrival of a new flow 6→3→5, new coding opportunities are introduced at node 3. Node 3 can sense this new opportunity as an intermediate node. Without the mechanism of self recommendation, node 1 will not be aware of the flow arrival, not to mention changing the route to utilize this coding opportunity. Our protocol specifies when intermediate nodes will send self recommendations, and how source nodes will compare the available routes. In our protocol, node 1 will re-route via 1→3→2, fully utilizing the coding opportunity.

Consider a wireless network that evolves along time. There can be multiple

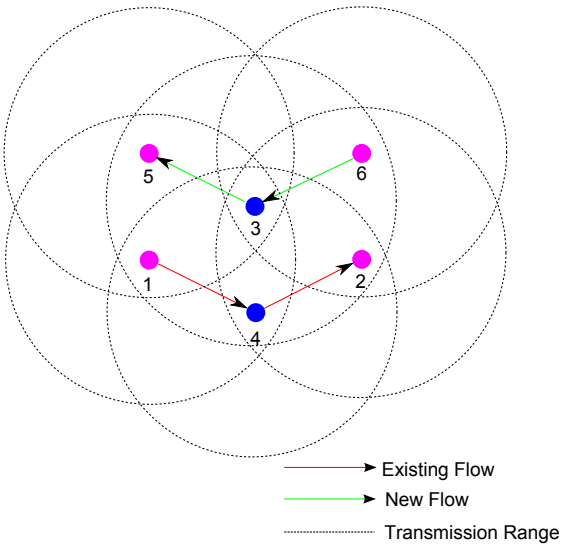


Figure 3.1: A Simple Test Scenario for SCAR

events that occur to the network, like the arrival and departure of flows, the traffic load changes, and possibly node movements. The ultimate routing decisions for all flows can be completely different if the sequence of these events is altered. Their overall performances can differ greatly as well. This is because existing routing protocols consider only information that is obtainable at the time when flows joined the network. After that, no further route maintenances are done. In our work, the *self recommendation* enables a new mechanism to revise existing route decisions. Such a revision is non-trivial mainly because of two reasons. First, the routing metric should be modified to remove a bias brought in by the currently-in-use route, which we term as *in-use-path bias*. The static Modified Queue-length (MQ)s of the nodes in the currently-in-use route are overestimated but their dynamic MQs are fair. This would inflate the Modified Interference Queue Length (MIQ) of those neighbouring nodes while keeping the MIQ calculation for the currently-in-use route correct (MIQ is a concept proposed in

DCAR [28], we will produce a digest of that paper in Section 3.2.1). This essentially puts all other routes, except for the currently-in-use route, at a competitive disadvantage. In our work, this bias is removed by making adjustment on the biased metric, and thus we term it *unbiased metric* in this work. The second reason why this revision is non-trivial is that, new routes should be tested before the final adoption. Route changes often involve more than one route. In the simplest setting where only one route is subject to change, the route selected according to the unbiased CRM would generally be better than any other routes. But this may not be true if any other routes underwent the same Route-Change procedure. We introduce a probation period to test the validity of such route changes.

We abstracted the above described scenario into several problems. The first is how to discover the arrival of new flows and new coding opportunities. Secondly, we need to know how to evaluate these opportunities and decide whether to change existing routes. To complete the work, we also need a third step to test whether the newly introduced complexity is well compensated. Here we first briefly answer these questions. The network dynamics are monitored by intermediate nodes. Like most routing protocols, our protocol also employs a Route REQuest Packet (RREQ)-Route REsPonse Packet (RREP) procedure to discover routes. The benchmark protocol is the DSR [89] protocol and parts of the DCAR protocol are also included. In an on-demand routing protocol, RREQ/RREP packets suggest new incoming flows. This discovered information is encoded in a SR packet and is sent to the source node, triggering a

Route-Change (RC) procedure to handle it. The Route-Change procedure employs a new unbiased routing metric to evaluate all possible routes, and makes updates to the routing table. We stress again here that "unbiased" is termed against the in-use-path bias. We will come back to this in Section 3.3.2. On the other hand, we have carefully analysed the causes for the throughput gain, and devised a series of indicators to predict the gain. Through extensive simulations on various topologies, we conclude empirically that throughput gains are observed whenever the indicators imply so, thus strengthening their applicability.

The contributions of this work are:

1. We have proposed a practical coding-aware routing protocol that enables coordination among flows. This coordination is done with the help of Self Recommendations (SRs) from intermediate nodes. The Route-Change (RC) procedure is devised to synchronize different nodes.

2. We have studied the Coding-aware Routing Metric (CRM) for quantifying the merits of candidate routes. The concepts of biased and unbiased CRM, as well as the methods to convert biased CRM to unbiased CRM, are introduced. We design packet formats and a route-maintenance procedure to gather necessary information to make the conversion possible.

3. We have analysed various coding structures and propose a series of indicators. These indicators can be used to estimate how much throughput gain is achievable under SCAR. They can also be used as a guide for modifying network topologies to improve throughput.

Compared to previous coding-aware routing protocols [90, 91, 92, 93], the performance evaluation of our protocol shows that our protocol can significantly improve throughput in many network topologies. It is also shown that our proposed protocol is robust against different traffic patterns. Our protocol is indifferent to the order of arrivals of flows.

The rest of the chapter is organized as such: Section 3.2 gives an overview of our SCAR protocol, with Section 3.3 detailing the Route-Change procedure in our protocol. Section 3.4 analyses coding structures and proposes a series of indicators to estimate throughput gain. The performance evaluation is given in Section 3.5 and we conclude in Section 3.6.

3.2 Protocol Overview

In the following text we explain the major components of our SCAR protocol. In order to fully understand and appreciate the rationale and the methods we took in designing the protocol, a digest of the DCAR paper [28] is provided for a quick reference.

3.2.1 Digest of the DCAR protocol

The major contribution of DCAR is the design of the CRM. To better appreciate the extension we have done in this thesis, we will describe the derivation of DCAR's CRM first.

The choice of routing metric is pivotal in the design of a routing protocol. A good coding-aware routing metric should quantify the merits between

coding-possible and coding-impossible routes. DCAR reviewed existing coding metric including Hop-count-based routing metric, Expected Transmission Count (ETX)-based routing metric, and load-based routing metric. The proposed metric in the DCAR paper collectively considered load information, packet loss rate, and coding benefits.

The evolution of DCAR's CRM starts from the packet queue length. Larger queue length indicates more packets to send, higher delays and lower per-flow throughput. The queue lengths are averaged in order to remove spikes.

Queue length itself does not consider the coding benefits. Or more precisely, the free-ride benefit is ignored so the queue length overestimates the actual cost of choosing a particular route. To account for the coding benefit, queue lengths are modified to arrive at MQ. Two cases of MQs are considered—the stable case and the dynamic case. DCAR uses the stable MQ (cf. Algorithm 1) to estimate the overall level of busyness of a node, and uses the dynamic MQ to estimate the level of busyness of a node seen by a newly-arriving flow (cf. Algorithm 2). In the pseudo-code of the algorithms, $N(v)$ denotes the neighbour set of vertex v. Subtracting a vertex set from another vertex set yields the difference set of vertices. Subtracting a vertex set from a graph, on the other hand, is removing all vertices in the vertex set and removing all edges that connect to the vertices in the vertex set. Note that the calculation of static modified queue length involves two randomness. The first is the random selection of v. The other is the random selection of maximal clique that contains v. So different runs of Algorithm 1 can yield different results. The calculation of modified queue length is not optimal

in any sense. We will come back to this statement in next chapter as we will be revising this algorithm.

```
1 Function Static-Modified-Queue-Length(G)
    Input: A weighted graph G = (V, E, w(·))
    Output: MQ_s
2   MQ_s ← 0;
3   repeat
4     Randomly select v from V;
5     V' ← the vertex set of a maximal clique that contains v;
6     MQ_s ← MQ_s + max_{i∈V'}{w(i)};
7     V ← V − V';
8   until V is empty;
9   return MQ_s;
```

Algorithm 1: Calculate Static Modified Queue Length in DCAR

```
1 Function Dynamic-Modified-Queue-Length(G, v)
    Input: A weighted graph G = (V, E, w(·)), the vertex that denotes
           the newly-arriving flow v
    Output: MQ_d
2   V' ← {v} ∪ N(v);
3   G' ← G − V';
4   return Static-Modified-Queue-Length (G');
```

Algorithm 2: Calculate Dynamic Modified Queue Length in DCAR

The modified queue length of a node, however, is not sufficient to estimate its available bandwidth in the wireless network either. A node with very short queue length can still be congested if its interfering nodes have a lot of packets to send. MIQ is calculated to address this factor. Let $I(c)$ denote the set of node c's interfering nodes, the MIQ of node c is defined as:

$$MIQ(c) = MQ_d(c) + \sum_{i \in I(c)} MQ_s(i) \tag{3.1}$$

The last factor to consider is packet loss rate. The CRM metric of link l is

calculated as:

$$CRM_l = \frac{1 + MIQ(S(l))}{1 - P_l} \tag{3.2}$$

where P_l denotes the packet loss rate on link l, $S(l)$ denotes the source node of link l. CRM_l estimates the expected number of transmissions to successfully transmit the existing packets as well as one incoming packet for the new flow. The routing metric of a given route can then be calculated as the sum of CRMs for all links in this route:

$$CRM_L = \sum_{l \in L} CRM_l \tag{3.3}$$

Another contribution of DCAR is the formulation of a distributed "Coding+Routing" procedure. In this procedure, enough information can be gathered so that CRM can be calculated at the source node upon receiving all RREPs.

For each node a in a wireless network, it maintains a list of one-hop neighbours, denoted by $N(a)$. It also keeps track of its packet loss probability on all of its *outgoing* links, denoted by $P(a, b)$ where b is one of its neighbour node. When a new flow arrives to the network, the source node initiates the following steps to discover possible routes for the flow:

Step 1. The source node a broadcasts a RREQ packet. RREQ packet contains following information:

- A list of "who-can-overhear" nodes. For now, this is essentially the list of one-hop neighbours for node a.

- The path that it has traversed, as is done in any other source routing protocols.

Step 2. Upon receiving an RREQ, an intermediate node c first detect if a loop in traversed path exists. If so, the RREQ packet is discarded. If not, node c then:

- Temporarily stores the RREQ for future use.
- Updates the "who-can-overhear" list by appending its own one-hop neighbours to it.
- Node c then continues to broadcast the updated version of RREQ.

Step 3. When an RREQ arrives at the destination node b, node b then replies with an RREP packet with a CRM value of zero, using the reversed path back to source node a.

Step 4. Upon receiving an RREP, an intermediate node c first extracts the path from RREP and the "who-can-overhear" information from previously temporarily stored RREQ. Combining the available information, node c is able to determine whether this new arriving flow is coding possible with existing flows that passes by node c. Thus, node c can draw the coding graph from which the dynamic MQ can be calculated. Adding the MQ from neighbouring nodes to this dynamic MQ, node c gets the MIQ value. It then adds the MIQ to the CRM value of the RREP packet and send it to next hop.

Step 5. Upon receiving multiple RREPs from different routes, the source node a is now able to compare the CRM values across different routes. The route

with the smallest CRM value is chosen and this concludes the route discovery procedure.

3.2.2 Two tables to store flow information

We continue our work from here after describing the DCAR routing metric.

The route-discovery procedure in DCAR [28] has already collected enough information for future routing coordination, but DCAR simply dumps it after the routes are constructed. In the SCAR protocol, each node maintains two tables: potential-flow table and relayed-flow table.

When an RREP passes by a node, it will store the route information in the potential-flow table. A potential-flow-table entry can be moved to the relayed-flow table when data packets are received from this route. It will be moved back if the node fails to receive data packets from this route for a period of time. In a traditional coding-aware routing protocol, a potential-flow-table entry is only used for calculating the coding-aware metric when forwarding RREP back to the source. In our protocol, this potential-flow information is stored for a longer time, so that this intermediate node can possibly recommend to the source node to use this route in the future. Obviously, a route can be either in the potential- or the relayed-flow table, but cannot be in both simultaneously. SRs would only be done for potential flows (because relayed flows are already chosen by their sources).

3.2.3 Self recommendations

With two tables storing the necessary information for making SRs, the next question is when to initiate the SR procedure. The SR procedure should be triggered in two situations. The first is when a new flow joins the network and a potential coding opportunity is found. The second is when an existing flow is terminated. In either case, the source should also be notified of the change and the other routes should be updated in accord with the new flow pattern.

In the first case, upon receiving an RREP of flow f_n, an intermediate node will check whether this new flow can be coded with any of the potential flows it has stored. If a potential flow f_p can be coded with f_n, this node can send a SR to the source node of flow f_p "later", indicating a potential coding opportunity. It should be sent "later" because we need to wait for this new flow f_n to stabilize. If after a delay, this new flow f_n is moved to the relayed-flow table, we can safely recommend f_p to the source of f_n. Otherwise, f_n remains in the potential-flow table. This indicates that the source of f_n has found a better route than the one via this intermediate node. Without this delay mechanism, we will have to send out two recommendations, both to the source node of f_n and f_p, to utilize the coding opportunity. This can cause turbulence to the network because more flows are involved in such a route change. For now, our protocol will send a SR only when f_n is later moved to the relayed-flow table in order to reduce the overhead.

In the second case, when an intermediate node detects the queue length for a certain relayed flow f_1 is drastically decreasing, and this decrease has continued

for some time, it then checks whether there is any other existing flow f_2 being coded with f_1. If so, the source of f_2 will be notified of the change by an SR packet, and its route should be updated.

3.2.4 Handling Self Recommendations

Not all recommendations should be accepted. The source retains its autonomy in determining which route to use. In order to compare this recommended route with other existing routes, the source node will send Modified RREQ Packet (MRQ)s to update the CRM of all existing routes as well as the recommended route. Unlike RREQ, modified RREQs are unicast packets that use source routing. This modified RREQ will request the destination and relay nodes to append the CRM value in their reply packets, Modified RREP Packet (MRP)s.

3.2.5 Decision Making

Upon receiving the MRP, the source node updates the CRM values of all routes. Ideally, the source node can compare these routes and choose the route with the minimum CRM. However, the CRM values received are biased. They will favour the currently-in-use route most of the time. In the next section we will discuss how to remove this bias and improve the effectiveness of SRs.

Altering the routing decision is not the end of the story. The routing decision we have just made can be wrong sometimes because of two reasons:

1. The CRM metric proposed in [28] is only a heuristic indicator for the suitability of a route being used in the context of network coding. There

is no guarantee that using a route with higher CRM will result in a lower throughput.

2. The CRM values calculated can be imprecise. The max-clique problem inherent in the CRM calculation is NP-complete. In real deployments, a greedy algorithm is usually used instead of a precise solver.

In order to discover and recover from these errors, each recommended route is given a probation period. After its probation, the source node will refresh its routing table through the modified RREQ/RREP procedure again. Monitoring this routing table refresh can give us hints as to whether recommending that route is a right decision.

3.2.6 Controlling the Frequency of Self Recommendations

For a particular flow, there might be several intermediate nodes that will send SRs to the source node. An intermediate node may also send multiple SRs for the same potential flow. Thus the source will constantly be flooded with requests to update. To prevent this from happening, a way to control the frequency of SRs is much needed. This control is done in two ways:

1. After an intermediate node sends a SR for a certain path, it is blocked from sending this specific SR for a period of time.

2. Whenever a source receives a new SR for a particular destination, the Route-Change procedure is initiated. Subsequent SRs for the same destination will be taken care of, but will not trigger sending of modified RREQ

again. Details for the RC procedure are explained in the next section.

3.3 Route-Change Procedure

This section describes how SR packets can alter existing routes. The route changes happen in a distributed but ordered manner.

3.3.1 Procedure Timeline

Why is a timeline important? In the simple scenario shown in Fig 3.1, synchronization is not a must. However, as the topology incorporates more nodes and the network accommodates more flows, modified RREPs may not immediately follow their initiating modified RREQs. They can be severely delayed or simply lost. We need to clarify how to handle these delayed packets. Moreover, an intermediate nodes can receive more than one SR in the same time frame, therefore, triggering even more modified RREQ/RREP packets. We need a synchronization method to coordinate all the SRs, modified RREQ, modified RREP packets so as to achieve a robust and consistent response. This synchronization method is a common timeline.

The Route-Change procedure starts with the arrival of an SR packet. SR packets can either initiate a Route-Change procedure, or, they can join an existing procedure. For a specific source node, there can be several parallel RC processing instances going on, each for a different destination. Modified RREQ packets are sent at certain time points in the procedure and the modified RREP packets are handled depending on the time they are received. Fig 3.2 shows the

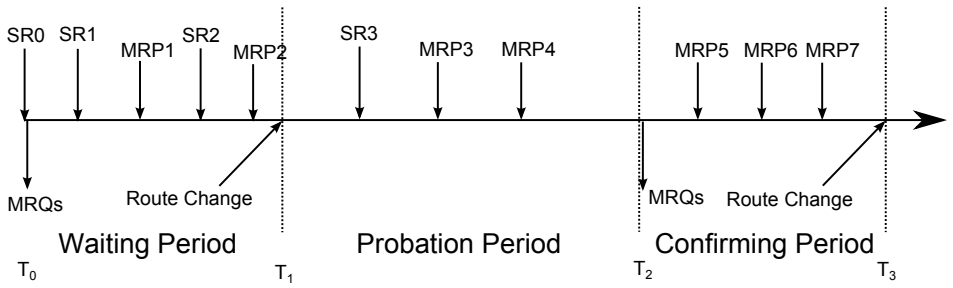


Figure 3.2: Route-Change Procedure Timeline (SR: Self Recommendation. MRQ: Modified RREQ Packet. MRP: Modified RREP Packet.)

timeline for one of such RC instances.

An RC procedure consists of three periods, and we discuss them separately as follows:

1. Waiting Period. Upon receiving SR0 for destination D, if there has not been any RC instance for D, a new RC instance is initiated. This node then sends several MRQs as shown in Fig 3.2), each for a known route to D. During the Waiting Period, any SR packets and any MRP packets for this destination will be temporarily stored.

2. Probation Period. At T_1, the node examines the SR/MRP packets received so far, compares the previous in-use route with the recommended routes, and decides which route is going to be adopted. Data packets are then sent through the newly adopted route. SR and MRP packets received during this period are simply discarded.

3. Confirming Period. At the end of Probation Period, this node will again send modified RREQs for each route, and gathers modified RREPs from these routes. This Confirming Period provides the chance to revert pre-

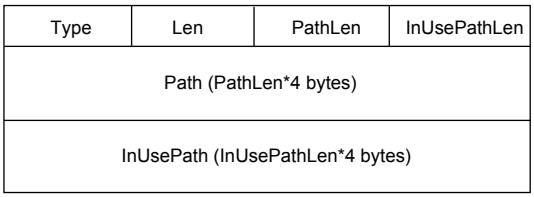
vious route changes if the recommended route has higher CRM than the previously-in-use route.

3.3.2 The Unbiased CRM Metric

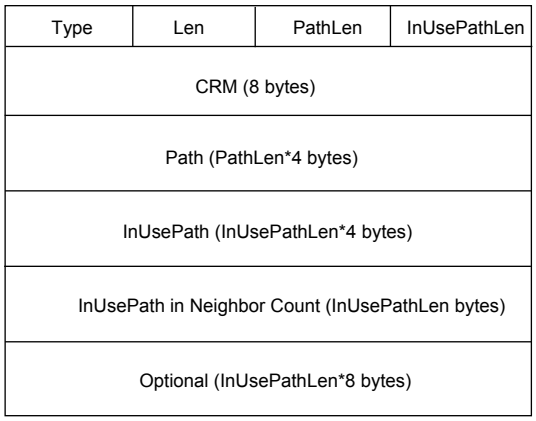
In the last section we mentioned that the CRM values in the modified RREP packets received are biased. They will favour the currently-in-use route. We justify this statement here.

The CRM values returned for the previously-in-use route $r_{\text{in-use}}$ and the recommended route r_{sr} are unbiased only if there is no data packet sent through $r_{\text{in-use}}$. The unbiased CRMs will weigh two routes equally. Consider a scenario shown in Fig 3.1. The flow (1→4→2) has existed for some time. As the new flow (6→3→5) arrives, node 3 will send a SR to node 1. The previously-in-use route is $r_{\text{in-use}}$ (1→4→2 in Fig 3.1), and the recommended route is r_{sr} (1→3→2). In Fig 3.1, unfortunately, there have been data packets sent through (1→4→2) before the SR is received by node 1. This traffic will increase the static modified queue length (This is a component of the CRM metric, refer to [28] for more details) of node 4 from 0 to some value *diff*. This modified queue length is transmitted in node 4's hello message, so node 3 will account for this *diff* when calculating the CRM for (1→3→2). Ironically, flow (1→4→2) will not account for *diff* in calculating its own CRM. As a result, node 1 receives a higher CRM for flow (1→3→2), which is unfair. Such a comparison between two flows makes no sense and also leads to a wrong routing decision.

The remedy for the problem is a more sophisticated modified RREQ-RREP



(a) Modified RREQ



(b) Modified RREP

Figure 3.3: Modified RREQ/RREP Packet Formats

procedure. In this scheme, though the CRM values are calculated as usual, node 1 will pre-process the values and make sure that the final comparison removes the In-use-path bias.

The packet formats of the modified RREQ/RREP are shown in Fig 3.3 where *Path* denotes the route of interest, and *InUsePath* the previously-in-use route. In addition to the normal CRM calculation, when the modified RREP is transmitted back towards the source node, the intermediate nodes update the *Count* of the packet. If a node in *InUsePath* happens to be its neighbour, the corresponding count is incremented by 1. Again using the simple scenario in Fig 3.1, the modified RREP for flow (1→3→2) will have a count of 1. This means node 4, which is in the *InUsePath* is node 3's neighbour. Moreover, the modified RREP

packet has an extra optional field. This field will be present only when *Path* is exactly the same as *InUsePath*. This field contains the difference between the static modified queue lengths of the two cases: calculated with the in-use route and without the in-use route.

When the source node receives the modified RREPs for both $r_{\text{in-use}}$ and r_{sr}, we have enough information to adjust the CRM values to the unbiased values. The modified RREP from $r_{\text{in-use}}$ has the optional field, and the modified RREP from r_{sr} has the count field. For each intermediate node in the previously-in-use route, subtract *count* $\times$ *difference* from the CRM value of r_{sr}. Therefore, the CRM values of both flows are the net effect of the previously-in-use route traffic, and they can be compared directly to decide which route should be chosen. Mathematically,

$$\widehat{CRM_{r_{\text{sr}}}} = CRM_{r_{\text{sr}}} - \sum_{i \in r_{\text{in-use}}} count_i * difference_i \tag{3.4}$$

, where $\widehat{CRM_{r_{\text{sr}}}}$ denotes the unbiased CRM for route r_{sr}, $CRM_{r_{\text{sr}}}$ denotes the CRM in RREP of route r_{sr}. $count_i$ and $difference_i$ denotes the count and difference in RREP of r_{sr} for node i, respectively.

3.4 Topology Analysis

In last section, we have demonstrated that, in certain circumstances, SCAR can discover coding opportunities that are overlooked by other coding-aware routing protocols. But what exactly characterizes these circumstances? Apparently they

Table 3.1: Definition of a Few Terminologies

Link	Undirected multi-hop path connecting one node to another
Route	Directed multi-hop path connecting a source node to a destination node
Flow	An ordered pair of source and destination nodes, without specifying the route chosen
Coding-possible route	This is a legacy definition. Given a set of routes that belong to different flows, if the packets of all the flows can be coded together and their respective destinations can decode the coded packets, these routes are coding-possible
Coding-possible flow	This is a new definition used in this chapter. A set of n flows are said to be coding-possible if there exist n coding-possible routes that respectively belong to the n flows

are related to the network topology, but how can we identify these "structures" and test their ubiquity? Furthermore, how can we quantify these structures' impact on the finalised throughput gain over other coding-aware protocols.

In this section we try to answer all the above questions.

3.4.1 Theoretical Induction of Indicators

Before describing the indicators, we first define some terms used as shown in Table 3.1.

The performance of a coding-aware routing protocol is highly dependent on the network topology and network traffic patterns. Some of the factors that affect whether the current flow can be coded with other flows are external, e.g., the routing decision for another route. We can still use the example topology shown in Fig 3.1. The flow (6-5) can be coded with other flows only when another flow, (1-2), has taken the route (1-3-2) instead of (1-4-2). In such a topology, the probability that SCAR can outperform DCAR is thus 50% given that flow (1-2)

comes earlier than the flow (6-5). The probability 50% is the value we want to capture using our indicators.

