Network Coded Media Distribution in Infrastructure Wireless Mesh Networks

by

Surachai Chieochan

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg

Abstract

Infrastructure wireless mesh networks (IWMNs) provide inexpensive deployment, flexible extension of wireless infrastructure, and easy access to the Internet. With multiple radios at each node, a capacity per node improves by transmitting over these radios simultaneously using orthogonal channels. However, without properly addressing the problem of channel assignment and routing for those nodes that form wireless infrastructures, the resulting network throughput and reliability are unlikely to meet the requirements of those highly demanding, media distribution applications. On a particular channel, poor resource allocation at a given access point/gateway of the underlying IWMN can amplify the problem even further. Motivated by these problems, we develop, based on the theory of network coding, a set of alternative solutions that addresses the above issues. We first introduce a sub-optimal solution to the joint problem of network coding, channel assignment and link scheduling for throughput optimization in the multi-channel multi-radio IWMN. We mathematically formulate the problem as a linear program, taking into account opportunistic overhearing, among other constraints. Based on this formulation, we develop a sub-optimal, auction-based algorithm for network throughput optimization. Simulation results reveal the effectiveness of our algorithm in exploiting multiple radios and channels while coping with fairness issues arising from auctions. The proposed solution also shows promising gains over traditional routing solutions. Our experimental results on an 802.11 testbed further confirm these results. The second part of this thesis then presents three AP/gateway-oriented solutions that address the link-level issues related to radio resource allocation at a particular AP/gateway node of the underlying IWMN, which operates on a

given channel serving a set of wireless clients. Since the last-hop wireless link is normally a bottleneck of the IWMN, the key idea underlying all the proposed solutions is to use a version of network coding at the bottlenecked AP/gateway. We use Markov chains and the probability theory to derive several performance measures related to media distribution for both uplink and downlink applications. Via extensive simulations, we show the promising delay and reliability gains of the network-coding based schemes over the traditional schemes without network coding.

Acknowledgments

I am greatly indebted to my advisor, Dr. Ekram Hossain, for his continuing guidance and support throughout the program. His tremendous patience with my work and progress has made sure I didn't fade during my burnouts. At times, I wonder though how someone can accomplish so much and manage so many things, yet remain so clam and relaxed. With his friendly skepticism as well as constant theoretical insights, he has taught me how to work around many research problems and get things done properly. I've learned a lot from him. He has given me a tremendous amount but I can only thank him for a subset.

I would also like to thank Jeff Diamond for his insightful suggestions and discussions on some of my work. His advice has always allowed me to see things from both academic- and industry-oriented perspectives. His patience with my writing has also taught me the importance of being precise. My grateful thanks also go to TR*Labs* and NSERC for their financial support. Without them, this thesis as well as my study at University of Manitoba would not have been possible.

My committee members, Dr. Lin Cai, Dr. Robert D. McLeod, Dr. Rasit Eskicioglu and Dr. Jun Cai , also made many invaluable suggestions during the candidacy and oral exam, many points of which have helped improve this thesis. My grateful thanks also go to the rest of the faculty members and all of the staff members in the ECE department and TR*Labs* for their constant support and collaboration. Judy, Amy, Nicole and Patricia, in particular, have always kindly taken care of my paperwork in all occasions.

My sincere thanks go to my friends and colleagues as well for their help and support throughout my time both in Winnipeg and back in Thailand. I have learned much

and received many helpful feedbacks from Tao and Teerawat when we collaborated on the Globecomm'09 paper which has made a chapter in this thesis. An, Ming, Bharat and Phond have sincerely helped me out in many occasions. I thank them for their jokes and laughters during our workouts, lunch, small gatherings and vacation, which have provided much needed relief from work for me. The same gratitude goes to P'Bu, P'Dee, P'Duang, P'Yui, Kook, Tan, N'Sai$_1$, N'Sai$_2$and N'Meaw. Jay and Zee, who had helped oversee the construction of my house back in Thailand, have also deserved my honest gratitude.

And finally, yet most importantly, my parents who have been an inspiration throughout my life. I thank them for what they are, and all they have done for me. I thank them for so many sacrifices they have made for educating me. My wife, Aead, with her selfless love, care and encouragement, has always been my constant moral support throughout. Four years of our long-distance relationship, thousand miles apart, have put her in charge of so many things for my family back in Thailand. This thesis is dedicated to her and my family.

To my family

Contents

List of Tables **x**

List of Figures **xi**

1 Introduction **1**
1.1 Network-wide Solution: Joint CA, Link Scheduling and Network Coding . . 5
1.2 AP/Gateway-oriented Solution I: Dynamic Buffer Allocation and Opportunistic Network Coding 7
1.3 AP/Gateway-oriented Solution II: Uplink Streaming with Wireless Fountain Coding 9
1.4 AP/Gateway-oriented Solution III: Downlink Streaming with Wireless Fountain Coding 12

2 Background **15**
2.1 Linear Network Coding 15
2.1.1 Linear Combining 19
2.1.2 Decoding 22
2.2 Infrastructure Wireless Mesh Networks (IWMNs) 23
2.2.1 Channelization 24
2.2.2 Medium Access Control 26

I Network-wide Solution: Joint Channel Assignment, Scheduling and Network Coding **29**

3 Joint Channel Assignment, Link Scheduling and Network Coding for Throughput Optimization **30**
3.1 Related Work 35
3.2 System Model and Assumptions 37
3.2.1 System Architecture and Operations 37
3.2.2 Interference Model 40
3.2.3 A Flow Network for A Multi-channel Multi-radio IWMN 42

3.2.4 Distributed Random Network Coding . . . 45
3.3 Problem Formulation . . . 49
3.3.1 Flow and Coding constraints . . . 49
3.3.2 Interference-free Broadcast Link Scheduling . . . 51
3.3.3 Node-Radio Constraints . . . 52
3.3.4 The Joint Channel Assignment and Network Coding Problem . . . 53
3.4 Refining Channel Assignment: Overview . . . 55
3.5 Refining Channel Assignment: Phase 1 . . . 59
3.6 Refining Channel Assignment: Phase 2 . . . 61
3.6.1 Formulation of CA Problem for Remaining Channels . . . 61
3.6.2 The Two-sided Multi-assignment Problem for Channel Assignment . 65
3.6.3 Converting TMP to Minimum Cost Flow Problem . . . 66
3.7 The Main Channel Assignment Algorithm . . . 69
3.8 Complexity of the Main Channel Assignment Algorithm . . . 74
3.9 Performance Evaluation . . . 78
3.9.1 Methodology . . . 78
3.9.2 Performance Metrics . . . 79
3.9.3 Simulation Evaluation . . . 80
3.9.4 Experimental Evaluation . . . 89
3.10 Conclusion . . . 94

4 iCORE: Experimental Platform for Multi-channel Multi-radio Coded IWMNs 95
4.1 Motivating Example . . . 98
4.2 Challenges . . . 99
4.3 Overview of iCORE . . . 101
4.4 Roles of Nodes . . . 104
4.4.1 Roles of Source-only Node (Intra-flow Coding) . . . 104
4.4.2 Roles of Repeater (Inter-flow Coding) . . . 106
4.4.3 Roles of Destination (Decoding) . . . 107
4.5 Implementation Details . . . 108
4.5.1 Packet Format . . . 108
4.5.2 Transmit Credits . . . 109
4.5.3 Node State . . . 110
4.5.4 Interface Cooperation . . . 110
4.5.5 Control Flow . . . 111
4.5.6 Piggybacking Acknowledgments . . . 114
4.6 Performance Evaluation of iCORE . . . 114
4.6.1 Implementation Testbed . . . 114
4.6.2 Performance Metrics . . . 115
4.6.3 Experimental Setup . . . 116
4.6.4 Experimental Results . . . 117
4.7 Conclusion . . . 120

II AP/Gateway-oriented Solutions: Resource Allocation/Error Control 122

5 Dynamic Buffer Allocation and Opportunistic Network Coding for Unicast Exchange 123
5.1 System Model and Assumptions . . . 126
5.1.1 Queueing Assumptions . . . 129
5.2 Discrete Time Markov Chain Model . . . 131
5.3 Performance Measures . . . 133
5.3.1 Coding Opportunity . . . 134
5.3.2 Average Packet Delivery Delay . . . 134
5.4 Numerical and Simulation Results . . . 135
5.4.1 Simulation Scenarios and Model Validation . . . 135
5.4.2 Comparison with Classical Scheduling . . . 136
5.5 Buffer-Equalized ONC (BE-ONC) . . . 139
5.6 Conclusion . . . 141

6 Wireless Fountain Coding with IEEE 802.11e Block ACK for Uplink Streaming 143
6.1 Motivating Example . . . 148
6.2 System Description . . . 151
6.2.1 Source Model and FEC . . . 152
6.2.2 Wireless Fountain Coding . . . 152
6.2.3 Modified B-ACK Model . . . 154
6.2.4 Application-layer ARQ . . . 156
6.2.5 Channel Model . . . 156
6.2.6 Wireline Delay Model . . . 157
6.3 System Analysis I: Streaming with Modified B-ACK . . . 158
6.3.1 Analysis of Wireless Delay . . . 158
6.3.2 Packet loss/late probability with FEC only . . . 161
6.3.3 Packet loss/late probability with FEC and application-layer ARQ . . . 162
6.4 System Analysis II: Streaming with Traditional B-ACK . . . 163
6.4.1 Analysis of Wireless Delay . . . 164
6.4.2 Packet Loss/Late Probability . . . 167
6.4.3 Accuracy of Extended Model . . . 169
6.5 Performance Evaluation Framework . . . 172
6.5.1 Packetization of Video Data . . . 172
6.5.2 Loss-Distortion Model . . . 174
6.5.3 Simulation Framework . . . 175
6.6 Simulation Results . . . 178
6.6.1 Model Validation . . . 178
6.6.2 Effects of block size K and FEC protection strength n/k . . . 183
6.6.3 The Hybrid Scheme . . . 187
6.7 Conclusion . . . 190

7 Downlink Media Streaming with Wireless Fountain Code 191
7.1 System Description . . . 196
7.1.1 Overview . . . 196
7.1.2 Source Model and FEC . . . 197
7.1.3 Wireless Fountain Coding . . . 198
7.1.4 Channel Model . . . 200
7.2 System Analysis . . . 200
7.2.1 Wireline Delay Model . . . 200
7.2.2 Delay Analysis of Wireless Transmission . . . 201
7.2.3 End-to-End Packet Loss/Late Probability . . . 205
7.3 Performance Evaluation Framework . . . 206
7.3.1 Packetization of Video Data . . . 206
7.3.2 Simulation Framework . . . 207
7.4 Simulation Results . . . 210
7.4.1 Effects of Number of Flows . . . 211
7.4.2 Effects of Maximum Allowed Delay (MAD) . . . 215
7.4.3 Average Video Frame Delay . . . 216
7.5 Conclusion . . . 218

8 Summary and Discussions 219
8.1 Summary of Contributions . . . 220
8.2 Future Work . . . 224

Bibliography 226

A Inner Matrices of DTMC for a Lossy Wireless Unicast Exchange Network 246
A.0.1 Descriptions of Matrix $\mathbf{B}$ . . . 246
A.0.2 Descriptions of Matrix $\mathbf{C}$ . . . 248
A.0.3 Descriptions of Matrix $\mathbf{A}_0$ . . . 249
A.0.4 Descriptions of Matrix $\mathbf{A}_2$ . . . 251
A.0.5 Descriptions of Matrices $\mathbf{A}_1$ and $\mathbf{A}_{1b}$ . . . 254

B Forward Auction Algorithm 257

List of Tables

4.1 Interface cooperation allows repeater B to inject its own packets as well as help others relay traffic. 100
4.2 Flows in opposite directions: With interface cooperation, an idle interface can be exploited to help a busier flow. 103

5.1 Coding Opportunity versus Normalized Arrival Rates. 137

6.1 Parameters for validation of the extended model. 171
6.2 Simulation parameters. 180

7.1 Simulation parameters. 212

A.1 Elements of Matrix B 248
A.2 Elements of Matrix C 249
A.3 Elements of Matrix $\mathbf{A}_0$ 251
A.4 Elements of Matrix $\mathbf{A}_2$ 253
A.5 Elements of Matrix $\mathbf{A}_1$ 255

List of Figures

1.1 Interactions between opportunistic network coding (ONC) and dynamic buffer allocation (DBA). (a & b) Network coding allows A and B to exchange a pair of packets using only 3 transmissions instead of 4 (numbers on arrows show the order of transmission). (c) Dynamic buffer allocation provides fairness and delay benefits among flows with random packet arrival and departure. It operates similarly to a pair of water tanks connected by a valve. 8
1.2 Reliability benefits of network coding can be realized in a single unicast flow in the presence of unreliable channels. 10
1.3 Network coding can be used to combat packet loss in multiple unicasts by exploiting the broadcast nature of wireless medium. Overheard packets are no longer useless. 12

2.1 Canonical example of network coding: With unit-capacity links and, hence, a min-cut of 2 information units per unit time to each destination, network coding can establish a multicast connection of rate 2 information units per unit time to each destination. This rate is possible because node y is allowed to algebraically combine incoming information units before forwarding. Without network coding, however, the information cannot be multicast simultaneously to both sinks with this max-flow rate because link yz must be shared by both information units. 17
2.2 Illustration of how global and local encoding vectors are formed in a coded network. A global encoding vector of a coded packet must be known to destinations in order for them to decode the intended information. In practical settings, a global encoding vector is appended to the header of a coded packet. 21
2.3 A typical infrastructure wireless mesh network. 24
2.4 IEEE 802.11 channels in the 2.4 GHz ISM band: Only three orthogonal channels are available (i.e., channels 1 6 and 11) 25
2.5 Power spectrum mask of the 2.4 GHz ISM channels illustrated with a particular signal spectrum density $|\frac{\sin(x)}{x}|$. 26
2.6 Two lower subbands of the 5 GHz UNII band: 8 orthogonal channels are shown. 26

2.7 Power spectrum mask of the 5 GHz UNII channel illustrated with some random signal spectrum density. 27

3.1 The system architecture of the multi-radio IWMN. We focus on the backbone portion of the IWMN in this chapter (boxed). 38
3.2 Example of a multi-layered hypergraph for $M = 2$ channels: (Left) The original IWMN $\mathbb{H}$, (Right) Flow network for 2 channels (multi-layered hypergraph). 42
3.3 Subset of nodes on H forming a tandem network from source s to gateway gw and finally to a sink t. 47
3.4 Example of how our channel assignment in both phases works. 58
3.5 Viewing a two-sided multi-assignment problem as a minimum cost flow problem: Node s' is connected to all nodes $l = 1, 2, ..., L$ with zero-cost virtual arcs (s', l), each arc with a feasible flow range $[0, \alpha_l - 1]$. Nodes $k = 1, 2, ..., \tau$ are connected back to node s' via zero-cost virtual arcs (k, s'), each arc with feasible flow range $[0, \alpha_k - 1]$. All other arcs $(l, k), l = 1, 2, ..., L, k = 1, 2, ..., \tau$, have a feasible flow range $[0, 1]$. 67
3.6 Intuitively, per-connection throughput decreases with the number of connections that must be supported. However, it tends to approach the upperbound value as the competition soars in the auction stage, resulting in reduced fairness. 81
3.7 Throughput nearly doubles for one additional radio added at each node. . . 83
3.8 Addressing fairness issues: (a) Fairness improves as the number of channels available in the system increases. (b) Fairness also improves if the batch size is properly adjusted. 84
3.9 On average, RNC-AucCA yields a throughput gain over the single-path routing solution with naive channel assignment (SP-RandCA) and the multi-path routing solutions with more intelligent channel assignment (MP-RCL and MP-AucCA) by 40% and 28.5%, respectively. 85
3.10 Auction-based solutions fail to give fairness when the number of available channels is limited. Fairness improves however if more channels are available. 87
3.11 RNC-AucCA outperforms SP-RandCA by approximately 42% while throughput gains over MP-AucCA and MP-RCL vary between 22% and 35%. . . . 88
3.12 Our experiment results are upper-bounded by the simulation results due to lack of synchronization in the 802.11 MAC. 91
3.13 As the number of channels is fixed, RNC-AucCA is able to effectively exploit the increasing number of radios. 92
3.14 Cumulative distribution of average throughput: As the number of channels increases, RNC-AucCA effectively utilizes the additional radio, compared to SP-RandCA. 93

4.1 Toy example showing how MORE works. 97

4.2 Nodes A, B, C and D form a chain topology using one of their wireless interfaces operating on an orthogonal channel. The number on each link represents the percentage of the successful packet delivery on that link. Packets are transmitted in broadcast mode with omni-directional antennae. All nodes are in promiscuous mode. The goal is to deliver Flow 1, containing packets **x** and **y**, from node A to node C, and Flow 2, containing packet **u**, from node B to node D. 98
4.3 Example of how iCORE works: The goal is to deliver Flow 1, containing packets **x** and **y**, from node A to node C, and Flow 2, containing packet **u** and **v**, from node C back to node A. 102
4.4 iCORE header. 105
4.5 Interface cooperation in iCORE is done via our channel and interface coordinator. 111
4.6 Control flow of iCORE. 112
4.7 Our testbed node: A net4521 Soekris board [148] with two IEEE 802.11b/g Dlink-WNA2330 interfaces [150]. 115
4.8 Possible node locations on our lab floor plan. 116
4.9 CDF of throughput for iCORE and MORE. 117
4.10 Throughput gain with iCORE. 118
4.11 Sustaining throughputs: iCORE is able to sustain the throughputs as offered load increases. 119
4.12 Packet delivery ratio as a batch size is varied. One can see that iCORE is very sensitive to a batch size. 121

5.1 Unicast exchange at an AP/Gateway node facilitating by network coding. . 124
5.2 (a) Buffer dynamics of our ONC model can be viewed as two water tanks with a static open value at the bottom. When a packet arrives, it pushes the water to the other buffer through the valve. The water however must not ever spill from the two tanks. (b) Buffer dynamics of BE-ONC is viewed on the other hand as two empty tanks, where the valve now is allowed to slide up and down to move liquid (packets) between two tanks. (c) Our notion of "time slot." 127
5.3 Expanded view of the queueing model of a wireless unicast exchange communication with opportunistic network coding and dynamic buffer allocation. While the dashed lines represent lossy wireless links with certain packet loss probabilities, the solid lines stand for virtual communication links with no packet error (i.e., each source is viewed as connected to itself). 128

5.4 Example of a discrete time Markov chain queueing model of a wireless unicast exchange network with opportunistic network coding and dynamic buffer allocation. The entire state space and *only some of all possible transitions* are shown for a case of $N_1 = N_2 = 0$ and $N_r = 2$. While the dotted lines represents normal transitions, the solid lines emphasize those transitions that occur after network coding is performed. Transition probabilities are determined by arrival parameters α_1 and α_2 and service parameters β_1 and β_2, corresponding to sources s_1 and s_2, respectively. θ is a service parameter for AP/gateway r. 132
5.5 Model validation for three sets of channel realizations. 137
5.6 ONC in comparison with classical scheduling and BE-ONC. 138

6.1 Example scenario of in-home uplink streaming. 144
6.2 Wireless fountain coding potentially offers delay benefits in streaming in the presence of unreliable wireless channels: (a) Actual system (b) Analogy. . . 149
6.3 Packetization of media data and forward error correction. 152
6.4 Immediate block acknowledgement schemes: (a) Traditional version without wireless fountain coding, and (b) Modified version with wireless fountain coding. The differences between both versions lie in Data-Exchange phase. . 155
6.5 Depending on K and i, the number of BAR/BA exchanges as well as the patterns they occur may vary. This example shows all the possible patterns of BAR/BA exchanges for $K = 3$ and $i = 0, 1, 2, 3$. 165
6.6 Comparison between the extended model and the original model developed in [95]. 172
6.7 (a) Packetization and FEC: The variable number of slices may be mapped into each transport packet because each slice contains the varying number of data bits that need to be encoded. We show a particular instance where a "green" slice contains more data bits than any other slices. Systematic RS code is applied to a group of $k = 2$ source packets to produce $n = 3$ transport packets in this example. (b) Mapping PSNR to VQI as suggested in [89]: A case of the *Foreman* video sequence. 173
6.8 Simulation framework. 176
6.9 Uplink media streaming in a combined wireline/802.11 network. 179
6.10 Model validation: Loss/late probability as a function of FEC protection strength, n/k. 181
6.11 Simulation results: Packet loss/late probability as a function of block sizes K and FEC protection strength n/k. 184
6.12 Simulation results: PSNR as a function of block sizes and FEC protection strength n/k. 184
6.13 Packet loss/late probability as a function of block sizes for fixed FEC protection strength: (a) When the forward channel is better than the backward channel, Modified B-ACK outperforms Traditional B-ACK, but (b) When the backward channel is as good as or better than the forward channel, however, Modified B-ACK prevails only for a certain range of block size K. . . 186

6.14 Performance of the hybrid scheme when the *Foreman* video is transmitted: (a) Packet loss/late probability and (b) Video quality index (VQI). 188
6.15 Performance of the hybrid scheme when the *Mother and Daughter* video is transmitted: (a) Packet loss/late probability and (b) Video quality index (VQI). 189

7.1 Example scenario of in-home downlink streaming. 192
7.2 Packetization of media data and forward error protection (FEC). 198
7.3 (a) Packetization and FEC: The variable number of slices may be mapped into each transport packet because each slice contains the varying number of data bits that need to be encoded. We show a particular instance where a "green" slice contains more data bits than any other slices. Systematic RS code is applied to a group of $k = 2$ source packets to produce $n = 3$ transport packets in this example. (b) Mapping PSNR to VQI as suggested in [89]: A case of the *Foreman* video sequence. 207
7.4 Simulation framework. 208
7.5 Downlink media streaming in a combined wireline/802.11 network. 210
7.6 Performance of the downlink video unicast/multicast as the number of multicast flows increases: (a) WFC helps accommodate more multicast flows without a significant increase in packet loss/late probability, (b) No significant increase in video distortion is seen when WFC is used. 214
7.7 Performance of the downlink video unicast/multicast as the maximum allowed delay of video packets varies: (a) There is a limit on the maximum allowed delay of video packets up to which WFC is more beneficial in terms of packet loss/late probability than streaming without WFC, (b) Mapping packet loss/late probability to video quality index averaged over all flows results in the same conclusion. 216
7.8 Video frame delay for 2 extreme cases: (a) Streaming with WFC outperforms streaming without WFC (MAD = 160 ms), (b)Streaming without WFC outperforms streaming with WFC (MAD = 280 ms). 217

Chapter 1

Introduction

In recent years, infrastructure wireless mesh networks (IWMNs) have been increasingly deployed and widely used for various purposes such as broadband Internet access or mobile telephony backhauling. Compared to traditional single-hop wireless networks, the IWMN deployments are more flexible and self-configured. Wireless coverage can be easily extended by adding more wireless mesh routers to form a wireless infrastructure. Such routers can be additionally equipped with a gateway functionality to interface with a wired infrastructure such as legacy LANs or the Internet. If further equipped with an access point functionality, a wireless router can serve also as an ingress/egress point for mobile/wireless clients' traffic.

With the decline in commodity IEEE 802.11 equipment costs, a node in the IEEE 802.11-based IWMNs is also likely to be equipped with multiple radios to simultaneously communicate over these radios using multiple orthogonal channels. With multiple radios per node, a capacity gain cannot be fully realized, however, if the issues related to routing,

link scheduling and mapping of channels to radios (i.e., channel assignment (CA)) at each wireless router are not properly addressed. In fact, routing and CA are mutually dependent and normally considered jointly. While CA determines the capacity of each link in a network, routing, on the other hand, determines the traffic rate at each link. CA decisions thus affect routing decisions inevitably.

In their current form, IEEE 802.11-based IWMNs are known to provide cheap and moderately fast connectivity for normal Internet browsing activities (e.g., checking emails, Twittering, Facebooking, etc.) for consumers in office/residential buildings and public hotspots. However, with the explosive growth of multimedia content providers on the Internet, consumers now expect to able to access those bandwidth-hungry multimedia contents such as videos, large files, high definition multimedia, among other things, or share their own contents with the same experience they have with normal Internet browsing. Accessing or sharing such contents require high throughput and low delay, which current IEEE 802.11-based IWMNs still struggle to provide.

Motivated by these problems, in this thesis, we ultimately aim to develop an optimal framework for channel assignment and related techniques for multimedia content distribution in the multi-channel multi-radio IEEE 802.11 IWMNs. To this end, we propose an efficient solution for joint channel assignment, link scheduling and routing for nodes that form a wireless infrastructure in a multi-channel multi-radio IWMN. For a given CA, link scheduling and routing solution, we also develop the solutions related to dynamic resource allocation (DRA) for wireless clients attached to a node[1] that has access point and/or gateway functionalities in the underlying IWMN. The key idea underlying all of our

[1]We now refer to such a node as either *access point/gateway* or *AP/gateway*.

proposed solutions is to allow nodes to code packets before forwarding them, i.e., to perform network coding.

Introduced by Ahlswede et al. [4], network coding is a new transmission paradigm that allows intermediate nodes in a network to not only forward but also algebraically combine multiple incoming packets into one or multiple output packets before forwarding them onto outgoing links. In doing so, theoretically, a max-flow min-cut capacity can be achieved by any source-sink pair of a particular multicast session in the network as if it had sole access to network resources [4]. Koetter [6] and Li and Cai [5] later established that such combining can be linear in order to achieve the maximum multicast capacity of a given network. To arrive at the maximum multicast capacity, Jaggi et al. [123] proved that such linear coding and decoding can be performed in polynomial time. Ho et al. [7] subsequently showed that random coefficients, rather than the deterministic ones as shown in [123], can also be used to achieve the same capacity. While the theoretical foundations of network coding have advanced very close to maturity [4–8], a search for its applications is still evolving, especially in the wireless arena [10,11]. Our solutions are part of this ongoing quest.