These indicators can be used to estimate how much throughput gain that is obtainable over a typical coding-enabled routing protocol. There have been much discourse on the maximum achievable coding gain [4, 8]. Note that the "coding gain" in their works refer to the throughput improvement of a coding-enabled protocol over other coding-disabled protocol. However, the "throughput gain" in our work here, refer to the throughput improvement of our protocol over a coding-enabled protocol. The baseline is elevated to incorporate network coding, but still excludes the awareness of network dynamics (flow arrivals and departures).

The series of indicators are named as CN, with N starting from 2. The definition is as follows. For *two* two-hop flows that are coding-possible, say $(S_1 - D_1, S_2 - D_2)$, we assume the number of possible routes for $S_1 - D_1$ is larger than or equal to that for $S_2 - D_2$. If we choose a route R_1 from all the possible routes for $S_1 - D_1$ randomly, the probability that there exists a route R_2 for $S_2 - D_2$ that is coding-possible with R_1 is defined as $C2$ for $S_1 - D_1, S_2 - D_2$. Suppose we define $R(S - D)$ as the set of possible routes for source destination pair $S - D$. And we use $r_1 \sim r_2$ to denote a flow taking the route r_1 can be coded with another flow taking the route r_2. Mathematically, $C2$ is calculated

as:

$$C2(S_1 - D_1, S_2 - D_2) = C2(S_2 - D_2, S_1 - D_1)$$
$$= \frac{|\{r \in R(S_1 - D_1) \mid \exists r' \in R(S_2 - D_2), r \sim r'\}|}{|R(S_1 - D_1)|} \quad (3.5)$$

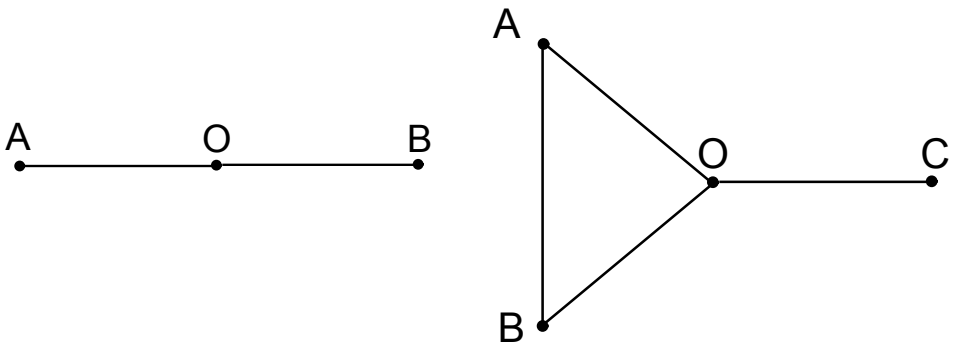
, assuming $|R(S_1 - D_1)| > |R(S_2 - D_2)|$. For comparison purpose, this $C2$ value is then adjusted as $C2_{adj} = 2 \times |C2 - 0.5|$, so that the range of $C2_{adj}$ still falls between zero and one. The rationale of doing this adjustment is discussed later in this section. As an illustration, we revisit the topology shown in Fig 3.1 and derive the $C2$ value for flow pair (1-2) and (6-5). There are two possible routes for flow (1-2), and only one possible route for (6-5). When choosing the route (1-4-2) for flow (1-2), there is no possible route for flow (6-5) to code. But if we choose the route (1-3-2) for flow (1-2), there is. So following the definition of $C2$, if we randomly choose a route for flow (1-2), we choose route (1-4-2) and route (1-3-2) with equal opportunity. The probability that there exists a route for flow (6-5) (in this case, we have only one possible route for flow (6-5), i.e., route (6-4-5)) that is coding possible with flow (1-2) is 50%. So the $C2$ value for these two flows is 0.5, and after adjustment, 0. The $C2$ value of the topology is calculated as an average of all the adjusted $C2$ values of every pair of coding-possible flows.

$C3$ is defined similarly with the only exception that we consider *three* two-hop flows, while still randomly selecting one route for the first flow. The adjustment of $C3$ is the same, $C3_{adj} = 2 \times |C3 - 0.5|$. Note that we consider only two-hop

flows for now. This is because in our simulation, coding opportunities are mostly found in two-hop scenarios. Although we have also formulated the indicators for multi-hop routes, the simulation results did not agree with the indicators. Multi-hop transmissions are prone to packet loss and they usually have outrageous delays compared to two-hop transmissions. Given the definition of CN, if we can find a systematic way of listing all coding-possible route pairs/triples, $C2$ and $C3$ can be easily calculated. We then provide a graph-based algorithm to enumerate all coding-possible route pairs/triples.

Consider 3 possible node placements as shown in Fig 3.4. The three sub-figures consider how many end nodes these two routes have in common. Fig 3.4a shows the case when two routes have both their end nodes in common, i.e., any two-hop route can be coded with its reverse route. In Fig 3.4b, route A-O-C can be coded with route C-O-B. Similarly, route B-O-C can be coded with C-O-A. In this topology, two route coding pairs are found. Note that we do not count A-O-C and C-O-A in this figure, because this pair is already counted in Fig 3.4a. In Fig 3.4c, where two routes share no common end node, four route coding pairs are found. They are A-O-C with D-O-B, A-O-D with C-O-B, C-O-A with B-O-D, and C-O-B with A-O-D. Similarly, the pair A-O-C and C-O-B is not counted here as it has been counted in the second case. In summary, our problem is reduced to a problem of listing these structures, then list the route pairs as above as long as the respective sources and destinations of these routes are not neighbours.

So far, the work seems trivial. But as we generalize to coding-possible route



(a) A route coded with its reverse

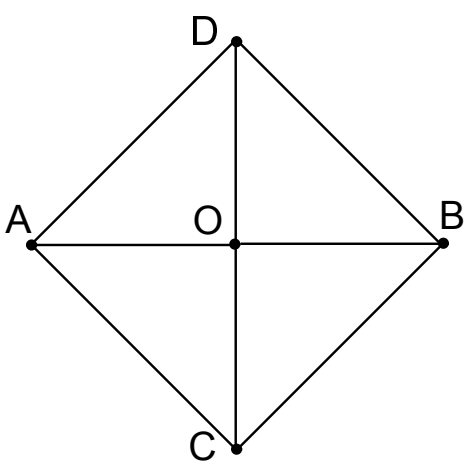
(b) Routes share an end node

(c) Routes with distinct end nodes

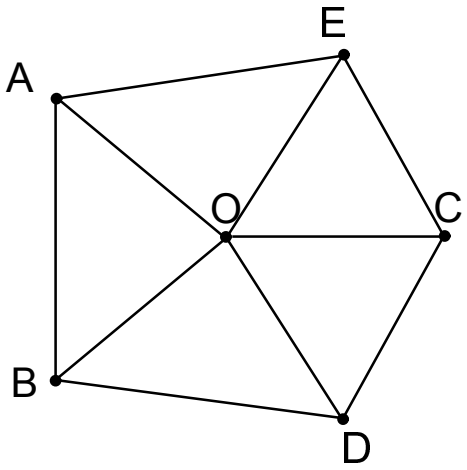
Figure 3.4: Possible Topologies for 2 Coding-possible Route Pairs

triples, we begin to find some patterns. A coding-possible route triple can be seen as a coding-possible route pair with an additional route. The additional route should meet certain criteria, i.e., the destination/source of the additional route should be a neighbor to the Source/Destination of either of the other two routes (Consider a sharing node as a special case of neighbourhood). So we derive the three possible topologies for coding-possible route triples from Fig 3.4, yielding Fig 3.5.

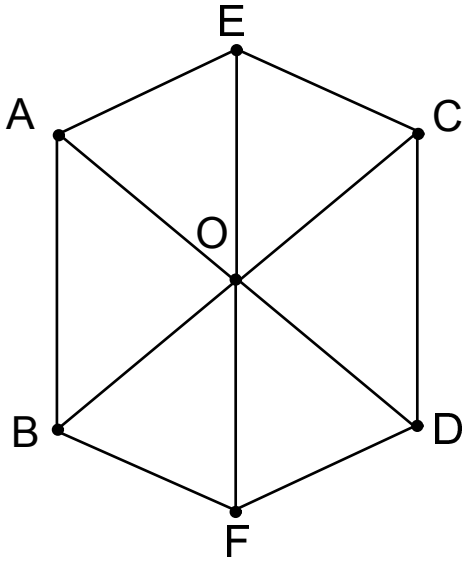
Extending Fig 3.4a, the additional route cannot share the end node with the previous routes, yielding a 4-node structure with 4 triangles (Fig 3.5a). With the initial two routes A-O-B and B-O-A, the new route can be either C-O-D or D-O-C. In addition, this topology can be rotated 90 degrees, with C-O-D and D-O-C chosen to be the initial routes. So we can find $2 \times 2 = 4$ route triples. Extending Fig 3.4b, the additional route again cannot share end nodes with the previous two, introducing the additional nodes D and E in Fig 3.5b. With the link O-C shared by the initial two routes, there are two coding triples in this topology: (A-O-C, C-O-B, D-O-E) and (B-O-C, C-O-A, E-O-D). Again, due to the symmetry of the topology, the number of route triples in this topology should be multiplied by 5, giving $2 \times 5 = 10$. Extending Fig 3.4c is the most complex one. The additional route can potentially share end nodes with initial routes. However, further investigation reveals that sharing end nodes will reduce the situation back to the one we have covered in Fig 3.5a and Fig 3.5b. The only new situation we should consider is shown in Fig 3.5c, where altogether 6 end nodes are considered. There are two route triples in this topology: (A-O-D,



(a) Case 1



(b) Case 2



(c) Case 3

Figure 3.5: Possible Topologies for 3 Coding-possible Route Triples

C-O-B, F-O-E) and (B-O-C, D-O-A, E-O-F). Rotating the topology will not give new route triples.

Now we can see the importance of triangles in these topologies. As long as we list all triangles in a topology, by examining the adjacency of these triangles, we can easily list all route pairs and route triples. This deduction can be carried forward for 4, 5 or even more routes. Based on their definitions, $C2$ and $C3$ can be easily calculated. We then discuss what are the implications of these indicators.

The improvement of SCAR over DCAR lies in the route-maintenance phase. SCAR provides a mechanism to revise already established routes. Therefore, if prior routes are poorly selected, they can be changed later to take into consideration the new flows in the network. Having a $C2$ between 0 and 1 leaves the possibility of choosing the "wrong" route initially. DCAR is more likely to choose the "right" route if $C2$ is closer to 1, and it is less likely to choose the right route if $C2$ is closer to 0. On the other hand, as the $C2$ value increases from 0.5 to 1, the throughput gain SCAR can provide over DCAR decreases as a result of better DCAR performance. As the $C2$ value decreases from 0.5 to 0, theoretically SCAR could still find the "right" route after a Route-Change procedure, and thus should provide a higher throughput gain. However in practice, SCAR's performance also suffers. A decreasing $C2$ value usually means more potential routes and less "correct" routes. With more routes in the network, the modified RREQ-RREP packets are prone to get lost, leading to incorrect route change decisions. So the intuition is: $C2$ value being too big or too small

can limit SCAR's performance over DCAR. So an adjusted $C2$ is proposed as $C2_{adj} = 2 \times |C2 - 0.5|$. $C2_{adj}$ is scaled such that it varies from 0 to 1. Smaller values imply higher throughput gain.

We argue that:

1. $C2$ represents the level of confidence that DCAR will find a coding structure in a given topology. $C3$ and higher rank indicators denotes the likelihood for DCAR to find multi-packet coding opportunities.

2. Adjusted $C2$, or $C2_{adj}$, represents the extent of throughput gain we can expect from choosing to use SCAR over DCAR as the routing protocol. Smaller $C2_{adj}$ implies higher throughput gain.

3. Adjusted $C3$ or higher-rank indicators offer possibility of higher performance as well, but the volatility of throughput gain is higher.

We validate our proposed scheme by simulations. The objective is to test and justify the effectiveness of using CN indicators to predict SCAR's throughput gain over DCAR. The approach we take is: First, to prove that the $C2$ can indeed reflect SCAR's ability to improve throughput, and then to verify that $C3$ and higher-rank indicators for flows can reflect SCAR's ability to discover multi-packet coding opportunities. In order to test our hypotheses, we designed a series of topologies where different coding structures with different CN values can be spotted. We then evaluate the throughput gains of SCAR over DCAR in these coding structures and plot them against the CN value as shown in the next section.

3.4.2 Simulation Results

3.4.2.1 Simulation 1

With an artificially designed topology where flows with different CN values can be found (See Fig 3.6), we choose 16 flow pairs for each of the $C2$ values to be examined, namely, 0, 1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5 and 1. For each flow pair, we measure the network throughput of both routing protocols under 6 different offered loads. The throughput measured at the stationary state of each simulation instance is averaged over 10 runs with random seeds. Fig 3.7 is the scatter plot of the throughput gain for each flow pair against the value of $C2$. The mean throughput gain for each $C2$ value is denoted as a triangle symbol in the figure. The throughput gains for both pre-adjusted C2 and adjusted C2 are averaged over the data points to yield Fig 3.8 and Fig 3.9. The error bars denote 90% confidence intervals. From these results, we observe that a $C2$ value between 0 and 1 implies a throughput gain significantly different from zero, while a $C2$ value of zero or one, implies a throughput gain near zero. It is worth noting that $C2$ being 0.5 gives the highest throughput gain. This justifies the adjustment on $C2$. The adjusted $C2$ values are readily comparable, and thus can be averaged to get the $C2$ for the whole topology. Although the correlation between throughput gain and adjusted C2 values is not strictly linear, it is safe to assume a pseudo-linear relationship within the granularity we are discussing.

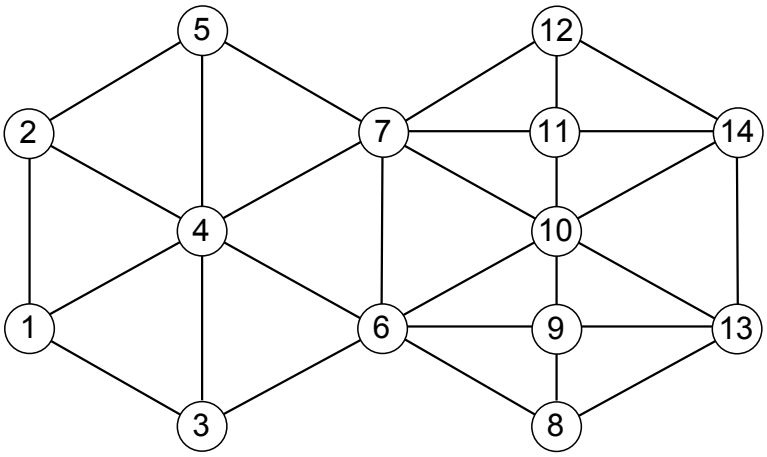


Figure 3.6: An Artificially Designed Topology to Measure the Effectiveness of CN Indicators

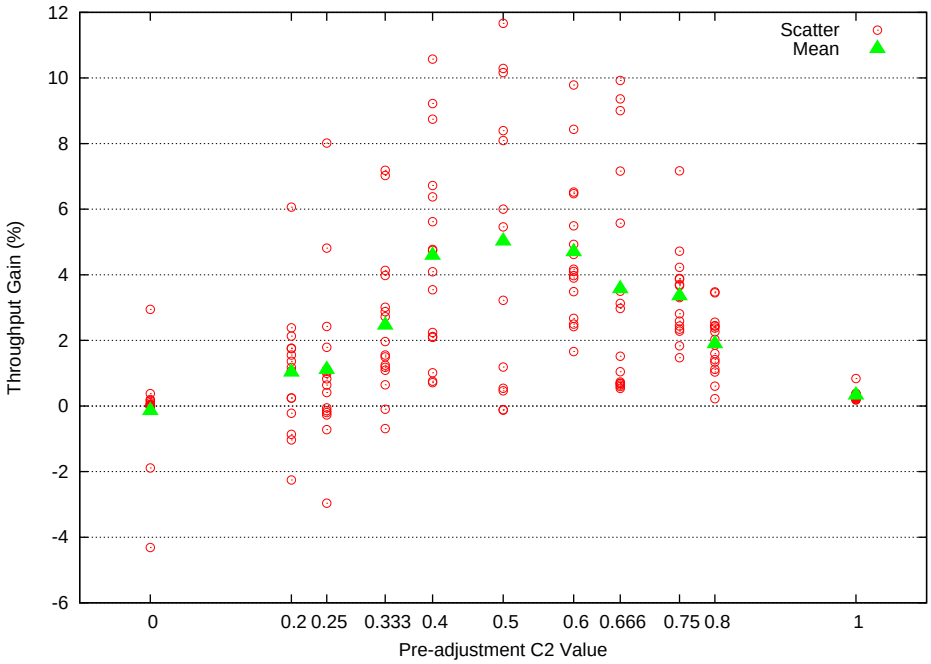


Figure 3.7: Scatter Plot of Throughput Gain with Different $C2$ Values

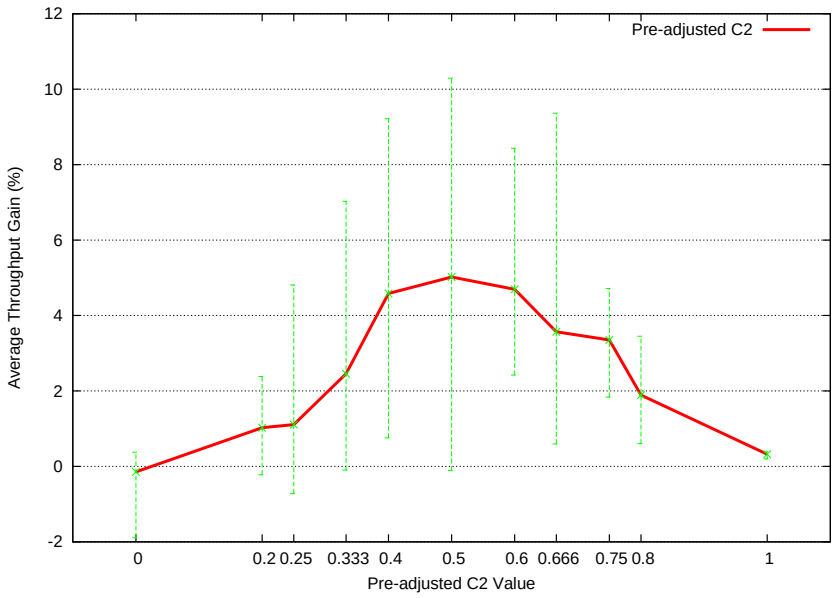


Figure 3.8: Average Throughput Gain versus Pre-adjusted $C2$ Values

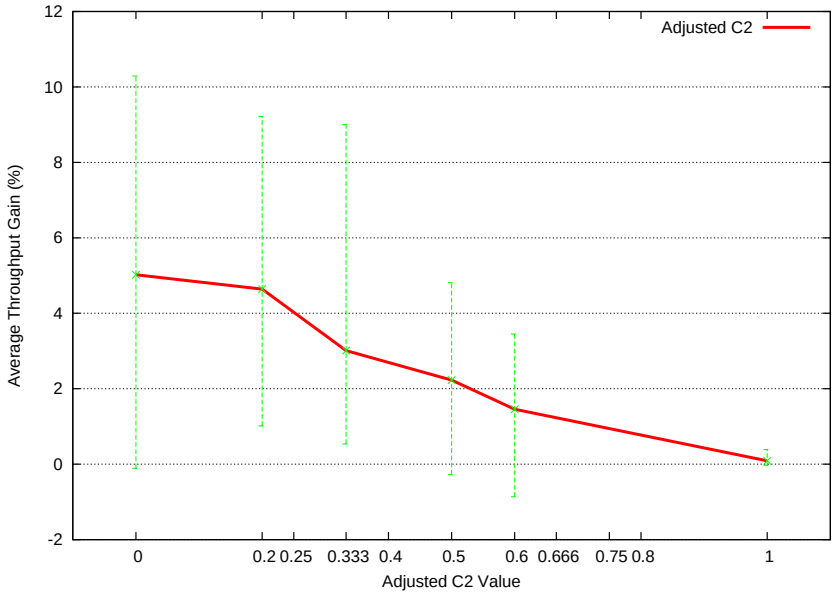


Figure 3.9: Average Throughput Gain versus Adjusted $C2$ Values

3.4.2.2 Simulation 2

Topologies with a specified $C3$ value are difficult to construct. In fact, the topology shown in Fig 3.6 gives 30 non-zero $C3$ value, but 26 of them have a value of 1/3. In order to test with other $C3$ values, we have randomly generated tens of topologies, usually each one can produce only 2-3 valid $C3$ values for our evaluation. Finally we collected altogether 144 different flow triples taking $C3$ values in $(0, 0.25, 1/3, 0.5, 2/3, 0.75, 1)$. The throughput gain with different $C3$ values is plotted in Fig 3.10, with the average throughput gain shown in Fig 3.11 for pre-adjusted $C3$ values and in Fig 3.12 for adjusted $C3$ values. The error bars denote 90% confidence intervals. We can find some patterns within the scatter plot. The scatter plot for each non-zero and non-one $C3$ value in Fig 3.10 can be basically divided into two clusters. One in the $\pm 10\%$ throughput gain range, and the other cluster above 10%. The first cluster is roughly symmetric about the zero point and we can explain it as a normal variation. In these cases, SCAR simply brings in a dynamic scheme for the sources to reconsider those already established routes, adding some deviations to the mean throughput. The second cluster, however, are the cases where SCAR manages to find multiple-packet coding possibilities. In fact, after thorough examination of the simulation log file for the data points in this cluster, SCAR tends to find 3-packet coding opportunities while DCAR only finds 2-packet coding opportunities or no coding opportunities at all due to the order of flow arrivals. In conclusion, a $C3$ value between 0 and 1 offers multiple-packet coding opportunities that can be discovered by SCAR. However, SCAR is not guaranteed to find them and the performance can

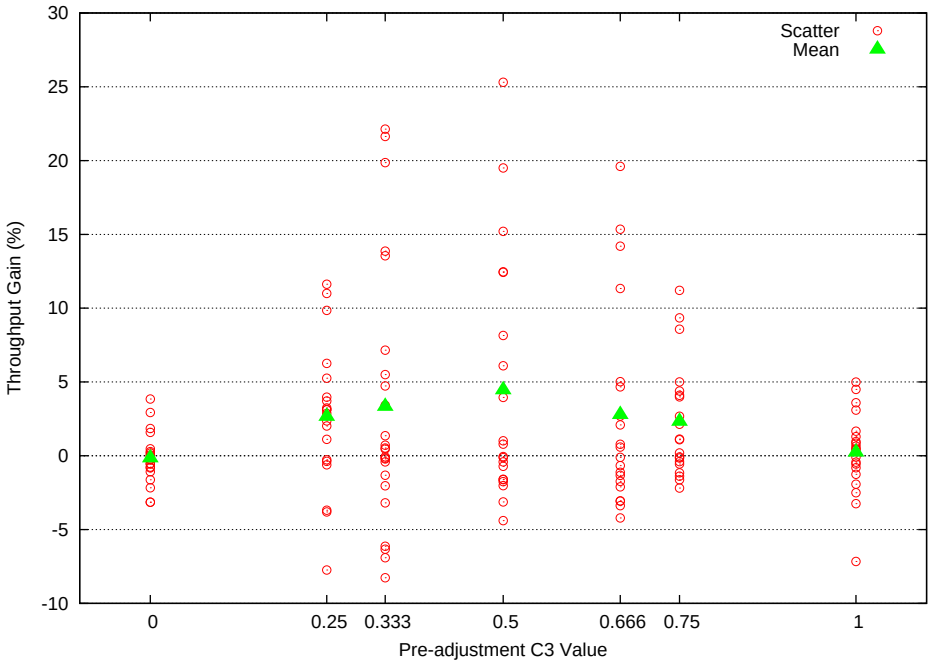


Figure 3.10: Scatter Plot of Throughput Gain with Different $C3$ Values

fluctuate within a relatively large variation depending on the traffic condition.

3.5 Evaluation

In last section we have argued that it is fairly common to have a topology where SCAR has an advantage over existing coding-aware routing protocols. In particular, when the adjusted CN indicator signifies a value close to zero, higher throughput gain can be expected.

In this section, we proceed to measure the performance of SCAR in network simulations rather than mathematical formulations. Three sets of simulations are done on the Qualnet[1] simulator, revealing different aspects of features of our proposed protocol SCAR.