While most promising results have been extensively reported in prior work for multicast, the benefits of network coding for unicast still remain largely unexplored, especially in the multi-channel multi-radio IWMNs context. In this context, the broadcast nature of wireless media on multiple channels turns out to be very useful for extracting the benefits of network coding for unicast. In general, a single wireless transmission is often received by more than one node. Nodes located farther than a one-hop distance may

overhear transmissions on one channel and help forward the received packets for previous hops on another channel operating on another radio. *Opportunistic overhearing/listening* has been extensively studied in conjunction with inter-flow network coding[2] in [29] for a single-channel single-radio wireless mesh network (WMN). Promising unicast throughput gains, when intra-flow network coding is used instead, have also been shown in [35] through experiments for a single-channel single-radio WMN.

In this thesis, we first introduce a sub-optimal scheme that performs joint channel assignment, broadcast link scheduling and routing[3] for multi-radio nodes with access point and/or gateway functionalities (i.e., for nodes that form a wireless infrastructure in the IWMN). To this end, we systematically formulate, for throughput optimization, the problem as a linear program, explicitly taking into account opportunistic listening and lossiness of broadcast links. We also develop a miniature IEEE 802.11b/g testbed, called iCORE, as an experimental platform for empirically evaluating our sub-optimal solution. We give the details of iCORE and illustrate its basic functionality as a repeater-based multi-channel multi-radio WiFi in Chapter 4. Relying on iCORE as a platform for experimental evaluation, the joint CA, link scheduling and network coding solution for the IWMN is presented in Chapter 3.

In the second part of this thesis, we introduce three Ap/Gateway-oriented link-level solutions related to dynamic buffer allocation and error control (Chapters 5–7). Aiming for a low-delay/low-loss performance in delivering multimedia contents at the combined wired/wireless edge portion of the underlying IWMN, these solutions will be performed at

[2]Packets are combined across flows.

[3]Since network coding embraces routing as a special case, we sometimes refer to this problem as *joint channel assignment, link scheduling and network coding*.

nodes attached to a particular access point/gateway node on a given channel of the underlying IWMN. Our results in both parts suggest promising gains over traditional methods without network coding for media content distribution in multi-channel multi-radio IWMNs.

We provide an overview of each solution as follows.

1.1 Network-wide Solution: Joint CA, Link Scheduling and Network Coding

We address in Chapter 3 the joint problem of channel assignment, broadcast link scheduling and routing in the IWMNs where intra-flow network coding is employed for unicast throughput optimization.

Compared to single-hop networks, multi-hop IWMNs can potentially embrace the broadcast benefits of a wireless medium in a more flexible manner. Rather than being point-to-point, links in the IWMNs may originate from a single node and reach more than one other node. Nodes located farther than a one-hop distance and overhearing such transmissions may opportunistically help relay packets for previous hops. This phenomenon is called *opportunistic overhearing/listening*. With multiple radios, a node can also improve its capacity by transmitting over multiple radios simultaneously using orthogonal channels. A packet received on one channel by a node may leave the node on another channel using the same or another radio. Capitalizing on these potential advantages thus requires effective routing and efficient mapping of channels to radios (channel assignment). While efficient channel assignment can greatly reduce interference from nearby transmitters, effective routing can potentially relieve congestion on paths to the infrastructure. Routing

however requires that only packets pertaining to a particular connection be routed on a pre-determined route. Random network coding breaks this constraint by allowing nodes to randomly mix packets overheard so far before forwarding. A relay node thus only needs to know *how many packets*, and not *which packets*, it should send. We mathematically formulate the joint problem of random network coding, channel assignment and broadcast link scheduling, taking into account opportunistic overhearing, the interference constraints, the coding constraints, the number of orthogonal channels, the number of radios per node, and fairness among unicast connections. Based on this formulation, we develop a sub-optimal, auction-based solution for overall network throughput optimization.

Performance evaluation results show that our algorithm can effectively exploit multiple radios and channels and can cope with fairness issues arising from auctions. Our algorithm also shows promising gains over traditional routing solutions in which various channel assignment strategies are used. Our experimental results further confirm this finding. We use iCORE as our experimental platform. iCORE – the **i**nterface **CO**operation **R**epeater-aided network coding **E**ngine – is designed and implemented to utilize idle interfaces opportunistically. iCORE allows packets to move across physical radios and be coded either within a flow or across flows whenever it sees more transmit opportunities. iCORE sits on top of the IEEE 802.11b/g MAC, making it independent from device drivers. We give the implementation details of iCORE as well as illustrate its basic functionality in Chapter 4.

1.2 AP/Gateway-oriented Solution I: Dynamic Buffer Allocation and Opportunistic Network Coding

Chapter 5 investigates, from the queueing perspective, the link layer interactions between opportunistic network coding (ONC) and dynamic buffer allocation (DBA) for a simple unicast exchange at a particular access point (AP) of the IWMN, as shown in Fig. 1.1a/b. While a traditional forwarding solution would require at least four transmissions to allow nodes A and B to exchange a pair of packets, network coding needs just three transmissions to complete the exchange. With network coding, the AP combines incoming packets using a simple eXclusive OR (XOR) operator and broadcasts the combined packet to both parties. Nodes A and B then obtain each other's packet by XOR-ing the combined packet with their own packet. Due to random packet arrival/departure/erasure, however, the AP may or may not have packets from both flows to combine. This is where opportunistic network coding (ONC) comes in. With ONC, the AP combines packets whenever packets from both flows are present, otherwise it just transmits the head-of-line (HOL) packet uncoded.

This chapter deals with this type of network coding as well as with dynamic buffer allocation (DBA). In DBA, a virtual pair of queues at the AP is dynamically maintained while accommodating packets coming from both flows, as shown in Fig. 1.1c. Such dynamic buffer allocation in our context operates much like two water tanks positioned side by side and connected at the bottom by a *static open valve*. We develop a discrete-time Markov queueing model to capture this phenomenon. Motivated by an increasing demand for real-time applications, we show the delay benefit of network coding for a lossy wireless unicast

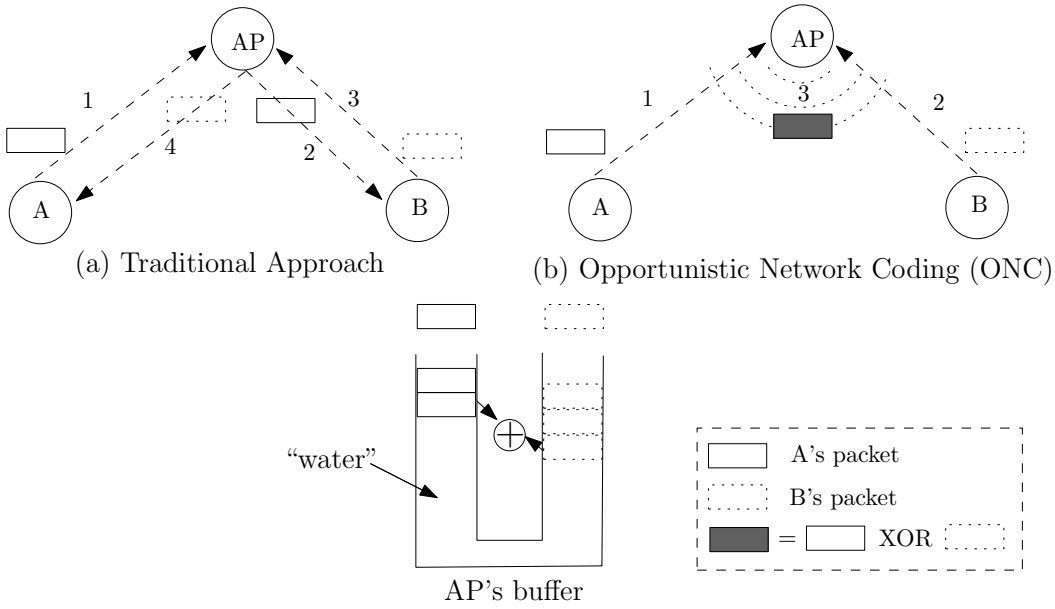


Figure 1.1: Interactions between opportunistic network coding (ONC) and dynamic buffer allocation (DBA). (a & b) Network coding allows A and B to exchange a pair of packets using only 3 transmissions instead of 4 (numbers on arrows show the order of transmission). (c) Dynamic buffer allocation provides fairness and delay benefits among flows with random packet arrival and departure. It operates similarly to a pair of water tanks connected by a valve.

exchange network over that of the classical scheduling, no-coding schemes such as first-in first-out and round-robin. To improve the performance further, we also propose a simple scheduling algorithm, called *buffer equalized opportunistic network coding*, which operates analogously to a pair of water tanks with a *sliding open valve* in the middle to allow packets to move across "tanks". We show that our *sliding open valve* scheme improves in terms of delay over the *static open valve* scheme.

1.3 AP/Gateway-oriented Solution II: Uplink Streaming with Wireless Fountain Coding

Chapter 6 addresses the problem of unicasting a packetized multimedia stream over a lossy combined wireline/802.11 portion of the underlying IWMN in an uplink direction. We use a version of network coding, which we term *wireless fountain coding (WFC)*, to ease the transmission mechanisms at the IEEE 802.11e MAC layer. WFC is a packet-level coding scheme which codes packets in a similar manner to intra-session network coding but delivers them in a manner similar to fountain coding. WFC allows a wireless node to linearly combine packets and then transmit the combined packets via an AP/gateway to a wired destination.

To quickly quantify the benefits of using this version of intra-flow network coding in this scenario, consider, for example, Fig. 1.2 in which node A attempts to unicast a flow of n packets, $\mathbf{p}_1, ..., \mathbf{p}_n$, to the AP. For simplicity, suppose the links in both directions are 50% reliable. According to the current IEEE 802.11 standard, four transmissions will be needed to successfully deliver each packet to AP, i.e., two transmissions for the packet itself and the

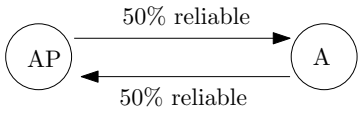


Figure 1.2: Reliability benefits of network coding can be realized in a single unicast flow in the presence of unreliable channels.

other two for acknowledgement. The total of $4n$ transmissions then will be required for the delivery of the whole flow. On the other hand, when intra-flow network coding is used, only $2n + 2$ transmissions – $2n$ transmissions for delivering n combined packets and 2 attempts for a batch-acknowledgement – are required with a relatively less complex protocol. That is, for each transmission i, node A sends a random linear combination of the n packets

$$\mathbf{x}_i = c_{i,1}\mathbf{p}_1 + c_{i,2}\mathbf{p}_2 + ... + c_{i,n}\mathbf{p}_n,$$

where $\mathbf{p}_i$ is the ith packet in the flow, and $c_{i,j}$'s are coefficients randomly generated for the ith transmission. Once AP has received n such combined packets and has successfully solved (1.1), it sends a positive batch-acknowledgement back to node A. Without network coding, a batch acknowledgement could also be implemented such that each batch acknowledgement packet will have to inform node A of missing packets. This relatively more complex scheme requires at least $2n + 2\log_2 n$ transmissions to complete sending the whole flow. Clearly, this number, $2n + 2\log_2 n$, is still greater than that in the coding case $(2n + 2)$.

$$\begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_n \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ c_{2,1} & \cdots & c_{2,n} \\ \vdots & \ddots & \vdots \\ c_{n,1} & \cdots & c_{n,n} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix}, \tag{1.1}$$

In this chapter, we analyze this reliability benefit of intra-flow network coding when employed in conjunction with other classical error control techniques at different layers in a combined wireline/802.11 network. Specifically, we develop performance models for delay-sensitive uplink media streaming over a combined wireline/802.11 network. Since the wireless channel is normally a bottleneck for such streaming, we modify the traditional 802.11e block acknowledgement (B-ACK) scheme to work with *wireless fountain coding (WFC)*. By using this modified B-ACK scheme, protocol complexity and wireless link-layer delay can be potentially reduced. We analytically quantify this delay and use it to derive end-to-end packet loss/late probabilities when automatic repeat request (ARQ) and forward error correction (FEC) are jointly employed at the application layer. We develop an integrated ns-3/EvalVid simulator [97,98] to validate our models and compare them with the case when the traditional 802.11e B-ACK scheme is employed. Through simulations of video streaming, we observe that the modified B-ACK scheme does not always perform better than the traditional B-ACK scheme in terms of end-to-end packet loss/late probability and video distortion under certain conditions of the wireless channel. This observation leads us to propose a hybrid scheme that switches between the modified and traditional B-ACK strategies according to the conditions of the wireless channel and the number of packets to transmit in a block. Via simulations, we show the benefits of the hybrid scheme when compared with the traditional IEEE 802.11e B-ACK scheme under various network settings.

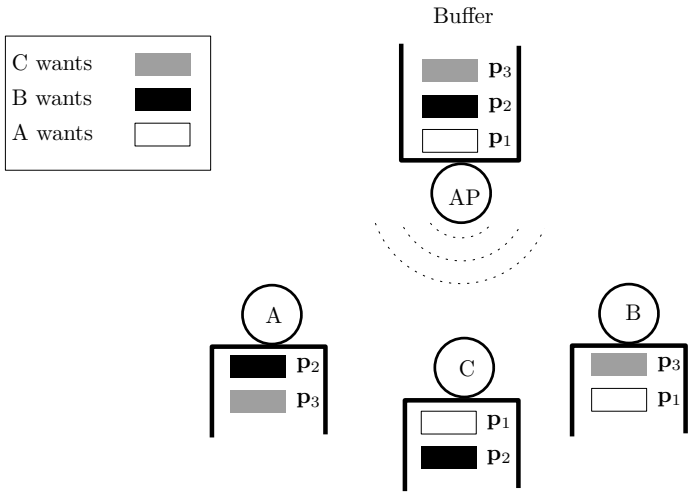


Figure 1.3: Network coding can be used to combat packet loss in multiple unicasts by exploiting the broadcast nature of wireless medium. Overheard packets are no longer useless.

1.4 AP/Gateway-oriented Solution III: Downlink Streaming with Wireless Fountain Coding

In Chapter 7, we still deal with packetized multimedia streaming in a combined wireline/802.11 network, but this time in a downlink direction. Both unicast and multicast streams are to be jointly supported at a particular AP/gateway. We employ wireless fountain coding (WFC) at the wireless link to alleviate the problem of unreliable downlink receptions by observing that wireless receptions at different nodes located at different locations are normally highly independent and error-prone. This is best illustrated by an example.

Consider the network shown in Fig. 1.3. Wireless nodes A, B and C each demand a different packet, as shown in the legend box. By exploiting the broadcast nature of omni-

directional transmission, wireless receivers listen opportunistically to ongoing transmissions from AP and store overheard packets in their buffers. After a certain period, they overheard the packets as shown in their respective buffer in Fig. 1.3, but none of these is destined for them. If the nodes are required to inform the AP of their buffer contents, the AP can use this information to determine which packets to combine in order for the combined packet to be useful to as many wireless receivers as possible in a single coded transmission. From Fig. 1.3, given the current contents of the buffers at every node, the AP finds that the best coding option is to combine $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ so that all the nodes can decode to recover their desired packet. Combining $\mathbf{p}_1$ and $\mathbf{p}_2$ is also possible but only nodes A and B can decode, resulting in a lower aggregate throughput. The worst option is to send any of the three packets uncoded which is useful only to the node that wants it, resulting in an even lower aggregate throughput.

In this chapter, we analyze the reliability benefit of such network coding for concurrent unicast and multicast multimedia streaming over a combined wireline/802.11 network in a downlink direction for an arbitrary number of wireless receivers. Since the wireless channel is normally the bottleneck for media streaming, we propose that WFC be used over the wireless downlink in order to efficiently utilize the wireless bandwidth and exploit the broadcast nature of the channel. Forward error correction (FEC) is also used to combat errors at the application-layer. We analytically obtain the moment generating function (MGF) for the wireless link-layer delay incurred by WFC. With the MGF, the expected value of this wireless link-layer delay is found and used by AP, who has no knowledge of the buffer contents of wireless receivers, to make a coding-based decision. We then derive the

end-to-end packet loss/delay probability based on the MGF. Similar to the uplink streaming evaluation, we develop an integrated ns-3/EvalVid simulator to evaluate our proposed system and compare it with the traditional 802.11e scheme which is without WFC capability but equipped with application- and link-layer retransmission mechanisms. Through extensive simulations of video streaming, we show that streaming with WFC is able to support more concurrent video flows compared to the traditional scheme. When the deadlines imposed on video packets are relatively stringent, streaming with WFC also shows superior performance in terms of packet loss/delay probability, video distortion, and video frame delay, over the traditional scheme.

Chapter 2

Background

In this chapter, we give an overview of network coding and current IWMNs. We begin in Section 2.1 with an introduction of linear network coding. Current IEEE 802.11-based IWMNs are described in Section 2.2.

2.1 Linear Network Coding

Network coding theory originates from a very simple observation as described by Ahlswede et al. [4] as

> "From the information-theoretic point of view, there is no reason to restrict the function of a [communication] node to that of a switch [which only relays information from an input link to an output link, or replicates information received from an input link and sends it to a certain set of output links]."

Prior to this observation, communication networks are thought to resemble transportation systems in that information units (e.g., data packets) are treated as *physically independent* entities, just like vehicles, that traverse the networks to their intended destinations. Whereas

links that carry information are physical, the information itself is not. Given this "non-physical" nature of information, there is nothing that prevents us from compressing it or combining it with other information as it traverses from node to node, as long as we have a mechanism that can recover it at a destination. This is the perspective that gives rise to the network coding theory. In fact, two fundamental concepts that inspire the birth of network coding theory also include *multilevel diversity coding* (MDC) and *distributed source coding* (DSC) proposed a few years earlier in [12–14] and [15], respectively. Network coding is, however, more general than MDC and DSC as it encompasses these two concepts as special cases.

To appreciate the beauty of network coding, let us consider a network represented by a directed acyclic graph shown in Fig. 2.1. The goal is to multicast two information units, say, packets $\mathbf{p}_1$ and $\mathbf{p}_2$, to common destinations t_1 and t_2 [1]. Suppose each link (edge) of this network (graph) has unit capacity (i.e., one information unit/unit time). From the much celebrated min-cut max-flow theorem [16], we know that a min-cut between source s to each sink t_i, $i \in \{1, 2\}$, is exactly 2. That is, information can be sent from the source to each destination at the maximum rate of 2 information units per unit time when the network's resources are dedicated solely for that destination. Equivalently, there exist exactly 2 edge-disjoint paths, shown as thick dashed and solid routes in Fig. 2.1, between s and t_i, through which two different information units may traverse to t_i. Evidently, this rate is impossible if we were to allow both t_1 and t_2 to use the network simultaneously as t_1 and t_2 will have to share the link yz. Here is where network coding comes in. If we were to

[1] Although this research deals mostly with unicast communications in lossy wireless networks, understanding network coding from a multicast communications perspective for lossless wired networks (where the communications benefits resulting from network coding are already known) is a reasonable starting point.

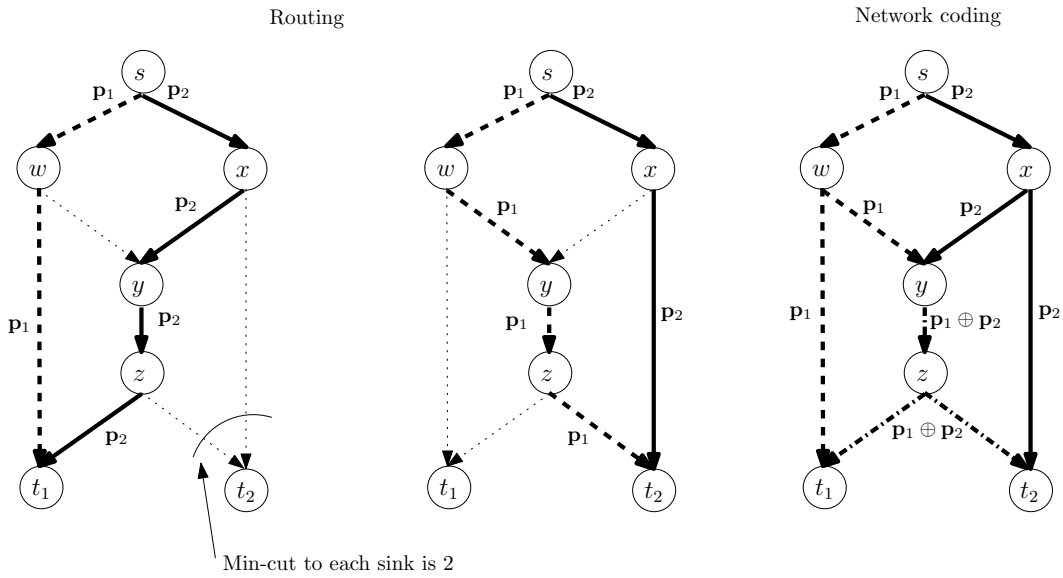


Figure 2.1: Canonical example of network coding: With unit-capacity links and, hence, a min-cut of 2 information units per unit time to each destination, network coding can establish a multicast connection of rate 2 information units per unit time to each destination. This rate is possible because node y is allowed to algebraically combine incoming information units before forwarding. Without network coding, however, the information cannot be multicast simultaneously to both sinks with this max-flow rate because link yz must be shared by both information units.

allow the intermediate node y to not only *forward* but also *combine* (e.g., using eXclusive OR) its incoming information units from nodes w and x, the information can be delivered to both t_1 and t_2 simultaneously with the rate as if each destination t_i had sole access to the network resources [17]. We shown this combining in the far right copy of the network (See Fig. 2.1).

In general, if we let a directed acyclic graph $G = (V, E)$ represent any communication network, where V is a set of nodes (vertices) and E a set of links (edges), then the main theorem of network coding states that:

Theorem 2.1.1 (Main Theorem of Network Coding [17]) *Given (1) the graph $G = (V, E)$ with unit capacity edges, (2) h unit rate sources located on the same vertex $s \in V$ of the graph, and (3) a set of T sinks, $\{t_1, ..., t_T\} \subset V$. Suppose the min-cut from s to each sink is h. Then there exist a multicast transmission scheme over a large enough finite field $\mathbf{GF}(q)$, in which intermediate network nodes linearly combine their incoming information symbols over $\mathbf{GF}(q)$, that delivers the information from the source simultaneously to each sink at a rate equal to h.*

In other words, to achieve the multicast rate of h, it is sufficient for intermediate network nodes to *linearly* combine the incoming information symbols before forwarding to the next nodes [5]. The theorem also states the existence of linear network codes over some large enough finite field $\mathbf{GF}(q)$. The complexity of employing network coding is thus due in part to the size of the finite field $\mathbf{GF}(q)$. For the complete proof of this theorem from the information-theoretic point of view, the reader is referred to [4, 5]. The sketch of this proof can also be found in [17, chap. 3]. The proof from the algebraic point of view is presented

in [17, chap. 2].

In the following, we explain how information units (data packets) are linearly combined and recovered, or decoded, when network coding is applied to a network for multicast.

2.1.1 Linear Combining

We restrict our attention to linear combining in which multiplication and addition over a finite field suffice. Our explanation on linear combining in this subsection applies to both *inter-flow* and *intra-flow* network coding. Whereas the former involves coding packets from different flows (i.e., packets destined to different destinations), the latter involves coding packets within a flow (i.e., packets destined to the same destination). This research involves both intra-flow and inter-flow network coding.

Let an information flow be represented by a sequence of data packets. Each data packet in turn consists of a sequence of symbols each of which is one of the elements in the finite field $\mathbf{GF}(q)$. That is, each symbol is a group of $\log_2 q$ consecutive bits. If L_{bit} is the total number of bits in each data packet, then a vector of $\frac{\mathsf{L}_{bit}}{\log_2 q}$ symbols can be used to represent each data packet. Refer to this vector as an information vector.

In a network using linear network coding, an output packet on a particular outgoing link of any node can be written as

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{p}_i,$$

where $\mathbf{c} = [c_1, \ldots, c_n]$ is a sequence of coefficients (i.e., symbols drawn from $\mathbf{GF}(q)$) called an encoding vector, and $\mathbf{p}_i$ is an information vector received from an ith incoming link of

the node. The addition and multiplication are performed over $\mathbf{GF}(q)$. The coefficients c_i can be either deterministically determined in a centralized manner [18] or randomly selected by individual nodes [7, 9, 91]. For the deterministic coefficients, this implies that there is no need for a coded packet to carry an encoding vector $\mathbf{c}$ in addition to its information vector because such an encoding vector is fixed and known at every node. The polynomial-time algorithm for determining such a code can be found in [18]. On the other hand, for the random coefficients, each packet in the network must carry an encoding vector along with its information vector. Since coding may take place repetitively over subsequent intermediate nodes, the distinction between a local encoding vector and a global one in this case must be made. This is best illustrated by an example. Denote by $(\mathbf{c}_j, \mathbf{x}_j)$ a coded packet for the j outgoing link, where $\mathbf{c}_j$ is the global encoding vector.