[1]http://web.scalable-networks.com/content/qualnet

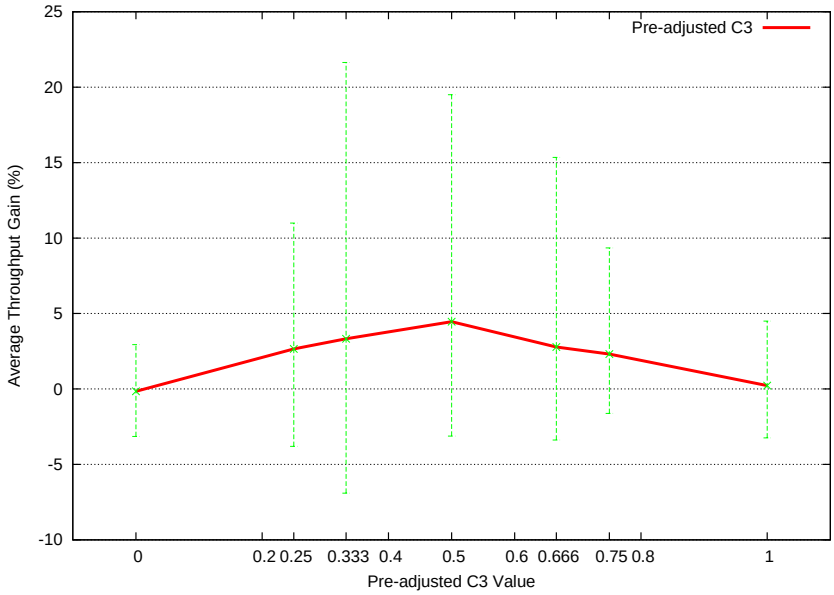


Figure 3.11: Average Throughput Gain versus Pre-adjusted $C3$ Values

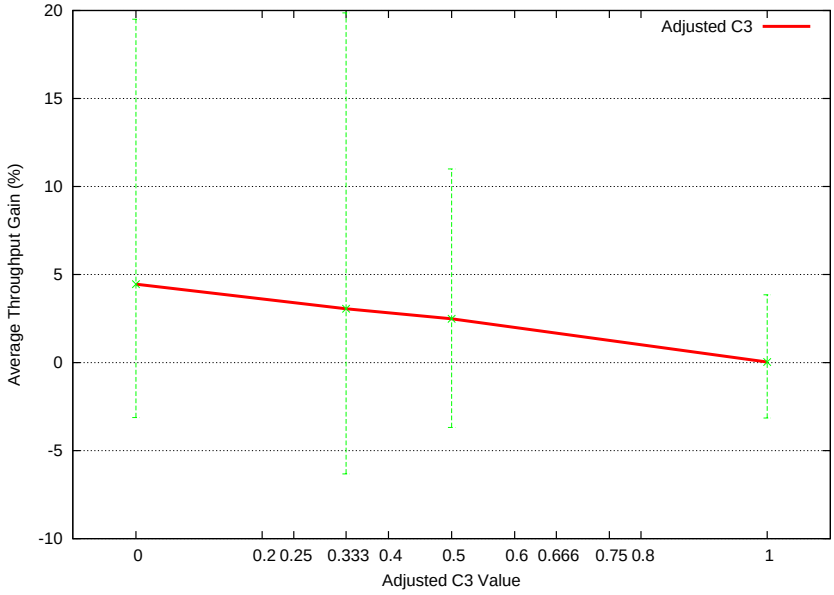
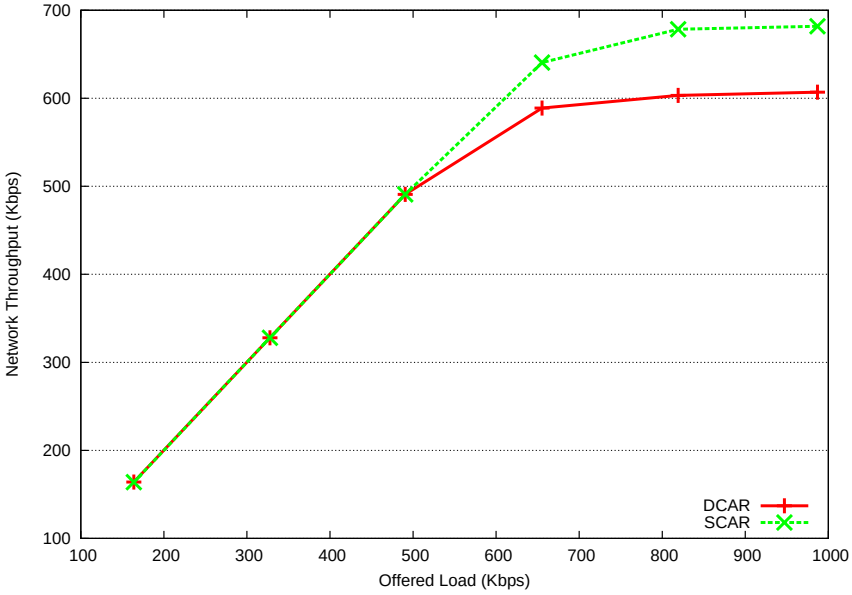


Figure 3.12: Average Throughput Gain versus Adjusted $C3$ Values

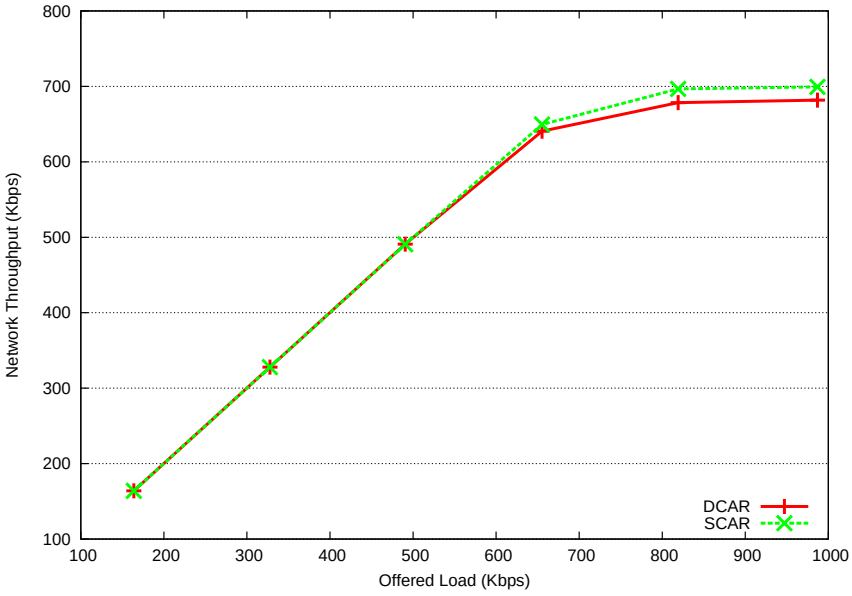
3.5.1 Simulation 1. Simple Topology

Using the simple scenario in Fig 3.1, we illustrate how DCAR is vulnerable to flow-arrival times and how our proposed protocol SCAR can withstand this variability. We start with a flow from node 1 to node 2, and then, add a new flow from node 6 to 5. This is defined as starting order 1. The starting order is then reversed and is defined as starting order 2. The flows are given the same traffic load. For starting order 1, whether there is a coding structure depends on the route selection for flow 1-2. The paths 1-3-2 and 1-4-2 would have the same routing metric value at the beginning. Under DCAR, if node 1 randomly chooses path 1-4-2, there will be no coding structure formed in the network. For starting order 2, DCAR can also choose the right routes to use at the onset. In contrast, for SCAR, no matter which flow node 1 chooses at the beginning, it is able to adjust itself to form a coding structure and improves the throughput of the network. In terms of CN indicators, the flow pair (1-2) and (6-5) has a $C2$ value of 0.5, which equals an adjusted $C2$ value of 0. This implies that it is very likely to have a throughput gain with these flow pairs. The whole topology, when considering all possible flow pairs, has an adjusted $C2$ value of 0.46. There is no multiple-packet coding opportunity in this topology. Anyway, this topology is simply to serve as a demonstration, so we do not test with more flow pairs.

We vary the offered load and plot the end-to-end throughput in Fig 3.13. The two starting orders mentioned above are considered for DCAR and SCAR. Each data point in the figure is an average throughput over 10 different random seeds. We observe that SCAR could always choose the path 1-3-2 after a period



(a) Starting order 1



(b) Starting order 2

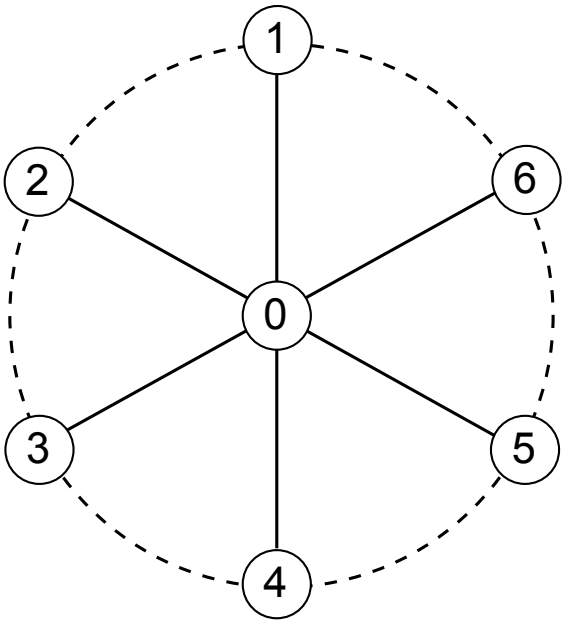
Figure 3.13: Simulation Results for the Simple Test Topology

of time and forms a coding structure in the network, while DCAR could only choose between the paths 1-3-2 and 1-4-2 randomly under starting order 1. The throughput gain at the highest offered load averages to 12.3%. However, under starting order 2, the performance of DCAR and SCAR are quite similar. The confidence interval, or the minimum/maximum value in the simulation, is not plotted in the figure for clarity. In fact, the minimum and maximum values are quite similar and DCAR's and SCAR's confidence interval severely overlap. The reason why SCAR's average performance is better than DCAR's, is that SCAR has higher probability of reaching the maximum value.

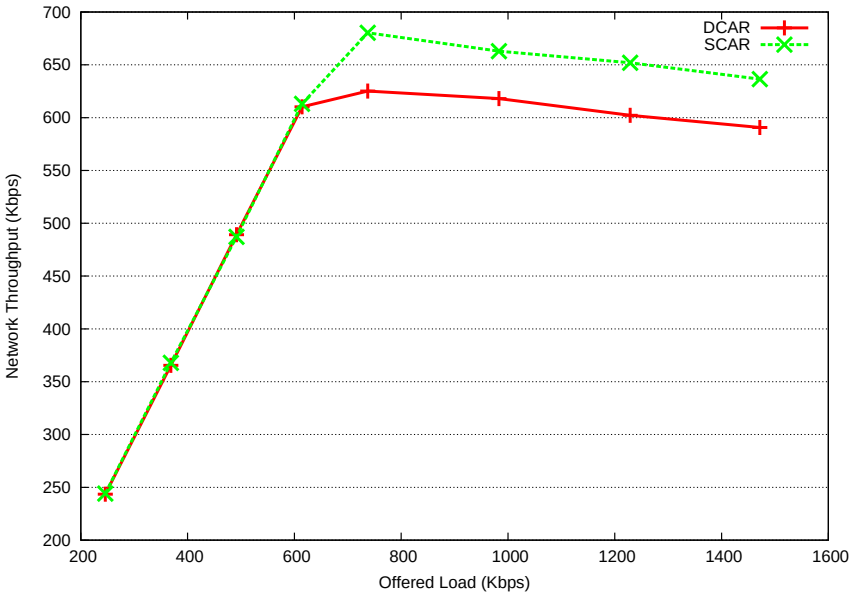
3.5.2 Simulation 2. "Wheel" Topology

The "Wheel" topology as shown in Fig 3.14a is an interesting and efficient topology to study the performance of coding-aware routing protocol. A central node (0) is surrounded by six nodes (1-6) evenly distributed along the cycle. Each node can reach all the other nodes except for the node on the opposite end of the diameter (e.g. node 1 can reach all nodes except node 4, and vice versa). Each node tries to send data to the node at the opposite end of the diameter. There are many coding opportunities in this scenario as studied in [28]. In our topology analysis, this topology has an adjusted $C2$ value of 0.36 and an adjusted $C3$ value of 0.33. This suggest that it is likely to see a throughput gain by exploiting the $C2$ value, while the throughput gain may not be consistent because the $C3$ value brings in more uncertainty as well as an expected throughput gain.

The average throughput of the network under varying traffic load is plotted



(a) "Wheel" topology



(b) Average throughput of the network

Figure 3.14: Simulation Results for "Wheel" Topology

in Fig 3.14b. When the network is saturated, the throughput gain averages to 8.02%. Further inspection into the simulation log reveals that SCAR tends to find 4-packet coding opportunities when DCAR is unlikely to find any. This is because of DCAR's inability to change route once the route was established. In such a topology, DCAR usually will match the first two arriving flows to share a common intermediate node. But the third arriving flow will be routed through a different intermediate node to avoid congestion. After DCAR routes the fourth arriving flow, these four flows are routed as two groups with each group sharing a common intermediate node. Therefore, in the case of DCAR, it is very unlikely, if at all possible, to find 4-packet coding opportunities. DCAR's routing mechanism will forbid the third flow to be routed through the same intermediate node for the first two flows. However, in the case of SCAR, the Route-Change procedure brings in more possibilities. SCAR enables a mechanism to revise existing routes even if the protocol firstly routes the 4 flows as two groups. SCAR also allows multiple concurrent Route-Change procedures, so it is also possible to change routes for two flows simultaneously. For example, suppose flow (1-4) and (4-1) have both chosen node 6 as the intermediate node. And we have the third flow (2-5) choosing node (3) as the intermediate node. Now the flow (5-2) comes in. Node 6 notices flow (5-2)'s arrival, and it initiates the Route-Change procedure for all flows including (1-4), (4-1), and (2-5). If flow (2-5) opted to use node 6 as the intermediate node after the Route-Change procedure, we are now having a 4-packet coding opportunity. Anyway, it is worth noting that SCAR does not guarantee a better routing decision. In complex network situations,

the interference of other flows may cause the Route-Change procedure to fail. It is also possible that multiple intermediate nodes are equally weighed by the CRM. Therefore, whether multiple flows would choose the same intermediate node is flow pattern sensitive and opportunistic. Anyway, SCAR have a higher probability of finding such opportunities. We have also tested different starting orders and find that this does not have any influence on the performance of DCAR and SCAR. The reason is that this is a totally symmetric scenario with symmetric topology and flows.

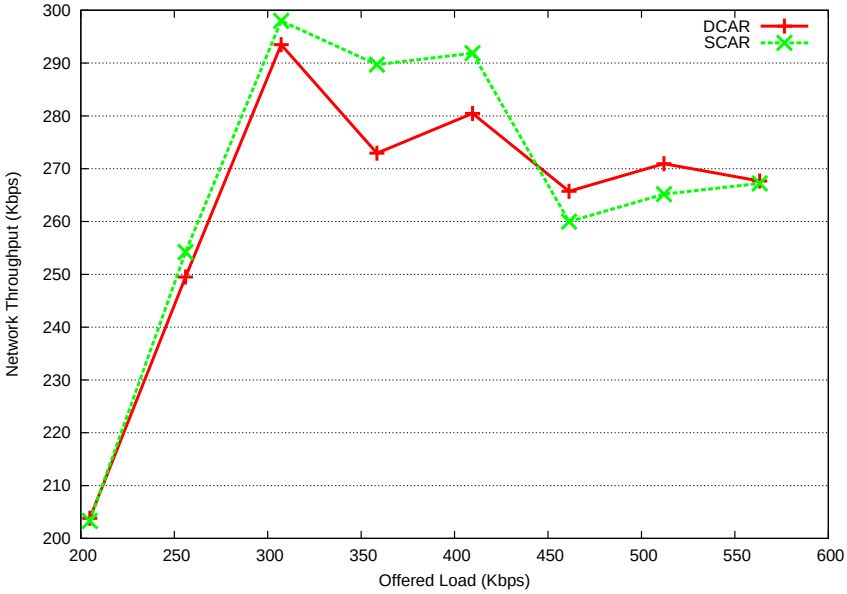
3.5.3 Simulation 3. Grid Topology

Next we construct the same 4×4 topology as mention in [28] where each node can only reach its northern, southern, eastern and western nodes. The simulation is run for 10 rounds and at each round, five flows (each with 2-5 hops) are randomly added into the network. The adjusted $C2$ and $C3$ values of this topology is 1, suggesting that there would be no throughput gain from switching the routing protocol. This is because in the deduction of the CN indicators, we limit the coding structure to two-hop ones. But in this grid topology, only multi-hop coding structures exist. The average end-to-end throughput of the network achieved by DCAR and SCAR is shown in Fig 3.15a. In this simulation, SCAR is only slightly better than DCAR because the multi-hop coding opportunities discovered by SCAR are prone to errors, therefore the throughput gain is compromised. When the offered load continues to increase, the throughput gain can be negative sometimes. The average throughput gain in a saturated network is

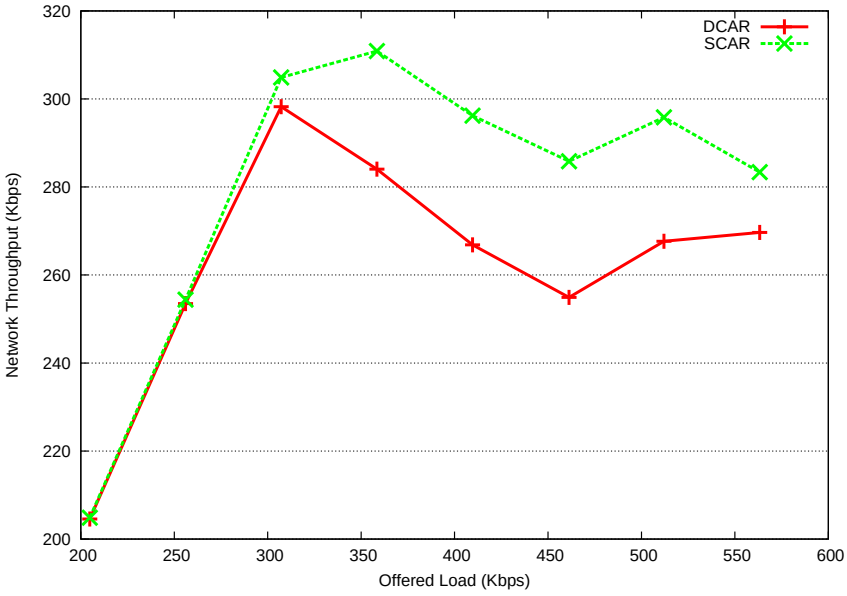
1.15%.

However, by modifying the topology only a little bit, we can achieve totally different results. By increasing the transmission range of the nodes to reach all its 8 neighbouring nodes, the CN indicators change dramatically. After the modification, the adjusted $C2$ value is 0.38 suggesting high probability of throughput gain, and the adjusted $C3$ value is 0.48, suggesting an expected throughput gain with variation. The simulation results are shown in Fig 3.15b and the throughput gain is obvious. The average throughput gain reaches 9.63% without changing any flow settings. In both figures, the average throughput went up and down. This is because at higher offered load, the packet error rate is also higher, leading to retransmissions and packet drops. In a severely saturated network, packet collisions are more often and thus more retransmissions are required, reducing the effective throughput.

The comparison between these two grid topology settings further justifies the effectiveness of CN indicators. $C2$ can be used as a reliable predictor for the expected throughput gain. Though $C3$ is not as reliable as $C2$ for a single run of simulation, it still identifies multiple-packet coding opportunities and provides an expected throughput gain across a large number of runs of simulations. Additionally, one can leverage on the knowledge of CN to modify the network topology, hoping to drastically improve the overall throughput.



(a) Average throughput of grid topology



(b) Average throughput of modified grid topology

Figure 3.15: Simulation Results for Grid Topology

3.6 Chapter Summary

In this chapter, we first introduced SCAR, a coding-aware routing protocol with Self-Recommendation for wireless ad hoc networks. The mechanism of SR enables the protocol to discover hidden coding opportunities that have been overlooked by other routing protocols. Whenever a new flow joins the network with new coding opportunity, prior routing decisions are revised through a Route-Change procedure. In such an environment, direct application of the routing metric will wrongly favour the currently-in-use route. To remove this bias, the Route-Change procedure triggered by the SR employs an adjustment method to remove such bias.

Moreover, in order to estimate how much performance we can gain from exploiting the SRs, a series of indicators are introduced. These indicators are calculated using graph theory, and they reflect how much throughput gain can be expected when using our protocol in a given topology. Testing these indicators on multiple topologies reveals that the throughput gain is not a coincidence and many practical topologies come with positive indicators. In other words, a fair number of network topologies are susceptible to changes in the order of flow arrival. Thus, the robustness we emphasize is an important metric when evaluating coding-aware routing protocols.

Further simulations are done in Qualnet to evaluate our protocol. It is shown that our protocol can significantly outperform other coding-aware routing protocols and the predictions given by the indicators are generally correct. The dynamic nature of our protocol results in a higher invariability against network

changes.

Chapter 4

Improving Coding Efficiency and Fairness by Network-layer Packet Scheduling Algorithm

4.1 Introduction

Packet scheduling, as it is often used by researchers on network coding, generally refers to the behaviour to conduct certain transmission task in an omniscient and ordered way in a given network [94]. The transmission task is usually to deliver several packets from one or more source nodes to one or more destination nodes. The scheduling is done with the full knowledge of the topology and the transmission capacity. In addition, all nodes are expected to act under the coordination of the *supervisor*.

In real deployments, the above mentioned requirements are seldom practical.

Instead, coding-aware routing protocols are often used to judiciously route the flows so that coding benefits can be exploited. Coding-aware routing protocols relax the requirement of centralised control and make the coding opportunities occur opportunistically. However, the problem of per-flow fairness and further throughput improvements are less analysed. As far as to the author's knowledge, there has been no in-depth analysis on per-flow fairness with network coding enabled. Also, how to select the proper packet in a node's backlog so that the overall draining rate of the packet queues of this node can be maximized has not been deeply studied.

In this chapter, we argue that by adopting a well-designed network-layer packet scheduling algorithm, we can have better control over other performance metrics of wireless ad hoc networks. Packet scheduling in this chapter refers to the decisions made at each node when it is ready to send packet. With multiple packets backlogging, each node should determine which packet to send first. When network coding techniques are considered, this decision also involves finding which are the other packets that should be coded together with the above selected packet.

In studying the packet scheduling problem, we can set different optimization objectives. One of these may be to maximize overall throughput. In our research it is observed that realizing this target is relatively straightforward but the solution suffers from severe starvation for some of the flows. In fact, only the flows that have coding counterparts would be able to transmit packet while flows without coding counterpart are completely ignored. This is generally undesirable in

actual deployments of the protocol. So we can instead set the target to a more meaningful one. We want to find a criteria that jointly considers transmission efficiency as well as per-flow fairness. Minimum per-flow throughput turns out to be a good proxy for such purpose. When this metric is maximized, all flows would have at least this same amount of throughput, yet some flows that are advantageous in exploiting the coding benefit could have higher throughputs.

In this chapter, our target is to maximize the minimum per-flow throughput first. When this target is fulfilled there may still be multiple choices, the solution that can maximize the average per-flow throughput is then selected. This two-step maximization can lead to quite acceptable performance as we will demonstrate in the evaluation section of this chapter. In addition, the derivation of the solution to this relatively complex target function can cover all the essentials of the problem.

Existing protocols that utilize the power of network coding usually take the most straightforward means to do network-packet scheduling, namely, Round-Robin (RR) scheduling. Basically an intermediate node will construct many queues, each storing packets from a specific flow/session. When the MAC layer seizes the chance to transmit after channel contention, the network layer selects the packet to be sent in a round-robin fashion across all queues. When a certain queue is selected, the first packet in that queue is dequeued and it goes through a check for coding opportunity. Coding opportunities are depicted in a *coding graph* [28]. In this graph, each vertex denotes a flow that passes by this intermediate node. An edge that links two vertices means packets from the respective two flows can

be coded together. When three flows can be coded together, it is shown as a triangle. Similarly, N coding-possible flows are denoted as a complete N-vertex sub-graph, aka. *N-clique*.

However, with this scheduling we do not have much control over the performance of the network (namely, per-flow throughput and fairness). Some flows may be repeatedly transmitted because they can be coded with many other flows, while some flows with less or no coding counterparts can starve. In addition, it is possible that we can increase the throughput by intelligently selecting the set of packets to transmit.

In this work, we first study a simplified and static form of the scheduling problem. When given a set of packets from multiple flows waiting to be sent, we analyse how to code and send the packets with the minimum number of transmissions. This problem is abstracted and formulated as a WCCP. WCCP is *NP-hard* in general, but some special cases of it can be solved in polynomial time. In order to reduce complexity and to make it more amenable for general use, a search algorithm with pruning and approximation is proposed. With this algorithm, the solution can be calculated much faster with known error bounds.

The solution to this static problem can then provide guidance to find the solution to the real problem, i.e., the dynamic form of the problem. In this form, we consider a node to be an intermediate node for multiple flows. A number of fixed-capacity queues are embedded in this node with packets stochastically arriving. The processing speed is also limited. We estimate the fairness with two relevant performance metrics: one is the minimum throughput and the other is

the variance of the throughputs of all flows. A higher minimum throughput, and a smaller throughput variance imply better fairness. Our objective is to find an optimal scheduling scheme which first maximizes fairness, then maximizes throughput. We use a heuristic scheduling method for this dynamic form of the problem.

Section 4.2 formulates the static scheduling problem as a WCCP and proposes an approximation algorithm. Section 4.3 describes the dynamic form of the problem and we propose here a heuristic scheduling scheme. Sections 4.2 and Section 4.3 contain some preliminary evaluations. Section 4.4 presents the scheduling scheme in the form of a routing protocol and a series of network simulations is done to evaluate the performance of this protocol. Lastly, we conclude in Section 4.5.

4.2 The Static Form of the Problem

4.2.1 Weighted Clique Cover Problem

In this section, we discuss the packet scheduling method within a single node. It is assumed in this section that no new packets will arrive, so this form is referred to as the *static* form of the scheduling problem. This scheduling method determines, from among all the packet queues, which queue to transmit and how the packet from this queue is encoded with other packets from other flows.

Consider a node in a network. This node has multiple packet queues and the number of packets inside each queue is known. Packets from some of the queues

can be coded together while others cannot. This relationship is depicted in a coding graph where each vertex represents one queue and each edge represents a coding-possible relationship between two queues. Each vertex is assigned with a weight. The weight of a vertex is equivalent to the number of packets that are backlogged in the packet queue corresponding to the vertex. A set of k packets can be coded and sent in one transmission if and only if they come from k distinct queues (k vertices) and each pair of queues is coding-possible ($\binom{k}{2}$ edges). The objective is to find a scheduling algorithm for this Weighted Coding Graph (WCG) that minimizes the number of transmissions to dump all packet queues. This problem is called *static* in the sense that it deals with packets already in queues and there are no new packet arrivals.

This problem is mathematically formulated as a weighted clique cover problem (WCCP) as below:

Definition 1 (Weighted Graph)**.** A *weighted graph* is a graph $G = \{V, E\}$ and a weight function $w : V \to \mathbb{Z}^+$. V is the set of vertices, E is the set of edges, and the weight function maps each vertex in V to a positive integer.

Definition 2 (Clique)**.** A *clique* is a complete sub-graph of the parent graph G. Mathematically, a clique is a set of vertices V_q, such that $\forall v \in V_q, v \in V$, and $\forall e \in V_q^2, e \in E$. V_q^2 denotes the set of edges that connect each pair of vertices in V_q.

Definition 3 (Weighted Clique)**.** A *weighted clique* w.r.t. the graph G is defined as $Q = \{V_q, w_q\}$, where V_q is a clique in G and w_q is a positive integer that

satisfies:

$$\forall v \in V_q, w_q \leq w(v) \tag{4.1}$$

Definition 4 (Weighted Clique Cover)**.** A *weighted clique cover* w.r.t. the weighted graph is a set of *weighted cliques* $\mathbf{Q} = \{Q_i = \{V_{q_i}, w_{q_i}\}, i = 1, 2 \ldots, n\}$ such that:

$$\bigcup_{i=1,2,\ldots,n} V_{q_i} = V \tag{4.2}$$

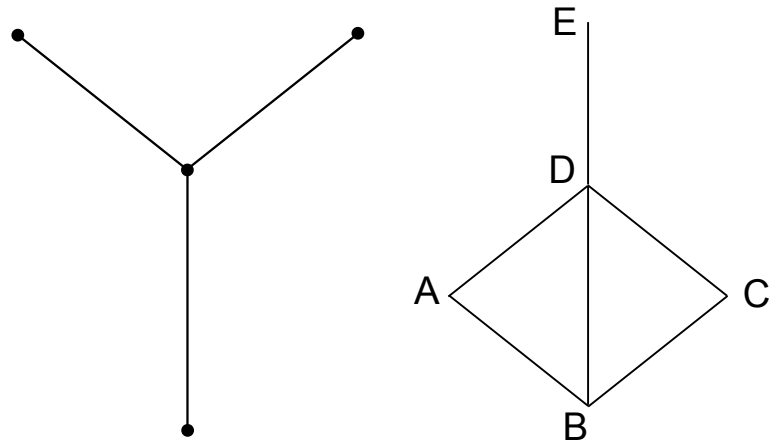
$$\forall v \in V, w(v) = \sum_{q_i : v \in V_{q_i}} w_{q_i} \tag{4.3}$$

Definition 5 (Weighted Clique Cover Problem)**.** Given a graph $G = \{V, E\}$ and a weight function $w : V \to \mathbb{Z}^+$. *Weighted clique cover problem* is to minimize $\sum_{i=1,\ldots,n} w_{q_i}$ among all possible weighted clique covers for graph G.

To facilitate the understanding of the definitions, we describe some of the symbols and terms here. For a graph $G = (V, E)$ where V is the vertex set and E is the edge set, there can be multiple cliques in this graph. Cliques are complete sub-graphs of a given parent graph. For the purpose of clarification, the vertex set of a clique is denoted as V_q. Similarly, we use V_{q_i} to list all cliques of the graph $G = (V, E)$, where i is the index that iterate through all cliques. In the scope of this thesis, a weighted clique is defined as a clique when assigned with a uniform weight to all its vertices. The weight that is assigned to clique V_{q_i} is then denoted by w_{q_i}.

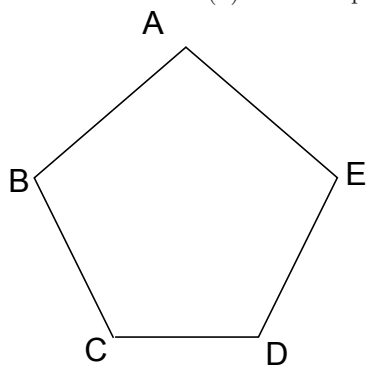
As one of the 21 Karp's NP-complete problems [95], Clique Cover Problem

(CCP) is NP-complete. WCCP, as a generalized form of CCP is known to be NP-complete as well and researchers have been working on a practical solution to it. It is shown by Hsu [96, 97] that the WCCP problem can be solved in polynomial time if the graph is a claw-free perfect graph. A claw is the shape shown in Fig 4.1a. It is often referred to as $K_{1,3}$ because it is essentially 1,3-bipartite. Bipartite is a type of graph where vertices are partitioned into two disjoint sets, and every edge connects from a vertex in the first set to a vertex in the other set. A claw-free graph is a graph where none of its sub-graphs is a claw. A perfect graph is a concept raised in the 1970s by Claude Berge [98]. A graph is said to be perfect if for each of its induced sub-graphs, the chromatic number equals to the size of its largest clique of the sub-graph. The chromatic number of a graph is the minimum number of colours required to colour the vertices of the graph such that no adjacent vertices share the same colour. Take Fig 4.1b as an example. Nodes A, B, D, E can form a sub-graph. The corresponding induced sub-graph contains the four nodes, as well as the 4 edges that connect them. In this induced sub-graph, the chromatic number is 3 because we can colour nodes A and E as red, node D as green and node B as yellow such that no adjacent nodes have the same colour. The size of the maximum clique in this sub-graph is also 3 as nodes A, B and D form a 3-node clique. The simplest imperfect graph is a ring of 5 nodes shown in Fig 4.1c. The size of the maximum clique is 2 but the chromatic number is 3. A more recent paper [99] summarizes the problem and some recent advances on the topic.



(a) Illustration of a Claw

(b) An Example of Perfect Graph



(c) An Example of Imperfect Graph

Figure 4.1: There Exists Polynomial Solutions to WCCP for Claw-free Perfect Graphs

4.2.2 Solution to WCCP

As the size of the problem (i.e., the number of vertices, the number of edges, and the amount of weights) scales up, WCCP becomes very hard to tackle. Existing polynomial solutions focus only on the claw-free perfect graphs. As is shown in Fig 4.1a, a claw is a very basic structure and it is observed in our simulations that such a structure appear very often. For practical networks, we need a definitive method to solve them, irrespective of whether they are claw-free.

Here we propose a search-based algorithm to solve WCCP for all graphs, which is more general than the existing algorithms. With the problem being NP-complete, the worst-case computational complexity of our algorithm is expected to be exponential. In order to solve the problem with practical time limits, a search-based algorithm with pruning rules is proposed in this work. The major improvement is the introduction of pruning rules. Applying the pruning rules can drastically decrease the amount of branches in search tree. We will come to see the effect of pruning rules in Section 4.4 Note that pruning can be applied even when the graph has claw sub-graphs and is imperfect. In very complex graphs, pruning rules may not be directly applicable, we therefore introduce an approximation method to keep the run time invariant to the amount of weights.

In essence, our algorithm is a search algorithm that structures all possible solutions to WCCP into a tree structure. Consider the WCCP in such a way: For each vertex in the graph, its weight equals the sum of the weights of all cliques that this vertex is part of. If we can list all cliques that one vertex is part of, the search for weighted clique covers can be transformed into the search for

allocation methods that allocate weights of vertices to a set of cliques.

The transformation of "covering" to "allocation" constructs the search space for our algorithm. We iteratively allocate weights for each vertex. When all vertices have their weights allocated, the solution to WCCP is determined. The search tree is then equivalent to a decision tree.

The root node of the search tree denotes the initial state of the problem. A total of $|V|$ un-allocated weights are pending allocation at the root node. Then one vertex v is selected. There can be multiple ways to allocate $w(v)$ to v's *participating cliques*.

Definition 6 (Participating Clique)**.** In a graph $G = (V, E)$, a participating clique of a given vertex $v \in V$ is any clique $G' \subset G$ such that $v \in G'$.

For each way of allocation, a new node is created in the search tree as a child node of the root node. In this new layer of nodes, v has finished its allocation of weight and thus is removed from the graph. After $|V|$ levels, the graph is empty and all allocation decisions are made. The path from root node to each leaf node determines a series of allocation decisions. This series of decisions then determines one *weighted clique cover* of the graph. Recall that our objective is to find a solution to the WCCP that minimizes the target function. This is then translated as, to find a leaf node in this search tree, that minimizes the target function.

For example, consider the coding graph shown in Fig 4.2. The numbers in parentheses denote the weights corresponding to the vertices. The first search decision can be made at the vertex D. Its weight (1) should be allocated to one

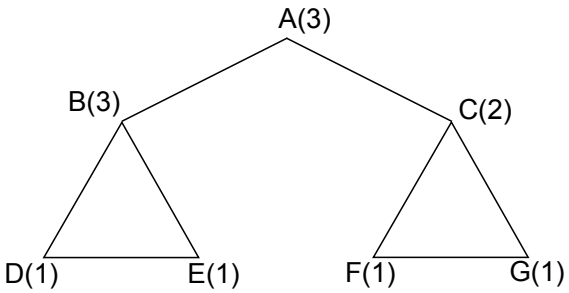


Figure 4.2: One Example of Coding Graph

of its four adjacent cliques (D), (B, D), (D, E) and (B, D, E). So in the first layer, we have four child nodes, each of which denotes one choice of possible allocations.

If we continue the search in this manner, it is a brute-force algorithm. To analyse the computational complexity of this brute-force algorithm, we carefully examine the case at one node in the search tree. Suppose the node under consideration is v and its weight is W. Suppose D is the number of participating cliques of v. Consider the problem to allocate W items to D piles. There are a total of $\binom{W+D-1}{D-1}$ allocation methods, and here is a brief proof. The problem can be converted to another equivalent problem: What is the number of possible allocation methods if we are to allocate $W + D$ items to D piles where each pile has at least one item. If we put these $W + D$ items in a line, inserting $D - 1$ non-overlapping breaks can delimit these items into D piles, and each pile has at least one item. There are altogether $W + D - 1$ places to insert break and we are to insert $D - 1$ breaks. The total number of possible delimitation is thus $\binom{W+D-1}{D-1}$.

The above analysis is on one layer of the search tree only, and the total

search tree has $|V|$ levels. Taking the average for all vertices, the computational complexity of the brute-force algorithm is estimated to be $O(\binom{W+D-1}{D-1}^{|V|})$, where $W = \max_{v \in V} w(v)$ and D is the average number of participating cliques that one vertex can be in. In real cases, since $W \gg D$, we take D as a constant and hence $\binom{W+D-1}{D-1} \approx W^D$. The complexity can be simplified as $O(\binom{W+D-1}{D-1}^{|V|}) \approx O(W^{D|V|})$. This is obviously unacceptable as the scale factors are both on the base and the exponent. So we seek a better algorithm through pruning and approximation.

The detailed pruning rule and the approximation method are described in following subsections. We first give the overall block diagram of the algorithm in Fig 4.3 for reference. This diagram shows how the contents are joined to a total algorithm.

4.2.2.1 Pruning

Before branching on one vertex we need to get all of its participating cliques, and the number of branches is dependent on the number of ways to allocate weights to these participating cliques. The pruning rule below demonstrates that we only need to consider a subset of these participating cliques in certain cases, and hence we can significantly reduce the amount of computation. $N(v)$ denotes the set of neighbouring vertices of vertex v.

Theorem 1 (Pruning Rule)**.** *For a vertex $v \in V$ where $\forall v_i \in N(v), w(v) \leq w(v_i)$, if a participating clique Q_j of v is subsumed by another participating clique Q_k, i.e., $V_{Q_j} \subset V_{Q_k}$, child nodes in the search tree where Q_j is allocated with*

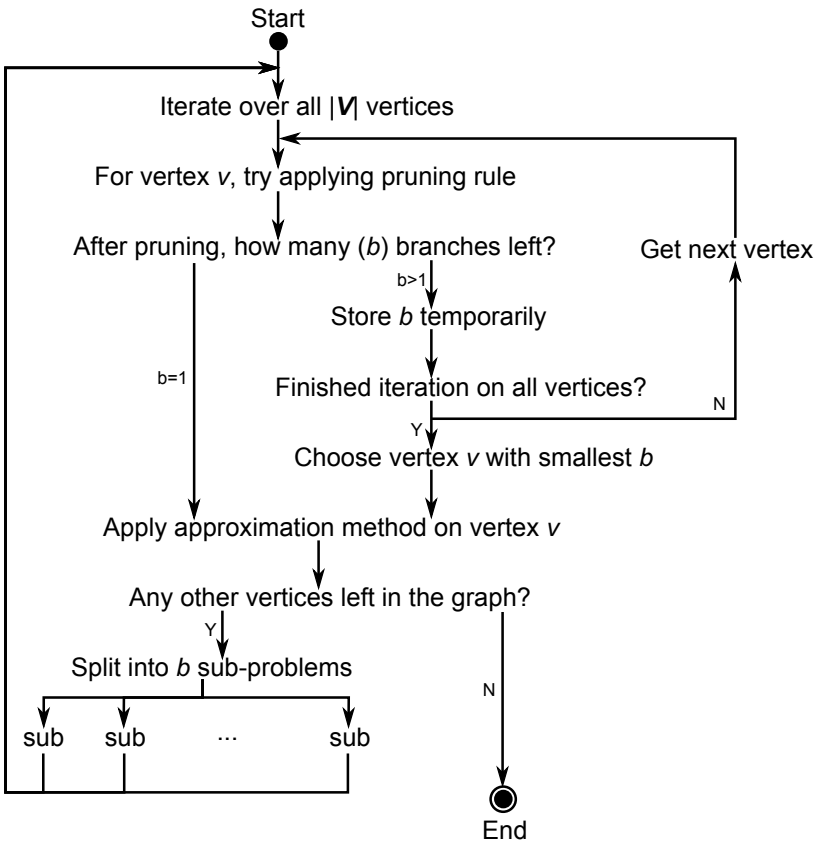


Figure 4.3: The Overall Block Diagram of the Algorithm for WCCP

non-zero weight can be pruned. Stated another way, we only need to consider allocation methods that allocates zero weights to Q_j.

Proof. Suppose in one optimal allocation, vertex v allocates $w^*(0 < w^* \leq w(v))$ to clique Q_j. We use $V_{Q_k} - V_{Q_j}$ to denote the difference vertices set between clique V_{Q_k} and V_{Q_j}. For each $v_l \in V_{Q_k} - V_{Q_j}$, we assert that v_l must have allocated at least $w' = w^* + w(v_l) - w(v) \geq w^*$ to some of its participating cliques where the vertex v is excluded. The reason is stated as below: for every unit of weight of v_l, its allocation may either fall in a clique that involves v, or fall in a clique that does not involve v. With $w(v_l) \geq w(v)$, v_l can allocate at most $w(v)$ to the cliques that involve v. Furthermore, with our assumption that v has allocated w^* to Q_j where v_l is not part of, v_l can then allocate at most $w(v) - w^*$ to the clique that involve v. Therefore, v_l must allocate at least $w(v_l) - (w(v) - w^*)$ to the cliques that does not involve v, just as we have asserted. Consider the following reallocation: For each v_l, we withdraw w^* weights from cliques that do not involve v. This withdrawal either shrinks the vertex set for some cliques, or it totally eliminates some cliques with only one vertex v_l inside. In fact, the latter case will never happen if the prior allocation is optimal. Then these withdrawn weights are merged into the weighted clique Q_j with w^* weights, yielding a weighted clique Q_k with w^* weights. This reallocation will never increase the sum of weights of weighted cliques, so the new allocation is also optimal. This justifies the pruning rule because at least the new allocation, which is optimal, is in the search tree after pruning. □

In the following we give several corollaries to further demonstrate the usage of this pruning rule.

Corollary 2 (One-degree Vertex)**.** *In WCCP, if there exists a degree-one vertex v_j connecting to another vertex v_k, we can prune all other branches and leave only the ones that allocate* $\min(w(v_j), w(v_k))$ *to the two-vertex clique* (v_j, v_k)*.*

Proof. The vertex v_j has only one neighbour v_k, so its participating cliques are (v_j) and (v_j, v_k). Since the former clique is subsumed by the latter one, we can allocate zero weight to the one-node clique (v_j) if $w(v_j) \leq w(v_k)$. For the cases when $w(v_j) > w(v_k)$, we can allocate at most $\min(w(v_j), w(v_k))$ to the clique (v_j, v_k). Applying Theorem 1, we can derive our conclusion in this corollary. □

Corollary 3 (m-degree Vertex)**.** *In WCCP, if there exists an m-degree ($m \geq 1$) vertex v_j, and v_j together with its m neighbours form a $(m+1)$-vertex clique Q_j, we can prune all other branches and leave only the ones that allocate* $\min_{v \in V(Q_j)}(w(v))$ *to the clique Q_j.*

The proof of this corollary follows the same logic as the proof of Corollary 2.

4.2.2.2 Approximation

So far we have discussed the pruning rule. When the pruning rule is carried out at a node, the number of branches at this node in the search tree is significantly reduced. But what happens when no pruning rule is applicable? We use quantization to remove the scale effect of the weights on vertices.

We first determine a parameter Q in the algorithm. Q denotes the level of

quantization of weights used in the search algorithm. Consider a vertex with a weight of W and D participating cliques. In the brute-force search algorithm, we will have $\binom{W+D-1}{D-1}$ branches at this node because there are a total of $\binom{W+D-1}{D-1}$ ways to allocate weights to the participating cliques. In our approximation algorithm, a weight $W(= Q \times m)$ is split into Q equal shares and these Q shares are allocated to participating cliques instead. Therefore, the number of branches at this node is reduced to $\binom{Q+D-1}{D-1}$.

We then analyse the amount of error introduced in the approximation step. We argue that the error involved in the approximation step is upper bounded by the W/Q where W is the sum of weights across all nodes and Q is the level of quantization. The proof is as follows: For a single vertex, its weight can be arbitrarily allocated to its participating cliques. But due to quantization, each participating clique can only be assigned an integer multiple of m as its weight. Suppose that a real optimal allocation exists and is denoted by Σ^*, and the "optimal" allocation found by our search algorithm is Σ. We derive $\Sigma - \Sigma^*$ by comparing the allocated weight for each participating clique. Consider the transformation from Σ^* to Σ. Any negative value in $\Sigma - \Sigma^*$ implies a revocation of weight for the clique in allocation Σ^*; Any positive value in $\Sigma - \Sigma^*$ implies a reallocation to the corresponding clique. Note that any revocation or reallocation involves at most one share of quantized weight, and this is exactly $1/Q$ of the weight in the original problem. The size of the problem is reduced to $1/Q$ of the original problem and we can derive that the error involved is limited to W/Q.

4.2.3 Scalability and Error Analysis

4.2.3.1 Scalability

We use Gilbert's random graph model [100] to randomly generate test coding graphs. This model defines a graph generator that takes two arguments. The generator itself is denoted as $G(n, p)$ in the paper, where n denotes the number of vertices and p denotes the probability by which each edge is selected. For each set of parameter we generate 25 random graphs and test our algorithm as well as a Non-Approximation (NA) algorithm. The *non-approximation* algorithm also has the feature of pruning unnecessary branches, but it does not quantize the weights into blocks. Therefore, it needs to consider *a lot* more branches when no pruning rule is applicable, but the solution it provides is an accurate answer.

Since WCCP solves the problem on a *weighted* graph, weight information should be attached to the randomly generated graphs. In this evaluation, we set the weights assigned to each vertex to be a random variable conforming to a log-normal distribution. By adjusting the mean and standard deviation of this log-normal distribution, we are able to test the algorithm with different scales of weights. The reason we choose *log-normal distribution* is that it is the simplest finite variance distribution with only positive values.

The final objective of this sub-section is to validate our analysis in previous sections. We are interested to test whether our stated algorithm can indeed accelerate the search for a solution to WCCP. Despite we have derived the computational complexity of our algorithm as well as a brute-force algorithm, a

validation in practice can be helpful because we can have a hands-on feeling of the complexity difference. In addition, we are also interested in testing how much error we have introduced in the *approximation* step. We have two criteria, one is the *error rate*, indicating how often an error occurs. The other is *average error*, indicating when an error is seen, how big it is.

In order to make the evaluations, we implemented our algorithm as well as the comparison algorithm with *approximation* mechanism disabled. The performance of our algorithm is then tested with extravagantly complex inputs. The reason we deliberately magnify the size of problem in our experiments is that we can only measure the elapsed time with higher confidence when the solver has run for sufficiently long time. Otherwise, the result can be easily affected by other out-of-the-system factors like operating system scheduling, system load, etc.

We run the solver in a computer with the following setting: Intel i5-3570 3.40 GHz, 8 GB memory, Windows 7 Enterprise 64-bit SP1. Because some of the test graphs are so complicated that it is impractical to wait until their finish, we opt to estimate their finishing time according to elapsed time and finished proportion. When the program has run for 10 minutes and still running, we use the following method to estimate the total time required to finish the search algorithm. Consider the search tree is constituted of several layers. We started searching from the root node and searched in a depth-first manner. At the end of 10 minutes, a proportion of the nodes in the search tree have been visited while the others are not. We first estimate the time required to finish

the inner-most search sub-tree by dividing the elapsed time in this sub-tree by the amount of nodes visited in this sub-tree. This estimation is done recursively to outer nodes and finally reaches the root node. For example, suppose we are currently working on the fourth layer, finished *two* branches but has *three* more to go under this decision node when the limit of 10 minute is hit. We first estimate the total amount of time required to finish this node by multiplying $(2 + 3)/2 = 5/2$ to the elapsed time on this node. Since all branches in this layer-4 node were finished, we go back to its parent node, a layer-3 node, repeat the above method to estimate the total amount of time required to finish this layer-3 node. This method is to be repeated recursively back to the root node and the total amount of time required to finish the search tree is calculated. The underlying assumption of this method is that the searching complexity in the first 10 minutes is an effective estimator for the overall searching complexity. In each layer, the time required to process the branches we have yet to reach is assumed to be proportional to the number of these branches and the average time to finish one branch in the past.