Consider the network shown in Fig. 2.2. Assuming unit capacity (e.g., one packet per unit time) on each link of the network, it can easily be verified that the min-cut from source s to any sink $t_i, i \in \{1, 2\}$, is exactly 2. The goal is to simultaneously multicast data packets $\mathbf{p}_1$ and $\mathbf{p}_2$ to nodes t_1 and t_2 at the rate equal to this min-cut. Without network coding, this rate is not achievable because the capacities on links BD, DG and GH will have to be shared by separate transmissions of $\mathbf{p}_1$ and $\mathbf{p}_2$, in some fashion. However, if network coding is used, $\mathbf{p}_1$ and $\mathbf{p}_2$ can be linearly combined at node B into one coded packet which is subsequently forwarded to nodes D, G and t_2. From Fig. 2.2 we see that the local encoding vector of node B is $[c_1, c_2]$ which happens to be a global encoding vector as well. On the other hand; while the local encoding vector of node G is $[c_3, c_4]$, its global encoding vector is different and given by $[c_3 + c_1c_4, c_2c_4]$ because, at node G, packet $\mathbf{p}_1$ is linearly combined

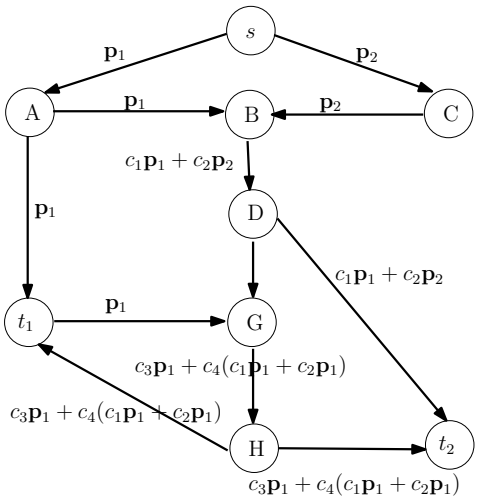


Figure 2.2: Illustration of how global and local encoding vectors are formed in a coded network. A global encoding vector of a coded packet must be known to destinations in order for them to decode the intended information. In practical settings, a global encoding vector is appended to the header of a coded packet.

with the previously coded packet $c_1\mathbf{p}_1 + c_2\mathbf{p}_2$. Note that a source packet $\mathbf{p}_i$, $i \in \{1, 2\}$, is also associated with an encoding vector $\mathbf{c}_i = [0, \ldots, 0, 1, 0, \ldots, 0]$, where 1 appears at the ith position.

2.1.2 Decoding

Decoding takes place individually at each destination and requires solving a system of linear equations. Suppose a destination has received k coded packets $(\mathbf{c}_1, \mathbf{x}_1), (\mathbf{c}_2, \mathbf{x}_2)$, $\ldots, (\mathbf{c}_k, \mathbf{x}_k)$. To recover the original n data packets[2], $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$, the destination solves the following system of linear equations

$$\mathbf{x}_1 = c_{1,1}\mathbf{p}_1 + ... + c_{1,n}\mathbf{p}_n \tag{2.1}$$

$$\mathbf{x}_2 = c_{2,1}\mathbf{p}_1 + ... + c_{2,n}\mathbf{p}_n \tag{2.2}$$

$$\cdots \tag{2.3}$$

$$\mathbf{x}_k = c_{k,1}\mathbf{p}_1 + ... + c_{k,n}\mathbf{p}_n, \tag{2.4}$$

where $c_{j,i}$ is the ith element of the encoding vector associated with the jth coded packet received. In matrix form, the above system of equations is represented as

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_k \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ c_{2,1} & \cdots & c_{2,n} \\ \vdots & \ddots & \vdots \\ c_{k,1} & \cdots & c_{k,n} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_n \end{pmatrix} \quad \Rightarrow \quad \mathbf{X} = \mathbf{CP}, \tag{2.5}$$

which yields a consistent and unique solution if and only if coefficient matrix $\mathbf{C}$ has full row rank [5]. The condition $k \geq n$ is therefore only necessary but not sufficient because

[2] n is equal to 2 for the example in Fig. 2.2.

even if $k \geq n$, it is still possible for a coefficient matrix $\mathbf{C}$ to have linearly dependent rows, which will in effect reduce the rank of $\mathbf{C}$. The probability of having such linearly dependent row depends on the size of the finite field $\mathbf{GF}(q)$ [7, 9, 91]. It is reported in [19] that, for most practical settings with relatively small field sizes, this probability is still very small. Otherwise specified, in this thesis, we specifically deal with random linear network coding with the field size of 256 (i.e., 8 bits per symbol).

Now, returning to the example in Fig. 2.2, destination t_2, for instance, recovers source packets $\mathbf{p}_1$ and $\mathbf{p}_2$ by solving:

$$\begin{pmatrix} \mathbf{x}_1 \\ \\ \mathbf{x}_1 \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ \\ c_3 + c_1 c_4 & c_2 c_4 \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \\ \mathbf{p}_2 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \mathbf{p}_1 \\ \\ \mathbf{p}_2 \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ \\ c_3 + c_1 c_4 & c_2 c_4 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{x}_1 \\ \\ \mathbf{x}_2 \end{pmatrix}.$$

2.2 Infrastructure Wireless Mesh Networks (IWMNs)

Our research focuses on an IEEE 802.11-based infrastructure mesh network (IWMN) where some nodes may be equipped with access point and/or gateway functionalities, as shown in Fig. 2.3. Each node with access point functionality will serve as an ingress or egress for the aggregate traffic associated with the wireless/mobile clients in its coverage area. Nodes with gateway functionality will interface directly with a wired infrastructure such as legacy LANs or the Internet. Packets may be routed via multiple wireless hops in and out of the wired infrastructure. With possibly multiple radios installed at each node, an orthogonal channel will have to be assigned to each existing radio. We next describe the channelization of the existing IEEE 802.11 standards.

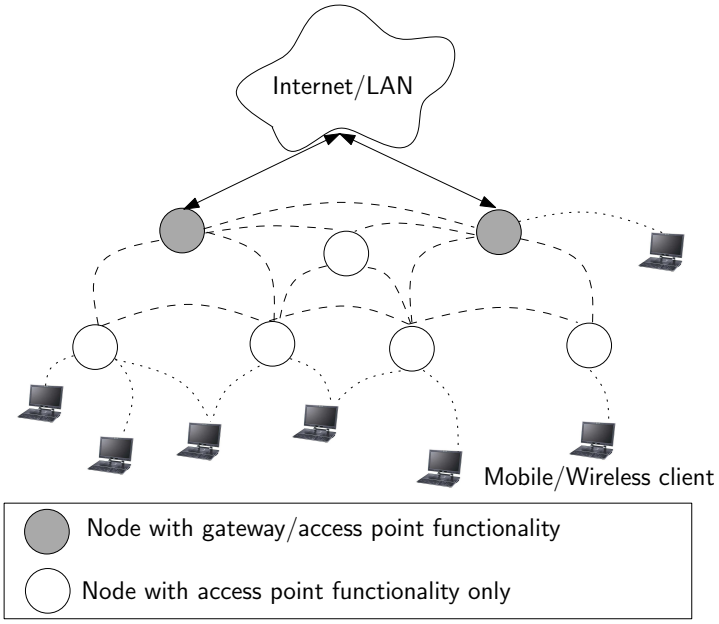


Figure 2.3: A typical infrastructure wireless mesh network.

2.2.1 Channelization

Currently, two unlicensed frequency spectrum bands are available for use in IEEE 802.11 networks: 1) 2.4 GHz Industrial, Scientific, and Medical (ISM) band, and 2) 5 GHz Unlicensed National Information Infrastructure (UNII) band, [1] and [2]. While the legacy IEEE 802.11 and enhanced IEEE 802.11b/g based networks operate on the 2.4-GHz band, the IEEE 802.11a based networks employ the 5-GHz band. Both bands are available internationally. The number of allowable channels however varies from country to country due to each country's regulations on radio spectrum allocation. In particular, while most European countries and Australia allow channels 1 up to 13 in the 802.11b/g band, most North, Central and South American countries only allow channels 1 to 11 in the same band [3]. In Japan, all 14 channels are allowed. These regulations are however subject to

change.

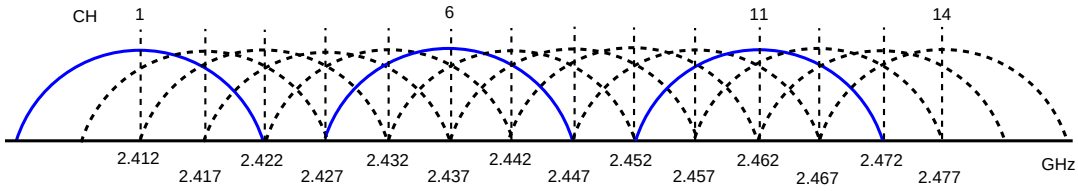


Figure 2.4: IEEE 802.11 channels in the 2.4 GHz ISM band: Only three orthogonal channels are available (i.e., channels 1 6 and 11)

As shown in Fig. 2.4, the 2.4-GHz band consists of 14 overlapping channels each of which occupies a bandwidth of 22 MHz. Due to the spectral overlaps of channels within this band, the standard also specifies the allowable levels of power overlaps between overlapping channels. Specifically, as shown in Fig. 2.5, the signal must drop 30 dB and 50 dB below its peak power when operating at ± 11 MHz and ± 22 MHz apart from the center frequency, respectively [1]. Such power constraints, however, can be relaxed[3] when operating on orthogonal channels, i.e., channels 1, 6, and 11.

The 5-GHz UNII band contains three subbands referred to as *low*, *middle*, and *high*, each of which further consists of four non-overlapping (orthogonal) channels as shown in Fig. 2.6. Each channel occupies a bandwidth of 20 MHz. Currently, the 5 GHz band is still significantly less populated than the 2.4 GHz because 1) 802.11a equipments are not so widespread as 802.11b/g equipment, and 2) due to the higher frequency, 802.11a signals cannot penetrate as far as 802.11b/g signals and are absorbed more readily by obstacles. As in the 2.4 GHz band, the power spectrum mask is also specified in the 5-GHz band [3].

[3]That is, when operating on different orthogonal channels, nodes may transmit with full power.

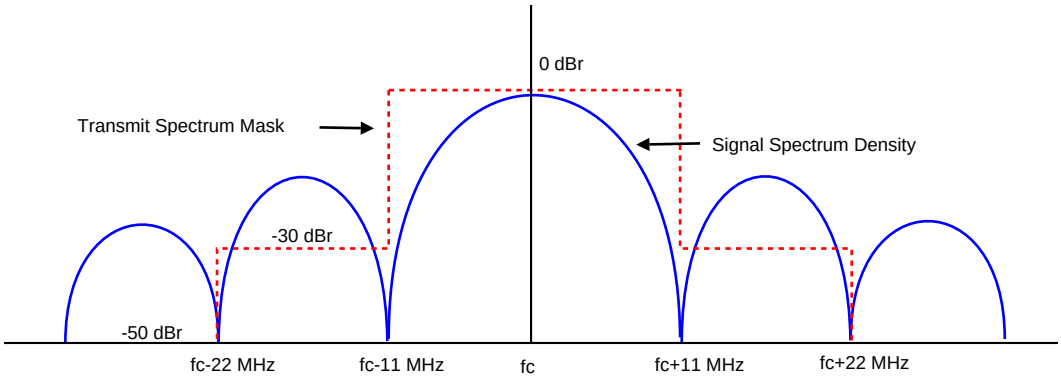


Figure 2.5: Power spectrum mask of the 2.4 GHz ISM channels illustrated with a particular signal spectrum density $|\frac{\sin(x)}{x}|$.

This is shown in Fig. 2.7 where the signal must drop 20 dB, 28 dB, and 40 dB at 11 MHz, 20 MHz, and 30 MHz apart from the center frequency, respectively.

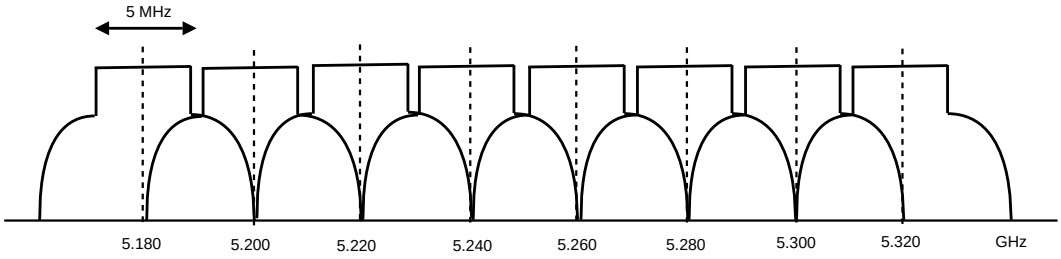


Figure 2.6: Two lower subbands of the 5 GHz UNII band: 8 orthogonal channels are shown.

2.2.2 Medium Access Control

IEEE 802.11 uses a default, contention-based mode of medium access control, called Distributed Coordination Function (DCF), to accommodate multiple wireless accesses on a particular channel. DCF is based on the carrier sense multiple access/collision avoidance (CSMA/CA) technique which works as follows. A node first senses the wireless

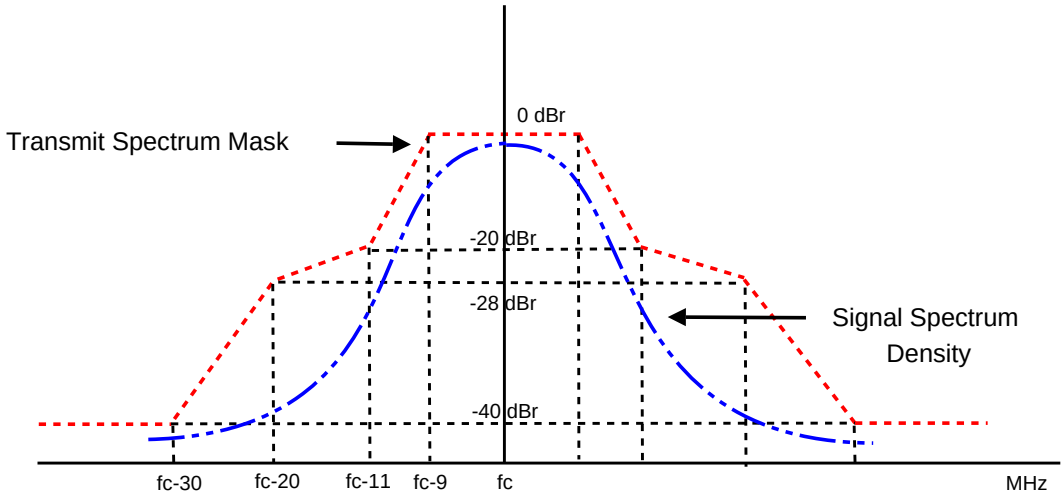


Figure 2.7: Power spectrum mask of the 5 GHz UNII channel illustrated with some random signal spectrum density.

medium for transmission opportunity. If the medium is idle, the node starts its transmission. Otherwise, it backs off and waits for a random period of time before contending again for the medium. In case two nodes sense an idle channel and start their transmissions simultaneously, *collision* is said to occur. If collided, both parties will have to back off for a random period of time and then retry in order to reduce the probability of further collision.

In order to lower the collision probability, wireless nodes may resort to an optional mode of handshake-based medium access mechanism known as Request-To-Send/Clear-To-Send (RTS/CTS) signaling. In the RTS/CTS mode, a node, prior to transmitting packets, broadcasts an RTS packet to reserve the medium. If the medium is idle, the destination responds with a CTS signal. The node then seizes the medium and starts transmitting. In addition to DCF, the IEEE 802.11 standard provides an alternative access method known as Point Coordination Function (PCF) in which a point coordinator, or an AP in our case,

determines which node has the right to transmit or receive.

Part I

Network-wide Solution: Joint Channel Assignment, Scheduling and Network Coding

Chapter 3

Joint Channel Assignment, Link Scheduling and Network Coding for Throughput Optimization

This chapter tackles the network-wide problem the IWMN faces when routing packets between multi-radio nodes that form a wireless infrastructure of the IWMN. As mentioned in Chapter 1, with multiple radios per node, a capacity gain cannot be fully realized if the issues related to routing, link scheduling and channel assignment (CA) (mapping of channels to radios at each node) are not properly addressed. Routing and CA are mutually dependent and must be considered jointly. While CA determines the capacity of each link in a network, routing determines the traffic rate at each link. CA decisions thus affect routing decisions inevitably. From a theoretical perspective, it has been shown that the joint problem of CA and routing is NP-complete. Various sub-optimal solutions have

been proposed in the literature to address this problem in the conventional routed IWMNs (See [109–116, 118–122]).

Since routing is a special case of network coding[1], the key problem we address in this chapter can be thought of as the (network) *coded* IWMNs analog of the joint problem of CA, routing and scheduling in the conventional *routed* IWMNs. Nodes in our system are allowed to combine packets based on the distributed random linear network coding scheme [7]. By doing such random combining, generally speaking, it does not matter *what* is received or lost at a destination, but it only matters that *enough* is received. Although most promising results on network coding have been presented for multicast in the wired domain, the broadcast nature of a lossy wireless medium turns out to be very useful for extracting the benefits of network coding for unicast as well. In general, a single wireless transmission is often received by more than one node. Nodes located farther than a one-hop distance may overhear transmissions and help relay packets for previous hops. Such *opportunistic overhearing/listening* has been extensively studied in conjunction with inter-flow network coding[2] in [29]. Substantial unicast throughput gains over traditional routing for single-channel single-radio IWMNs have been reported [29] through extensive experiments. Promising unicast throughput gains, when intra-flow network coding is used, have also been shown in [35] through experiments.

In this chapter, we explicitly take opportunistic listening into our optimization framework for the joint problem of CA, network coding and broadcast link scheduling in multi-radio IWMNs, aiming to maximize throughputs for multiple unicast connections in a

[1]That is, for each routed transmission there is only one packet to combine and coefficients for such combining are all ones.

[2]Packets are combined across flows.

fair manner. Although such explicit modeling of opportunistic listening may increase the computational complexity of the formulated optimization problem, we ask if the throughput gains are worth the computational efforts when compared with a traditional routing approach. The answer is positive. Also, it turns out that our resulting optimization framework is general enough to embrace multi-path routing as a special case. We only deal with the distributed version of intra-flow random network coding (RNC) [44] in this chapter. The choice of intra-flow coding[3] is desirable here because it tends to incur fixed and less overhead than inter-flow coding [35]. For inter-flow network coding, packets are coded and decoded hop by hop. Which packets to code depend on the unforeseen buffer contents of nexthops. Variable overhead and unreliability thus may penetrate from hop to hop. Reliability however is provided without feedback by intra-flow network coding, thus eliminating the need to schedule retransmissions and manage interactions between retransmission protocols at different layers. This feedforward approach to reliability in intra-flow network coding is different from the digital fountain approach where coding is performed only at the source. In intra-flow network coding, all nodes participate in coding as packets are relayed through a network.

Methodology

We use the following methodology to address the problem of joint CA, distributed RNC and scheduling for the multi-channel multi-radio IWMNs with opportunistic listening.

- **Modeling an IWMN as a flow network**: We first model the given multi-radio

[3]For intra-flow network coding, packets are combined only within their respective flow.

IWMN as a two-dimensional directed hypergraph whose vertices correspond to the nodes of the IWMN and hyperarcs correspond to the broadcast links of the IWMN. We then transform this hypergraph into a three-dimensional flow network where, except for the source, each horizontal layer corresponds to a copy of the original two-dimensional directed hypergraph: each layer for each of the orthogonal channels.

- **Formulating a linear program**: We then formulate the problem as a linear program (LP) that aims at optimizing the throughputs for all the connections proportionally, subject to the interference constraints, the coding constraints, the number of orthogonal channels available, the number of radios per node, and fairness. The solution to this linear program is the vector of packet injection rates, which basically specifies how many coded packets each node on the three-dimensional flow network should transmit.

- **Broadcast link scheduling for 1st channel**: The solution from the previous step however may not result in a feasible assignment of channels to radios at all nodes. Therefore, we propose a two-phase solution that ensures a feasible assignment of channels to radios while trying to maintain the throughputs arbitrarily close to the optimal output from the linear program. We formulate the first phase of CA as a broadcast link scheduling problem for a single channel.

- **Formulating CA as a two-sided multi-assignment problem for remaining channels**: Based on the channel assignment in the first phase, the second phase allocates transmit opportunities successively on each of the remaining channels to each node, taking into account the number of radios per node. More specifically, for

each remaining channel, we formulate the problem of assigning broadcast links to time slots, and vice versa, as a two-sided multi-assignment problem (TMP) where multiple broadcast links can share a single time slot and multiple time slots can be assigned to a single broadcast link. The objective of the TMP is to maximize the aggregate benefit of such assignment subject to interference constraints and the number of radios per node. We convert this TMP to a minimum cost flow problem (MCFP). The dual of this MCFP is solved by the auction algorithms based on which we develop our main channel assignment algorithm.

- **Simulation and experimental performance evaluation**: Finally, we evaluate our algorithm via simulations. We also evaluate our proposed system via experiments on a 10-node 802.11b/g testbed built on the iCORE implementation.

The rest of this chapter is organized as follows. We discuss the related work in Section 3.1. In Section 3.2 we describe the model and assumptions for every component in our system. We mathematically formulate the joint network coding, channel assignment and scheduling problem for throughput optimization in a multi-radio IWMN in Section 3.3. Based on the solution of the formulated optimization problem, we propose the two-step CA algorithm whose overview is given in Section 3.4. The development of the algorithm is detailed in Section 3.5, Section 3.6 and Section 3.7. We analyze this algorithm in Section 3.8. The evaluation of our proposed solution is presented in Section 3.9. Section 3.10 states the conclusion.

3.1 Related Work

Our work in this chapter is closely related to [129]- [130] in which throughput gains of both packet-level and signal-level inter-flow network coding in multi-radio ad hoc networks are mathematically characterized. The formulation in [130] however differs from ours in that lossy broadcast links were not explicitly modeled, and hence no opportunistic listening/overhearing was considered. In [131], the joint problem of CA, scheduling and wireless network coding was studied but the authors restricted their attention to only a one-hop information exchange, or Alice-Bob, scenario with simple binary coding (eXclusive OR). Opportunistic listening was not fully exploited. In [132], a heuristic for channel assignment was proposed in the inter-flow network coded multi-radio IWMNs, taking opportunistic listening into account. No optimization framework however was provided.

The capacity of the multi-radio IWMNs with respect to the number of radios and channels and number of nodes was charcterized in [133]. For the conventional routed multi-radio IWMNs, the CA problem has been studied extensively and jointly with routing, scheduling, congestion control, topology control, MAC, or rate selection [135,136]. In [109], the joint problem of CA, link scheduling and multipath routing was formulated as an LP for multiple (uplink) unicast connections. Unlike ours, the authors in [109] applied the novel flow transformation technique to the solution of the LP to ensure a feasible CA and interference-free scheduling from which the constant-factor approximation algorithm results. Channel assignment is updated in a less dynamic manner than ours. Like ours, fairness was explicitly taken into account in [109]. In [110], the authors proved that the CA problem in the multi-radio IWMNs is NP-hard and proceeded to propose a joint CA and multipath

routing heuristic, ensuring that each link's capacity is no less than its traffic load. No fairness was however considered. In [111], the joint routing, CA and scheduling problem was formulated as a multi-commodity flow problem: each commodity for each distinct source/sink pair. The resulting LP can incorporate a variety of linear objective functions related to traffic load and link capacity. Based on the conflict graph, both distributed and centralized algorithms were proposed for assigning channels to communication links in order to minimize network interference in [112]. In [113], the proposed CA algorithm, resulting from the integer LP formulation of joint CA and routing, aims to make a given set of flow rates on the IWMNs schedulable while maximizing the throughput at the potentially bottlenecked gateways. Other gateway-oriented solutions can be found in [114, 115].

The topology control and CA were jointly considered in [137] such that when channels are assigned to links, the maximum link conflict weights among all links is minimized. A similar approach was adopted in [116] by considering links between radios, instead of links between nodes as in [137]. Another topology control based CA can be found in [138]. The problem of joint CA, congestion control and scheduling was considered in [139] where queue lengths are used as a guide to arrive at a joint solution. Joint congestion control and CA was also studied in [140]. Channel assignment was also studied from the MAC perspective in [141]. The problem of joint CA and link rate selection was studied in [117]. Recently, the problem of joint scheduling, routing, power control and rate adaptation in a single-radio IWMN has been studied in [142] considering a physical interference model and spatial channel reuse.

In terms of implementation, a channel abstraction module for an operating system

kernel was presented in [143] in order to support interface switching in a multi-radio mesh network. This module and an example multichannel protocol using this module were implemented in a multichannel multi-interface testbed. Some experimental performance results for a conventional three-node multi-channel multi-interface (one, two, and three interfaces) wireless mesh network were presented in [144] considering TCP and UDP traffic.

In terms of execution of the solution, CA approaches can be classified into two main categories: centralized approaches [109–113, 116, 137] and distributed approaches [112, 114, 115, 118–122]. Under the centralized category, the formulation of the CA problem can be further classified into three main categories [136], namely, graph-based formulation [112,116, 137], network flow based formulation [109–111] and topology control based formulation [113]. Our work belongs to the centralized, network flow-based approach.

3.2 System Model and Assumptions

We describe various components and assumptions we use for modeling the multi-radio IWMNs in this section.

3.2.1 System Architecture and Operations

We consider an IWMN, shown in Fig. 3.1, consisting of a wired infrastructure (Internet, for example) and static wireless mesh routers. Each wireless mesh router with access point functionality serves as an ingress or egress for the aggregate traffic associated with the mobile/wireless clients in its coverage area. Such traffic is routed to and from the wired infrastructure via multiple wireless hops formed by the wireless mesh routers some of

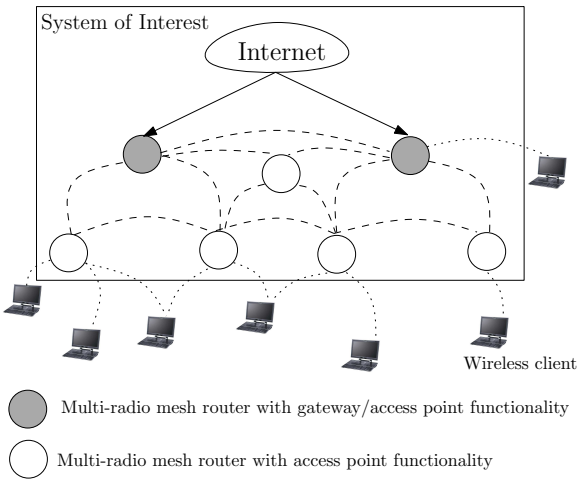


Figure 3.1: The system architecture of the multi-radio IWMN. We focus on the backbone portion of the IWMN in this chapter (boxed).

which also function as gateways to the wired infrastructure. Each wireless router may be equipped with multiple wireless interfaces (radios) each of which operates on an orthogonal channel. Each node in our system can algebraically combine incoming packets according to the random linear network coding (RNC) scheme before forwarding the resulting combined packets to other nodes via its broadcast link. We assume that wireless nodes are in promiscuous mode and that all wireless transmissions are in broadcast mode. Those wireless nodes that hear such transmissions may engage in packet forwarding. It is also assumed that our system operates synchronously in a time-slotted mode.