We have tested the validity of this estimation method for some of the scenarios when the estimation is above 10 minutes but below 2 hours, the estimation error is generally within 20%. Since we are more concerned about the order of magnitude instead of the absolute value, this error is acceptable. In addition, even though the 25 graphs are generated using the same set of parameters, their processing times can span from milliseconds to days or more. In order to capture this distribution, we take the median value among these 25 values and draw the

Table 4.1: Default Parameters for Performance Evaluation of the Static-Form Problem

Parameter	N	p	mean(w)	stdev(w)	Q
Value	20	0.15	100	50	10

figures.

The default parameters when running the experiments are shown in Table 4.1. This default set of parameters is chosen after weighing two factors. Firstly, we want the scenario to be of some complexity. Secondly, we want to keep the size of the maximum clique to be smaller than 6. This number is worked out in [13] as the theoretical maximum clique size in a coding graph. By setting the probability of generating a 6-node clique to be 5%, we work out this set of parameters. Then we evaluate the performance of the algorithm by adjusting each single parameter. Note that the ratio between the mean and the standard deviation is fixed at 2:1 for the log-normal distribution. Fig 4.4 and Fig 4.5 show that 1) the processing time grows for both algorithms when the size of the problem grows, 2) our algorithm consistently runs faster than the non-approximation algorithm, and 3) the improvement (as large as several orders of magnitude) is most significant when the scale of the problem is big. Fig 4.6 measures the performance with different mean weights. It is clear that our algorithm is roughly invariant to the weight factor while the non-approximation algorithm grows exponentially with the mean weights. Please be reminded that the indicated "processing" time in the graphs, whenever above 10 minutes, is not the real measured value, but a predictor of the true processing time.

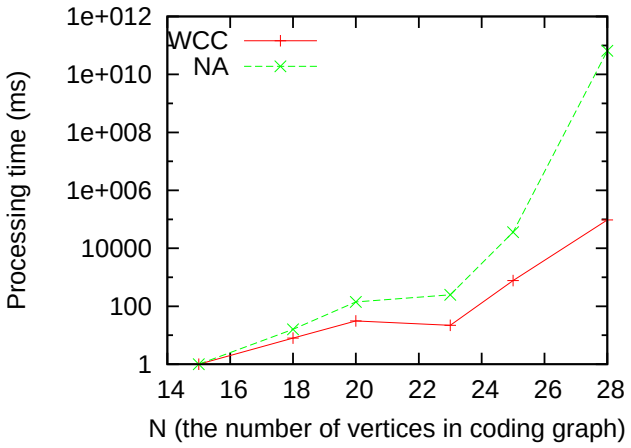


Figure 4.4: Processing Time Sensitivity to N

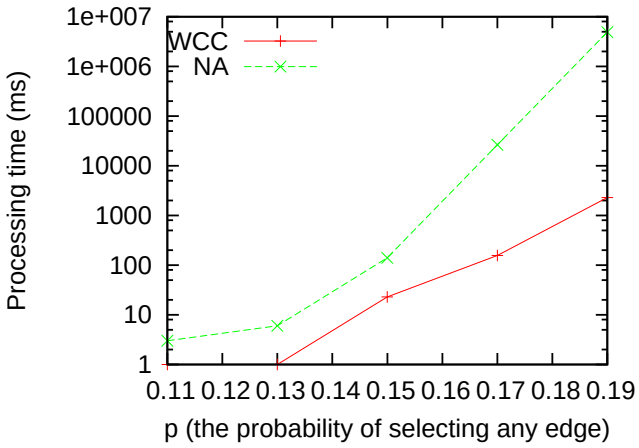


Figure 4.5: Processing Time Sensitivity to p

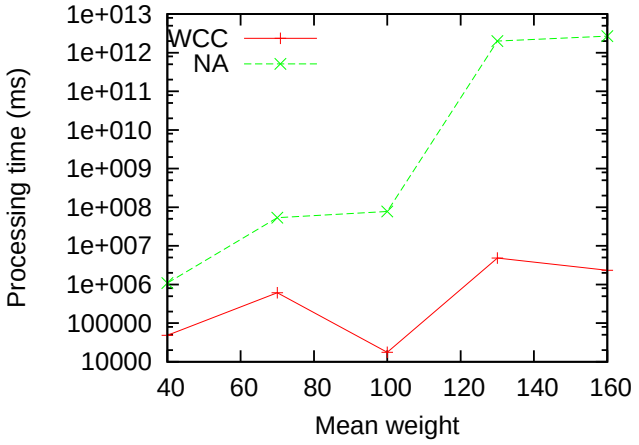


Figure 4.6: Processing Time Sensitivity to mean(w)

4.2.3.2 Error Analysis

Even though we have derived an upper bound for the approximation error introduced in Section 4.2.2.2, we are still interested in typical values. In fact, we observe in most cases the error level is significantly lower than the upper bound. Given a set of parameters, we define the *error rate* as the probability of getting a scenario that results in an error out of all the generated scenarios. The *average error* is defined as the average amount of error over all the cases. Since the program stops at the 10th minute, any scenario with a running time longer than 10 minutes does not come with a precise result value. Therefore, we only sample the ones that have results worked out within 10 minutes.

Surprisingly, the measured error is significantly lower than what we have expected. Within the three sets of experiments where we adjust N, p and mean(w), the error rates are 1.92%, 1.08%, and 6.38%, respectively. The average errors are also very small, at 0.17%, 0.09%, and 0.15% respectively. So the overall error rate is much smaller than the upper bound we set, which is 5%. Note that the overall error rate is given by the *error rate* multiplied by the *average error* as we have defined. Further analysis leads us to conclude that this is because of the limit we put on the processing time. For those test cases where the processing time is longer than 10 minutes, there is a higher probability of getting an error because of the greater amount of branching. Now that we have excluded those cases, the measured error level is underestimated. Nevertheless, this experiments at least gives us some hints on the level of error introduced. It also demonstrates that in real-life scenarios with real-time processing requirements,

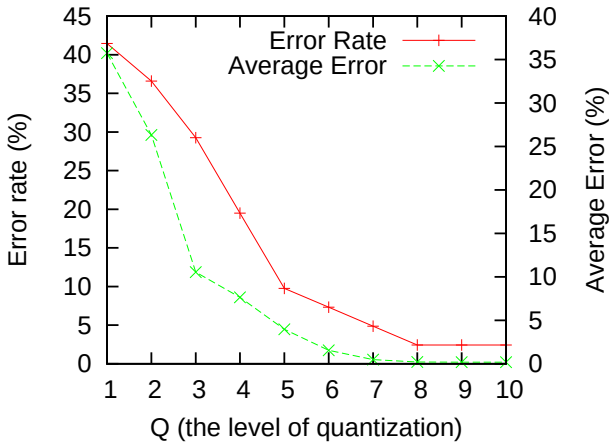


Figure 4.7: The Error Rate and Average Error with Different Q Values

the error introduced is negligible.

We also measure the amount of error introduced by adjusting the parameter Q. A larger Q means a finer quantization and usually less error as analysed in Section 4.2.2.2. A smaller Q will lead to higher levels of errors. The extreme case of $Q = 1$ is in essence a trivial "all-or-none" algorithm. From Fig 4.7 we observe 1) our algorithm performs significantly better than the trivial solution and 2) a Q value greater than 5 can give good-enough performance (with empirical errors smaller than 5%). The values in Fig 4.7 are obtained by running the experiment with default parameters 50 times. Both the error rate and the average error are presented in the figure.

4.3 The Dynamic Form of the Problem

So far we have analysed the scheduling problem where no new packets would arrive. In this section, we will relax this restriction and study the problem when new packets are arriving while queued packets are being processed. The

transmission capacity is also considered as it can be a restricting condition.

Before designing the scheduling method, we need to define the optimization objective. When taking into account transmission capacity, we need to consider the trade-off between throughput and fairness. Note that existing coding-ignorant scheduling methods (e.g. Round-robin) have no control over this trade-off at all. In order to avoid unnecessary complexity in describing the scheduling scheme, we discuss only two extreme cases. The first is putting maximal throughput at the highest priority. Among all possible ways to maximize throughput, we choose the one that achieves the best fairness. The second case is the other way round, putting fairness over throughput. In fact, the second case is more sophisticated and its solution reflects all the tools we use in designing the scheme for the first case. So only the second case is expanded in the following.

A simplified solution to the dynamic problem would be to blindly apply the solution from the static problem with slight modification. Consider that the algorithm for the static problem can be applied successively to the dynamic form of the scheduling problem periodically. We only need to substitute the number of packets in the packet queues with the expected number of packets that would arrive at each assessment instant. The scheduling decisions obtained this way can then be used to schedule packets dynamically.

However, this approach is subject to several problems. The adverse effect of dynamic packet arrivals is two-fold:

1. Some flows may have no queued packets when the node is ready to transmit. This waste of bandwidth would decrease the throughput.

2. Some flows may have packets coming faster than they are being processed, resulting in queue overflows. The static solution has never considered the transmitting capacity of nodes and thus cannot guarantee better performance when the network is saturated.

Dynamic packet arrivals may introduce opportunities as well. In the static setting, very often some "leftover" packets cannot be coded or fully-coded because there are no other packets left to be coded with. This situation is very much alleviated in the dynamic case as this shortage of packets will recover sooner or later.

There is another important change resulting from random packet arrivals. In the dynamic case, packet arrival intervals are random variables that conform to certain distribution, if at all we can model the arrivals. Since the inputs to the system are not deterministic but probabilistic, the outputs, or the solution can not be deterministic neither. Moreover, due to the fact that packet arrivals are usually bursty and error-prone, it is very difficult to work out an "optimal" solution even in the probabilistic sense. But fortunately, the lack of "optimal" solution does not mean nothing can be done to improve the round-robin scheduling. In fact, we take a heuristic approach to achieve a better scheduling decision.

4.3.1 Heuristic Scheduling

The objective of dynamic scheduling is to get higher throughput given the capacity of the network, and to maintain a relative fairness among all flows when the network is saturated.

The starting point of this scheduling method is the trivial solution we mentioned above. First we translate the offered loads of flows into the expected number of packets that would arrive in one second. These figures are used as the weight in WCCP and a static solution is obtained in the form of a set of weighted cliques. We will refer to this solution as the *initial plan*.

The first heuristic rule is to overcome the restriction of transmission capacity. The capacity of a node can be estimated by monitoring its transmission history. When the capacity is smaller than the arrival rate, flows with heavy traffic should be cut off while flows with less traffic remain intact. Basically we use the *Generate-Quota* algorithm shown in Algorithm 3 to allocate the limited transmission capacity. The final set of weights for all cliques obtained in this step is called a *quota*, and it is the starting point for further steps in solving the dynamic form of the scheduling problem.

Here we briefly explain the rationale behind the algorithm. We describe a simplified version of the *Generate-Quota* algorithm. The first step is to solve a WCCP where all vertices are assigned a weight of one ($\forall v, w(v) = 1$). If the transmission capacity is higher than the amount of transmissions required by this WCCP, we proceed to solve a WCCP where all vertices are assigned a weight of two ($\forall v, w(v) = 2$). This continues until either when we deplete the transmission capacity, or when the uniform weight u reaches the minimum weight among vertices. For the first case, we understand that we can transmit u packets for each flow with the limited transmission capacity, but cannot transmit $u + 1$ packets. So setting the quota to u for each flow maximizes the minimum

per-flow throughput. For the second case, we have spare transmission capacity after fulfilling the flow with minimum offered load. An amount of u weights are deducted from all vertices, yielding a weighted graph with less vertices (because there is at least one vertex that has only u weight). Then, the rest of the transmission capacities are allocated to the new smaller weighted graph following the same procedure as above to further reduce the per-flow throughput variance.

The above-described simplified version requires too many iterations, so we introduce the algorithm described in Algorithm 3. In Algorithm 3 we have better guesses of the uniform weight to start with, instead of the fixed scheme from 1 with step 1. For example, we jump directly to w_{min} if the given transmission capacity can cover the least weight vertex (cf. Line 9). If that is not the case, we also improve the search for maximum u by starting from $\lfloor \frac{C}{X_1} \rfloor + 1$ (cf. Line 15). The actual value of maximum u is always bigger than $\lfloor \frac{C}{X_1} \rfloor + 1$, because by duplicating (Γ_1, X_1) to $\lfloor \frac{C}{X_1} \rfloor + 1$ times, the total amount of transmission capacity C is still not exhausted. This improvement can significantly reduce the number of iterations especially when the average amount of weight is big. The characteristics that we have maximized the minimum per-flow throughput and minimized the per-flow throughput variance are retained after the performance tuning. In the pseudo-code of the algorithm, we use Σ and Γ to denote the quota and scheduling methods, respectively. But in essence, both of these two notations are sets of weighted cliques. Φ is used to denote the empty set.

The second heuristic rule is to exploit the order of packet arrival. In order to enforce a quota plan with out-of-order packet arrivals, we employ a proba-

```
1  Function Generate-Quota(G, C)
       Input: A weighted graph G = (V, E, w(·)), transmission capacity C
       Output: Quota Σ
2      if V is empty then
3          return Φ;
4      end
5      w_min ← the minimum vertex weight in G;
6      Γ_1 ← Uniform-WCCP(G, 1);
7      Γ_{w_min} ← Uniform-WCCP(G, w_min);
8      X_1, X_{w_min} = |Γ_1|, |Γ_{w_min}|;
9      if C ≥ X_{w_min} then
10         G' ← (G − Γ_{w_min});
11         C' ← (C − X_{w_min});
12         Σ_u ←Generate-Quota(G', C');
13         return Γ_{w_min} + Σ_u;
14     else if X_1 ≤ C < X_{w_min} then
15         for u = ⌊C/X_1⌋ + 1 to w_min do
16             Γ_u ← Uniform-WCCP(G, u);
17             X_u = |Γ_u|;
18             if X_u > C then
19                 G' ← (G − Γ_{w_{u−1}});
20                 C' ← (C − X_{w_min});
21                 return Γ_{w_{u−1}} + Generate-Random-Quota(G', C');
22             end
23         end
24     else
25         return Generate-Random-Quota (G, C);
26     end
27 Function Uniform-WCCP(G, u)
       Input: A weighted graph G = (V, E, w(·)), uniform weight u
       Output: Scheduling method Γ
28     modify w(·) in G such that all weights are set to u;
29     return the solution of WCCP with the modified G;
30 Function Generate-Random-Quota(G, C)
       Input: A weighted graph G = (V, E, w(·)), transmission capacity C
       Output: Scheduling method Γ
31     Γ ← Φ;
32     for i = 1 to C do
33         randomly select a maximal clique c in G;
34         Γ ← Γ + c;
35     end
36     return Γ;
```

Algorithm 3: Generate-Quota Algorithm

bilistic scheduling method. If multiple cliques are ready to transmit (i.e., their corresponding flows have packets in queue), each of them is given a probability to transmit in this round. The probability for one clique is its quota minus the amount that has been transmitted in this instance, and then this count is normalized by the sum across all cliques. The randomness achieved by this probabilistic scheduling method avoids many problems faced by a fixed scheduling method. There is no bias towards any flow due to the absence of a sequential decision model. Cliques with higher quota are given higher probabilities in the beginning. With more packets sent for this clique, its probability would decrease and other cliques will have higher probabilities. In the equilibrium state, the number of packets sent for each clique would be generally proportional to its quota. In cases when no clique has positive probability, the first packet in the queue is sent regardless of whether other packets in its clique are ready.

The third heuristic rule is to exploit the opportunity to code more "leftover" packets. In our scheduling scheme, packets are checked for on-the-spot coding opportunities before sending out. For example, there may be a single-vertex clique representing one flow in the initial plan and quota plan. When this clique is set to transmit as a result of the probabilistic scheduling, we check for hidden coding opportunities right before sending out packets from this flow, i.e., if this packet can be coded with another packet in the same queue, they are coded together and sent with the other flow marked with one packet reserve. In future schedules, flows with a reserve are eligible for probabilistic scheduling even if they do not have packets in their queue. The reserve works as if the packets

are not sent until the reserve is used. The mechanism of packet reserve can take advantage of the opportunities to code more packets, without disturbing the existing quota-based system. It also adds more flexibility to the scheduling scheme.

All heuristic rules work together toward the same objective. When the network is able to sustain all flows, we try to stick to the static scheduling scheme because this is the scheduling that maximizes the throughput. When the network cannot satisfy all flow traffic requirements, we aim to maximize the minimum throughput among all flows, which is the most frequently used metric for measuring fairness. It should be noted that the heuristic scheduling method may not yield the optimal solution, but we demonstrate that we can nevertheless achieve reasonable performance gain.

4.3.2 Performance with Poisson Arrivals

The effectiveness of our heuristic scheduling rules is examined next. In this subsection, we test our scheduling scheme with Poisson packet arrivals. MASON [101] is chosen to be the simulator used because it is a fast discrete-time Java-based simulator. In this simulation, we simply model the behaviour of a queuing system. We are interested to see whether, in an isolated environment, our scheduling scheme can perform better than scheduling schemes that are straignt-forwardly extended for coding awareness. A more extensive and more realistic simulation is done in Section 4.4 so as to evaluate the overall performance of our scheduling scheme with all network factors considered.

Two frequently used scheduling schemes are taken as the comparison group. One is First-In-First-Out (FIFO)-based, and the other is RR-based. Basically these two packet scheduling schemes cover most existing literatures on network coding protocols. In the FIFO-based scheduling scheme, all packets arrive at a single queue, each with a tag denoting which flow it is coming from. When the node obtains permission to transmit, the first packet in the queue is dequeued. The packet backlog then undergoes a check for maximal clique. Finally, all flows from this maximal clique dequeue one packet each, encode them and transmit to the medium. On the other hand, the RR-based scheduling scheme maintains multiple queues, each for a flow. The scheduler then polls these queues in a round-robin fashion. This scheme overcomes the problem in FIFO-based method that a busy flow can crowd out all other flows.

Among all generated coding graphs using the default parameters (cf. Table 4.1), the one that corresponds to the median processing time is used in this simulation. This choice guarantees that the generated coding graph is neither too plain, nor too complex. In this test bench, a total of 20 flows are present. The total offered load is 1868 Packet Per Second (pps), with the minimum flow being 48 pps and the maximum being 282 pps. Each simulation is run for 60 seconds (time in simulation). The average throughput and minimum throughput among all flows are sampled from the end of the 10th second until the end of the simulation. After 10 seconds, the system has already stabilised, so we choose to measure the throughputs from 10 seconds onward to the end of the simulation. All simulations are repeated with 10 different random seeds and

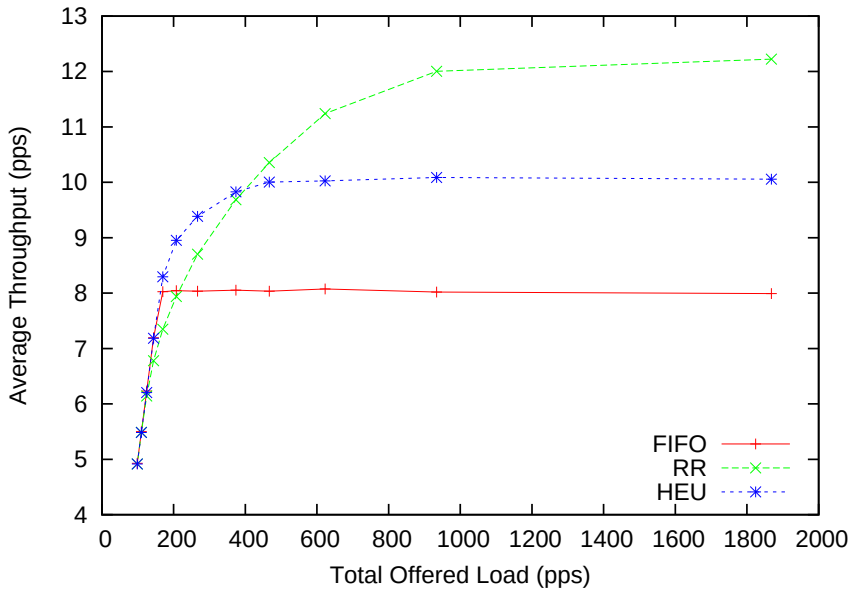


Figure 4.8: Average Throughput Performance with Poisson Arrivals

the average/minimum per-flow throughput is plotted in Fig 4.8 and Fig 4.9, respectively.

In the static version of this scheme, a round-robin based scheduling scheme requires 1198 transmissions to clear the queued packets. However, our static solution requires only 1010 transmissions. This is almost a 20% improvement.

In the dynamic case, the average throughput improvement is not as significant. We scale the offered packets with a multiplier and generate different offered loads. The distribution of service time follows exponential distribution. We set the mean service time to be 100 packets per second. It is observed that our Heuristic Scheduling Scheme (HEU) can outperform RR when the offered load is below 400. This is four times the transmission capacity. The maximum improvement can reach 13%. This amount is smaller than the 20% obtained in the static form. This is generally because our heuristic scheduling method bal-

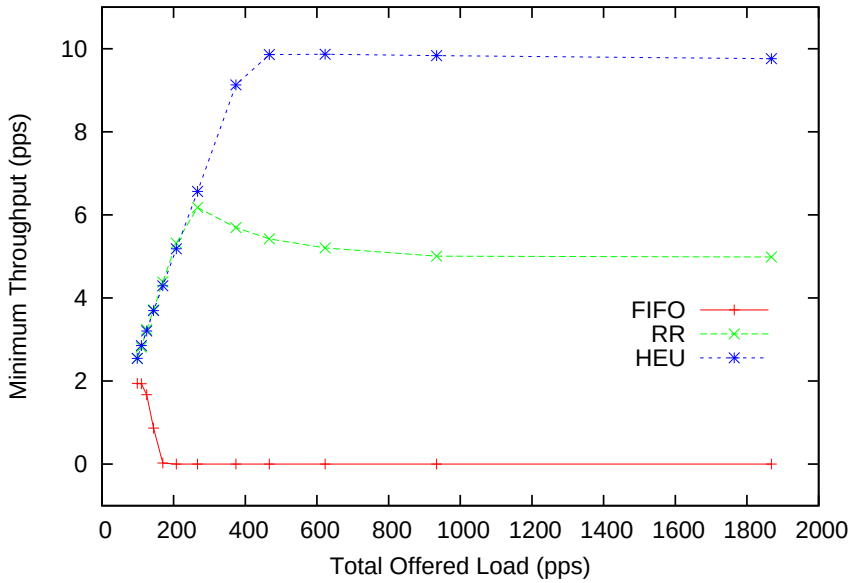


Figure 4.9: Minimum Throughput Performance with Poisson Arrivals

ances between better performance and higher anomaly tolerance. Only if we can always stick to the optimal scheduling scheme can we achieve the highest performance improvement. But a fixed scheme cannot adapt to the dynamic network conditions. With the introduction of probabilistic scheduling and the allowance mechanism, we achieved adaptability to network contingencies at a small cost of potential performance gain. When the offered load continues to rise above 400, RR provides higher average throughput as compared to HEU. At the same time, we turn to Fig 4.9 for more insight about this under-performance. One can soon discover that the minimum throughput of HEU is slightly below 10 pps while the average throughput is slightly above 10 pps. This indicates that our scheme achieves the fairness by allocating almost equal-throughput channel for each flow. The RR scheme can provide around 12 pps average throughput, but its minimum throughput is only around 5 pps. This is most conspicuous for the

FIFO scheme where the minimum throughput is minimal. In fact, from 250 pps offered load and above, HEU can provide better fairness than RR. But it only starts to under-perform in terms of average throughput after 400 pps.

To sum up, when the network is not severely saturated (as in this simulation, when the average transmission capacity is 100 packets per second and the total offered load is below 400 packets per second), our heuristic scheduling scheme for the dynamic case can provide higher average throughput across all flows. At the same time, our scheduling scheme is able to schedule packets in a way such that no flow suffers from starvation. In particular, we managed to achieve the highest minimum per-flow throughput. It overcomes the crowd-out effect of other scheduling schemes when the offered load is too high and some flows may suffer from low or even zero throughput.

4.4 Evaluation

In this section, we evaluate our work in a more complex network environment. Specifically, an environment with upstream/downstream nodes, contentions, and interferences. We first tune our heuristic scheduling scheme to counter the adverse effects we may encounter in real networks. Then the tuned scheduling scheme is tested in Qualnet simulations where a set of network topologies with multiple nodes are constructed.

There are three assumptions in the evaluation of our heuristic scheduling scheme. But they are not immediately valid in the real network settings.

1. The offered load of each flow is known and fixed.

2. The transmission capacity of each node is known and fixed.

3. The arrival of packets follows Poisson process.

Therefore, we have to make adjustments to fit the scheduling scheme in a real protocol. Hence we designed Dynamic Packet Scheduling Algorithm (DPSA), on top of the routing protocol proposed in Chapter 3—SCAR. SCAR is a routing protocol based on DSR [89]. Routes are selected in a on-demand fashion and source routing is used to navigate packets through the network. SCAR is able to discover coding opportunities and compare potential throughput gain among multiple paths.

The first adjustment is the design of a cross-layer channel and a rate control scheme. Using the cross-layer channel, application layer can give hints on the offered load to lower layers. The rate control mechanism, being an extension to congestion control mechanism, is able to adjust the offered load of flows for both congestion and transmission efficiency concerns. In short, this mechanism can cut off "excessive" offered load for some flows while encouraging higher offered load for other flows. Put in the context of Algorithm 3 (The *Generate-Quota Algorithm*), excessive offered load is the amount of weight of vertices that fail to get included in the scheduled quota. According to the calculated quota, rate control packets for certain flows are sent backwards from downstream nodes to upstream nodes. When receiving the rate control packet, the source node for the flow can decide to tune down its offered load. More information on the cross-

layer channel and the rate control scheme can be found in next Chapter, where we extensively study how can we exploit the synergy between network layer and application layer.

Since we mentioned rate control, it is worth noting that the TCP congestion control mechanism is disabled in our simulations. It is understood that TCP's window-based congestion control is not suitable for scenarios with network coding[28]. This is because packets are frequently out-of-order with the introduction of network coding. Out-of-order packets have the implication of a congestion in the TCP protocol, and TCP will limit the offered load based on this false alarm. This can cause a severe performance degradation, and thus we use our own rate control mechanism described above to replace the TCP congestion control mechanism.

The second adjustment is the delivery of offered load information from the upstream nodes to the downstream nodes. Consider a flow with three nodes, S, I and D. Node S can rely on the cross-layer channel to estimate the offered load. For intermediate nodes like I, a propagation of offered load information is required from the upstream node to the downstream nodes. The propagated offered load information, adjusted by packet drop rate, finally yields the offered load perceived by the intermediate nodes. An upstream node calculates the offered load for a downstream node by subtracting the amount of packets dropped from its own offered load, periodically. This delivery of information is encoded in the source routing header.

The third adjustment is the introduction of changing transmission capacity.

This is a critical value used in the heuristic scheduling scheme. Overstating this value will generally lead to unfair scheduling and understating this will lead to lower throughput gain. In our protocol, this value is measured as the number of packets processed per second, averaged through the recent 5 seconds. Therefore, it can to some extent reflect changes in the network condition, but filter out unnecessary noise.

4.4.1 Simulation 1

We choose a circle topology in Simulation 1 as shown in Fig 4.10. First we fix a central node O and the radius r of a circle. Then we randomly generate an angle θ_1 between 0° and 360°. Node A_1 is added on the circle such that the line segment that connects A_1 and O intercepts the positive axis with an angle θ_1. This process is repeated N times until N nodes are added on the circle. The traffic patterns are randomly generated as long as the source node and the destination node are not within transmission range (R) with each other [13]. A total of e flows are selected and all these flows use node O as the intermediate node. The experimental parameters are described in Table 4.2. The generated topology and the coding graph is shown in Fig 4.11 and Fig 4.12, respectively. For simplicity, we manually choose node O as the intermediate node for all flows. The 802.11 protocol stack is employed to construct the network.

The comparison group is chosen to be SCAR as described in Chapter 4, which uses round-robin scheduling. In this simulation, each run of simulation is repeated 10 times with different random seeds and the measurements are

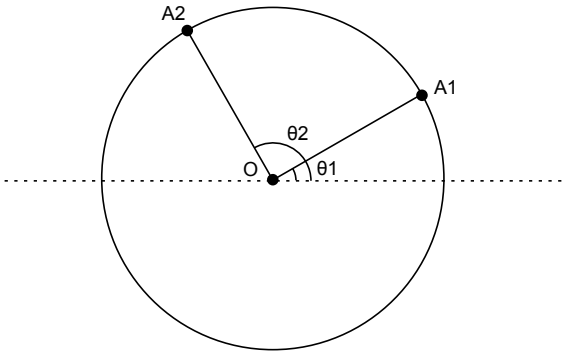


Figure 4.10: A Topology where Source and Destination Nodes are on the Same Circle

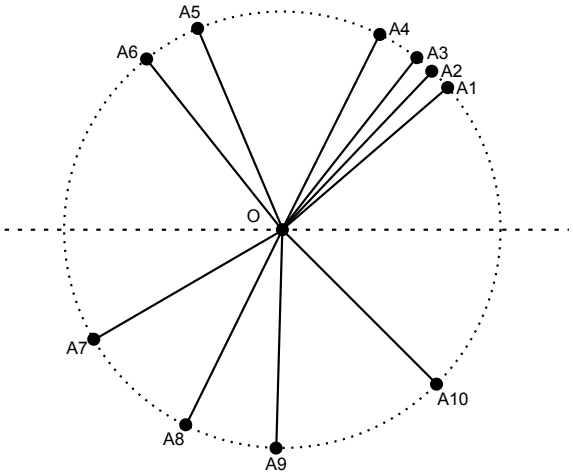


Figure 4.11: The Generated Topology for Simulation 1

Table 4.2: Simulation 1 Parameter Settings

Parameter	**Value**	**Parameter**	**Value**
PHY/MAC layer protocol	802.11b	NETWORK layer protocol	DPSA
# of nodes N	10	# of flows e	10
Radius of the circle topology r	100 m	Transmission range R	150 m

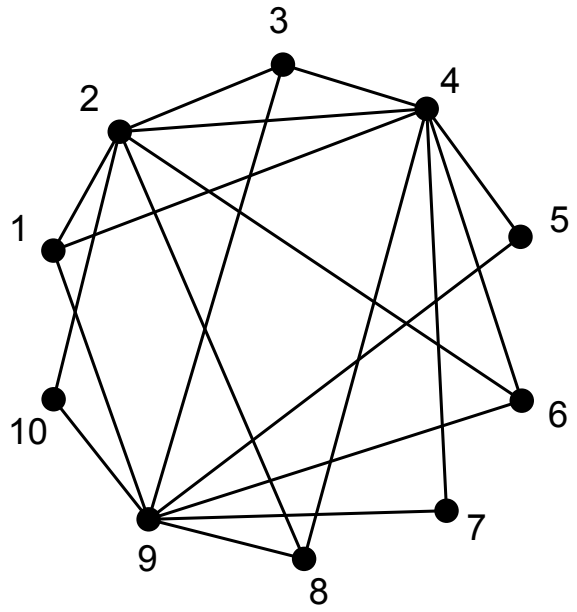


Figure 4.12: Coding Graph at Node O in Simulation 1

averaged.

The per-flow average throughput is shown in Fig 4.13, with the error bars showing the minimum and maximum per-flow throughput. Note that the points are slightly shifted along the x-axis for visual clearance. The exact x value should be the mid point between each pair of data points. It is observed that DPSA can initially provide higher throughput than SCAR. This is because the DPSA systematically schedules the packets to be coded together, minimizing the number of transmissions to clear packets in queue. In addition, DPSA almost provides a straight line connecting the zero point to its highest achievable throughput. On the other hand, the curvature of the SCAR line is much higher.

However, DPSA's maximum throughput level is lower than SCAR's. This happens when the network is fully saturated, i.e., the transmission capacity of the router node is smaller than the offered load. DPSA schedules packets in an

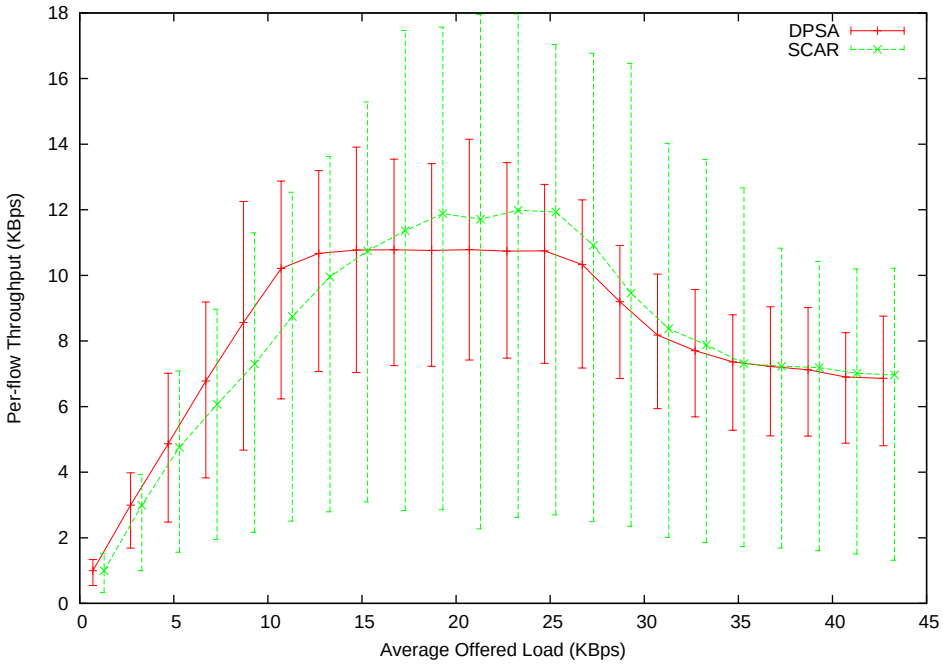


Figure 4.13: Per-flow Throughput in Simulation 1

order that prioritizes fairness. In terms of the coding graph, DPSA intentionally shifts transmission capacity from those well-connected vertices to the ones that have less neighbours. The result of this shifting leads to improved performance. The throughput span is significantly narrowed, meaning smaller variance and better fairness. DPSA can fairly allocate bandwidth to all flows—the minimum throughput among all flows is roughly 70% of the average throughput. On the other hand, some flows receive only around 20% of the average throughput in SCAR.

As the offered load continues to rise, the average throughput of both protocols decline. This is mainly because of MAC-layer contentions, collisions and retransmissions. The router node has decreasing transmission capacity because the media is more often occupied by the source nodes. The hidden terminal

problem is more severe at higher offered loads.

4.4.2 Simulation 2

In this simulation we take a random topology to mimic real-life ad hoc networks. The topology is shown in Fig 4.14. We randomly choose 10 S-D pairs and generate traffic according to a randomly generated ratio.

In Fig 4.15 we have plotted the average throughput versus the average offered load. In this simulation, we can still observe that DPSA provides higher throughput than SCAR when the network is not yet saturated. However, the difference in the average throughput of a saturated network almost vanishes. This can be explained by the cascading loss effect: In Simulation 2 most flows have more than two hops. The scheduling decision made at one node can affect the offered load of the next. Without proper planning (as in round-robin scheduling), the mismatch between offered load and transmission capacity would cause a great proportion of packets to be discarded along the way. The more hops one flow has, the bandwidth loss becomes more severe. This is a great waste of bandwidth, which effectively decreases the maximum throughput achievable by the round-robin scheduling. Also note that DPSA manages to schedule packets more fairly, and the scheduling maintains a good linkage between upstream nodes and downstream nodes, minimizing the cascading loss effect. In addition, the significant decrease of throughput as seen in Simulation 1 is not present. This is simply because the MAC-layer contention is not as severe as in Simulation 1's topology.

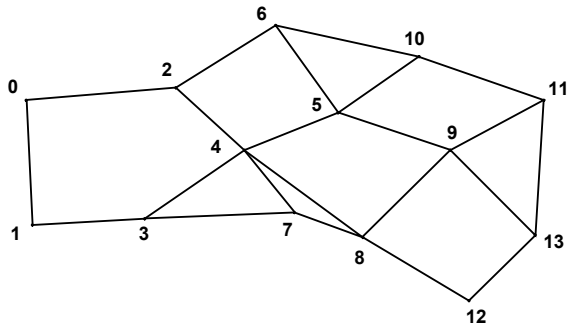


Figure 4.14: Random Topology

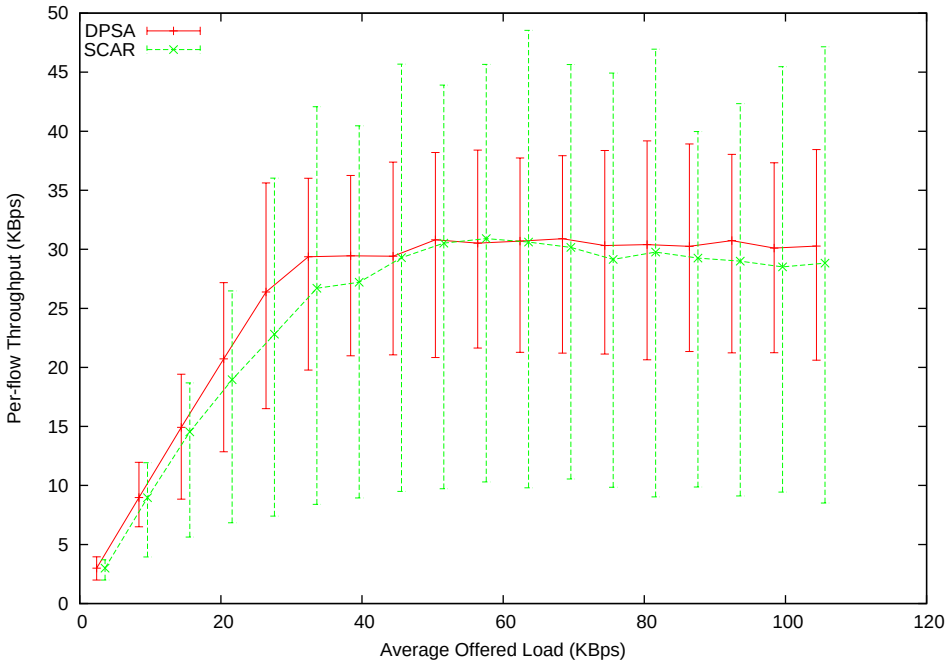


Figure 4.15: Per-flow Throughput of Simulation 2

4.5 Chapter Summary

In this chapter, we have raised an area of problem that was long been neglected—the packet scheduling scheme with network coding enabled. Although a trivial round-robin method can provide decent results, an optimized scheduling method can push the boundaries further to reach optimality. To make things worse, the MAC contention and interferences can significantly degrade the performance of round-robin scheduling. This gives rise to our work described in this chapter.

The optimal scheduling problem is tackled first in a simplified and static version. With our scheduling method, intermediate nodes can clear packet queues faster. This implies a higher throughput at one node. The scalability of the algorithm is also tested. The findings are then applied to the dynamic form of the scheduling problem where we consider multiple queueing systems. Although it is hard to analyse the system, we proposed a heuristic scheduling method addressing the challenges posed by a dynamic system. The solution to this dynamic problem is finally incorporated into a routing protocol. Simulation results for this routing protocol reveal that with proper scheduling, we can have better control over the fairness of the network. The coding efficiency is maximized without compromising the overall fairness. When fairness is not a concern, one can easily adapt our protocol to put more emphasis on those higher coding gain pairs to improve throughput.

Chapter 5

Content Distribution in Wireless Ad Hoc Networks with Network Coding

5.1 Introduction

Peer-to-peer file sharing has contributed a significant amount of network traffic on the Internet. Time has proven that P2P techniques are very effective methods for content distribution [80, 88]. Content distribution in wireless ad hoc networks is also an area that has been extensively studied [82, 87]. This chapter considers the problem of applying network coding techniques on wireless P2P transmissions.

Network coding is quite suitable for P2P transmissions because of the inherent existence of coding structures, such as reversed flows. A flow from node

S, through some route, arriving at node D, is definitely coding-possible with another flow from node D, through the same but reverse route, to node S. P2P networks are based on this tit-for-tat traffic pattern, thus have a lot of reversed-flow-like coding opportunities.

Although traffic patterns are amenable to network coding, directly applying network layer coding schemes on P2P protocols gives unsatisfactory results [81]. We need to examine the problem more carefully and work out a solution that maximizes the synergy between wireless P2P and network coding.

The synergy between the network layer and the application layer derives from the extra source of information. Specifically, when given the routing information in the network layer, the application layer can evaluate the appropriateness of choosing a certain peer better. In wired networks, this may not be as significant because all peers are relatively indifferent to one source node. But in wireless ad hoc networks, choosing a peer that is very far away can degrade the throughput performance. Given the routing information, it is now possible to choose peers that have better routes, thus enabling higher delivery efficiency. Secondly, routing metrics can now be measured more accurately due to the information from the application layer. This is because coding opportunities are highly dependent on traffic patterns. So are the routing metrics. With the offered load information from the application layer, intermediate nodes can predict future coding opportunities in a quantified manner. Lastly, the application layer can now adjust offered load dynamically according to the packet scheduling scheme at intermediate nodes. It can be expected that the adjusted offered load can

yield a weighted coding graph that is more friendly to packet scheduling, and improve the coding efficiency as well as fairness among flows.

In this chapter, we propose a cross-layer solution for the wireless P2P content distribution problem. In a wireless ad hoc network, the scenario we consider is that one node possesses the file to be shared among other nodes. The content discovery is done through broadcasts of meta-data of this file. Our cross-layer design aims to minimize the distribution time. The cross-layer considerations can be beneficial to the application layer by providing routing conditions for peer nodes. Therefore, peer-selection decisions can be done in a more informed way. In addition, the application layer can hint to the network layer on the offered load, thus enabling better routing decisions and packet scheduling order. Furthermore, the network layer could suggest to the application layer to adjust offered load for a particular flow only, with the aim to maximize network coding benefits.

The contributions of our proposed new scheme include:

1. Formulated and improved the coding-aware routing protocol. In particular, we point out the usually neglected self-interference in calculating the routing metric.

2. Applied the routing metric in the peer-selection strategy. This enables a peer-selection criterion that considers more than just the hop count. It also minimizes the reliance on optimistic unchoking to discover better peer choices.

3. Designed a full-fledged APP-cum-NETWORK layer protocol stack to realize wireless P2P content distribution based on the ideas proposed above. The evaluation of this solution is based on simulation.

The content of this chapter is organized as such: The key components of the protocol stack are analysed in Section 5.2 and Section 5.3 for the network layer and application layer, respectively. Further details of the protocol are described in Section 5.4 where simulation results are analysed as well. Finally we conclude in Section 5.5.

5.2 Routing Metric

The network layer can benefit from cross-layer considerations leading to better routing decisions. With the offered load information hinted by the application layer, the network layer can better quantify current coding benefits and predict future coding opportunities. These are reflected in a comprehensive evaluation metric. This metric should be able to weigh the benefits brought about by network coding against other conditions in the network. The routing metric we adopt is based on the one proposed in the Distributed Coding-Aware Routing (DCAR) scheme [28]. We extend this metric in two aspects:

1. We find that the calculation of the MQ in DCAR's formulation is incomplete and sub-optimal. This is extended with a mathematical formulation of the problem, a completeness-proved search algorithm, and a practical pruning and approximation implementation method.

2. Though DCAR takes interference into consideration, a significant part of interference has been ignored, namely self-interference. We add this part into the derivation of CRM (Coding-aware Routing Metric) and modify the RREQ-RREP procedure to update the change.

The first aspect has been elaborated in Chapter 4 and this work further incorporates the second improvement. The design of the CRM employed in our protocol is not the theme of this chapter; we nevertheless give a brief walkthrough for the derivation of the CRM to make this thesis self-contained. More details can be found in [28].

The gist of the CRM is to evaluate how busy each node is transmitting packets from other flows. In other words, we evaluate approximately how long a packet from the newly arriving flow need to wait before it can be processed by all the intermediate nodes and arrive at the destination. So the starting point of such a metric is the average queue length of a network interface. Since this queue length metric did not take coding benefits into account, it should be *modified* to yield the *modified queue length (MQ)*.

This modification is done on a WCG. This graph consists of vertices, edges and weights associated with those vertices. Vertices in a WCG correspond to flows that pass through an intermediate node. An edge between two vertices indicates that those two flows can be coded. The weight information associated with a vertex is the offered load of that flow. So the problem of finding the best packet scheduling scheme, is then reduced to a mathematical *weighted clique cover problem* (WCCP). WCCP studies how to *cover* a WCG with the minimum

number of cliques. Put in the context of networking, a clique in WCG represents a coded transmission. The vertices in that clique correspond to the flows from which the native packets are coming. Now it is easy to see that the minimum number of cliques to cover a WCG equals the minimum number of transmissions to clear all queued packets. This number, in effect, describes how busy a node is in transmitting packets, with network coding benefits considered. We have proved this problem is NP-hard but also gave a workable solution that produces sufficiently good results in Chapter 4.

Formally, the derivation above can be abstracted as:

$$MQ(v) = |\text{WCCP}(\mathbf{G}_v)| \tag{5.1}$$

, where v is the node under consideration, queue lengths from all relevant flows $Q(v)$ constitute the coding graph $\mathbf{G}_v$, and by solving the WCCP on $\mathbf{G}_v$ we get modified queue length $MQ(v)$.

So far the modified queue length reflects all conditions within a node. Next, we take interference into account and arrive at the MIQ. Interference comes from two sources. The first is the channel time occupied by adjacent nodes. When adjacent nodes are transmitting, the transmission for this node is blocked by the RTS signals to avoid collision. The second source is the interference that is brought about by the flow itself. When the upstream node of this node is transmitting packets for this flow, this node will be blocked as well. By ignoring this effect, one will arrive at inappropriate routing metrics as described in the

following example. Suppose we have two possible routes from a source node to a destination, one is two-hop and the other one is three-hop. Assume the network is all idling prior to the arrival of this flow. If self-interference is ignored, the metric for both flows will both equal to zero. But with the consideration of self interference, the two-hop route will be preferred as its metric is only equivalent to one times the offered load of this flow. The three-hop route will have a metric of two times the offered load, because the self-interference impacts twice, both by the source node and by the first hop node. Thus when self-interference is ignored, the impact of the packet path along the downstream nodes is not adequately taken into consideration.

Mathematically, modified interference queue length can be represented as:

$$MIQ(v) = MQ(v) + \Sigma_{u \in N(v)} MQ(u) + R \tag{5.2}$$

, where $N(v)$ denotes the set of neighbours of node v and R is the offered load of the newly-arriving flow.

Lastly, the effect of packet loss is considered. By dividing the MIQ by the *successful transmission rate*, we arrive at the final result of CRM. The *Successful transmission rate* equals one minus the packet loss rate for a given node.

$$CRM(v) = \frac{1 + MIQ(v)}{1 - P(v)} \tag{5.3}$$

, where $P(v)$ denotes the packet loss rate for the link at node v for the newly-arriving flow. The CRM metric for the flow as a whole is the sum of CRM metric

of all nodes along the selected route.

$$CRM = \Sigma_{v \in L} CRM(v) \tag{5.4}$$

, where L is the set of nodes along the selected route. In order to achieve higher throughput and lower delay, one should choose the least-busy route, or the route with the smallest metric value.

5.3 Application-layer Strategies

This section studies how the cross-layer design can help from the application layer's perspective. As pointed out in previous literatures [87], direct deployment of P2P protocols on wireless ad hoc network is undesirable. The major limitation is the topology constraint. For wired networks, the physical locations of the peers are transparent to the P2P protocol. The logical topology overrides the effect of physical location. However, physical location relationships are no longer irrelevant or unimportant in wireless ad hoc networks. Even without putting additional assumptions on the behaviour of a node, the choice of neighbour peers is never homogeneous. Their physical location imposes an intrinsic constraint, and this constraint can heavily impact the transmission performance.

As a result, P2P-based content distribution in wireless networks entails a discrimination of peers. The peer-selection algorithm is given even higher importance. For example, *BitHoc* [102] distinguishes near-by peers from far-away peers. Peers that are more than two hops away are considered far away. In our

work, the power of network coding is tapped, so a peer-selection algorithm that considers hop distance, coding benefits, and network traffic is called for. This is exactly the routing metric we discussed in Section 5.2.