We model the given multi-radio IWMN with a directed hypergraph $\mathbb{H} = (\mathbb{N}, \mathbb{A})$, where $\mathbb{N}$ is the set of nodes, and $\mathbb{A}$ is the set of hyperarcs. A hypergraph is a generalization of a graph, where each hyperarc connects a starting node with any number of ending nodes. That is, a hyperarc is a pair (i, J), where the starting node i is an element of $\mathbb{N}$, and J, the set of nodes, is a non-empty subset of $\mathbb{N}$. In the special case when the set J has only one

element, we simply refer to a pair (i, J) as an "arc" (instead of "hyperarc") which we write (i, j) instead of $(i, \{j\})$ for $j \in J$.

Except for source node s representing the wired infrastructure, the nodes of the hypergraph $\mathbb{H}$ correspond to individual wireless mesh routers. Each hyperarc (i, J) then represents a wireless broadcast link from starting node i to ending nodes in the non-empty set J.[4] A link between node s and each of the wireless mesh routers with gateway functionality is actually an arc representing a wired link which we assume has an unlimited capacity. For convenience, we define $\mathbb{H}' = (\mathbb{N}', \mathbb{A}')$ as the IWMN in which source node s and those arcs connecting node s and the wireless mesh gateways are removed from the original IWMN $\mathbb{H}$. More precisely, $\mathbb{N}'$ is the set consisting of nodes in the non-empty subset $\mathbb{N} \setminus \{s\}$, and $\mathbb{A}'$ is the set consisting of wireless broadcast links only, i.e., hyperarcs (i, J) where i is an element of $\mathbb{N}'$ and J is the non-empty subset of $\mathbb{N}'$. We denote the set of wireless mesh gateways by $\mathbb{N}^{gw}$.

We denote by $R(i)$ the number of radios that node $i \in \mathbb{N}'$ has. Each radio operates on an orthogonal channel chosen from the set $\mathbb{F}$ containing M orthogonal channels. We represent such channels by numbers from 1 to M. For every node $i \in \mathbb{N}'$, we denote by $F(i)$ the ordered set of $R(i)$ orthogonal channels assigned to node i so that the m-th radio of node i operates on the m-th channel in the set $F(i) \subset \mathbb{F}$. For each broadcast link (i, J), we denote by $\hat{F}_{iJ}$ the set of channels that starting node i has in common with every ending node $j \in J$.

Given the IWMN $\mathbb{H} = (\mathbb{N}, \mathbb{A})$, the goal of the system is to establish T unicast connections from node s to the set of sink nodes $\mathbb{T} \subset \mathbb{N}'$. We denote a traffic demand of

[4] We use the terms hyperarc and broadcast link interchangeably throughout this chapter.

connection n by $r_n, n = 1, 2, ..., T$. Our system operates in a batch of K packets. Given the traffic demand r_n, the sink of connection n wishes to wait for a time K/r_n. For every batch, source node s first algebraically combines the K source packets for each connection (i.e., intra-session network coding is employed) and keeps forwarding the combined packets to the wireless mesh gateways. Each mesh gateway further combines the incoming packets with those already in its memory and forwards the resulting combined packet (with coding coefficients embedded in its header) over its broadcast link. Each nexthop wireless mesh router that hears this transmission then checks the header of the overheard packet to determine whether it is allowed to perform network coding and forwarding for this packet. If so, it combines this packet with those already in its memory and transmits. Packets for each connection therefore are routed through multiple paths. This process continues until the corresponding sink of a particular connection receives the sufficient number of independent combined packets at which point the decoding process starts. We assume that there is a mechanism to signal such a successful reception to node s and all other nodes to stop further forwarding [5]. All the available channels and radios at each node in the set of wireless routers, $\mathbb{N}'$, will be utilized to ensure that throughputs are maximized proportionally for all unicast connections, subject to several constraints which we will describe in detail in Section 3.3.

3.2.2 Interference Model

For two nodes to directly communicate, they need to use a common channel and stay within the communication range of one another. Even if they cannot directly commu-

[5]For details on how we implement this mechanism, the reader is referred to Chapter 4 (Section 4.5)

nicate, any two nodes may however interfere with each other's communication given that they use the same channel and stay within the interference range of each other. According to the Protocol model of interference, an interference range is greater than a communication range by a factor $\beta \geq 1$.

Let Range_I be the interference range and Range_C be the communication range so that $\mathsf{Range}_I = \beta \cdot \mathsf{Range}_C$. Also let the distance between node i and node j be $\mathsf{Dist}(i, j)$. We say that broadcast link (i, J) is in the set of broadcast links, $\mathbb{A}$, of the IWMN $\mathbb{H}$ if and only if $\mathsf{Dist}(i, j) \leq \mathsf{Range}_C$ for every node $j \in J$, and there is at least one common channel among the corresponding sets of channels, $F(i)$ and $F(j)$, i.e., $\hat{F}_{iJ} \neq \emptyset$.

Simultaneous communications are possible over broadcast link (i, J) if starting node i has the number of channels in common with node j for all $j \in J$: one communication on each channel. We say that two broadcast links (i, J) and (i', J') interfere if either one of the following conditions holds:

1. At least one ending node of broadcast link (i, J) is in the interference range of the starting node of broadcast link (i', J'), i.e., $\mathsf{Dist}(i', j) \leq \mathsf{Range}_I, \quad \exists j \in J$, or

2. at least one ending node of broadcast link (i', J') is in the interference range of the starting node of broadcast link (i, J), i.e., $\mathsf{Dist}(i, j') \leq \mathsf{Range}_I, \quad \exists j' \in J'$.

We model the interference arising from the above situations on a particular channel using a conflict graph $G = (V, E)$ where V is the set of vertices and E the set of edges. Each vertex $v \in V$ then corresponds to a broadcast link $(i, J) \in \mathbb{A}'$. An edge (u, v) is an element of E if the broadcast links corresponding to nodes u and v interfere with each other for all $u, v \in V$.

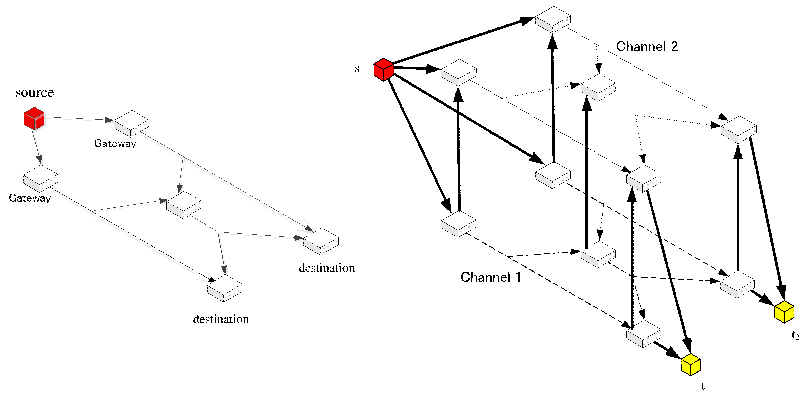


Figure 3.2: Example of a multi-layered hypergraph for $M = 2$ channels: (Left) The original IWMN $\mathbb{H}$, (Right) Flow network for 2 channels (multi-layered hypergraph).

3.2.3 A Flow Network for A Multi-channel Multi-radio IWMN

One element we need in the formulation of the problem is a flow network which represents all possible routes that traffic can flow, without exceeding link capacities, in a multi-radio IWMN when utilizing all the orthogonal channels. We explain in this section how the flow network for the IWMN $\mathbb{H} = (\mathbb{N}, \mathbb{A})$ is constructed. One can visualize a flow network as a three-dimensional multi-layered hypergraph[6] in which each layer represents a copy, except for source node s, of the IWMN $\mathbb{H}$ on each channel. We show for example in Fig. 3.2 how the original IWMN (left) can be transformed into a multi-layered hypergraph (right) when two channels are available.

The steps we use to construct the multi-layered hypergraph are as follows.

1. **Create horizontal layers:** Except for source node s representing the wired infrastructure, we create, for each channel $m, m = 1, 2, ..., M$, a copy of the original IWMN $\mathbb{H}$ and align them vertically in layers. Denote a hypergraph on layer m by $\mathbb{H}'(m)$

[6]We use the terms *multi-layered hypergraph* and *flow network* interchangeably throughout this chapter.

and the corresponding sets of nodes (including the sink nodes) and broadcast links by $\mathbb{N}'(m)$ and $\mathbb{A}'(m)$, respectively. Similarly, we denote the set of wireless mesh gateways on each layer m by $\mathbb{N}^{gw}(m)$ which is the non-empty subset of $\mathbb{N}'(m)$. For every node $i \in \mathbb{N}'$, we denote a copy of node i on layer m by $i(m)$, and a copy of broadcast link $(i, J) \in \mathbb{A}'$ on layer m by $(i(m), J(m))$ which is an element of $\mathbb{A}'(m)$. That is,

$$\mathbb{A}'(m) = \{(i(m), J(m)) \mid i(m) \in \mathbb{N}', J(m) \subset \mathbb{N}'\}$$

for $m = 1, 2, ..., M$.

2. **Connect the source to gateways on each layer:** We next connect source node s to every node $i(m) \in \mathbb{N}^{gw}(m), m = 1, 2, ..., M$. We define the set of the resulting directed arcs as

$$\mathbb{A}_s = \{(s, i(m)) \mid i(m) \in \mathbb{N}^{gw}(m), m = 1, 2, ..., M\}.$$

We assume that each such arc has an infinite capacity.

3. **Create dummy sinks and connect them to all copies of the actual sinks:** For each node $i \in \mathbb{T}$ as a sink for connection n, we introduce a dummy sink node t_n which is connected to all copies of the actual sink node i via virtual directed arcs $(i(1), t_n), ..., (i(M), t_n)$ (See Fig. 3.2 for example). We assume that each such arc has an infinite capacity so that packets received on any channel (copy) of the actual sink node $i \in \mathbb{T}$ are transferred simultaneously to its corresponding dummy sink. Denote the set of all such arcs for all $i \in \mathbb{T}$ by $\mathbb{A}_t$ where

$$\mathbb{A}_t = \{(i(m), t_n) \mid i \in \mathbb{T}, m = 1, 2, ..., M \text{ and } n = 1, 2, ..., T\}.$$

Finally, denote the set of dummy sinks by

$$\mathbb{T}' = \{t_n \mid n = 1, 2, ..., T\}.$$

4. **Connect layers to one another:** All copies of node $i \in \mathbb{N}'$ on different layers are connected via the set of $M(M-1)/2$ arcs where each arc has an infinite capacity. These arcs will enable packets to move from one channel (layer) to another within a node $i \in \mathbb{N}'$. Define the set of these arcs as

$$\mathbb{A}_i = \{(i(m), i(m')) \mid i \in \mathbb{N}', 1 \leq m, m' \leq M, m \neq m'\}.$$

5. **Define capacities of wireless broadcast links on each channel:** We assume that the capacity of broadcast link $(i(m), J(m)) \in \mathbb{A}'(m)$ on channel m is the minimum of the capacities of its component point-to-point links, which is given by

$$c_{iJ}(m) = \left(\prod_{j \in J} [1 - \mathsf{Loss}_{iJj}(m)] \right) \min_{j \in J} \mathsf{Rate}_{iJj}(m)$$

where $\mathsf{Rate}_{iJj}(m)$ is the individual transmission rate from the actual node $i \in \mathbb{N}'$ to node $j \in J, J \subset \mathbb{N}'$, on channel m, and $\mathsf{Loss}_{iJj}(m)$ is the packet loss probability on the corresponding link.

For convenience, we simply write c_{iJ} without a channel index m to represent a capacity on those arcs that do not involve actual transmissions on physical wireless channels, i.e., every arc

$$(i, J) \in \mathbb{A}_s \bigcup \mathbb{A}_t \bigcup_{i \in \mathbb{N}'} \mathbb{A}_i.$$

The resulting multi-layered hypergraph is denoted by $H = (N, A)$ where

$$N = \{s\} \bigcup \mathbb{T}' \bigcup_{m=1}^{M} \mathbb{N}'(m) \quad \text{and} \quad A = \bigcup_{m=1}^{M} \mathbb{A}'(m) \bigcup \mathbb{A}_s \bigcup \mathbb{A}_t \bigcup_{i \in \mathbb{N}'} \mathbb{A}_i.$$

In general, we represent by $f_{iJj}^{(n)}$ the flow rate of connection n routed from node $i \in N$ to node $j \in J$ where $(i, J) \in A$ and $J \subset N$. However, we specifically denote by $f_{iJj}^{(n)}(m)$ the flow rate of connection n routed from node $i \in N$ to node $j \in J, J \subset \mathbb{N}'(m)$, when packets are actually transmitted on physical wireless broadcast link $(i, J) \in \mathbb{A}'(m)$, using channel m. The vector consisting of $f_{iJj}^{(n)}$ thus also includes $f_{iJj}^{(n)}(m)$. We denote by $f^{(n)}$ the flow vector consisting of $f_{iJj}^{(n)}$.

3.2.4 Distributed Random Network Coding

Every packet in our system is a linear combination of a given batch of source packets. More specifically, every packet that is transmitted on every link $(i, J) \in \bigcup_{m=1}^{M} \mathbb{A}'(m) \bigcup \mathbb{A}_s$ is coded using a distributed random network coding scheme (RNC). Note that the set $\mathbb{A}'(m)$ contains those broadcast links that lie virtually and horizontally on each layer (channel) of the multi-layered hypergraph H, and the set $\mathbb{A}_s$ contains those wired point-to-point links connecting source s to all copies of the wireless mesh gateways on all the layers of H. We describe in this section how packets are coded using RNC as they traverse these links.

Coding Operation

Given the multi-layered hypergraph H and the traffic demand r_n packets per unit time for connection n, $n = 1, 2, ..., T$, we view each demand as a unicast connection from node s to a dummy sink $t_n \in \mathbb{T}'$. Denote a unicast connection by a tuple (s, t_n, r_n), where s is the source corresponding to the wired infrastructure, and t_n is a dummy sink with traffic demand r_n packets per unit time. There are at most T connections that must be supported: $(s, t_1, r_1), ..., (s, t_n, r_n), ..., (s, t_T, r_T)$. For each connection, a batch of K packets,

$\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_K$, will be delivered. We assume that these K packets are stored in the memory of a central coding module attached to node s. All packets are assumed to have equal length of $\mathsf{L}_{\mathsf{bit}}$ bits. We represent each packet in our system by a vector of length $\mathsf{L}_{\mathsf{sym}}$ consisting of symbols drawn from the finite field $\mathbf{GF}(q)$ such that $\mathsf{L}_{\mathsf{bit}} = \mathsf{L}_{\mathsf{sym}} \cdot \log_2 q$.

Every node codes packets in the same manner. Received packets are stored in the node's memory. Whenever a transmission opportunity arises, all the packets in the memory are linearly combined, preparing for injection into an outgoing broadcast link attached to that node. The coefficients used for such combination are drawn uniformly at random from $\mathbf{GF}(q)$. Any coded packet in the network can always be expressed as a linear combination of the source packets $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_K$. To see this, consider a unicast connection on a tandem network of nodes formed on the multi-layered hypergraph H, as shown in Fig. 3.3. Assume that after a specific period the gateway gw has received from source s a total of K' packets $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{K'}$. Each of these packets can be expressed as

$$\mathbf{v}_j = \sum_{l=1}^{K} \mathsf{Coeff}_{jl} \cdot \mathbf{p}_l$$

where Coeff_{jl} is a coding coefficient drawn uniformly at random from the finite field $\mathbf{GF}(q)$ by node s for source packet $\mathbf{p}_l$ in order to form the jth coded packet $\mathbf{v}_j$. Any coded packet $\mathbf{u}$ forwarded by the gateway gw to the sink t is therefore a linear combination of packets $\mathbf{v}_j, j = 1, 2, ..., K'$, i.e.,

$$\mathbf{u} = \sum_{j=1}^{K'} \mathsf{Coeff}'_j \mathbf{v}_j = \sum_{l=1}^{K} \left(\sum_{j=1}^{K'} \mathsf{Coeff}_{jl} \cdot \mathsf{Coeff}'_j \right) \mathbf{p}_l$$

where Coeff'_j is a random coding coefficient drawn from $\mathbf{GF}(q)$ by the gateway gw for the jth coded packet $\mathbf{v}_j$. This linear combination yields a global encoding vector $\overrightarrow{\mathsf{Coeff}^{\mathsf{g}}} =$

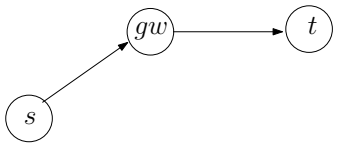


Figure 3.3: Subset of nodes on H forming a tandem network from source s to gateway gw and finally to a sink t.

$[\mathsf{Coeff}_1^{\mathsf{g}}, \mathsf{Coeff}_2^{\mathsf{g}}, ..., \mathsf{Coeff}_K^{\mathsf{g}}]$ the lth component of which is

$$\mathsf{Coeff}_l^{\mathsf{g}} = \left(\sum_{j=1}^{K'} \mathsf{Coeff}_{jl} \cdot \mathsf{Coeff}'_j \right).$$

A global encoding vector is the side information that must be appended to the header of every coded packet in order for the sink to decode. Define an innovative packet as a coded packet whose global coding vector is linearly independent of those global coding vectors corresponding to the packets already stored in the memory. Such innovative packets always contain new information and must be forwarded. We assume that every node in our system has a mechanism to check whether an incoming packet is innovative. Such linear independence checking ensures that only innovative packets are stored in the memory.

The role of any sink node $t_n \in \mathbb{T}$ is to keep collecting coded packets on its incoming links until K innovative coded packets have been received. In particular, each sink performs Gaussian Elimination on the set of global coding vectors corresponding to the received coded packets in its memory. If an inverse exists, it applies the inverse to the received packets to retrieve source packets $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_K$. Otherwise, a decoding error is said to occur.

Coding Subgraph

With distributed random network coding, each node only needs to know *how many*, and not *which*, packets it should forward or inject into its hyperarcs/broadcast links. In the network coding literature, the term *coding subgraph* is used to specify the frequencies and locations of such packet injection. Let us describe a coding subgraph on the multi-layered hypergraph H.

We associate with each broadcast link $(i, J) \in \bigcup_{m=1}^{M} \mathbb{A}'(m)$ a value $z_{iJ}(m)$ which determines how many packets node $i \in \mathbb{N}'(m)$ should forward on each channel m. In other words, it is the average rate at which packets are injected into broadcast link (i, J) on channel m, i.e.,

$$z_{iJ}(m) = \sum_{K \subset J} z_{iJK}(m), \tag{3.1}$$

where $z_{iJK}(m)$ is an average rate of the counting process b^m_{iJK} describing the arrival of packets injected on broadcast link (i, J) and received by exactly the set of nodes $K \subset J$ on channel m. That is, for time $\tau > 0$, the counting process $b^m_{iJK}(\tau)$ is the total number of packets that are injected on broadcast link (i, J) and received by all nodes in the set K between time 0 and time τ on channel m. As time τ approaches infinity, we thus have

$$\lim_{\tau \to \infty} \frac{b^m_{iJK}(\tau)}{\tau} = z_{iJK}(m).$$

We denote a coding subgraph on channel m by $z(m)$, consisting of all injection rates $z_{iJ}(m), (i, J) \in \mathbb{A}'(m), m = 1, 2, ..., M$.

3.3 Problem Formulation

Given the multi-layered hypergraph $H = (N, A)$ and unicast connections (s, t_n, r_n), $n = 1, 2, ..., T$, the problem is to find the coding subgraphs $z(m), m = 1, 2, ..., M$, such that the traffic demand for all connections are maximized. In general, the demands r_n may not be achievable for all connections. We thus aim to support only the proportional demand $\lambda \cdot r_n$ where λ is a scaling factor we want to maximize. In other words, we consider the problem of maximizing λ such that a $\lambda \cdot r_n$ portion of traffic can be delivered, via RNC, for each dummy sink $t_n \in \mathbb{T}'$. The fraction of demand that can be supported is thus the same for all dummy sinks $t_n \in \mathbb{T}'$. We now describe the constraints that the above optimization problem must satisfy as follows.

3.3.1 Flow and Coding constraints

Suppose we want to establish a connection (s, t_n, r_n) with rate arbitrarily close to $\lambda \cdot r_n$, i.e., to recover a batch of K source packets, sink t_n is willing to wait for a time τ which is only marginally greater than $K/(\lambda \cdot r_n)$. Suppose further that with RNC running over this period τ the connection (s, t_n, r_n) can be supported, i.e.,

$$\lambda \cdot r_n \leq \min_{\psi \in \Psi(s,t_n)} \left\{ \sum_{m=1}^{M} \sum_{(i,J) \in \digamma(\psi)} \sum_{K \not\subseteq \psi} z_{iJK}(m) \right\} \tag{3.2}$$

where $\Psi(s, t_n)$ is the set of all cuts between source s and sink t_n, and $\digamma(\psi)$, defined in (3.3), is the set of forward hyperarcs of the cut $\psi \in \Psi(s, t_n)$.

$$\digamma(\psi) := \{(i, J) \in A \mid i \in \psi \text{ and } J - \psi \neq \emptyset\} \tag{3.3}$$

In other words, given the coding subgraphs $z(m)$, specifying the frequencies and locations of coded packet injections on the multi-layered hypergraph H, this connection (s, t_n, r_n) is not bottlenecked if only a $\lambda \cdot r_n$ portion of traffic is required. Therefore, according to the max-flow min-cut theorem, there exists for each connection n a flow vector $f^{(n)}$ satisfying[7]

$$\sum_{\{J|(i,J)\in A\}} \sum_{j\in J} f_{iJj}^{(n)} - \sum_{\{j|(j,I)\in A, i\in I\}} f_{jIi}^{(n)} = \begin{cases} \lambda \cdot r_n, & \text{if } i = s \\ -\lambda \cdot r_n, & \text{if } i = t_n \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

for all nodes $i \in N$ and $n = 1, 2, ..., T$, and

$$\sum_{j\in K} f_{iJj}^{(n)}(m) \leq \sum_{\{L\subset J|L\cap K\neq\emptyset\}} y_{iJL}^{(n)}(m) \tag{3.5}$$

for all broadcast links $(i, J) \in \mathbb{A}'(m), K \subset J, n = 1, 2, ..., T$, and $m = 1, 2, ..., M$. Variable $y_{iJK}^{(n)}(m)$ represents the average rate of packets in connection n, that are injected on broadcast link (i, J) and received by exactly the set of nodes $K \subset J$ on channel m. Such receptions occur with average $z_{iJK}(m)$. On aggregate, the packet injection rates $y_{iJK}^{(n)}(m)$ for all connections cannot exceed $z_{iJK}(m)$. That is,

$$\sum_{n=1}^{T} y_{iJK}^{(n)}(m) \leq z_{iJK}(m), \tag{3.6}$$

for all broadcast links $(i, J) \in \mathbb{A}'(m), K \subset J, m = 1, 2, ..., M$. The following non-negativity constraints must also hold:

$$f_{iJj}^{(n)} \geq 0, \tag{3.7}$$

[7]Note that the notation (j, I) simply refers to any hyperarc/broadcast link other than (i, J). Otherwise specified, (i, J) denotes a hyperarc or a broadcast link.

for all links $(i, J) \in A$ and $n = 1, 2, ..., T$. For flows transmitted on each channel m, in particular, we also have the broadcast link capacity constraints as

$$0 \leq \sum_{n=1}^{T} \sum_{j \in K} f_{iJj}^{(n)}(m) \leq c_{iJ}(m), \tag{3.8}$$

for all broadcast links $(i, J) \in \mathbb{A}'(m), K \subset J$ and $m = 1, 2, ..., M$.

3.3.2 Interference-free Broadcast Link Scheduling

Recall that we model interference between broadcast links on each channel m as a conflict graph $G = (V, E)$ where a node v corresponds to a broadcast link in the set $\mathbb{A}'(m)$, and an edge (u, v) belongs to the edge set E if u and v interfere, for all $u, v \in V$.

Consider a clique $Q \subset V$ on a particular channel m. All nodes $v \in Q$ then correspond to broadcast links $(i, J) \in \mathbb{A}'(m)$ that interfere with each other. We write v_{iJ} to denote this correspondence. For interference-free broadcast link scheduling, it is required that at most one broadcast link (i, J), corresponding to a node in Q, be scheduled for transmission at any given time on a particular channel. We thus have

$$\sum_{v_{iJ} \in Q} \frac{z_{iJ}(m)}{c_{iJ}(m)} \leq 1, \tag{3.9}$$

for all cliques $Q \in G$ and $m = 1, 2, ..., M$.

The above constraints state that the sum of the fractional link utilization of all broadcast links $(i, J) \in \mathbb{A}'(m)$ that interfere with one another on channel m must not exceed its capacity $c_{iJ}(m)$. We denote each such fractional broadcast link utilization by

$$\mu_{iJ}(m) = \frac{z_{iJ}(m)}{c_{iJ}(m)}. \tag{3.10}$$

3.3.3 Node-Radio Constraints

Consider the original hypergraph $\mathbb{H} = (\mathbb{N}, \mathbb{A})$. Recall that each wireless node $i \in \mathbb{N}'$ has exactly $R(i)$ radios each of which will be assigned an orthogonal channel m. Define $W_{iJ}^{(m,k)}$, for all $(i, J) \in \mathbb{A}'(m), m \in \hat{F}_{iJ}$, as an indicator variable that is 1 if broadcast link (i, J) is active on channel m at time slot k, and 0 otherwise. Since each wireless node $i \in \mathbb{N}'$ can participate in at most $R(i)$ simultaneous communications, any valid link scheduling in any given time slot is free of interference if, for all $i \in \mathbb{N}'$,

$$\sum_{m=1}^{M} \sum_{(i,J)\in\mathbb{A}'(m)} W_{iJ}^{(m,k)} + \sum_{m=1}^{M} \sum_{\{(j,I)\in\mathbb{A}'(m)|i\in I\}} W_{jI}^{(m,k)} \leq R(i). \tag{3.11}$$

While the first term on the LHS of (3.11) indicates the number of simultaneous transmissions node i is participating, the second term indicates the number of its simultaneous receptions.

Given a link schedule of duration τ, summing (3.11) over time τ and then dividing by τ results in

$$\sum_{m=1}^{M} \sum_{(i,J)\in\mathbb{A}'(m)} \mu_{iJ}(m) + \sum_{m=1}^{M} \sum_{\{(j,I)\in\mathbb{A}'(m)|i\in I\}} \mu_{jI}(m) \leq R(i) \tag{3.12}$$

where $\mu_{iJ}(m)$, defined in (3.10), denotes the fractional link utilization for broadcast link (i, J) on channel m. We thus have the following constraints:

$$\sum_{m=1}^{M} \sum_{(i,J)\in\mathbb{A}'(m)} \frac{z_{iJ}(m)}{c_{iJ}(m)} + \sum_{m=1}^{M} \sum_{\{(j,I)\in\mathbb{A}'(m)|i\in I\}} \frac{z_{jI}(m)}{c_{jI}(m)} \leq R(i), \tag{3.13}$$

for all $i \in \mathbb{N}'$.

3.3.4 The Joint Channel Assignment and Network Coding Problem

Putting all the constraints together, we have the following linear program (LP1):

$$\begin{aligned}
&\text{maximize} \quad \lambda \\
&\text{subject to} \quad \sum_{\{J|(i,J)\in A\}} \sum_{j\in J} f_{iJj}^{(n)} - \sum_{\{j|(j,I)\in A, i\in I\}} f_{jIi}^{(n)} = \begin{cases} \lambda d_n, & \text{if } i = s \\ -\lambda d_n, & \text{if } i = t_n, \\ 0, & \text{otherwise} \end{cases} \\
&\qquad \forall i \in N, n = 1, 2..., T \\
&\sum_{n=1}^{T} y_{iJK}^{(n)}(m) \leq z_{iJK}(m), \quad \forall (i,J) \in \mathbb{A}'(m), K \subset J, m = 1, 2, ..., M \\
&\sum_{j\in K} f_{iJj}^{(n)}(m) \leq \sum_{\{L\subset J | L\cap K \neq \emptyset\}} y_{iJL}^{(n)}(m), \\
&\qquad \forall (i,J) \in \mathbb{A}'(m), K \subset J, n = 1, 2, ..., T, m = 1, 2, ..., M \\
&0 \leq \sum_{n=1}^{T} \sum_{j\in K} f_{iJj}^{(n)}(m) \leq c_{iJ}(m), \quad \forall (i,J) \in \mathbb{A}'(m), K \subset J, m = 1, 2, ..., M \\
&f_{iJj}^{(n)} \geq 0, \quad \forall (i,J) \in A, n = 1, 2..., T \\
&\sum_{v_{iJ}\in Q} \frac{z_{iJ}(m)}{c_{iJ}(m)} \leq 1, \quad \forall Q \in G, m = 1, 2, ..., M \\
&\sum_{m=1}^{M} \sum_{(i,J)\in\mathbb{A}'(m)} \frac{z_{iJ}(m)}{c_{iJ}(m)} + \sum_{m=1}^{M} \sum_{\{(j,I)\in\mathbb{A}'(m)|i\in I\}} \frac{z_{jI}(m)}{c_{jI}(m)} \leq R(i), \quad \forall i \in \mathbb{N}'.
\end{aligned} \tag{3.14}$$

The size of LP1 (i.e., the number of variables) depends on the number of individual point-to-point links of the multi-layered hypergraph H as well as on the extent to which the broadcast nature of the wireless medium is exploited. More specifically, we can express the size of LP1 in terms of the number of channels, M, the number of gateways, $|\mathbb{N}^{gw}|$, and

the number of connections, T, as

$$\mathsf{LP}_{\mathsf{size}} = M \times T \left(\sum_{(i,J)\in\bigcup_{m=1}^{M}\mathbb{A}'(m)} |J|(1+\frac{M-1}{2}) + |\mathbb{N}^{gw}| + T \right) + \ldots \left(\sum_{(i,J)\in\bigcup_{m=1}^{M}\mathbb{A}'(m)} 2^{|J|}M(T+1) \right) \tag{3.15}$$

where the first term on the RHS represents the number of individual point-to-point links of the multi-layered hypergraph H, and the second term corresponds to the number of (coded) packet injection rates $y_{iJK}^{(n)}(m)$ and $z_{iJK}(m)$. As mentioned earlier, the extent to which the broadcast nature of the wireless medium is exploited through opportunistic listening is governed by the number of ending nodes, $|J|$, on every broadcast link $(i,J) \in \bigcup_{m=1}^{M}\mathbb{A}'(m)$. The higher extent of such exploitation however causes the size of LP1 to grow exponentially. The worst cast scenario is that when $|J|$ is equal to $|\mathbb{N}|-2$, i.e., when a transmission is overheard by all nodes except the wired source node s and the transmitting node itself. Then, all possible subsets K of set J will have to be considered, resulting in the increasing number of variables $y_{iJK}^{(n)}(m)$ and $z_{iJK}(m)$ in LP1.

It is worth noting that LP1 is, in fact, general enough to encompass the conventional multi-path routing case (no network coding) but with only a slight modification of LP1's formulation. More precisely, without network coding, the coding constraints (3.5) and (3.6) will be discarded, and coding variables $z_{iJ}(m)$ in constraints (3.9) and (3.13) will be replaced by the term $\sum_{n=1}^{T}\sum_{j\in K} f_{iJj}^{(n)}$ for $K \subset J$. It can be verified that the size of the modified LP in this case is reduced to

$$\mathsf{LP}'_{\mathsf{size}} = \left(\sum_{(i,J)\in\bigcup_{m=1}^{M}\mathbb{A}'(m)} 2^{|J|}M(T+1) \right)$$

which is obviously smaller than that of the coding case in (3.15). This is the price we pay in

using network coding, but we will show in our evaluation that the increased computational complexity introduced by network coding can be traded off with the gain in throughput. In general, if LP1 is solved by the primal-dual interior point algorithm, the complexity is known to have a $O(\mathsf{LP}_{\mathsf{bits}} \cdot \mathsf{LP}_{\mathsf{size}})$ running time where $\mathsf{LP}_{\mathsf{bits}}$ is the number of bits used to store $\mathsf{LP}_{\mathsf{size}}$ variables [124].

The optimal solution to LP1 is a flow vector f and coding subgraphs $z(m), m = 1, 2, ..., M$, that maximize λ on the multi-layered hypergraph H. From the practical point of view, however, only the best possible solution, now called "upperbound", can be obtained, and we denote the traffic scaling factor corresponding to this solution by λ^*. Due to the node-radio constraints in (3.13) – which are the relaxed version of the binary constraints in (3.11) – the coding subgraphs $z(m), m = 1, 2, ..., M$, resulting from the upperbound solution do not guarantee a feasible mapping of radios to channels at all nodes. In other words, by modeling the number of radios on each node using the broadcast link utilization, the exact binary constraints in (3.11) become loose and may render channel assignment infeasible. In fact, a node with, say, 2 radios may be involved in 3 or more simultaneous communications while still satisfying the relaxed node-radio constraints in (3.13). We propose a two-step approach to refine such channel assignment in the following sections.

3.4 Refining Channel Assignment: Overview

We give an overview of a refinement of the possibly infeasible channel assignment output by LP1 in this section. The main objective of such refinement is to derive a feasible mapping of radios to channels for all nodes involved in packet injection of the broadcast

links[8] of the multi-layered hypergraph H. Such refinement must also ensure that, after the refinement, the new traffic scaling factor λ_n for all connections $t_n \in \mathbb{T}'$ deviates from the best possible λ^* as slightly as possible. The final solution is therefore sub-optimal to the upperbound solution output by LP1. We propose a two-step method to achieve this goal.

Our proposed method is executed in two phases. In Phase 1 we deal with assigning hyperarcs (i, J) with packet injection rate $z_{iJ}(1)$ to time slots on channel 1 (or equivalently on any given channel) such that no interference incurs (interference-free broadcast link scheduling). How many time slots should be assigned to a particular hyperarc depends strongly on the packet injection rate $z_{iJ}(1)$ output by LP1 and the corresponding broadcast link capacities.

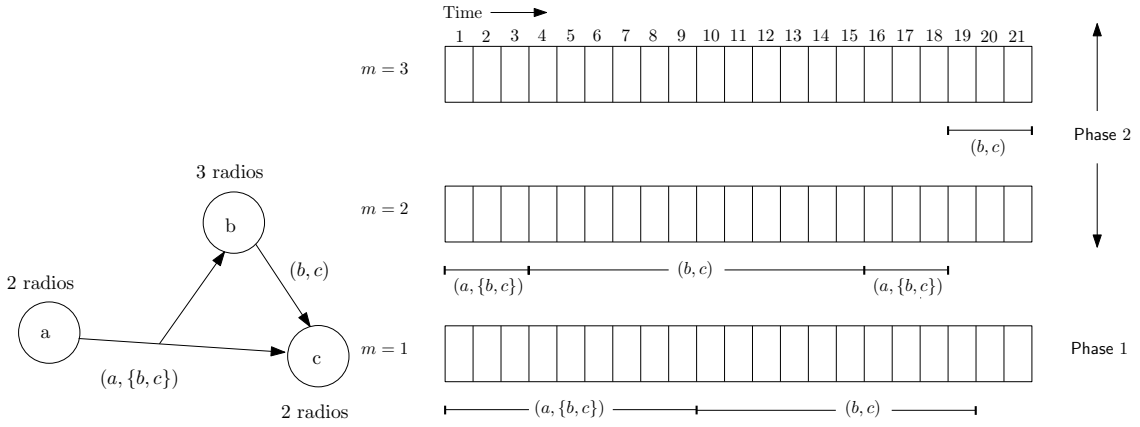
Using the output from Phase 1 as a reference and taking into account the number of radios at each node, Phase 2 next will attempt to assign hyperarcs $(i(m), J(m))$ with non-zero packet injection rates to time slots successively on the remaining channels $m = 2, ..., M$, by way of auction. In particular, we formulate the channel assignment problem in Phase 2 as a two-sided multi-assignment problem (TMP) [125] in which we try to assign hyperarcs to time slots, and vice versa, in order to maximize the benefit in terms of total packet injection rates. It turns out this TMP can be converted to a minimum cost flow problem (MCFP) whose dual can be solved by the auction-based algorithms [125]. To be assigned to certain time slots (interpreted as "objects"), a given hyperarc (interpreted as "person") has to bid for them and win the auction. On the other hand, to be assigned back to certain hyperarcs (now interpreted as "objects"), a given time slot ("person") has to submit the highest bid. Therefore, a single slot may be assigned to multiple hyperarcs if those hyperarcs do not

[8]For convenience, we now refer to *broadcast links* as *hyperarcs* throughout the rest of this chapter.

interfere with one another. Similarly, a single hyperarc can be assigned also to multiple non-contiguous time slots if the member nodes of that hyperarc have enough idle radios to support simultaneous communications in those time slots. As Phase 2 progresses, it will use the output from a previous stage (i.e., the mapping of hyperarcs to time slots on channel $m-1$) to guide its operations in a current stage for channel m.

We show the big picture of how the final solution to our two-step approach may look like in Fig. 3.4. Consider the network shown in Fig. 3.4(a) where only two hyperarcs exist. Node a is to deliver unicast packets to node c with the help of relay b. The number of radios per node are as shown in the figure. Assume that 3 orthogonal channels are available in the network. Time is divide into equally-spaced slots on all channels. Based on the coding subgraphs output by LP1, hyperarc $(a, \{b, c\})$ will need at least 9 time slots to inject its packets, and arc (b, c) will need 10. Phase 1 attempts to allocate the first available time slots 1-9 to hyperarc $(a, \{b, c\})$ and time slots 10-19 to arc (b, c) on channel 1. One radio is now tuned to channel 1 at all the nodes. On channel 2, both $(a, \{b, c\})$ and (b, c) will competitively bid for time slots in Phase 2. One possible assignment for channel 2 is as shown in the figure. All nodes now have two simultaneous communications, utilizing 2 radios per node. In channel 3, however, simultaneous communications are not allowed in the first 18 time slots, because nodes a and b do not have enough radios to do so. Both links then compete for the remaining time slots in which arc (b, c) wins the bid.

In summary, taking as input the vector of non-zero packet injection rates $z_{iJ}(m)$ on hyperarcs $(i, J) \in \mathbb{A}'(m)$ for all channels $m = 1, 2, ..., M$, our two-step channel assignment algorithm attempts to allocate time slots to the hyperarcs, and vice versa, such that



(a) Network with 2 hyperarcs. (b) Assignment of hyperarcs to time slots (3 channels available.)

Figure 3.4: Example of how our channel assignment in both phases works.

1. on each channel the packet injection rates $z_{iJ}(m)$ are satisfied for all $(i, J) \in \mathbb{A}'(m)$,

2. across all channels $m = 1, 2, ..., M$, the number of channels assigned, in a particular time slot, to each member node of a particular hyperarc never exceeds the number of its radios, and

3. for all channels and all time slots the aggregate benefit of such allocation is maximized, where we will model the benefit in terms of coded packet injection rates.

For the sake of algorithm development, we will denote by $z^l_{iJ}(m) \in z(m)$ the non-zero average packet injection rate on hyperarc $(i, J)_l, l = 1, 2, ..., L$, on channel m, where we order the hyperarcs with index l in a downstream direction, i.e., from those hyperarcs connected to source s to those hyperarcs connected to sinks $t_n \in \mathbb{T}'$. Also, denote the corresponding capacities of such hyperarcs by $c^l_{iJ}(m)$. Given a scheduling period τ, it follows that the number of time slots required to satisfy the packet injection rate for each

hyperarc $(i, J)_l$ is

$$\sum_{1 \le k \le \tau} W_{iJ}^{(m,k)} = \lceil \tau \frac{z_{iJ}^l(m)}{c_{iJ}^l(m)} \rceil$$

where $W_{iJ}^{(m,k)}$ is an indicator variable which is 1 if hyperarc (i, J) is active on channel m in time slot k, and 0 otherwise.

3.5 Refining Channel Assignment: Phase 1

Given the coding subgraph $z(1)$ and the corresponding conflict graph $G = (V, E)$, we want to find an interference-free assignment of hyperarcs to time slots, denoted by Φ_1, over a scheduling period of τ time slots. We let the value of τ be

$$\tau = \min_{n=1,...,T} K/(\lambda^* r_n) \tag{3.16}$$

where $K/(\lambda^* r_n)$ is a time each sink $t_n \in \mathbb{T}'$ is willing to wait to recover the K source packets at rate $\lambda^* r_n$. This choice of τ will favor all the connections.

We show the pseudo code for finding the interference-free assignment of hyperarcs to time slots on channel $m = 1$ in Algorithm 3.1. In the algorithm, each hyperarc will be allocated the first available $\lceil \tau \frac{z_{iJ}^l(1)}{c_{iJ}^l(1)} \rceil$ time slots such that the assignment is interference-free. Two hyperarcs that interfere must be assigned to distinct time slots. We denote by $S\{(i, J)_l\}$ the set of time slots in which hyperarc $(i, J)_l$ is scheduled in the assignment Φ_1. From the practical point of view, we assume that the duration of each time slot is large enough so that channel switching delay is negligible.

The output of Algorithm 3.1 will guide the assignment for the remaining channels, $m = 2, ..., M$, in Phase 2. We describe Phase 2 in the following section.

Algorithm 3.1: Phase 1: Interference-free assignment of hyperarcs to time slots for 1st channel.

input : $z_{iJ}(1), G = (V, E)$

output: Φ_1: Interference-free assignment of hyperarcs to time slots for channel $m = 1$

Initialization:

Set schedule $S\{(i, J)_l\} = \emptyset, \forall (i, J)_l, l = 1, ..., L$

Set $m = 1$

availableSlots $= \{1, 2, ..., \tau\}$

for $l = 1, ..., L$ **do**

  Set $u = (i, J)_l \in V$

  Set $S\{(i, J)_l\}$ to first available $\lceil \tau \frac{z^l_{iJ}(1)}{c^l_{iJ}(1)} \rceil$ time slots in availableSlots such that

$$S\{(i, J)_l\} \bigcap \bigcup_{\{v|(u,v)\in E, v=(j,I)_{l'}, i\in I, u\neq v\}} S\{(j, I)_{l'}\} = \emptyset$$

3.6 Refining Channel Assignment: Phase 2

The interference-free assignment Φ_1 output from Phase 1 will serve as a reference for assigning hyperarcs to time slots for the remaining channels. We formulate the channel assignment problem for *each* of the remaining channels, $m = 2, 3, ..., M$, as a Two-sided Multi-assignment Problem (TMP) in this section. We describe the formulation as follows.

3.6.1 Formulation of CA Problem for Remaining Channels

On a particular channel $m \neq 1$, we divide time into τ equally-spaced time slots (See Fig. 3.4, for example). There are L hyperarcs $(i, J)_l, l = 1, 2, ..., L$, that we want to assign to available time slots on that channel. The number of time slots that each hyperarc requires depends on its packet injection rate $z^l_{iJ}(m)$ output from LP1 in (3.14) as well as its channel capacity $(c^l_{iJ}(m))$ [9]. We view the problem of assigning hyperarcs to time slots on a particular channel $m \neq 1$ as a Two-sided Multi-assignment problem where not only hyperarcs can be assigned to time slots but also time slots can be assigned back to hyperarcs, depending on the interference and node-radio constraints.

In general, after such two-sided assignment, it is possible that multiple hyperarcs are assigned to the same time slot as long as the hyperarcs do not interfere with each other. Furthermore, it is also possible that multiple non-contiguous time slots are allocated back to a single hyperarc as long as such assignment does not create interference or over-utilize the radios of the nodes associated with this hyperarc (because any member node of this hyperarc may be a member of any other hyperarcs). Any feasible assignment then implies a certain amount of benefits that can be gained, i.e., the aggregate packet injection rate

[9] From now on, we refer to hyperarc $(i, J)_l$ without its associated packet injection rate for convenience.

in our case. Our objective is thus to find such assignment that maximizes the aggregate benefit, subject to the interference and node-radio constraints. To this end, we model several elements required to formulate our TMP problem as follows.

Modeling the Set of Allowable Time Slots: $C(l)$

Due to interference and the different number of radios per node, an allowable set of time slots that can be assigned to a particular hyperarc on a particular channel $m \neq 1$ varies from hyperarc to hyperarc. Denote the allowable set of time slots to which hyperarc l can be assigned by a nonempty set $C(l)$.

To ensure a feasible channel assignment, the set $C(l)$ must be governed by the interference-free assignment of hyperarc to time slots Φ_{m-1} output from a previous stage as well as by the remaining number of radios at each node associated with hyperarc l. For ease of exposition, we first start with $m = 2$ and find the set $C(l)$ based on the assignment Φ_1 output by Algorithm 3.1. We show the pseudo code for finding $C(l)$ for $m = 2$ in Algorithm 3.2 (We will discuss later how this algorithm is modified for $m > 2$.).

For each hyperarc $(i, J)_l$, Algorithm 3.2 first checks if the minimum number of radios at all nodes associated with hyperarc l is less than the value of $m = 2$. If so, then we know that not all time slots can be assigned to hyperarc l on channel $m = 2$. Under this condition, three cases must be considered. For each receiver $j \in J$ of hyperarc l, we check if the number of its radios is (1) equal to, (2) greater than or (3) less than the number of node i's radios (node i is the transmitter of hyperarc l). If equal, we find hyperarcs l' such that at least one of the nodes associated with l' is a member of the set of nodes associated with l. We then know that hyperarc l cannot be assigned to the same time slots as those assigned

to hyperarc l' on channel $m' < 2$ due to the insufficient number of radios. A similar line of reasoning applies for the other two cases.

Modeling the Set of Allowable hyperarcs: $B(k)$

In assigning time slots back to hyperarcs, we also need to determine the set of hyperarcs to which time slot k can be assigned is also a non-empty set $B(k)$. The set $B(k)$ is dependent on the set $C(l)$ and easy to find. Algorithm 3.3 shows how $B(k)$ can be determined. The algorithm first allows all hyperarcs to be assigned to each time slot $k = 1, 2, ..., \tau$, then iteratively checks whether time slot k is in $C(l)$. If false, then hyperarc l is removed from $B(k)$.

Modeling the Maximum Number of Allowable Time Slots and Hyperarcs: α_k, α_l

Since a single time slot can be assigned to multiple hyperarcs, we first ask: what is the maximum number of hyperarcs (denoted by α_k) we can assign to each time slot k? Given the conflict graph G, it is sensible to limit α_k to the number of cliques in G since only one hyperarc can be active on each clique to avoid broadcast link interference. We thus set α_k equal to the number of cliques in conflict graph G.

On the other hand, a single hyperarc may be assigned to multiple time slots. To satisfy the proportional traffic demands output from LP1, the maximum number of time slots to which each hyperarc l can be assigned is determined by the packet injection rate $z^l_{iJ}(m)$ on hyperarc l. That is,

$$\alpha_l = \lceil \tau \frac{z^l_{iJ}(m)}{c^l_{iJ}(m)} \rceil$$

for all $l = 1, 2, ..., L$.

Algorithm 3.2: Finding the set of allowable time slots, $C(l)$.

input : Interference-free assignment Φ_1, no. of radios at each node

$R(i), i \in \mathbb{N} - s$, channel $m \neq 1$

output: Sets of allowed slots $C(l), \forall (i,J)_l, l = 1, 2, ..., L$

Initialization:

Set $S\{(i,J)_l\} = \emptyset, \forall (i,J)_l, l = 1, 2, ..., L$

Set $m = 1$ Set $C(l) = \{1, 2, ..., \tau\}, \forall (i,J)_l, l = 1, 2, ..., L$

availableSlots $= \{1, 2, ..., T\}$

for $l = 1, ..., L$ **do**

  Set $\mathcal{L} = \emptyset$

  Set tx(l) := transmitter of hyperarc l, and rx(l) := set of receivers of hyperarc l

  Set $R^l_{\min} = \min_{i \in \{\text{tx}(l) \bigcup \text{rx}(l)\}} R(i)$ **if** $R^l_{\min} < m$ **then**

  for $j \in$ rx(l) **do**

    if $R(j) = R(\text{tx}(l))$ **then**

      Find $\mathcal{L}_j = \{l' \mid$ tx$(l') =$ tx(l) OR rx$(l') \bigcap$ rx$(l) \neq \emptyset$ OR tx$(l') \in$ rx(l) OR rx$(l') \ni$ tx$(l)\}$

    else if $R(j) > R(\text{tx}(l))$ **then**

      Find $\mathcal{L}_j = \{l' \mid$ tx$(l') =$ tx(l) OR rx$(l') \ni$ tx$(l)\}$

    else if $R(j) < R(\text{tx}(l))$ **then**

      Find $\mathcal{L}_j = \{l' \mid$ rx$(l') \bigcap$ rx$(l) \neq \emptyset$ OR tx$(l') \in$ rx$(l)\}$

  Set $\mathcal{L} = \bigcup_j \mathcal{L}_j$

  Update the set of allowable time slots for hyperarc l: $C(l) = C(l) - \bigcup_{l' \in L} C(l')$, where $C(l')$ is the set of time slots scheduled for hyperarc l' in Φ_1.

  else

    All time slots are available for hyperarc l

Algorithm 3.3: Finding the sets of allowed hyperarcs, $B(k)$.

input : Sets of allowable time slots $C(l), \forall (i,J)_l, l = 1, 2, ..., L$

output: Sets of allowable hyperarcs $B(k), k = 1, 2, ..., \tau$

```
Initialization:
Set B(k) = {1, 2, ..., L}
for l = 1, ..., L do
    for k = 1, 2, ..., τ do
        if k not in C(l) then
            B(k) = B(k) \ {l}
```

Modeling the Benefit: b_{lk}

Finally, we model the benefit for assigning hyperarc l to time slot k as the number of coded packets injected on that hyperarc using channel $m \neq 1$, i.e.,

$$b_{lk} = c_{iJ}(m)\bar{\tau},$$

where $\bar{\tau}$ is the duration of each time slot. Note that not all time slots, however, are allowed for hyperarc l. Therefore, for those time slots k that are not in $C(l)$ we set the corresponding benefits b_{lk} to zero for each $l = 1, 2, ..., L$.