Applying solely the CRM in the peer-selection algorithm is, however, not desirable as well. CRM is a good measure for the fitness of choosing a peer, but solely applying this metric can easily cause starvation for a lot of other nodes. Usually this problem is solved by introducing uncertainties, similar to the optimistic unchoking mechanism in BitTorrent. For example, we can choose peers mostly using CRM, but opt to other measures occasionally. The probability is hard to determine, but as a rule of thumb, we fixed it to 20%. For example, we can choose peers mostly using CRM, but opt to other measures occasionally. Setting this probability too high will yield degraded efficiency, while setting it too low can starve some nodes. As a rule of thumb, we fixed it to 20%. This means that when a node is about to choose a new peer, it will choose a peer with the smallest CRM value 80% of the time, while 20% of the time the node will randomly choose a peer from the rest of all connected nodes.

The cross-layer channel brings in another improvement for the application layer, i.e., the possibility of dynamic offered load adjustment based on coding opportunities. For most of the P2P protocols, the offered load of P2P flows are governed by the tit-for-tat reward scheme and the limitation of transmission capacity. This is essentially a contention where every flow is requesting for higher offered load. However, we argue that with some coordination to intentionally adjust offered loads among flows, all flows can be better off, including the flows

that yielded themselves to decrease offered load. The requests to adjust offered load (*INC* and *DEC*) are sent backward from the intermediate nodes to source nodes. Hence, it is not vulnerable to selfish nodes who want to increase offered load all the time. We then describe the mechanism in three parts: how load adjustment packets are generated at intermediate nodes; how they are transmitted backwards to the source nodes; and how the source nodes respond to such requests.

Load adjustment packets are generated according to the *Quota-Generating Algorithm* described in Chapter 4. The logic behind the generation of load adjustment packets is straightforward. For each flow, this intermediate node can have 3 possible opinions: INC, DEC, and no opinion. What is attached in the first two types of packets is a relevant amount. This amount denotes the suggested change in offered load. If the quota generated in Algorithm 3 equals the offered load of this flow, it means that all packets from this flow can be delivered with no problem. In such case, the opinion would be INC if this node's transmission capacity is not saturated. This node hints upstream nodes to increase the offered load to fully utilize its bandwidth. In this case, the suggested amount is the difference between transmission capacity and the cardinality of the WCCP solution, divided by the total number of flows. Otherwise if this node's transmission capacity is saturated, the node expresses no opinion for this flow. On the other hand when the quota is smaller than the offered load of this flow, it means that this node cannot handle all packets from this flow, and actively cuts off the overly-abundant supply of packets. Therefore, this node will hint

upstream node to decrease the offered load for this flow by sending DEC packets. The relevant amount is then the difference between its current offered load and its assigned quota. Be reminded that the quota will never be greater than the offered load according to the *Quota-Generating Algorithm*. The objective of such logic is to mediate the offered loads of all the flows such that the bandwidth is fully utilized and no received packets are dropped due to insufficient transmission capacity along the route.

The load adjustment packet is actually only a flag in the ACK packet from the TCP protocol. Upon receiving a load adjustment packet at an intermediate node, this node needs to consider its own opinion before forwarding to the upstream nodes. The *DEC* packet overrides the *INC* packets as a means to avoid congestion. For example, if a node determines that a *INC* is appropriate for a flow, but receives a *DEC* from the downstream node, it propagates this *DEC* to upstream nodes and discards its own *INC* opinion. In other words, an *INC* packet can reach the source node only when all intermediate nodes endorse the *INC* decision or have no opinion.

When we jointly consider the INC/DEC packet generating rule and the forwarding rule, we arrive at the overall flow chart depicting the generation of INC and DEC packets as shown in Fig 5.1.

The source nodes adopt the following scheme to respond to load adjustment packets: when the bandwidth is sufficient to accommodate higher offered load, all *INC* requests are satiated while *DEC* requests are discarded; when the bandwidth is saturated, *INC* requests are not processed until a *DEC* arrives. This

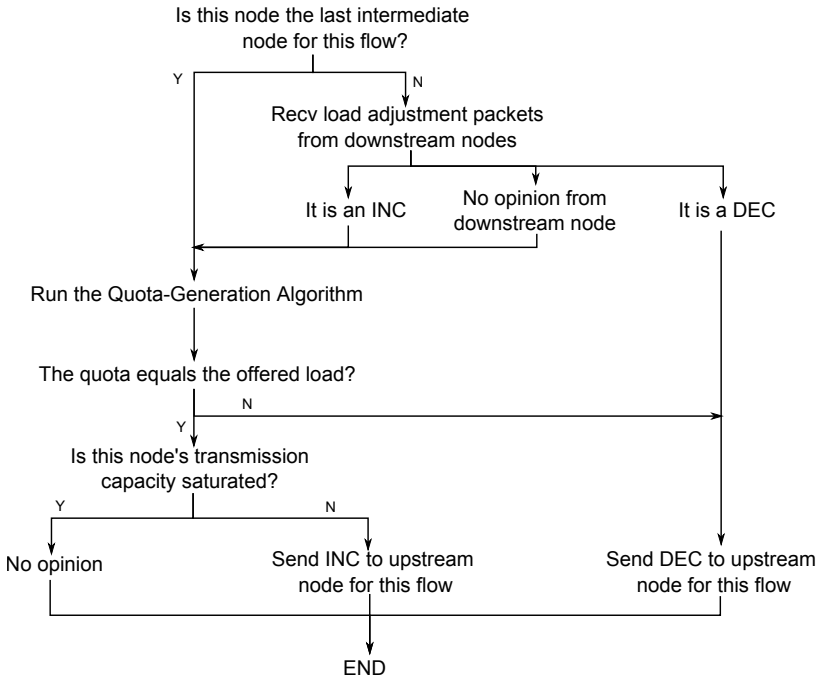


Figure 5.1: The flow chart depicting the generation of INC/DEC packets

scheme guarantees that a decrease in offered load for one flow is always accompanied with some increases for other flows.

5.4 Implementation

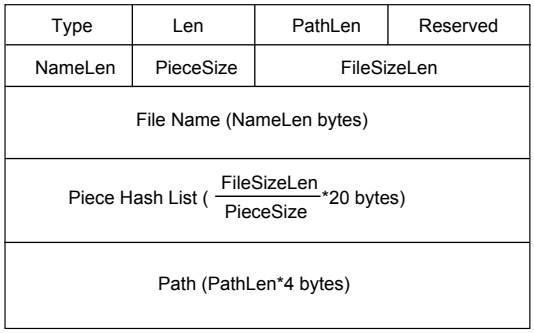
In this section, we discuss the implementation issues. The details are described methodically, from the construction of the ad hoc network to the termination of transmission.

5.4.1 Initial Setup

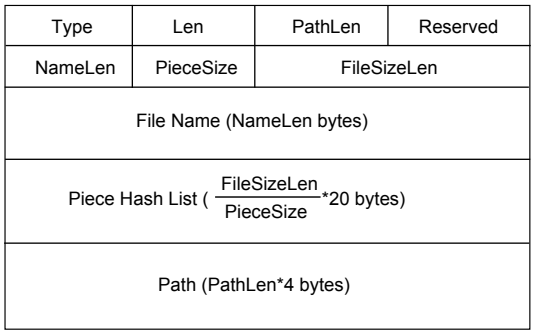
After the topology has been constructed, one node is assigned with a file to be distributed. This node is then referred to as the *seed node*. This node then acts as both a seed and a tracker. In order to make the file known to other nodes, it

broadcasts the meta-data of this file in a series of special RREQ messages with no specific intended recipient. Nodes that heard this message can determine if the file is of interest to them. If a node is interested in the file, it then sends a RREP message to the seed node, establishing the route from the seed to this node (leecher). The format of this special RREQ/RREP is shown in Fig 5.2a and Fig 5.2b, respectively. Most of the field names are self-descriptive, so we refrain from the tautology but only give some overall description. As for the length of fields, a single row is generally 4 bytes (32 bits) unless otherwise stated in parentheses. So one fourth of a row is one byte. Special fields such as *FileSizeLen* is of length two bytes. Some concepts are borrowed from BitTorrent's Torrent file format[103]. Note that the RREP contains only a 8-bit hash of the file name that is being shared. In our scenarios, we did not envision highly concurrent file sharing.

Unlike traditional RREQ/RREP procedures, the arrival of RREP packets at the seed node is not the end of the story. The seed node is responsible for supplying a list of peers to the leecher. Upon receiving this list of peers, the leecher node can then initiate connections to other peers in the list. When connections are established for a majority of the nodes in the network, the swarm of a P2P network is constructed and the routes between each pair of nodes are selected. Following that, the nodes in the network then start periodic exchanges of piece bitmaps. Each node keeps tracks of piece bitmaps of its two-hop neighbours employing a rarest-first piece-selection algorithm [104].



(a) Special RREQ



(b) Special RREP

Figure 5.2: Special RREQ/RREP Packet Formats

5.4.2 Cross-layer Dynamics

Once the initial peer connections are established, we then consider how the dynamics of the protocol stack fulfil the task of content distribution.

The first important issue is the update of offered load information. Initially, each peer establishes new connections with other peers with a fixed start-up offered load. This value is then adjusted by the mechanism we describe in Section 5.3. The planned offered load is inserted in the source routing header to inform the intermediate nodes. As such, intermediate nodes can plan their packet scheduling algorithm better. Every 30 seconds, the uploading node will initiate the RREQ/RREP procedure again to calculate the CRM again. The new CRM value is used to re-evaluate peer selection, and peers may opt to change peers if better routes are available.

From the observations of our simulation results, this re-evaluation of routes is pivotal. Without this re-evaluation, the performance of our protocol will be no better than existing coding-ignorant P2P protocols for ad hoc networks. This is because a route selected in the initial setup soon becomes obsolete when network traffic for other flows changes. The coding benefits may become less significant or be totally invalidated after such changes. The re-evaluation interval should not be set too short as well to avoid network turbulence.

5.4.3 Simulation Settings

We have implemented the above-described strategies collectively as a network-layer routing protocol and an application-layer P2P protocol in Qualnet [1]. Qualnet is a network simulator with the 7-layer OSI network structure implemented. We designed our protocols as modules and test the performance against the topology shown in Fig 4.14.

In this topology, we choose node 3 to be the seed node, and nodes 0, 4, 8, 10, 13 as leecher nodes. The file to be distributed is 50 mega-bytes, divided into 3200 16-KB pieces. The piece size chosen is smaller than typical values seen in BitTorrent due to the instability of wireless transmissions. A smaller piece size can significantly increase the chance of successfully transmitting a whole piece along multi-hop routes.

The performance of our protocol is compared against two other protocols. The first is a direct implementation of BitTorrent. The second is BitHoc, a version of BitTorrent tailored for wireless networks. For both protocols, network-layer opportunistic network coding is turned on. Hence these two protocols are referred to as "BitTorrent-NC" and "BitHoc-NC". Our protocol is named as Cross-layer solution with Network Coding (XLiNC). BitTorrent-NC, though with network coding enabled, is expected to have the worst performance. This is because the network topology of ad hoc networks is significantly different from the wired networks where BitTorrent is often employed. The heterogeneity between peers can quickly deteriorate the transmission efficiency. BitHoc-NC takes

[1] http://web.scalable-networks.com/content/qualnet

Table 5.1: Simulation Parameters

Parameter	**Value**	**Parameter**	**Value**
Terrain Dimension	1000x1000	Radio range	100 m
PHY layer	PHY802.11b	MAC Protocol	MACDOT11
Transmission Rate	2Mbps	Mobility	None
Auto-rate-fallback	Disabled	Packet Payload	512 Bytes
# of Unchoked Peers	2		

a step further. It is able to discriminate peers according to their relative distances in the network. Thus, the choice of peers can be done in a more bandwidth-efficient way. In the meantime, BitHoc-NC sometimes selects far-away peers to increase the transmission diversity. In XLiNC, a completely different peer selection scheme is employed. Peers are selected based on their offered-load-adjusted CRM metric. XLiNC also introduces more transmission diversity by randomly selecting other peers just like BitTorrent's opportunistic unchoking. Our intention is to show a cross-layer design can utilize a synergy between wireless P2P and network coding techniques.

Some of the simulation parameters are displayed in Table 5.1. Note that some of the terms used in the table are specific to Qualnet, but their names are self-descriptive.

5.4.4 Results

The finishing time is the most important evaluation criteria for a content distribution protocol. It measures the total amount of time required to finish the content distribution. Fig 5.3 shows the cumulative percentage of progress for all three protocols. The finishing time is 2653.79, 1490.87, and 1267.24 seconds for

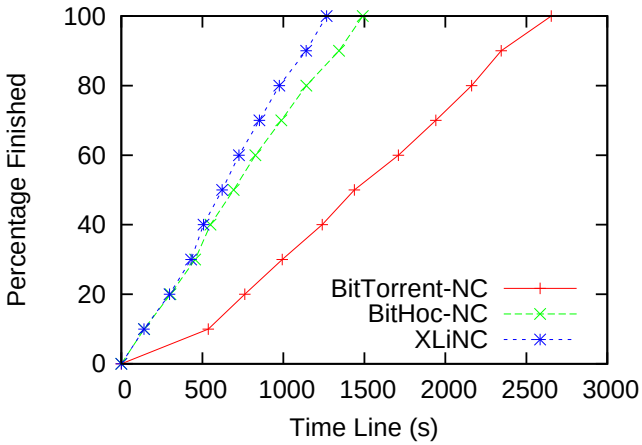


Figure 5.3: Cumulative Percentage Finished over Time

all three protocols respectively. It is observed that BitTorrent-NC has the longest finishing time. The initialization of the P2P network is also slower, depicted by the smoother slope at the beginning. This is because of an inappropriate peer-selection algorithm. Choosing a far-away node as a peer and uploading to that node at the initialization phase can significantly reduce the effectiveness of channel usage. The curve of BitHoc-NC and XLiNC coincide to overlap each other in the beginning. This is because in the beginning, only the seed node can transmit packets. Thus, the seed node becomes a bottleneck, restricting the overall throughput. As the transmission continues, the content of the file is dispersed across the network. With higher availability of the file, XLiNC can select peers more wisely and utilize the coding gain better.

We also measure the number of coded packets sent in the network. As depicted in Fig 5.4, XLiNC has the highest number of coded packets sent. BitHoc-NC sent less coded packets but their shapes are similar. BitTorrent-NC has the least number of coded packets sent. As compared to the other two proto-

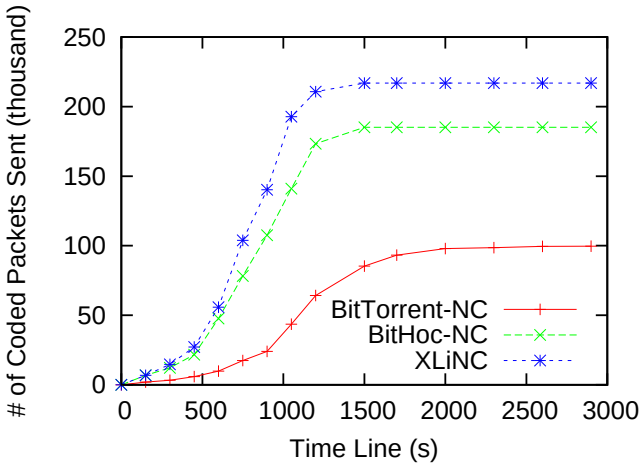


Figure 5.4: Number of Coded Packets Sent

cols, BitTorrent-NC has longer and flatter tails, indicating that network coding is rarely seen, both at the beginning and at the end of the transmission. On closer examination of this result, one may find that the smaller number of coded packets is not the only cause for BitTorrent-NC's *much longer* finishing time. In fact, a significantly higher dropped-packet count is the major cause for it. In our simulation, we measure the total amount of packets that are dropped. While in XLiNC the total dropped count is 12,774, BitTorrent-NC almost quadrupled this number, reaching a total of 44,933. To put this in context, note that the total number of coded packets is less than 100,000 in BitTorrent-NC. This relatively high drop rate can severely degrade performance. The reason for such high drop rate is that BitTorrent tends to choose longer routes/peers. Both the failed TCP transmissions and the mismatch between offered load and transmission capacity at intermediate nodes are attributable for the higher dropped-packet count.

Next we go deeper to examine the synergistic effect between wireless P2P and network coding techniques. BitTorrent-based P2P protocols usually share

the same mechanisms: opportunistic unchoking and anti-snubbing. Opportunistic unchoking is indispensable in discovering better peers, and anti-snubbing is making opportunistic unchoking more effective. Ideally, a peer can discover a "snubbing" as soon as its paired peer stops to upload, and then initiates opportunistic unchoking to exchange files with another peer. Anti-snubbing is most important in network-coding-enabled networks because an exchanging peer pair is naturally a coding opportunity. Maintaining an equal and fair peer relationship can make best use of network coding. When one peer stops to send, or lacks new pieces, the balance is broken and coding opportunities are gone.

Fig 5.5 presents the number of downloaded/uploaded packets for selected peer pairs. A peer pair denoted by $i - j$ represents the peer pair between node i and node j. For each pair, U_{ij} and U_{ji} are displayed for all three protocols, where U_{ij} denotes the amount uploaded by node i to node j. It is observed that, for example, XLiNC exhibits higher traffic but lower absolute difference for pair "$4 - 8$" and "$8 - 13$". This is explained as follows. From the view of node 8 in XLiNC, it evaluates all peers based on the CRM. Node 3, though being the seed node, takes two hops to arrive and the route from node 3 to node 8 involves a lot of interference from node 4. Node 10 takes 3 hops to arrive, so the CRM value is also higher. On the contrary, node 4 is only one hop away, and node 13, when taking the route $13 - 12 - 8$ is mostly noise-free. So node 4 and node 13 are the preferred peers for node 8, and node 8 tries to connect to them more often. On the other hand, since the routing layer can negotiate offered load with application layer, the protocol will try to maximize the free-rider effect,

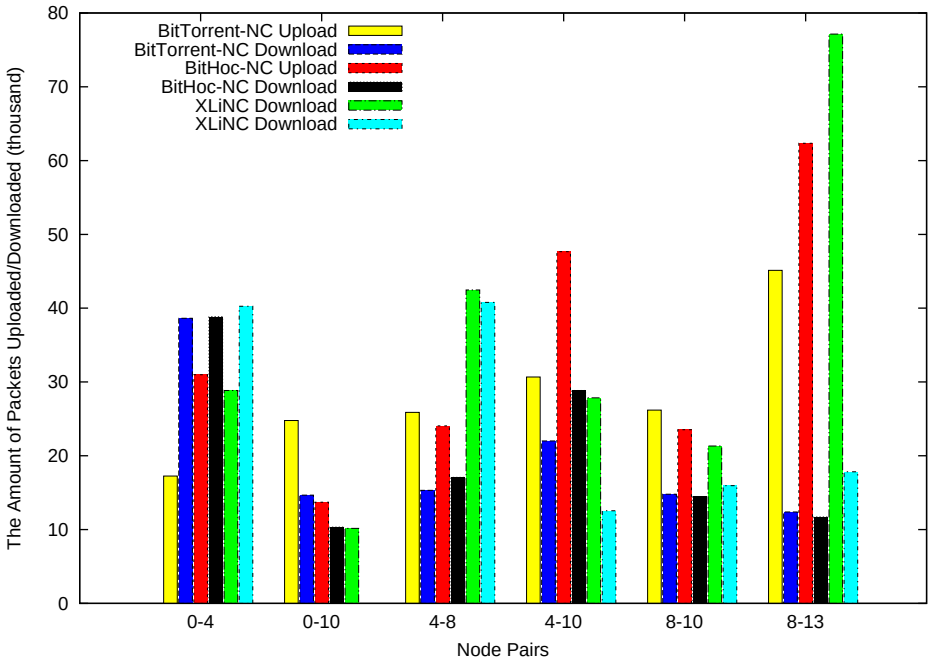


Figure 5.5: Upload/Download Amounts for Selected Pairs

or coding benefits, by reducing the disparity between the offered loads for both directions in a pair.

5.5 Chapter Summary

In this chapter, we have proposed a cross-layer solution for wireless content distribution utilizing network coding techniques. The advantages of introducing this cross-layer solution is that we can exploit the synergy between network layer and application layer. In wireless ad hoc networks, this cross-layer channel serves well to exchange routing information and traffic pattern information.

The novelty of this solution consists of the following points:

1. The peer-selection algorithm in P2P applications is broadened to consider specific routes. Given the routing metric information, peers are better

evaluated in terms of their contribution to overall throughput.

2. The routing metric can be calculated with precise offered load information. This value can be more predictive than the average queue length, which is the best guess given only information from the network layer.

3. A backward propagation of load adjustment packets is introduced to dynamically control the offered load. This avoids the bandwidth wastage during congestions and improves coding efficiency at intermediate nodes.

In addition to these novel concepts, many implementation details are considered as well. These include the initial setup of the system followed by adaptations to system dynamics. Simulation results show it is advantageous to employ such a cross-layer solution. The performance of our cross-layer solution outperforms another location-aware P2P protocol with network coding enabled.

Chapter 6

Conclusion

6.1 Thesis Contribution

In this thesis, we have investigated various issues on binding network coding techniques with wireless ad hoc networks.

Among other challenges we face, the first part of the thesis strives to improve coding-aware routing protocols' robustness against network dynamics. Specifically, the timing, and the order in which flows join or exit the network, as well as offered load changes should be coped with by the coding-aware routing protocol. The protocol should be able to detect coding opportunities and evaluate coding benefits regardless of the factors mentioned above. Hence, we have introduced SCAR, a coding-aware routing protocol with self recommendation in wireless ad hoc networks. The mechanism of self recommendation enables the protocol to discover hidden coding opportunities that have been overlooked by other routing protocols. Whenever a new flow joins the network and presents a new coding

opportunity, prior routing decisions are revised through a Route-Change procedure. The Route-Change procedure triggered by the self recommendation is able to adjust the biased routing metrics.

In order to gain further insights into how much our protocol can outperform protocols without such route-maintenance procedure, a series of indicators are introduced. Through topology analysis, the proposed indicator $C2$ estimates, how much throughput gain is achievable by revising prior route decisions for 2 flows. Similarly, $C3$ reflects how much throughput gain is achievable if we initiate a self-recommendation at the intermediate nodes for 3 flows. Testing these indicators on multiple topologies reveals that the throughput gain achieved by SCAR is not a coincidence and many practical topologies come with positive indicators. Further simulations are done in *Qualnet* to evaluate our protocol. It is shown that our protocol can significantly outperform other coding-aware routing protocols and the predictions given by the indicators are generally correct. The dynamic nature of our protocol results in a higher tolerance and adaptiveness against changes.

The second issue we study is coding-aware packet scheduling algorithms. To avoid misunderstanding, the *scheduling problem* considered in this thesis is not the same *scheduling problem* defined in most literatures. Prior works mostly define *scheduling* as the practice to coordinate all nodes' transmissions in a network. This scheduling can only be done with the full knowledge of the topology, the transmission capacities, the planned transmission tasks, and the full compliance of all nodes. Our thesis considers a totally different *schedul-*

ing problem. We consider the tactic chosen by a node to process its backlogged packets in network-layer. It is proved in this thesis that a well-designed network-layer packet scheduling algorithm can help us tune the performance of the network. Specifically, we set the performance target to maximize minimum per-flow throughput first, and then maximize the average per-flow throughput. This target, though being a bit complex, is very practical in real deployments. In addition, the tools we used in achieving this target are non-trivial and they cover the tools used for other targets, such as directly maximizing overall throughput.

In our work, the scheduling problem is tackled first in a simplified and static version. In this case, we abstract the static scheduling problem as an optimization problem called WCCP. Since this problem is proved to be NP-hard, we proposed pruning rules and approximation methods to keep this problem tractable, while maintaining acceptable error. With our scheduling method, intermediate nodes can clear packet queues faster and the scalability of the algorithm is also studied. The findings are then applied to the dynamic form of the problem where we consider multiple queueing systems with random packet arrivals and finite queue lengths. We proposed a heuristic scheduling method (HEU) to address the challenges posed. Since more uncertainties are brought into the problem, HEU provides only 13% improvement in a dynamic situation, compared to the 20% performance gain in the static form. But after all, we can still gain positive throughput gain and we managed to improve the fairness significantly. In the simulation we observe that the minimum per-flow throughput is almost at the same level as the average per-flow throughput. This is an indication of good fair-

ness control of the scheduling algorithm. The solution to this dynamic problem is then incorporated into a routing protocol. Simulation results for this routing protocol reveal that with proper scheduling, we can have better control over the fairness of the network. When fairness is not a concern, one can easily adapt our protocol to put more emphasis on those higher coding gain pairs to improve throughput.

After considering the robustness, efficiency and fairness of network coding techniques, we apply the findings to a full solution—P2P content distribution in wireless ad hoc networks. The scenario is characterized by typical traffic patterns in the form of multiple unicast flows. This pattern is generally suitable for inter-flow network coding techniques. To make better use of the information at hand, we adopt a cross-layer method to integrate the network layer and the application layer together. The novelty of this cross-layer solution is generally three-fold. Firstly, the application layer can give hints on the offered load information for flows. These hints are used by the network layer to produce more reliable routing metrics. Compared to our previous way of calculating routing metric, we no longer need to rely on the average queue length to predict future offered load. Secondly, the routing metric, while on one hand reflects the appropriateness of choosing the route for a flow, also foretells the potential achievable throughput of this flow. Thus, this information can be used by the application layer to wisely choose neighbour peers. By choosing peers with better routing metrics, the coding benefits can be exploited, and thus expedites the whole content distribution process. The third part of the novelty is the back propagation

of rate adjustment packets. Intermediate nodes, when doing packet scheduling, can have a clear understanding of how coding efficiency can be improved. By sending back rate adjustment packets, intermediate nodes can notify the source nodes about the network conditions. This interaction between network layer and application layer can help P2P content distribution systems utilize bandwidth more efficiently.

6.2 Future Work

This thesis solved various problems in the incorporation of network coding techniques into wireless ad hoc networks and reaches a cross-layer solution for a typical wireless application. However, there are still a number of directions to follow up:

1. A major limitation of XOR-based network coding is that ACK is not reliable. When multiple packets are coded together, at most one native packet can have reliable ACK packet from the TCP protocol. For other native packets that were transmitted in the same coded packet, there is no mechanism to ensure their successful transmissions. So one possible direction is to employ a circular scheme that rotates among these native flows to choose a "leading" flow. Moreover, the ACK of the "leading" flow is a modified ACK that acknowledges multiple packets in the past for this flow. This scheme can detect link failures much earlier than the currently employed one.

2. The hidden terminal problem impacts network coding severely as well. A great number of packet loss comes from interference of hidden terminals. The hidden terminal problem can be much alleviated by a more organized MAC-layer scheduling method. So this can be a future direction to improve network coding efficiency as well.

3. The cross-layer solution to content distribution in wireless networks can be improved further as well. The idea is to exploit the fact that file pieces are universal across the network. In other words, a node that has received a given piece previously can use the file to decode future coded packets. It is not imperative for a node to have received the exact native packet for decoding. As long as the node has received the corresponding piece, it can decode successfully. This can help us increase the amount of network coding opportunities, thus yielding higher throughputs.

Bibliography

[1] PricewaterhouseCoopers. *North American wireless industry survey*. PwC. 2013. URL: http://www.pwc.com/en_US/us/industry/communications/publications/assets/pwc-north-american-wireless-industy-survey-2012.pdf.

[2] T. Matsuda, T. Noguchi, and T. Takine. "Survey of network coding and its applications". In: *IEICE Transactions on Communications* 94.3 (2011), pp. 698–717.

[3] R. Ahlswede, N. Cai, S. Li, and R. Yeung. "Network information flow". In: *IEEE Transactions on Information Theory* 46.4 (2000), pp. 1204–1216.

[4] S. Li, R. Yeung, and N. Cai. "Linear network coding". In: *IEEE Transactions on Information Theory* 49.2 (2003), pp. 371–381.

[5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. "XORs in the air: Practical wireless network coding". In: *IEEE/ACM Transactions on Networking* 16.3 (2008), pp. 497–510.

[6] P. Chou, Y. Wu, and K. Jain. "Practical network coding". In: *Proceedings of Allerton Conference on Communication, Control, and Computing*. 2003, pp. 40–49.

[7] D. Lun, M. Médard, and R. Koetter. "Efficient operation of wireless packet networks using network coding". In: *Proceedings of International Workshop on Convergent Technologies*. 2005.

[8] A. Ramamoorthy, J. Shi, and R. Wesel. "On the capacity of network coding for random networks". In: *IEEE Transactions on Information Theory* 51.8 (2005), pp. 2878–2885.

[9] Z. Li and B. Li. "Network coding: The case of multiple unicast sessions". In: *Proceedings of Allerton Conference on Communications*. 2004.

[10] T. Ho and R. Koetter. "Online incremental network coding for multiple unicasts". In: *Proceedings of DIMACS Working Group on Network Coding*. 2005.

[11] T. Ho, Y. Chang, and K. Han. "On constructive network coding for multiple unicasts". In: *Proceedings of Allerton Conference on Communication, Control and Computing*. 2006.

[12] S. Rayanchu, S. Sen, J. Wu, S. Banerjee, and S. Sengupta. "Loss-aware network coding for unicast wireless sessions: Design, implementation, and performance evaluation". In: *Proceedings of ACM SIGMETRICS Performance Evaluation Review*. 2008, pp. 85–96.

[13] J. Le, J. Lui, and D. Chiu. "How many packets can we encode? - An analysis of practical wireless network coding". In: *Proceedings of the 27th IEEE International Conference on Computer Communications*. 2008, pp. 371–375.

[14] Z. Mobini, P. Sadeghi, M. Khabbazian, and S. Zokaei. "Power allocation and group assignment for reducing network coding noise in multi-unicast wireless systems". In: *IEEE Transactions on Vehicular Technology* 61.8 (2012), pp. 3615–3629.

[15] A. Shafieinejad, F. Hendessi, and F. Fekri. "Network coding for multiple unicast sessions in multi-channel/interface wireless networks". In: *Wireless Networks* 19.5 (2013), pp. 891–911.

[16] R. Koetter and M. Médard. "An algebraic approach to network coding". In: *IEEE/ACM Transactions on Networking* 11.5 (2003), pp. 782–795.

[17] D. Nguyen, T. Tran, T. Nguyen, and B. Bose. "Wireless broadcast using network coding". In: *IEEE Transactions on Vehicular Technology* 58.2 (2009), pp. 914–925.

[18] T. Tran, T. Nguyen, B. Bose, and V. Gopal. "A hybrid network coding technique for single-hop wireless networks". In: *IEEE Journal on Selected Areas in Communications* 27.5 (2009), pp. 685–698.

[19] D. Nguyen, T. Nguyen, and X. Yang. "Multimedia wireless transmission with network coding". In: *Proceedings of Packet Video*. 2007, pp. 326–335.

[20] T. Tran and T. Nguyen. “Prioritized wireless transmissions using random linear codes”. In: *Proceedings of IEEE International Symposium on Network Coding*. 2010, pp. 1–6.

[21] T. Tran, T. Nguyen, and R. Raich. “On achievable throughput region of prioritized transmissions via network coding”. In: *Proceedings of 18th Internatonal Conference on Computer Communications and Networks*. 2009, pp. 1–6.

[22] T. Tran, D. Nguyen, and T. Nguyen. “A case for joint network coding and power control in wireless linear networks”. In: *Proceedings of 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*. 2009, pp. 1–6.

[23] T. Tran, D. Nguyen, T. Nguyen, and D. Tran. “Joint network coding and power control for cellular radio networks”. In: *Proceedings of 2nd International Conference on Communications and Electronics*. 2008, pp. 1–6.

[24] Y. Wu, P. Chou, and S. Kung. “Minimum-energy multicast in mobile ad hoc networks using network coding”. In: *IEEE Transactions on Communications* 53.11 (2005), pp. 1906–1918.

[25] X. Liu, K. Fouli, R. Kang, and M. Maier. “Network-coding-based energy management for next-generation passive optical networks”. In: *Journal of Lightwave Technology* 30.6 (2012), pp. 864–875.

[26] M. Zhou, Q. Cui, R. Jantti, and X. Tao. "Energy-efficient relay selection and power allocation for two-way relay channel with analog network coding". In: *IEEE Communications Letters* 16.6 (2012), pp. 816–819.

[27] T. Cui, L. Chen, and T. Ho. "Energy efficient opportunistic network coding for wireless networks". In: *Proceedings of the 27th IEEE International Conference on Computer Communications*. 2008, pp. 361–365.

[28] J. Le, J. Lui, and D. Chiu. "DCAR: Distributed coding-aware routing in wireless networks". In: *IEEE Transactions on Mobile Computing* 9.4 (2010), pp. 596–608.

[29] S. Sengupta, S. Rayanchu, and S. Banerjee. "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing". In: *Proceedings of the 26th IEEE International Conference on Computer Communications*. 2007, pp. 1028–1036.

[30] S. Das, Y. Wu, R. Chandra, and Y. Hu. "Context-based routing: Technique, applications, and experience". In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. 2008, pp. 379–392.

[31] A. Khreishah, I. Khalil, and J. Wu. "Distributed network coding-based opportunistic routing for multicast". In: *Proceedings of the 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 2012, pp. 115–124.

[32] P. Chaporkar and A. Proutiere. “Adaptive network coding and scheduling for maximizing throughput in wireless networks”. In: *Proceedings of the 13th Annual International Conference on Mobile Computing and Networking*. 2007, pp. 135–146.

[33] B. Scheuermann, W. Hu, and J. Crowcroft. “Near-optimal co-ordinated coding in wireless multihop networks”. In: *Proceedings of the 3rd International Conference on Emerging Networking Experiments and Technologies*. 2007.

[34] D. Traskov, M. Heindlmaier, M. Médard, and R. Koetter. “Scheduling for network-coded multicast”. In: *IEEE/ACM Transactions on Networking* 20.5 (2012), pp. 1479–1488.

[35] L. Yang, Y. Sagduyu, and J. Li. “Adaptive network coding for scheduling real-time traffic with hard deadlines”. In: *Proceedings of the 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 2012, pp. 105–114.

[36] T. Kim, S. Vural, I. Broustis, D. Syrivelis, S. Krishnamurthy, and T. La Porta. “A framework for joint network coding and transmission rate control in wireless networks”. In: *Proceedings of the 29th IEEE International Conference on Computer Communications*. 2010, pp. 1–9.

[37] Y. Yan, B. Zhang, H. Mouftah, and J. Ma. “Practical coding-aware mechanism for opportunistic routing in wireless mesh networks”. In: *Pro-*

ceedings of IEEE International Conference on Communications. 2008, pp. 2871–2876.

[38] Y. Wu, P. Chou, and S. Kung. *Information exchange in wireless networks with network coding and physical-layer broadcast*. Tech. rep. MSR-TR-2004, 2004.

[39] A. Eryilmaz, A. Ozdaglar, and M. Medard. "On delay performance gains from network coding". In: *Proceedings of 40th Annual Conference on Information Sciences and Systems*. 2006, pp. 864–870.

[40] S. Hu and H. Médard. "The importance of being opportunistic: Practical network coding for wireless environments". In: *Proceedings of 43rd Allerton Conference on Communication, Control, and Computing*. 2005.

[41] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong. "A random linear network coding approach to multicast". In: *IEEE Transactions on Information Theory* 52.10 (2006), pp. 4413–4430.

[42] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed diffusion: A scalable and robust communication paradigm for sensor networks". In: *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*. 2000, pp. 56–67.

[43] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. "Energy-efficient communication protocol for wireless microsensor networks". In: *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*. 2000, pp. 1–10.

[44] Q. Li, J. Aslam, and D. Rus. “Hierarchical power-aware routing in sensor networks”. In: *Proceedings of the DIMACS Workshop on Pervasive Networking*. 2001.

[45] T. Singh, N. Gupta, and J. Minj. “Hierarchical cluster based routing protocol with high throughput for wireless sensor networks”. In: *Proceedings of IEEE International Conference on Signal Processing, Computing and Control*. 2013, pp. 1–6.

[46] Y. Xu, J. Heidemann, and D. Estrin. “Geography-informed energy conservation for ad hoc routing”. In: *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*. 2001, pp. 70–84.

[47] C. Perkins and E. Royer. “Ad-hoc on-demand distance vector routing”. In: *Proceedings of 2nd Workshop on Mobile Computing Systems and Applications*. 1999, pp. 90–100.

[48] D. Johnson and D. Maltz. “Dynamic source routing in ad hoc wireless networks”. In: *Mobile Computing*. Springer, 1996, pp. 153–181.

[49] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot. “Optimized link state routing protocol”. In: *RFC 3626* (2003).

[50] D. De Couto, D. Aguayo, B. Chambers, and R. Morris. “Performance of multihop wireless networks: Shortest path is not enough”. In: *ACM SIGCOMM Computer Communication Review* 33.1 (2003), pp. 83–88.

[51] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. "Link-level measurements from an 802.11 b mesh network". In: *Proceedings of ACM SIGCOMM Computer Communication Review*. 2004, pp. 121–132.

[52] M. Halloush and H. Radha. "Network coding with multi-generation mixing: A generalized framework for practical network coding". In: *IEEE Transactions on Wireless Communications* 10.2 (2011), pp. 466–473.

[53] S. Kim. "Concatenated random parity forwarding in large-scale multi-hop relay networks". In: *Proceedings of Military Communications Conference*. 2007, pp. 1–7.

[54] J. Jin and B. Li. "Adaptive random network coding in WiMAX". In: *Proceedings of IEEE International Conference on Communications*. 2008, pp. 2576–2580.

[55] R. Prasad, H. Wu, D. Perkins, and N. Tzeng. "Local topology assisted XOR coding in wireless mesh networks". In: *Proceedings of 28th International Conference on Distributed Computing Systems Workshops*. 2008, pp. 156–161.

[56] C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul. "Wireless network coding: Opportunities & Challenges". In: *Proceedings of Military Communications Conference*. 2007, pp. 1–8.

[57] Y. Sagduyu and A. Ephremides. "Joint scheduling and wireless network coding". In: *Proceedings of WINMEE, RAWNET and NETCOD 2005 Workshops*. 2005.

[58] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. "On MAC scheduling and packet combination strategies for practical random network coding". In: *Proceedings of IEEE International Conference on Communications*. 2007, pp. 3582–3589.

[59] S. Shabdanov, C. Rosenberg, and P. Mitran. "Joint routing, scheduling, and network coding for wireless multihop networks". In: *Proceedings of 9th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*. 2011, pp. 33–40.

[60] N. Jones, B. Shrader, and E. Modiano. "Optimal routing and scheduling for a simple network coding scheme". In: *Proceedings of 31st IEEE International Conference on Computer Communications*. 2012, pp. 352–360.

[61] J. Ghaderi, T. Ji, and R. Srikant. "Connection-level scheduling in wireless networks using only MAC-layer information". In: *Proceedings of the 31st IEEE International Conference on Computer Communications*. 2012.

[62] S. Wang, Q. Song, X. Wang, and A. Jamalipour. "Distributed MAC protocol supporting physical-layer network coding". In: *IEEE Transactions on Mobile Computing* 12.5 (2013), pp. 1023–1036.

[63] H. Yomo and P. Popovski. "Opportunistic scheduling for wireless network coding". In: *Proceedings of IEEE International Conference on Communications*. 2007, pp. 5610–5615.

[64] B. Ni, N. Santhapuri, C. Gray, and S. Nelakuditi. "Selection of bit-rate for wireless network coding". In: *Proceedings of the 5th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*. 2008, pp. 1–6.

[65] M. Zhao and Y. Yang. "Packet scheduling with joint design of MIMO and network coding". In: *Journal of Parallel and Distributed Computing* 72.3 (2009), pp. 227–236.

[66] V. Srivastava and M. Motani. "Cross-layer design: A survey and the road ahead". In: *IEEE Communications Magazine* 43.12 (2005), pp. 112–119.

[67] S. Buzzi, H. Poor, and D. Saturnino. "Adaptive cross-layer distributed energy-efficient resource allocation algorithms for wireless data networks". In: *EURASIP Journal on Advances in Signal Processing* (2009).

[68] G. Carneiro, J. Ruela, and M. Ricardo. "Cross-layer design in 4G wireless terminals". In: *IEEE Wireless Communications* 11.2 (2004), pp. 7–13.

[69] L. Chen, S. Low, M. Chiang, and J. Doyle. "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks". In: *Proceedings of the 25th IEEE International Conference on Computer Communications*. 2006, pp. 1–13.

[70] M. Conti, S. Giordano, G. Maselli, and G. Turi. "Mobileman: Mobile metropolitan ad hoc networks". In: *Proceedings of Personal Wireless Communications*. 2003, pp. 169–174.

[71] V. Raisinghani and S. Iyer. "ECLAIR: An efficient cross layer architecture for wireless protocol stacks". In: *Proceedings of World Wireless Congress*. 2004.

[72] W. Su and T. Lim. "Cross-layer design and optimisation for wireless sensor networks". In: *International Journal of Sensor Networks* 6.1 (2009), pp. 3–12.

[73] J. Yuan, Z. Li, W. Yu, and B. Li. "A cross-layer optimization framework for multicast in multi-hop wireless networks". In: *Proceedings of the 1st International Conference on Wireless Internet*. 2005, pp. 47–54.

[74] L. Song and D. Hatzinakos. "A cross-layer architecture of wireless sensor networks for target tracking". In: *IEEE/ACM Transactions on Networking* 15.1 (2007), pp. 145–158.

[75] N. Zhao and L. Sun. "Research on cross-layer frameworks design in wireless sensor networks". In: *Proceedings of the 3rd International Conference on Wireless and Mobile Communications*. 2007.

[76] M. Razzaque, S. Dobson, and P. Nixon. "Cross-layer architectures for autonomic communications". In: *Journal of Network and Systems Management* 15.1 (2007), pp. 13–27.

[77] J. Kuo, C. Shih, C. Ho, and Y. Chen. "A cross-layer approach for real-time multimedia streaming on wireless peer-to-peer ad hoc network". In: *Ad Hoc Networks* 11.1 (2013), pp. 339–354.

[78] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets. "GRACE: Cross-layer adaptation for multimedia quality and battery energy". In: *IEEE Transactions on Mobile Computing* 5.7 (2006), pp. 799–815.

[79] R. Madan, S. Cui, S. Lall, and A. Goldsmith. "Cross-layer design for lifetime maximization in interference-limited wireless sensor networks". In: *IEEE Transactions on Wireless Communications* 5.11 (2006), pp. 3142–3152.

[80] R. Xia and J. Muppala. "A survey of BitTorrent performance". In: *IEEE Communications Surveys & Tutorials* 12.2 (2010), pp. 140–158.

[81] D. Chiu, R. Yeung, J. Huang, and B. Fan. "Can network coding help in P2P networks?" In: *Proceedings of 2nd Workshop on Network Coding, Theory, and Applications (in Conjunction with WiOpt 2006)*. 2006, pp. 1–5.

[82] C. Gkantsidis and P. Rodriguez. "Network coding for large scale content distribution". In: *Proceedings of the 24th IEEE International Conference on Computer Communications*. 2005, pp. 2235–2245.

[83] C. Huang, T. Hsu, and M. Hsu. "Network-aware P2P file sharing over the wireless mobile networks". In: *IEEE Journal on Selected Areas in Communications* 25.1 (2007), pp. 204–210.

[84] Y. Zhu, B. Li, and J. Guo. "Multicast with network coding in application-layer overlay networks". In: *IEEE Journal on Selected Areas in Communications* 22.1 (2004), pp. 107–120.

[85] Q. Yan, M. Li, Z. Yang, W. Lou, and H. Zhai. "Throughput analysis of cooperative mobile content distribution in vehicular network using symbol level network coding". In: *IEEE Journal on Selected Areas in Communications* 30.2 (2012), pp. 484–492.

[86] Y. Kao, C. Lee, P. Wu, and H. Kao. "A network coding equivalent content distribution scheme for efficient peer-to-peer interactive VoD streaming". In: *IEEE Transactions on Parallel and Distributed Systems* 23.6 (2012), pp. 985–994.

[87] M. Conti, E. Gregori, and G. Turi. "A cross-layer optimization of gnutella for mobile ad hoc networks". In: *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 2005, pp. 343–354.

[88] U. Lee, S. Lee, K. Lee, and M. Gerla. "Understanding processing overheads of network coding-based content distribution in VANETs". In: *IEEE Transactions on Parallel and Distributed Systems* 24.11 (2013), pp. 2304–2318.

[89] D. Johnson, D. Maltz, and J. Broch. "DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks". In: *Ad Hoc Networking* 5 (2001), pp. 139–172.

[90] Y. Yan, Z. Zhao, B. Zhang, H. Mouftah, and J. Ma. "Rate-adaptive coding-aware multiple path routing for wireless mesh networks". In: *Proceedings of IEEE Global Communications Conference*. 2008, pp. 1–5.

[91] J. Zhang and Q. Zhang. “Cooperative network coding-aware routing for multi-rate wireless networks”. In: *Proceedings of the 28th IEEE International Conference on Computer Communications*. 2009, pp. 181–189.

[92] Y. Yan, B. Zhang, J. Zheng, and J. Ma. “Core: A coding-aware opportunistic routing mechanism for wireless mesh networks”. In: *IEEE Wireless Communications* 17.3 (2010), pp. 96–103.

[93] S. Sengupta, S. Rayanchu, and S. Banerjee. “Network coding-aware routing in wireless networks”. In: *IEEE/ACM Transactions on Networking* 18.4 (2010), pp. 1158–1170.

[94] J.-S. Park, D. S. Lun, F. Soldo, M. Gerla, and M. Medard. “Performance of Network Coding in Ad Hoc Networks”. In: *Proceedings of IEEE Military Communications Conference (MILCOM 2006)*. 2006.

[95] R. Karp. “Reducibility among combinatorial problems”. In: *50 Years of Integer Programming 1958-2008* (2010), pp. 219–241.

[96] W. Hsu and G. Nemhauser. “A polynomial algorithm for the minimum weighted clique cover problem on claw-free perfect graphs”. In: *Discrete Mathematics* 38.1 (1982), pp. 65 –71.

[97] W. Hsu and G. Nemhauser. “Algorithms for maximum weight cliques, minimum weighted clique covers and minimum colorings of claw-free perfect graphs”. In: *Topics on Perfect Graphs* 88 (1984), pp. 357–369.

[98] C. Berge and E. Minieka. *Graphs and hypergraphs*. Vol. 7. North-Holland Publishing Company Amsterdam, 1973.

[99] F. Bonomo, G. Oriolo, and C. Snels. "Minimum weighted clique cover on strip-composed perfect graphs". In: *Graph-Theoretic Concepts in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 22–33.

[100] E. Gilbert. "Random graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144.

[101] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. "Mason: A multi-agent simulation environment". In: *Simulation: Transactions of the Society for Modeling and Simulation International* 81.7 (2005), pp. 517–527.

[102] A. Krifa, M. Sbai, C. Barakat, and T. Turletti. "BitHoc: A content sharing application for wireless ad hoc networks". In: *Proceedings of the 7th IEEE International Conference on Pervasive Computing and Communications*. 2009, pp. 1–3.

[103] B. Cohen. *The BitTorrent protocol specification*. Jan. 2008. URL: http://www.bittorrent.org/beps/bep_0003.html.

[104] B. Cohen. "Incentives build robustness in BitTorrent". In: *Proceedings of Workshop on Economics of Peer-to-Peer Systems*. 2003, pp. 68–72.

List of Publications

[1] J. Wang, C. Zhu, T. Y. Chai, and W. Wong. "SCAR: A coding-aware routing protocol with self recommendation in static wireless ad hoc networks". In: *Journal of Computer Networks and Communications* 2014.637278 (2014), pp. 1–12.

[2] J. Wang, T. Y. Chai, and W. Wong. "Towards a fair and efficient packet scheduling scheme in inter-flow network coding". In: *Journal of Sensor and Actuator Networks* 2014.4 (2014), pp. 274–296.

[3] J. Wang, T. Y. Chai, and W. Wong. "Content Distribution in wireless ad hoc networks with network coding". In: *Proceedings of the 14th International Conference on Communication Systems*. Macau, Nov. 2014.

[4] J. Wang, T. Y. Chai, and W. Wong. "Optimizing packet scheduling decisions with network coding". In: *Proceedings of International Conference on Telecommunications*. Lisbon, Portugal, May 2014.

[5] J. Wang, C. Zhu, Q. Guo, T. Y. Chai, and W. Wong. "SCAR: A dynamic coding-aware routing protocol". In: *Proceedings of the 6th Inter-*

national Conference on Signal Processing and Communication Systems. Gold Coast, Australia, Dec. 2012.

[6] C. Zhu, C. Guo, J. Wang, and T. Tay. "Towards scalability issue in ontology-based context-aware systems". In: *Proceedings of 2012 International Conference on Software and Computer Applications*. June 2012, pp. 127–131.