3.6.2 The Two-sided Multi-assignment Problem for Channel Assignment

We now have all the elements to formally state the TMP for channel assignment in each of the remaining channels $m = 2, 3, ..., M$. We state the TMP for channel assignment more formally as follows. An assignment Φ_m is a set of hyperarc-time slot pairs (l, k) such that time slot k is in the allowable set of time slots $(C(l))$, and hyperarc l is in the set of

allowable hyperarcs ($B(k)$) for all pairs $(l,k) \in \Phi_m$ on channel $m \neq 1$. For each hyperarc l, there can be at most α_l pairs $(l,k) \in \Phi_m$, and for every time slot there can be at most α_k pairs $(l,k) \in \Phi_m$. Given an assignment Φ_m, we say that hyperarc l' is *assigned* if there exists at least one pair $(l',k) \in \Phi_m$; otherwise, we say that hyperarc l' is *unassigned*. Similarly, time slot k' is *assigned* if there exists at least one pair $(l,k') \in \Phi_m$; otherwise, we say that time slot k' is *unassigned*. We say that an assignment Φ_m is feasible if every hyperarc, as well as every time slot, is assigned. An assignment is partial if at least one hyperarc or time slot is unassigned. Assigning a hyperarc l to time slot k incurs a benefit b_{lk}. The objective of the problem is to find an assignment Φ_m for each channel $m \neq 1$ such that the aggregate benefit, $\sum_{(l,k)\in\Phi} b_{lk}$, of such assignment is maximized. We write the TMP as follows (TMP1):

$$\begin{aligned} \text{maximize} \quad & \sum_{(l,k)\in\Theta} b_{lk}x_{lk} && (3.17)\\ \text{subject to} \quad & 1 \leq \sum_{k\in C(l)} x_{lk} \leq \alpha_l, \qquad \forall l = 1,...,L \\ & 1 \leq \sum_{l\in B(k)} x_{lk} \leq \alpha_k, \quad \forall k = 1,2,...,\tau \\ & 0 \leq x_{lk} \leq 1, \qquad\qquad \forall (l,k) \in \Theta \end{aligned}$$

where x_{lk} is a decision variable, and Θ is the set of all possible pairs (l,k). We assume that the number of time slots is greater than the number of hyperarcs, i.e., $\tau > L$.

3.6.3 Converting TMP to Minimum Cost Flow Problem

Since an assignment problem is a special case of the minimum cost flow problem (MCFP) [9], we can convert TMP1 into the standard MCFP. Introducing a virtual supply

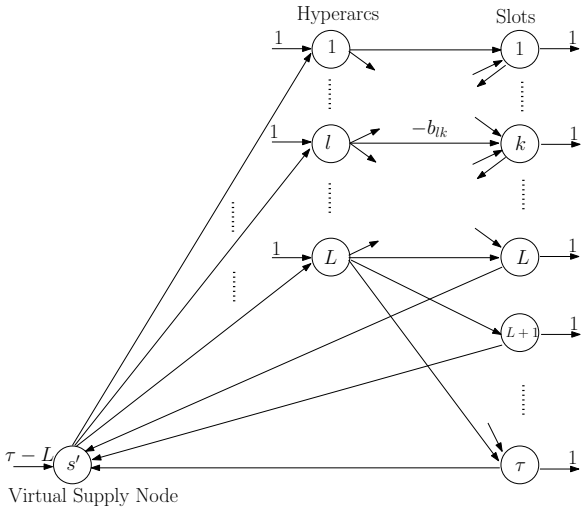


Figure 3.5: Viewing a two-sided multi-assignment problem as a minimum cost flow problem: Node s' is connected to all nodes $l = 1, 2, ..., L$ with zero-cost virtual arcs (s', l), each arc with a feasible flow range $[0, \alpha_l - 1]$. Nodes $k = 1, 2, ..., \tau$ are connected back to node s' via zero-cost virtual arcs (k, s'), each arc with feasible flow range $[0, \alpha_k - 1]$. All other arcs $(l, k), l = 1, 2, ..., L, k = 1, 2, ..., \tau$, have a feasible flow range $[0, 1]$.

node s' with supply $\tau - L$, we can think of hyperarcs and time slots as demand/supply nodes. We can then think of arcs connecting the nodes as transportation links with certain transportation costs. The problem is thus to find a set of arc flows that minimizes a linear cost function, subject to the flow conservation constraints and the capacity constraints.

In particular, as shown in Fig. 3.5, we view each hyperarc l as a node with unit supply and each time slot k as a node with unit demand. A decision variable x_{lk} can be interpreted as a flow of product from node l to node k, with the cost $-b_{lk}$ per unit product transported. In Fig. 3.5, we also connect node s' to all nodes $l = 1, 2, ..., L$ with the zero-cost virtual arcs (s', l), each arc with a feasible flow range $[0, \alpha_l - 1]$. Nodes $k = 1, 2, ..., \tau$ are

connected to node s' via the zero-cost virtual arcs (k, s'), each arc with feasible flow range $[0, \alpha_k - 1]$. All other arcs $(l, k), l = 1, 2, ..., L, k = 1, 2, ..., \tau$, have a feasible flow range $[0, 1]$. We formally write the MCFP as follows (MCFP1):

$$\begin{aligned}
\text{minimize} \quad & \sum_{(l,k)\in\Theta} (-b_{lk} x_{lk}) \\
\text{subject to} \quad & \sum_{k\in C(l)} x_{lk} - x_{s'l} = 1, \quad \forall l = 1, 2, ..., L \\
& \sum_{l\in B(k)} x_{lk} - x_{ks'} = 1, \quad \forall k = 1, 2, ..., \tau \\
& 0 \le x_{s'l} \le \alpha_l - 1, \qquad \forall l = 1, 2, ..., L \\
& 0 \le x_{ks'} \le \alpha_k - 1, \qquad \forall k = 1, 2, ..., \tau \\
& 0 \le x_{lk} \le 1, \qquad\qquad \forall (l, k) \in \Theta.
\end{aligned} \tag{3.18}$$

Using the duality theory, the dual problem of MCFP1 can be written as

$$\begin{aligned}
\text{minimize} \sum_{(l,k)\in\Theta} \max\{0, b_{lk} - p_k - \pi_l\} + \rho(\tau - L) + \sum_{l=1}^{L} (\pi_l + \max\{0, (\alpha_l - 1) \ldots \\
(\rho - \pi_l)\}) + \sum_{k=1}^{\tau} (p_k + \max\{0, (\alpha_k - 1)(\rho + p_k)\})
\end{aligned}$$

subject to no constraints, (3.19)

where we introduce the dual variables ρ, π_l, and p_k.

The dual problem in (3.19) can be solved by an auction-based algorithm [125] which tries to find the best possible aggregate benefit to MCFP1 by way of auction. In our context, dual variable ρ will be interpreted as the price of the virtual source s' of MCFP1, variable $-\pi_l$ as the price of hyperarc l (π_l is hence a profit), and variable p_k as the price of time slot k. Each hyperarc then may bid for any time slots in its allowable set, and, in

turn, each time slot may bid for any hyperarcs in its allowable set.

More specifically, for hyperarc l to use time slot $k \in C(l)$, it must submit the highest bid among those bids submitted by other hyperarcs l' (whose sets of allowable time slots also include time slot k, i.e., $C(l') \ni k$). We refer to this type of auction as a *forward auction*. On the other hand, for time slot k to take over hyperarc $l \in B(k)$, it must submit the highest bid among those bid submitted by other time slots k (whose sets of allowable hyperarcs also include hyperarc l, i.e., $B(k') \ni l$). We refer to this type of auction as a *reverse auction*. Our main channel assignment algorithm, described in the next section, is developed based on such forward and reverse auctions.

3.7 The Main Channel Assignment Algorithm

The objective of the main channel assignment algorithm is to find the best possible assignment of hyperarcs to time slots on all channel $m = 1, 2, ..., M$. By "best possible," we mean the assignment that yields the highest possible aggregate benefit which in our context is the highest aggregate packet injection rate on the multi-layered hypergraph H. It achieves this by first determining the interference-free assignment Φ_1 of hyperarcs to time slots for channel 1 (or, equivalently any channel). It then solves MCFP1 through its dual problem (3.19) by way of combined forward and reverse auction such that the best possible assignment of hyperarcs to time slots on the remaining channel $m = 2, ..., M$ is obtained. Because the main channel assignment algorithm takes as input the coding subgraphs $z(m), m = 1, 2, ..., M$, output by LP1, the algorithm must be run after LP1 is solved. A feasible channel assignment obtained from the main channel assignment algorithm

however does not guarantee that all the unicast connections will maintain the same best possible traffic scaling factor λ^* output for them by LP1. The traffic scaling factors of some connections then may deviate from λ^*, causing unfairness. We discuss this unfairness in the evaluation of our system.

We show the main channel assignment algorithm in Algorithm 3.5. It runs for every batch of K packets. For each run, it first finds the interference-free assignment Φ_1 for channel 1 (or, equivalently any channel) and calculates the initial set of allowable time slots, $C(l)$, for every hyperarc l. Using the assignment Φ_1 as a reference, it then iterates over channels $m = 2, ..., M$ and outputs a feasible assignment of hyperarcs to time slots with the maximum total packet injection on hyperarcs (as the aggregate benefit).

For each $m = 2, ..., M$, Algorithm 3.5 first sets up the appropriate environment for performing forward and reverse auctions. More specifically, it determines in Steps 6–9:

1. $C(l)$: The set of allowable time slots to which each hyperarc l can be assigned,
2. α_l: The maximum number of time slots to which each hyperarc l can be assigned
3. $B(k)$: The set of allowable hyperarcs to which each time slot k can be assigned, and
4. α_k: The maximum number of hyperarcs to which each time slot can be assigned.

It then starts with a partial assignment Φ such that at most one hyperarc is assigned to one time slot, and vice versa. Additionally, such a partial assignment must yield the initial prices of all time slots (p_k) and of all hyperarcs (π_l) such that they satisfy the first two $\epsilon-$Complementary Slackness ($\epsilon-$CS) conditions for a general asymmetric assignment problem [125] [10]. These conditions will ensure that the feasible assignment output by the

[10] In an asymmetric assignment there can be at most one object assigned to a person and at most one

main channel assignment algorithm is within $\tau\epsilon$ of being an optimal solution of the dual problem (3.19). Intuitively, the second condition states as follows. For all hyperarc-time slot pairs (l, k) in the partial assignment Φ, it must be the case that the profit made when hyperarc l wins the bid for time slot k is the benefit of assigning hyperarc l to time slot k less the price of time slot k itself, i.e., $\pi_l = b_{lk} - p_k$.[11] Now, for all possible pairs $(l, k) \in \Theta$ that may not be in the partial assignment Φ, it must be the case that the profit made when hyperarc l bids for time slot k is at least the benefit of assigning hyperarc l to time slot k less the price of time slot k itself and a bid increment ϵ, i.e., $\pi_l \geq b_{lk} - p_k - \epsilon$. This is the first $\epsilon-$CS condition.

With assignment Φ from the previous step, a standard forward auction algorithm, shown in Appendix 1, is executed to yield a new assignment Φ such that each hyperarc is assigned to a single distinct time slot, and that the first two $\epsilon-$CS conditions for a two-sided multi-assignment problem are satisfied [125] in Step 11. This in effect yields an interference-free broadcast link scheduling. However, since the conflict graph may contain multiple cliques, multiple hyperarcs may share the same time slot. The next step is therefore to assign time slots back to hyperarcs.

Step 12 first sets the price ρ of the virtual source s' to the maximum profit as seen by all hyperarcs $l = 1, 2, ..., L$ in Step 11. As a result, any time slot k with price p_k plus the maximum profit ρ greater than zero should be assigned to as many hyperarcs as possible. We modify a standard reverse auction algorithm to assign time slots back to hyperarcs (Step 13). Candidate time slots must be unassigned or assigned with less than α_k

person assigned to an object, given that the number of objects is greater than the number of persons. Therefore, some objects are allowed to remain unassigned.

[11] Recall that we interpret $-\pi_l$ as a price of hyperarc l.

hyperarcs. In both cases, the price of each candidate time slot p_k plus the maximum profit ρ must be greater than zero. We show the pseudo code for our modified reverse auction algorithm in Algorithm 3.5. The output of Algorithm 3.5 is a feasible assignment such that each time slot k is assigned to at most α_k hyperarcs, and each hyperarc is assigned to at most α_l time slots. There are two main cases that we have to consider in Algorithm 3.5. The first case shown in Algorithm 3.6 occurs when the bidding time slot is unassigned. The second case shown in Algorithm 3.7 occurs when the bidding time slot is already assigned, but the number of hyperarcs assigned to it is still less than the maximum allowed (α_k), and its price p_k plus the maximum profit ρ is greater than zero. In this case, either time slot k is assigned to an additional hyperarc, or else its price p_k is set to $-\rho$. For both cases, the maximum number of allowable time slots for hyperarc l (α_l) and the maximum number of allowable hyperarcs for time slot k (α_k) will be updated. In addition, if the maximum number of allowable time slots for hyperarc l (α_l) is equal to or less than zero, then the packet injection rate on hyperarc l is satisfied, and hyperarc l can be removed from the set of allowable hyperarcs for time slot k ($B(k)$) for $k = 1, 2, ..., \tau$. Algorithm 3.5 continues as long as there is at least one time slot available for assignment, and the packet injection rate on each hyperarc is not satisfied.

Upon exiting from the modified reverse auction algorithm (Algorithm 3.5), our main channel assignment algorithm re-calculates $C(l)$, the set of allowable time slots to which hyperarc l can be assigned. It takes as input a current assignment Φ and a current $C(l)$ to Algorithm 3.2. Taking this current $C(l)$ implies that Step 2 in Algorithm 3.2 must be skipped. That is, not all time slots now will be available for hyperarc l on the new channel.

Algorithm 3.4: The main channel assignment algorithm.

input : $z_{iJ}(m), m = 2, ..., M$ and conflict graph $G = (V, E)$

output: Assignment $\Phi_1, ..., \Phi_M$

Initialization:

Run **Algorithm 3.1** to get the interference-free assignment Φ_1 for channel $m = 1$

Given Φ_1 from Step 2, run **Algorithm 3.2** to get $C(l), l = 1, 2, ..., L$

Set $\epsilon = 1/\tau$

for $m = 2, ..., M$ **do**

  Given $C(l), l = 1, 2, ..., L$, run **Algorithm 3.3** to get $B(k), k = 1, 2, ..., \tau$

  Set α_k to no. of cliques in G for all $k = 1, 2, ..., \tau$

  Set $\alpha_l = \text{ceil}(\tau \frac{z_{iJ}(m)}{c_{iJ}(m)}), \quad \forall (i, J)_l, l = 1, 2, ..., L$

  Set $b_{lk} = \frac{c_{iJ}(m)\bar{\tau}}{\chi}, \quad \forall (i, J)_l, l = 1, 2, ..., L, k = 1, 2, ..., \tau$

  Start with a random assignment Φ such that at most one hyperarc is assigned to one time slot, and vice versa, with $p = (p_1, ..., p_k, ..., p_\tau)$ and $\pi = (\pi_1, ..., \pi_l, ..., \pi_L)$ satisfying $\pi_l + p_k \geq b_{lk} - \epsilon, \forall (l, k) \in \Theta$ and $\pi_l + p_k = b_{lk}, \forall (l, k) \in \Phi$

  Run **Forward Auction Algorithm (Algorithm B.1)** with input $(p, \pi, b_{lk}, C(l), \epsilon, \Phi)$ obtained from previous steps until each hyperarc is assigned to a single distinct time slot while satisfying $\pi_l + p_k \geq b_{lk} - \epsilon, \forall (l, k) \in \Theta$ and $\pi_l + p_k \leq b_{lk}, \forall (l, k) \in \Phi$

  Set $\rho = \max_{l=1,2,...,L} \pi_l$

  Run **Modified Reverse Auction Algorithm (Algorithm 3.5)** with input $(b_{lk}, p, \pi, \rho, C(l), B(k), \epsilon, \Phi)$

  if $m \neq M$ **then**

    Run **Algorithm 3.2** with input Φ (instead of Φ_1) and $C(l)$ from previous steps (i.e., skip step 2 in **Algorithm 3.2**) to get new $C(l), l = 1, 2, ..., L$

  Set $\Phi_m = \Phi$

In other words, as we move to assign hyperarcs to time slots in a new channel, the number of available time slots to which hyperarc l can be assigned becomes more limited. At the end of the execution, the main channel assignment algorithm (Algorithm 3.4) outputs the feasible channel assignment that may deviate the upperbound proportional traffic demands $(\lambda^* r_n)$ for all nodes $t_n \in \mathbb{T}'$. This tradeoff is present because auctions introduce competition. We will show in the evaluation that this issue can be partially dealt with.

3.8 Complexity of the Main Channel Assignment Algorithm

We analyze the complexity of the main channel assignment algorithm (Algorithm 4) in this section. We start with the complexity of the initialization phase. Computing the interference-free assignment Φ_1 in Algorithm 3.1 has a time complexity of $O(L^2)$ since every hyperarc has to be visited and checked if it interferes with any other hyperarcs. Computing the set of allowable time slots for all hyperarcs in Algorithm 3.2 has a complexity of $O(L(|\mathbb{N}'|-1))$ since we have to check for every hyperarc whether its constituent wireless nodes have a sufficient number of radios to activate additional simultaneous communications in a particular time slot. For both cases, in the worst-case scenario, the number of hyperarcs with non-zero packet injection rates, L, is equal to the number of wireless nodes, $|\mathbb{N}'|-1$. In other words, it is the scenario where every wireless node acts as a relay. The initialization phase of Algorithm 3.4 thus has a $O((|\mathbb{N}|-1)^2)$ worst case running time. The **for** loop of Algorithm 3.4 is repeated exactly $(M-1)$ times, and the complexity of each iteration depends on the complexity of Algorithms 3.2, 3.3, 3.5, and B.1. The complexity of Algorithm 3.3 can be shown to be of $O(\tau L)$ since it has to loop through every hyperarc and

Algorithm 3.5: Modified reverse auction algorithm.

input : $b_{lk}, p, \pi, \rho, C(l), B(k), \epsilon, \Phi$

output: Assignment $\Phi = \{(l,k) \mid$ each hyperarc l is assigned to at most α_l time slots, and each time slot is assigned to at most α_k $\}$

Initialization:

Set $\Phi = \emptyset$

availableSlots $= \{k \mid k$ unassigned or assigned to at least one but less than α_k links, and $p_k > -\rho$ $\}$

while availableSlots *NOT empty & Packet injection rate on each hyperarc NOT satisfied* **do**

  for $k \in$ availableSlots **do**

    if $B(k)$ *NOT empty* **then**

      Find the best hyperarc $l^* = \arg\max_{l \in B(k), (l,k) \notin \Phi} \{b_{lk} - \pi_l\}$

      Find the corresponding price of the best hyperarc $\eta_k = \max_{l \in B(k), (l,k) \notin \Phi} \{b_{lk} - \pi_l\}$

      if $\{l \mid l \in B(k), (l,k) \notin \Phi, l \neq l^*\}$ *is empty* **then**

        Set the price of the second best time slot $w_k = -\infty$

      else

        Set $w_k = \max_{l \in B(k), (l,k) \notin \Phi, l \neq l^*} \{b_{lk} - \pi_l\}$

      if k *is unassigned* **then**

        Execute code in **Algorithm 3.6**

      else if k *is assigned to at least one but less than* α_k *hyperarcs, and* $p_k + \rho > 0$ **then**

        Execute code in **Algorithm 3.7**

    else

      $p_k = -\rho$

Algorithm 3.6: Code snippet to be executed if k is unassigned.

Set $\sigma = \min\{\rho - \pi_{l^*}, \eta_k - w_k + \epsilon\}$

if $\sigma > 0$ **then**

    Remove (l^*, k') from Φ, where k' was previously assigned to l^* under Φ.

Add (l^*, k) to assignment Φ, and set

$p_k := w_k - \epsilon; \pi_{l^*} := \pi_{l^*} + \sigma;$

$\alpha_{l^*} = \alpha_{l^*} - 1$ and $\alpha_k = \alpha_k - 1$

if $\alpha_{l^*} \leq 0$ **then**

    Remove l^* from $B(k), k = 1, 2, ..., \tau$

every time slot. Assigning L hyperarcs to exactly L distinct time slots in Algorithm B.1 requires a worst case running time of $O\left(\tau L^2(1 + \frac{b^{\max}}{\epsilon})\right)$, where $b^{\max} = \max_{(l,k)\in\Theta} b_{lk}$ and $\tau > L$ [125]. Similarly, it can be shown that multiassigning hyperarcs to time slots, and vice versa, in Algorithm 3.5 takes an $O\left(\tau^2 L(1 + \frac{b^{\max}}{\epsilon})\right)$ worst case running time. In the worst case scenario, where all hyperarcs on all channels have to participate in forwarding, L will be replaced by $|\mathbb{N}| - 1$. Additionally, since τ is greater than L, we conclude that Algorithm 4 runs in polynomial time of $O\left((M-1)\tau^2(|\mathbb{N}|-1)(1 + \frac{b^{\max}}{\epsilon})\right)$. Note that the complexity of polynomial-time channel assignment algorithms, which have been proposed for the conventional routed IWMNs, also depend on the number of nodes, links, channels and radios (see [109, 117], for example). The complexity of our proposed algorithm is however independent of the number of radios.

Algorithm 3.7: Code snippet to be executed if k is assigned to at least one but less than α_k hyperarcs, and $p_k + \rho > 0$.

Set $\sigma = \min\{\rho - \pi_{l^*}, \eta_k - w_k + \epsilon, \eta_k + \rho\}$

if $\sigma < \eta_k + \rho$ **then**

  Add (l^*, k) to assignment Φ, and set

  $p_k := \max\{w_k - \epsilon, -\rho\}$, $\pi_{l^*} := \pi_{l^*} + \sigma$ and

  $\pi_l := \min\{b_{lk} - \max\{w_k - \epsilon, -\rho\}, \rho\}, \forall l$ such that $(l, k) \in \Phi$

  $\alpha_{l^*} = \alpha_{l^*} - 1$ and $\alpha_k = \alpha_k - 1$

  if $\alpha_{l^*} \leq 0$ **then**

    Remove l^* from $B(k), k = 1, 2, ..., \tau$

  if $\sigma > 0$ **then**

    Remove (l^*, k') from Φ, where k' was previously assigned to l^* under Φ.

if $\sigma = \eta_k + \rho$ **then**

  Set $p_k := -\rho$

  $\pi_{l^*} := \pi_{l^*} + \max\{0, \sigma\}$ and

  $\pi_l := \min\{b_{lk} + \rho, \rho\}, \forall l$ such that $(l, k) \in \Phi$

  $\alpha_{l^*} = \alpha_{l^*} - 1$ and $\alpha_k = \alpha_k - 1$

  if $\alpha_{l^*} \leq 0$ **then**

    Remove l^* from $B(k), k = 1, 2, ..., \tau$

  if $\sigma > 0$ **then**

    Add (l^*, k) to assignment Φ

    Remove (l^*, k') from Φ, where k' was previously assigned to l^* under Φ.

3.9 Performance Evaluation

3.9.1 Methodology

We first evaluate our algorithm via simulations in Matlab [147] and later via experiments on an IEEE 802.11b/g testbed. In both cases, we will look at the impact of using multiple channels and multiple radios on throughput and fairness performances. We compare our proposed scheme, denoted by **RNC-AucCA**, with the following non-coding schemes:

1. **Single path routing with random channel assignment** (**SP-RandCA**) which employs a state-of-the-art best path routing protocol and assigns channels to radios uniformly at random. The state-of-the-art best path routing protocol uses the Dijkstra's shortest path algorithm to find the best route. Link costs are calculated based on the ETX metric [128] which is the expected number of transmissions for delivering a packet from any node to a sink.

2. **Multi-path routing with our auction-based channel assignment and link scheduling scheme** (**MP-AucCA**) in which packets are routed concurrently via multiple routes. Flow rates on the links along all the routes are determined by LP1 (3.14). The flow vector $f : N \times N \rightarrow R$ is then fed as input to our auction-based channel assignment algorithm in which time slots bid for *point-to-point links* (instead of *broadcast links* (hyperarcs) as in the network coding case) and vice versa.

3. **Multi-path routing with combined channel and link scheduling** (**MP-RCL**) as proposed in [109], which jointly solves routing, channel assignment and link schedul-

ing in a multi-channel multi-radio IWMN. This scheme first transforms the given network $\mathbb{H}$ into a flow network H and then solves an LP similar to (3.14), yielding the routes in terms of the flow vector $f : N \times N \rightarrow R$. Based on f, channel assignment is performed by observing that such assignment is trivial if the number of channels is equal to the minimum number of radios per node ($R_{\min}$): each node is simply assigned all the channels. In particular, the given network $\mathbb{H}$ is transformed into a network $\mathbb{H}'$ such that each node $i \in \mathbb{H}'$ has approximately $R_{\min}$ radios, and the total link utilization of all links incident on $i \in \mathbb{H}'$ is at most $R(i)$ where $R(i)$ is the number of radios per node. The $R_{\min}$ channels are then assigned such that the node-radio constraints, similar to (3.13), are still satisfied. Next, the flow vector f is readjusted, trying to spread the interference uniformly across all channels so that the maximum interference on each channel is bounded. Channel assignment is later revised using the remaining $M - R_{\min}$ channels while trying to minimize the maximum interference on each channel. Finally, channel assignment is mapped back to the original IWMN $\mathbb{H}$, and the links sharing the same channel are scheduled to use different time slots. Channel assignment is re-executed if the network topology and/or traffic demands change.

3.9.2 Performance Metrics

We use the following metrics for performance evaluation:

- **Average throughput** refers to a per-connection throughput averaged over all connections. Each connection is a tuple (s, t_n, r_n) denoting a unicast connection from

source s to a sink $t_n \in \mathbb{T}'$ with traffic demand r_n packets per unit time with an achieved throughput $\lambda_n r_n$ output by the auction-based channel assignment algorithm. When compared with the traditional multi- and single-path routing schemes, we also consider the network coding overhead in our proposed scheme (RNC-AucCA) for fair comparison.

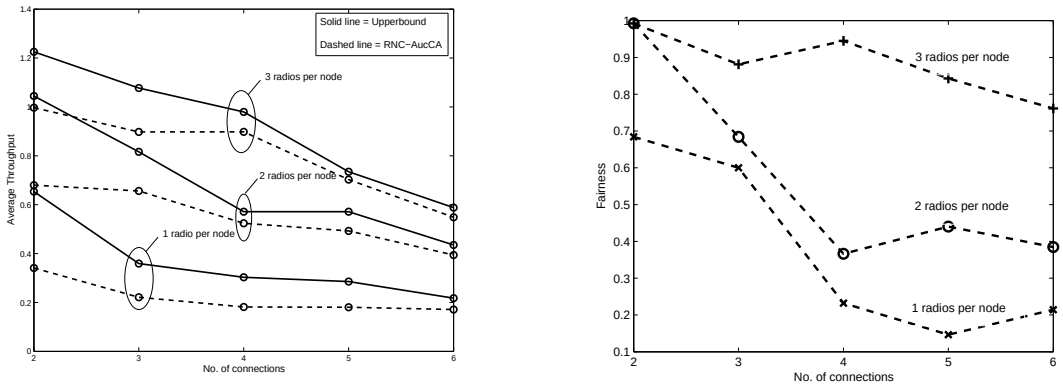
- **Fairness** defines how fairly each connection is served compared to others. Recall that LP1 (3.14) outputs the same and upperbound λ^* for all connections. However, after channel assignment has been made feasible in the auction stage, each connection may not necessarily have the same value of λ. The deviation of λ from the upperbound value occurs because of the competition in the auction stage. Based on the Jain's fairness index, we thus define fairness as follows:

$$\text{fairness } = \frac{\left(\sum_{n=1}^{T}(\lambda^* - \lambda_n)\right)^2}{T\sum_{n=1}^{T}(\lambda^* - \lambda_n)^2},$$

where λ_n is the value of the traffic scaling factor output by Algorithm 3.4. Ideally, every connection would have the same λ^* output by LP1, which corresponds to fairness having the value of 1 (best case). On the other hand, when λ_n greatly deviates from λ^*, fairness approaches $1/T$ (worst case).

3.9.3 Simulation Evaluation

We evaluate our algorithm using an indoor IWMN. We assume that the system operates synchronously in a time-slotted mode. We use a log-normal model to predict radio propagation inside a building and consider random topologies with a total of 15 nodes. We assume the transmission range of 20 meters and the interference range of 40 meters.



(a) (b)

Figure 3.6: Intuitively, per-connection throughput decreases with the number of connections that must be supported. However, it tends to approach the upperbound value as the competition soars in the auction stage, resulting in reduced fairness.

The number of radios per node varies from 1 to 3. We fix the number of gateways to 2. For every batch of K packets, a central controller solves LP1 (3.14) and inputs the coding subgraphs $z(m), m = 1, 2, ..., M$, to Algorithm 3.4. The feasible assignment of hyperarcs to time slots and the coding subgraphs are then distributed to all nodes. We assume the batch size of packets $K = 64$ for all scenarios where each packet is 1,500 bytes long, and coding operations occur in the finite field $\mathbf{GF}(2^8)$.

Impact of Multiple Radios

In this evaluation, we refer to the upperbound throughput as the solution output from LP1 (3.14), which does not necessarily yield a feasible channel assignment but provides upperbounds on flows and coding subgraphs on the multi-layered hypergraph H. We use the default large-scale linear programming solver [145] – a variant of a primal-dual interior-

point method – provided in Matlab to obtain the best possible traffic scaling factor (λ^*) for LP1. This λ^* is same for all connections and serves as a scaling factor that yields the upperbound throughput $\lambda^* r_n, n = 1, 2, ..., T$.

We vary the number of connections in the 15-node random topologies to see its impact on the average throughput and fairness. Each connection has a normalized traffic demand of 1. The number of channels is fixed at 3. Each data point is averaged over 10 random topologies. We show the results in Fig. 3.6(a) for 3 settings with the number of radios per node varying uniformly from 1 to 3. For both upperbound and RNC-AucCA solutions, as expected, the average throughput decreases as the number of connections increases. As the number of radios uniformly increases, the throughput generally increases. We see that our algorithm can effectively exploit the number of radios available.

More interestingly, we also observe that our sub-optimal auction-based solution (RNC-AucCA) tends to approach the upperbound solution as the number of connections increases. This can be explained as follows. When the number of connections increases, generally the number of hyperarcs used to inject coded packets also increases[12]; in other words, time slots now have more hyperarcs to bid for. The set of available hyperarcs that the time slots can bid however differs from time slot to time slot, depending on the number of radios and interference constraints explained in Algorithms 3.2 and 3.3. Some hyperarcs therefore are assigned to time slots more often than needed while some might starve. And, the average throughput does not necessarily divert from the upperbound value while the individual throughput might, resulting in unfairness. We show this unfairness in Fig. 3.6(b)

[12]Recall that in terms of random network coding, it does not matter which packet is injected on a hyperarc. A node stops packet injection only when it is told to do so.

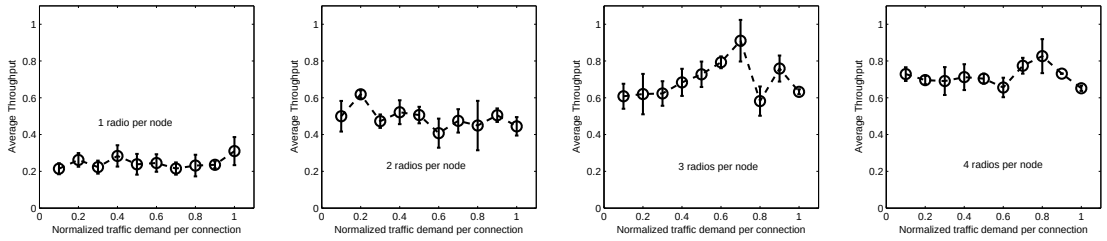


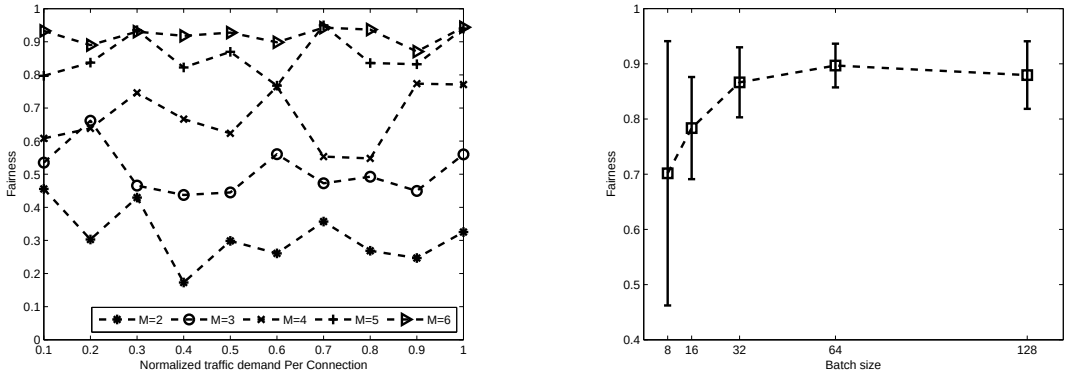
Figure 3.7: Throughput nearly doubles for one additional radio added at each node.

where we plot only the results for RNC-AucCA since, for the upperbound case, fairness is 1.

As we fix the number of connections at 6 connections and let traffic demands vary uniformly for all the connections, we obtain the results shown in Fig. 3.7 for 4 settings depending on the number of radios per node. We fix the number of channels at 6. Each data point on each setting is averaged over 10 random topologies. We see that for one additional radio added at each node, the average throughput nearly doubles systemwise. This result shows the effectiveness of RNC-AucCA in exploiting the increased number of radios per node.

Addressing the Fairness Issue

Intuitively, unfairness seen in Fig. 3.6(b) can be mitigated if more channels are available because employing more channels effectively increases the number of time slots that will bid for the same number of hyperarcs. In this evaluation, we let the number of connections and the number of radios per node be 6 and 2, respectively. We show the results in Fig. 3.8(a) for 5 settings with the varying number of channels available. Each data point is averaged over 10 random topologies. We see that fairness increases significantly as the



(a) (b)

Figure 3.8: Addressing fairness issues: (a) Fairness improves as the number of channels available in the system increases. (b) Fairness also improves if the batch size is properly adjusted.

number of available channels increases. For example, with a normalized demand of 1, we see that when moving from 2 to 6 channels fairness increases from 0.33 to 0.94, a 180% increase.

For a fixed number of channels available, we can also address the fairness issue by properly adjusting the network coding parameter, namely, the batch size of packets (K) to code. In this evaluation, we fix the number of connections, channels and radios at 6, 5 and 2, respectively. All connections uniformly demand a normalized traffic of 1. We show the result in Fig. 3.8(b) where each data point is averaged over 10 random topologies. We observe that as K increases, fairness improves. For example, we see a 21% improvement as K is increased from 8 to 32 packets. Recall that hyperarcs bid for time slots over a period of τ time slots on each channel and that this τ increases proportionally with K. Increasing K

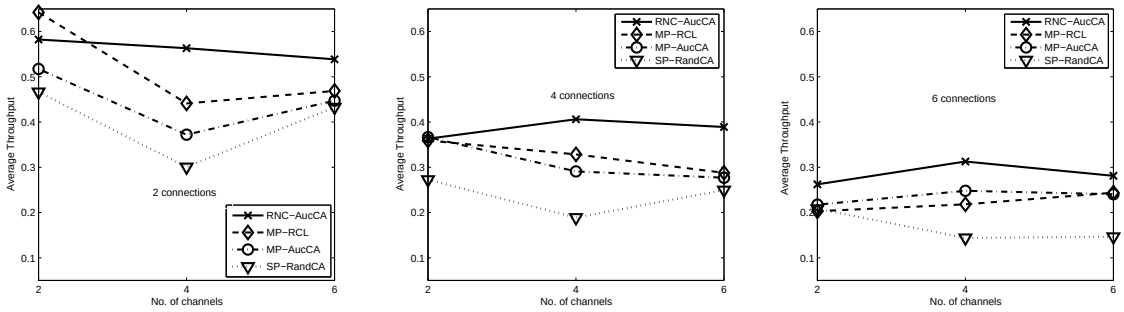


Figure 3.9: On average, RNC-AucCA yields a throughput gain over the single-path routing solution with naive channel assignment (SP-RandCA) and the multi-path routing solutions with more intelligent channel assignment (MP-RCL and MP-AucCA) by 40% and 28.5%, respectively.

thus effectively increases the number of time slots each of which has a finer grain than that with a lower K. This higher number of finer-grained time slots implies that all hyperarcs now have more choices in bidding for their desired time slots. In other words, the size of the set of allowable time slots ($C(l)$) for each hyperarc effectively increases. Further increasing τ however may not improve fairness since as time slots become small, channel switching occurs too frequently and becomes computationally cumbersome. For the rest of this discussion, otherwise specified, we set K equal to 64 .

These two sets of results collectively justify the effectiveness of our auction-based channel assignment algorithm (RNC-AucCA) in resolving the fairness issue by exploiting the increasing number of channels as well as fine-tuning the network coding parameter.

Comparison with Routing

In this evaluation, the goal is to compare our proposed scheme (RNC-AucCA) with the traditional routing schemes in which various channel assignment strategies are employed. We fix the number of radios per node at 2. We plot the results in Fig. 3.9 for 3 settings with the varying number of connections. Each connection has a normalized traffic demand of 1. Each data point is averaged over 10 topologies. First of all, we see that as the number of connections increases the average throughputs obtained by all the schemes drop. This is expected as we have already seen in Fig. 3.6(a). On average, RNC-AucCA outperforms the single-path routing solution with naive channel assignment (SP-RandCA) by 40% in terms of throughput. This is due to the fact that, in contrast to SP-RandCA our RNC-AucCA system can fully exploit the broadcast nature of wireless channels by allowing packets to be transmitted on multiple alternative paths. In addition, when radios switch channels randomly in SP-RandCA, it is more likely to partition the network into several disconnected sub-networks. RNC-AucCA, on the other hand, relies on the coding subgraphs and flows output from LP1 (3.14), which ensures that all nodes on the coding subgraphs remain connected after channel assignment.

We also see that, compared with the multi-path routing schemes MP-RCL [109] and MP-AucCA, RNC-AucCA improves the throughput by 28% and 29%, respectively, on average. The improvement over MP-RCL (which embeds a static channel assignment scheme) is due to the fact that our proposed system performs channel assignment in a more dynamic manner while taking channel conditions (via broadcast link capacities) into account. That is, in every batch of K packets, the bidding benefit that each hyperarc

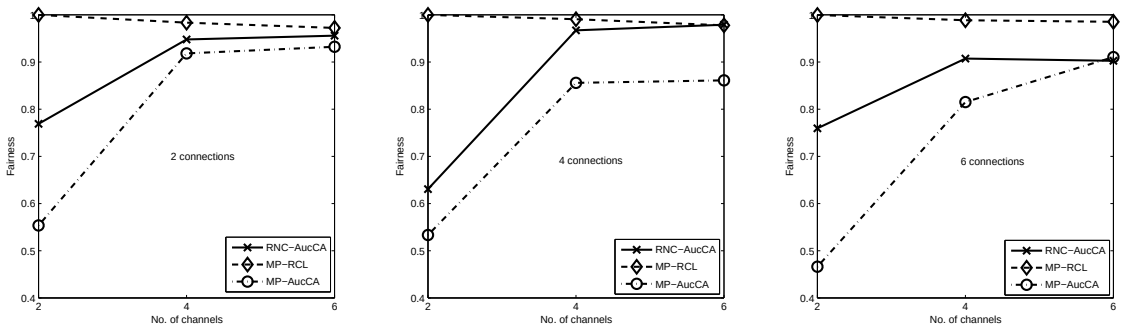


Figure 3.10: Auction-based solutions fail to give fairness when the number of available channels is limited. Fairness improves however if more channels are available.

achieves from winning a particular time slot on a certain channel varies from time to time as the capacities of hyperarcs (broadcast links) change. The throughput improvement over both MP-AucCA and MP-RCL is also due in part to the way packets are routed. In traditional routing, a packet of a particular connection must be transmitted on a certain route. In random network coding, however, this constraint is eliminated by allowing a non-empty subset of ending nodes of each broadcast link to randomly mix packets and transmit through opportunistic listening/overhearing.

The flexibility of random network coding in routing packets however comes with a price in fairness as we show in Fig. 3.10 for the corresponding 3 settings where we omit the result for SP-RandCA as it tends to fluctuate randomly without patterns. While fairness in MP-RCL is nearly insured, RNC-AucCA and MP-AucCA, whose channel assignment is calculated based on auctions, experience unfairness, especially when the number of available channels is limited. This can be expected since auctions introduce competition into the channel assignment phase where hyperarcs (or point-to-point links in case of MP-AucCA) greedily bid for their desired time slots and vice versa. This unfairness however substantially

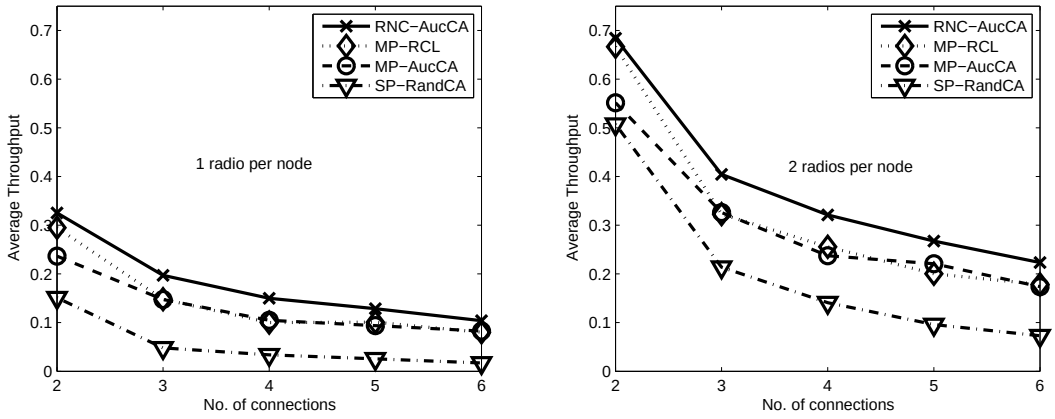


Figure 3.11: RNC-AucCA outperforms SP-RandCA by approximately 42% while throughput gains over MP-AucCA and MP-RCL vary between 22% and 35%.

improves once the number of available channels increases. For example, with 4 connections, fairness for MP-AucCA improves by approximately 60% while fairness for RNC-AucCA also increases by 100% when the number of channels increases from 2 to 6 channels. This result is consistent with the results in Fig. 3.8(a).

For a fixed number of channels (4 channels), we also show the performance comparison among the above schemes in Fig. 3.11 for 2 settings with the varying number of radios per node. Each connection demands a normalized traffic of 1. We see that RNC-AucCA outperforms SP-RandCA by approximately 42% while improvements over MP-AucCA and MP-RCL range between 22% and 35%. This result complements the previous finding in Fig. 3.9.

3.9.4 Experimental Evaluation

We evaluate our algorithm using a 10-node 802.11b/g testbed. Each node in our testbed is a net4521 Soekris board [148] running a Debian-based Linux [149]. Each board is equipped with two 802.11b/g wireless interfaces (DLink-WNA2330 model [150]). Each interface is connected to an omni-directional antenna and set to ad-hoc mode without RTS/CTS capability. We set the transmit power of all the wireless interfaces to 10 dBm. The bitrate of each interface is set at 11 Mb/s. All transmissions on each channel are in broadcast mode. Using the userlevel Click framework [126], we implement random network coding as a userspace daemon which sends and receives raw 802.11 frames from the bonding interface using a libcap-like interface [151]. For the source node, UDP traffic is generated by a Click utility program `udpgen`. Each connection requests a file whose size varies between 10 to 20 MB. A packet size is 1500 bytes. We fix the batch size at $K = 64$ for all connections. We assume a uniform traffic demand for all nodes. For more details on our random network coding module, the reader is referred to [127].

The system starts with a random channel assignment for the first batch while the central controller, attached to the source node (representing the Internet), solves LP1 (3.14) and executes Algorithm 3.4 for the following batch. The header of each packet specifies the source and destination IP addresses, a batch sequence number, a coding vector, and a list of forwarders. In addition, for each forwarder, the header also specifies a list of channels and the corresponding numbers of time slots output from Algorithm 3.4. Since the 802.11 MAC is not synchronized, each forwarder only knows how much it should send. The decision of when to access each channel is handled by the channel and interface coordinator module

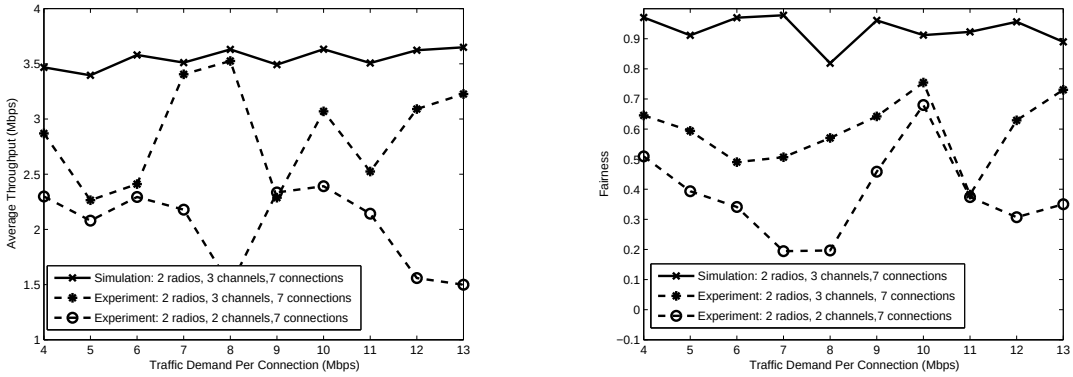
which we briefly explain below.

Channel switching is handled by our channel and interface coordinator module as described in Chapter 4. This module communicates with the network coding module and the bonding device. Interface bonding allows the aggregation of multiple network interfaces into a single logical one such that user applications are oblivious to multiple physical interfaces or IP addresses. Interface bonding provides *features* for throughput aggregation, load balancing and fault-tolerance. Based on [143], we implement a new bonding feature aimed specifically for channel switching. Using the channel and interface coordinator module, a node learns about the channels used by its neighbors through a special "hello" message sent out periodically by its neighbors. The node then updates the entries of the lookup table through IOCTL calls to the bonding device. The role of the lookup table is to map a coded packet to a channel and a physical wireless interface if there are listening neighbors.

We place testbed nodes across our 40m×40m office floor so that we have The node locations are constrained by office privacy and the availability of LAN outlets through which we send experiment-driving commands and collect data. This results in a random topology with paths between the nodes ranging between 1 to 4 hops and the link loss rates varying between 0 and 40%. We use the ETX implementation provided by the Roofnet software [128] to periodically measure a packet delivery probability for each link. We assume that the packet delivery probability report is ready before the system starts.

Experimental Results

We first compare our experimental results with the simulation results. For simulation, we set the number of nodes to 10 with 2 radios per node. The number of channels and



(a) (b)

Figure 3.12: Our experiment results are upper-bounded by the simulation results due to lack of synchronization in the 802.11 MAC.

the number of connections are fixed at 3 and 7, respectively. For experiments, we have two settings. The first setting resembles the simulation case, and the second setting reduces the number of channels available to 2. The results are shown in Fig. 3.12. Each data point of the simulation results is averaged over 10 simulation runs on a random topology that resembles our testbed topology. For each simulation run, a fixed number of random connections is chosen. Each data point of the experimental results is averaged over 30 experiments. In each experiment, a fixed number of connections is randomly selected. We see that our experimental results are bounded above by the simulation results for all traffic demands. This can be expected because our simulation assumes that the system operates synchronously in a time slotted mode. In our experiment, however, there is no such synchronization in the 802.11 MAC. Comparing between the two experimental settings, we also see that our algorithm can effectively exploit the increasing number of available channels. For example,

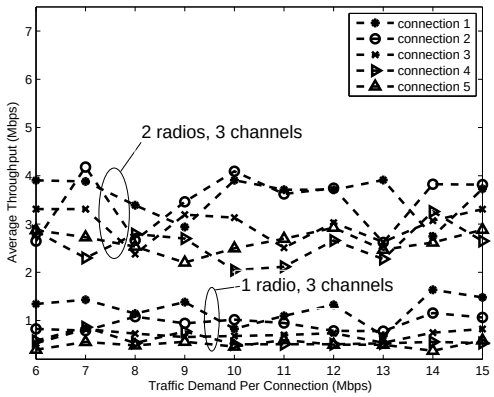
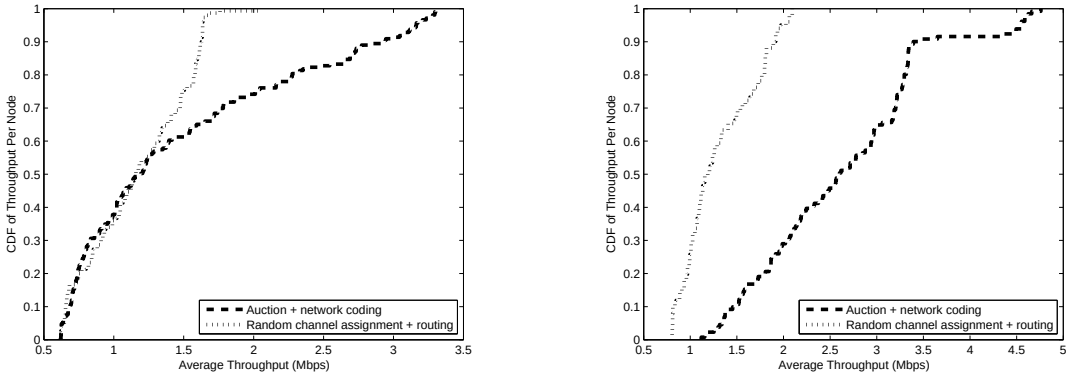


Figure 3.13: As the number of channels is fixed, RNC-AucCA is able to effectively exploit the increasing number of radios.

at the demand of 12 Mbps, the improvement accounts for a 95% increase in throughput and for a 113% increase in fairness. This result confirms that RNC-AucCA can effectively exploit additional orthogonal channels if available.

We next evaluate our system to see the impact of multiple radios on average throughput. We fix the number of available channels at 3. The number of connections is fixed at 5: the source-destination pair is fixed for each connection. The results are shown in Fig. 3.13 for 2 settings with the varying number of radios per node. Each data point is averaged over 30 experiments for each connection. As expected, with the higher number of radios available, the average throughput is significantly improved. On average, this improvement accounts for a 56% increase. This result confirms our simulation result in Section 3.9.3.

Finally, we compare RNC-AucNC with SP-RandCA the implementation of which is available for public use [128]. In this evaluation, we fix the number of radios at 2. The



(a) 1 channel available

(b) 3 channels available

Figure 3.14: Cumulative distribution of average throughput: As the number of channels increases, RNC-AucCA effectively utilizes the additional radio, compared to SP-RandCA.

number of connections is also fixed at 7, each connection with a uniform traffic demand of 10 Mbps. Fig. 3.14 shows the cumulative distributions of average throughput achieved by RNC-AucCA and by SP-RandCA, for 2 settings with the varying number of available channels. To obtain this result, we collect data from 200 experiments. In each experiment, RNC-AucCA is executed after SP-RandCA, and the throughput is calculated by averaging over all the connections. We see that empirically the average throughput achieved by RNC-AucCA significantly increases as two more channels become available for use, while the throughput for SP-RandCA does not necessarily increase. For example, we see that 50% of the time both schemes have throughput no greater than 1.25 Mbps when one channel is available. However, when three channels are available, RNC-AucCA pushes this limit to 2.75 Mbps while the throughput of SP-RandCA marginally increases.

3.10 Conclusion

In this chapter, we have presented the network-wide solution to the joint problem of channel assignment, random network coding and broadcast link scheduling for an infrastructure multi-channel multi-radio wireless mesh network. The main objective is to optimize throughputs for multiple unicast connections in a fair manner. We have first formulated the problem as a linear program that finds the optimal coding subgraphs for all connections. Based on the solution to this linear program, we have proposed a centralized, auction-based algorithm that computes a feasible channel assignment for all nodes. We have demonstrated through simulations and a miniature 802.11b/g testbed that our proposed solution can effectively exploit the increasing number of channels and radios. Our solution also shows promising gains over several traditional routing schemes in which various channel assignment strategies are employed. Through simulation, we also show that unfairness arising from auctions can also be managed by properly tuning a batch size of packets and utilizing more channels.

Chapter 4

iCORE: Experimental Platform for Multi-channel Multi-radio Coded IWMNs

The recent trend in providing wireless access has extended beyond treating wireless networks merely as one-hop networks. Idle nodes may be invited to relay traffic for those nodes sitting far away from an access point/gateway node and possibly suffering from weak signals. This chapter proposes a solution that facilitates such transport from a practical point of view. As a result, a 4-node 802.11b/g testbed is designed and implemented to evaluate the solution. This testbed serves also as a platform for the experimental evaluation of the network-wide solution discussed in Chapter 3.

The key idea of the present solution is to look at opportunistic routing from the network coding perspective. Introduced by [134], opportunistic routing exploits multi-user

diversity by allowing intermediate nodes, who overhear wireless transmissions, to opportunistically forward overheard packets to intended destinations. To realize a throughput benefit over a traditional best-path routing scheme such as Srcr [128], strict link scheduling is required to prevent multiple intermediate nodes from accessing the medium simultaneously. However, Chachulski and Katti [35] showed that this strict link scheduling can be eliminated and that throughput can be increased further, when intra-flow random network coding is used hand in hand with opportunistic routing. That is, for a particular batch of packets, intermediate nodes who overhear transmissions only need to know how many transmissions of coded packets they have to make, but not when. And after having received a sufficient number of coded packets, a destination starts decoding and sends an acknowledgment (ACK) to stop its source and forwarders from transmitting any more coded packets. This scheme is termed MORE [35]. Figure 4.1 shows a simple example of how MORE works. From the figure, after the first two transmissions, the destination overheard only packet $\mathbf{p}_1$ and the repeater heard both. Using network coding, the repeater r does not need to know which packet to forward. It simply sends $\mathbf{p}_1 \oplus \mathbf{p}_2$ to the destination. The destination recovers $\mathbf{p}_2$ by XORing $\mathbf{p}_1 \oplus \mathbf{p}_2$ with the previously received $\mathbf{p}_1$. The MORE protocol sits on top of the 802.11 MAC and below the IP layer and was designed and implemented on an 802.11 testbed to realize a throughput increase over pure opportunistic routing, EoTX [134]. Only intra-flow network coding was considered and multiple concurrent flows may coexist but must be supported independently. Repeaters are also refrained from transmitting their own packets.

In contrast to MORE, we explore in this chapter the benefits of combining both

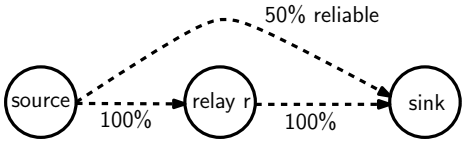


Figure 4.1: Toy example showing how MORE works.

intra-flow and inter-flow network coding by allowing packets to be coded across flows. Data flows are treated jointly to take advantage of idle interfaces/channels. Our work is also related to [131]. But, in contrast to [131], we focus on the implementation aspect of integrating network coding with channel/interface management. Unlike [131], we also exploit opportunistic listening in order to find more useful coding opportunities. We designed and implemented iCORE, the **i**nterface **CO**operation **R**epeater-aided network coding **E**ngine – to realize these important differences. iCORE allows packets to move across interfaces whenever it sees more transmit opportunities. It sits on top of the MAC, thus making it independent from device drivers. We evaluate iCORE on a 4-node chain-like topology testbed and compare it to MORE. Our experimental results show promising avenues for extending iCORE to more general topologies.

We organize this chapter as follows. Section 4.1 provides a motivating example and gives a rough idea of how iCORE works. Section 4.2 highlights the important challenges we face in making iCORE work. The overview of iCORE is given in Section 4.3. The roles of nodes are discussed in Section 4.4. We give the implementation details of iCORE in Section 4.5. Our experimental results and how we set up the experiments follow in Section 4.6. We conclude this chapter in Section 4.7.

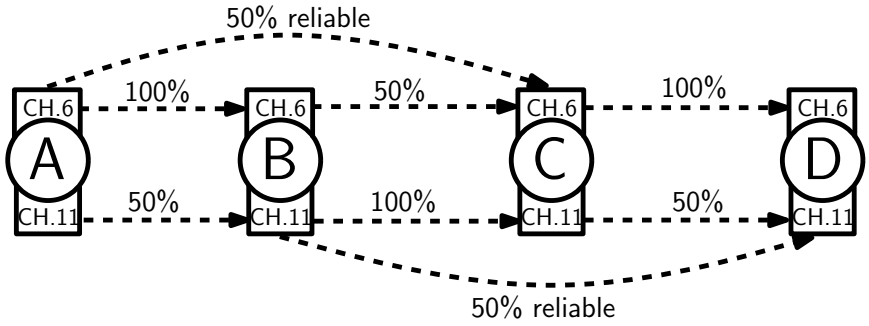


Figure 4.2: Nodes A, B, C and D form a chain topology using one of their wireless interfaces operating on an orthogonal channel. The number on each link represents the percentage of the successful packet delivery on that link. Packets are transmitted in broadcast mode with omni-directional antennae. All nodes are in promiscuous mode. The goal is to deliver Flow 1, containing packets **x** and **y**, from node A to node C, and Flow 2, containing packet **u**, from node B to node D.

4.1 Motivating Example

We show the synergy between interface cooperation and network coding in Figure 4.2 and Table 4.1. For convenience, we represent the interface of node, say, A operating on channel 6 as A.6. Furthermore, let's assume that links are symmetric and that Flow 2 is shorter, in the number of packets, than Flow 1. The goal of the communication in Figure 4.2 is to deliver Flow 1 from A to C, and Flow 2 from B to D. Applying MORE independently to each flow will result in 6 timeslots being used for Flow 1 (4 timeslots for data and 2 for ACK) and 4 timeslots for Flow 2 (2 timeslots for data and another 2 for ACK). The MORE aggregate throughput is thus $(2/6 + 1/4) = 7/12$.

However, if we allow interface cooperation to work with network coding, only 4 timeslots are required to deliver both flows, resulting in the aggregate throughput of 3/4.

This is shown in Table 4.1. In the 1st timeslot, nodes A.6 and B.11 transmit a packet independently. Due to link unreliability, only nodes B.6 and C.11 receive the corresponding packet. The interface cooperation is triggered in the 2nd timeslot when node B.11 linearly combines packet $\mathbf{u}$ with the coded packet, $\mathbf{x}+\mathbf{y}$, previously received on B.6 in the 1st timeslot. Node B.11 is now able to deliver the resulting coded packet, $\mathbf{u}+\mathbf{x}+\mathbf{y}$, to node C.11 because the link between nodes B.11 and C.11 is perfect. At the same time node A.6 also delivers another coded packet, $\mathbf{x}+2\mathbf{y}$, to node C.6 after failing to deliver packet $\mathbf{x}+\mathbf{y}$ in the 1st timeslot due to link unreliability. Thus, after 2 timeslots node C has enough coded packets (3 coded packets) to start the decoding process. Node C acknowledges node A about this in the 3rd timeslot via C.6 when A_1 is piggybacked on packet $\mathbf{u}$. A_1 however does not reach node A.6 until the 4th timeslot due to the link unreliability between nodes C.6 and A.6. A_2 can be sent immediately in the 3rd timeslot on node D.11 once node D receives packet $\mathbf{u}$ from node C.6. The delivery of A_2 is however not successful until the 4th timeslot. The aggregate throughput is therefore 3/4, a 29% increase over MORE.

4.2 Challenges

To realize a throughput gain such as illustrated in Section 4.1, we have to address the following challenges:

1. In MORE, only packets going in the same direction and belonging to the same flow/batch are coded (intra-flow network coding). In iCORE however packets may be coded across flows depending on the transmit opportunities signaled by device drivers. The first question is how do we keep track of packets we code across flows? Further-

Table 4.1: Interface cooperation allows repeater B to inject its own packets as well as help others relay traffic.

	Time Slot			
	1	2	3	4
CH.6				
A [source]	tx: $\mathbf{x}+\mathbf{y}$	tx: $\mathbf{x}+2\mathbf{y}$	–	rx: $\mathbf{u}+\mathsf{A}_1$
B	rx: $\mathbf{x}+\mathbf{y}$	rx: $\mathbf{x}+2\mathbf{y}$	–	–
C [dest.]	–	rx: $\mathbf{x}+2\mathbf{y}$	tx: $\mathbf{u}+\mathsf{A}_1$	tx: $\mathbf{u}+\mathsf{A}_1$
D	–	–	rx: $\mathbf{u}+\mathsf{A}_1$	rx: $\mathbf{u}+\mathsf{A}_1$
CH.11				
B [source]	tx: $\mathbf{u}$	tx: $\mathbf{u}+\mathbf{x}+\mathbf{y}$	–	rx: A_2
C	rx: $\mathbf{u}$	rx: $\mathbf{u}+\mathbf{x}+\mathbf{y}$	–	–
D [dest.]	–	rx: $\mathbf{u}+\mathbf{x}+\mathbf{y}$	tx: A_2	tx: A_2

A_1 = ACK for Flow 1; A_2 = ACK for Flow 2

more, since now packets going in the opposite directions may be coded together, the second question is how do we determine how many transmissions a repeater should make in delivering a packet from source to destination?

2. Piggybacking acknowledgements has two approaches. One is to treat each ACK packet as if it were a data packet, and then code it with other packets heading toward the source waiting for the ACK. The other approach is to implement piggybacking as an extra header attached to every coded packet. Which approach is most suitable for our system?

3. In current IP-based networks each network interface of a node is assigned a particular

IP address. Each network entity identified by other nodes is therefore not the node itself but a network interface attached to that node. Assume a node wants to send packets, generated by its FTP application, to another node. Even if the node has multiple interfaces available for use at once, only one of them can be utilized to transmit the packets because only one link formed by a pair of source and destination IP addresses can be used by any application at a time. The question is how do we shield multiple interfaces/IP's from user applications and make them appear as a single logical interface?

4.3 Overview of iCORE

In iCORE, a node can be either a source, a repeater or a destination at the same time. The role of the source is to keep broadcasting the linear combinations of source packets until it is told to stop. The source knows packet delivery probabilities between nodes and selects certain nodes as its potential repeaters based on these probabilities. All nodes are in promiscuous mode and keep listening to ongoing transmissions. If a node is included in the repeater list of the source, it re-combines a received packet with previously received packets and forwards the combined packet to nexthops. The header of a coded packet is updated as it traverses from node to node. The role of the destination is to keep collecting a sufficient number of coded packets before starting the decoding process. An acknowledgement is then transmitted back to the corresponding source by piggybacking it on another packet that needs to be transmitted by the destination.

To see how interface cooperation works in conjunction with network coding, con-

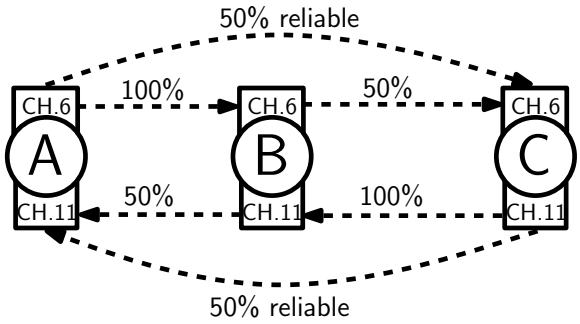


Figure 4.3: Example of how iCORE works: The goal is to deliver Flow 1, containing packets **x** and **y**, from node A to node C, and Flow 2, containing packet **u** and **v**, from node C back to node A.

sider two parallel chain networks shown in Fig.4.3. Each node is represented by a dotted rectangle in which two wireless interfaces represented by circles are embedded. Nodes A, B and C form each chain topology using one of their two wireless interfaces. The number inside each circle represents the operating channel. For convenience we represent the interface of node, say, A operating on channel 6 as A.6. The number on each link represents the packet delivery probability[1] on that link[2]. We assume that the links are symmetric and that packets are transmitted in broadcast mode with omni-directional antennae.

The goal is to deliver Flow 1, containing packets **x** and **y**, from node A.6 to C.6, and Flow 2, containing packets **u** and **v**, from node C.11 to A.11. Using MORE, these two flows will be delivered independently. For each flow, 6 time slots will be required on average, i.e., 4 timeslots for delivering 2 coded packets and 2 timeslots for ACK. If Flow 2 is allowed to start at the 4th timeslot, the total of 9 timeslots will be required to deliver

[1]For example, the packet delivery probability of 0.5 means two time slots are needed to successfully deliver a packet.

[2]We use the ETX (Expected Transmission Count) implementation provided by the Roofnet software [128] to measure the packet delivery probability on each link.

both flows, resulting in the aggregate throughput of 4/9.

Table 4.2: Flows in opposite directions: With interface cooperation, an idle interface can be exploited to help a busier flow.

	Time Slot						
	1	2	3	4	5	6	7
CH.6							
A [source]	tx: $\mathbf{x}+\mathbf{y}$	tx: $\mathbf{x}+2\mathbf{y}$	–	–	rx: $\mathbf{u}+\mathbf{v}$	tx: A_2	tx: A_2
B	rx: $\mathbf{x}+\mathbf{y}$	rx: $\mathbf{x}+2\mathbf{y}$	tx: $3\mathbf{x}+4\mathbf{y}$	tx: $4\mathbf{x}+5\mathbf{y}$	tx: $\mathbf{u}+\mathbf{v}$	rx: A_2	
C [dest.]	–	rx: $\mathbf{x}+2\mathbf{y}$	–	rx: $4\mathbf{x}+5\mathbf{y}$	–	–	rx: A_2
CH.11							
C [source]	–	–	–	tx: $\mathbf{u}+\mathbf{v}$	tx: $\mathbf{u}+3\mathbf{v}+\mathsf{A}_1$	–	rx: A_2
B	–	–	–	rx: $\mathbf{u}+\mathbf{v}$	rx: $\mathbf{u}+3\mathbf{v}+\mathsf{A}_1$	–	
A [dest.]	–	–	–	–	rx: $\mathbf{u}+3\mathbf{v}+\mathsf{A}_1$	tx: A_2	tx: A_2

tx = transmits; rx = receives; A_1 = ACK for Flow 1; A_2 = ACK for Flow 2

However, if the two interfaces on each node are allowed to cooperate, the aggregate throughput can be improved by up to 28% as shown in Table 4.2. From the table, at the end of the 4th time slot, node C.6 has a sufficient number of coded packets to start decoding. At the same time, node B.11 has received a coded packet $\mathbf{u}+\mathbf{v}$ on channel 11. Interface cooperation is then triggered at the 5th timeslot when node B decides to forward $\mathbf{u}+\mathbf{v}$ on channel 6 instead and lets node C tune to channel 11 to acknowledge node A upon the completion of Flow 1. Instead of sending a lone A_1 back to node A, A_1 is piggybacked[3] on a

[3]ACK is not actually coded together with other packets, but implemented as an extra header field.

coded packet of Flow 2 and transmitted by node C.11 (We represent this piggybacking by "...+A_1" in Table 4.2). This transmission ensures a successful delivery to node A.11 because the previous transmission from node C.11 to node A.11 has already failed. With the packet delivery probability of 0.5 on this link, this transmission therefore must be successful. At the same time (i.e., at the 5th timeslot), node B.6 can be used to relay a coded packet ($\mathbf{u}+\mathbf{v}$) for Flow 2. Since the link between nodes B.6 and A.6 are perfect, node A will have two coded packets ($\mathbf{u}+\mathbf{v}$ and $\mathbf{u}+3\mathbf{u}$), which are sufficient to start decoding. Time slots 6 and 7 are used solely by both channels to deliver A_2 for Flow 2. From this example, we see that 7 timeslots are required to deliver both flows. This results in the aggregate throughput of 4/7 which is 28% higher than naively applying MORE to each interface domain. Note that higher gains can be realized if interface cooperation is triggered earlier than the 5th timeslot.

4.4 Roles of Nodes

We explain in detail the roles of each node in this section.

4.4.1 Roles of Source-only Node (Intra-flow Coding)

In iCORE, a source-only node is a node that only generates traffic and does not relay traffic for others. It codes packets only within the same flow. A data file to be transmitted in a flow is first divided into batches of K equal-length packets – denoted as $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_K$. Each packet is a vector of symbols. Each symbol is one of the elements in the finite field $\mathbf{GF}(2^8)$. When a transmission opportunity at the MAC arises, the source-only

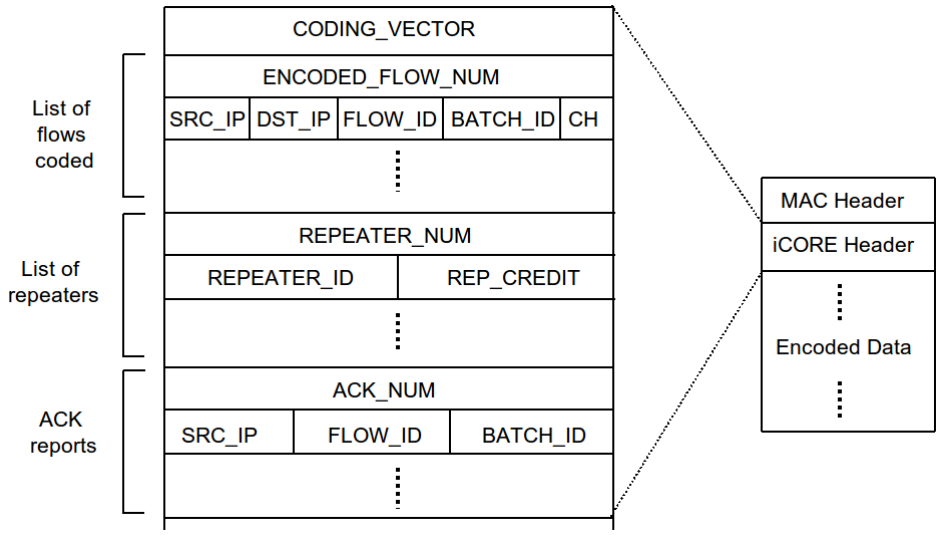


Figure 4.4: iCORE header.

node linearly combines the K packets in the current batch, i.e., $\mathbf{x}_j = \sum_{i=1}^{K} c_{ij}\mathbf{p}_i$, where $\mathbf{x}_j$ is a coded packet and c_{ij} is a coefficient randomly draw from $\mathbf{GF}(2^8)$. The role of the source-only node is to keep broadcasting such linearly coded packets in a current batch until it is notified to stop by an ACK piggybacked on one of the packets it overhears. The source-only node proceeds to transmit the next batch after the reception of the current batch is acknowledged or stops if it reaches the end of the file.

Each iCORE packet has a variable-length header[4] as shown in Fig. 4.4. The vector $\mathbf{c}_j = \{c_{ij} : i = 1, 2, ..., K\}$ is a coding vector and must be added to the header of the packet. A flow ID, a batch ID, source and destination IP addresses and a channel to use, must also be added. The list of repeaters that will be used to relay the packet is next added. Repeaters in the list are ordered according to their transmit credits (described in Section 4.5) calculated based on how reliably the repeaters can help deliver a packet from

[4]The exact length of the header of each packet depends on how many flows are coded together as well as on the batch size. In our case, the overhead varies between 4%–10% of the payload.

source to destination. Finally, if the source has pending ACKs to deliver, it adds the ACK reports to the header.

4.4.2 Roles of Repeater (Inter-flow Coding)

iCORE allows repeaters (i.e., all other nodes besides source-only nodes) to generate traffic as well as relay traffic for others. If a repeater has its own packets to transmit, it codes them with the innovative coded packets received from other flows. To see this, assume that the repeater has received two innovative packets from Flow 1: $\mathbf{x}_1 = \sum_{i=1}^{K_1} c_{i1}\mathbf{p}_i$, and $\mathbf{x}_2 = \sum_{i=1}^{K_1} c_{i2}\mathbf{p}_i$, where K_1 is the batch size of Flow 1. Let $\mathbf{p'}_i$ be a native packet of the repeater and K_2 its batch size. A new coded packet is formed by

$$\begin{aligned} \mathbf{y}_j &= \sum_{i=1}^{K_2} b_{ij}\mathbf{p'}_i + b_{(K_2+1)j}\mathbf{x}_1 + b_{(K_2+2)j}\mathbf{x}_2 \\ &= \sum_{i=1}^{K_2} b_{ij}\mathbf{p'}_i + \sum_{l=1}^{K_1}\left(\sum_{i=K_2+1}^{K_2+2} b_{ij}c_{l(i-K_2)}\right)\mathbf{p}_i \end{aligned} \tag{4.1}$$

which is basically a linear combination of the native packets, i.e., $\mathbf{p}_i$ and $\mathbf{p'}_i$. Before sending $\mathbf{y}_j$, its header must be updated, i.e., the header fields related to a coding vector, the list of flows coded together, and the list of repeaters.

The number of transmissions a repeater is required to make depends on how reliable it is, as viewed by the source, in relaying a packet to the destination (see Section 4.5.2). Each repeater reduces its transmit credits each time it transmits. Which interface and channel to transmit depends on transmit opportunities signaled by our bonding device (see Section 4.5.4).

When a packet is received, the repeater checks if it is included in the list of repeaters. If yes, it checks if the packet is outdated. It does this by comparing the flow

and batch IDs in the packet's header with those which the repeater is currently serving. If at least one flow/batch ID matches, the packet is regarded as useful. The repeater next updates its transmit credits by checking if it is closer, in an ETX sense, to the majority of the destinations than the previous hop. If it is closer, its transmit credits are increased, meaning that it should become more participative in forwarding packets. The repeater still needs to check if the packet is innovative, i.e., if the packet is linearly independent with the ones received previously. If so the packet will be stored and pre-encoded with other innovative packets before forwarding. All of the non-innovative packets will be discarded.

4.4.3 Roles of Destination (Decoding)

Decoding is performed at the destination. The role of the destination is to collect at least $N = K_1 + K_2 + ... + K_k$ innovative packets, assuming that k flows are coded together. The destination decodes $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N$ by solving (4.2).

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,N} \\ c_{2,1} & \cdots & c_{2,N} \\ \vdots & \ddots & \vdots \\ c_{N,1} & \cdots & c_{N,N} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_N \end{pmatrix}. \tag{4.2}$$

After decoding, the destination extracts only the batch, say, $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{K_1}$, meant for it and forwards the remaining decoded batches to the output queue. If the destination also takes a repeater or source role, it piggybacks an ACK for this batch on a new coded packet whenever the first transmit opportunity signaled by the MAC arises.

4.5 Implementation Details

In this section, we give the details of our implementation.

4.5.1 Packet Format

As shown in Fig. 4.4, iCORE adds a variable-length header to each packet. iCORE's header sits between the MAC header and the encoded data and consists of the following four portions.

- **Coding vector**: This portion contains a vector of random coefficients used to form a coded packet. These coefficients are picked locally and uniformly at random by sources and repeaters.

- **List of flows coded together**: This portion starts with the number of flows coded together in the packet. Each item of the list specifies the source and corresponding destination IP addresses, a flow ID, a batch ID, and a channel used to carry this flow in the previous hop.

- **List of repeaters**: This portion starts with the number of potential repeaters. Each item of the list specifies a repeater's ID and its transmit credit computed according to its proximity to the sources.

- **ACK reports**: The ACK reports start with the number of ACKs piggybacked on the packet. Each report entry tells the recipients of the packet about the transmission completion of a particular batch of a particular flow whose origin is specified by the source IP address field. Repeaters that overhear the packet with ACK reports

matching all of its currently serving batches flush the packets belonging to those batches.

4.5.2 Transmit Credits

A transmit credit, TX-credit, is used by each repeater. It is defined as the number of transmissions that a node should make for every packet it receives from a node farther from the destination in the ETX metric [35]. The transmit credit for node i is given by

$$\text{TX-credit}_i = \frac{z_i}{\sum_{j>i}(1-\epsilon_{ji})z_j} \tag{4.3}$$

where z_j is the number of transmissions made by node j and ϵ_{ji} is the link loss probability from node j to node i. The inequality $j > i$ denotes that node j is farther from the destination than node i, or equivalently, node j has a higher ETX than i.

Generally speaking, a transmit credit determines how reliable each repeater is in relaying a packet to the destination, and more reliable repeaters should relay more traffic. We leverage the MORE approach to calculating the transmit credit for each repeater but modify it to fit our framework. In MORE, transmit credits are calculated by the source for a uni-directional flow. As a packet traverses the network, the transmit credits are incremented and decremented by the repeaters via a counter called CreditCounter. In iCORE, however, the repeaters forward coded packets in an omni-directional fashion. Transmit credits thus must be adjusted accordingly by each repeater. The way the source calculates TX-credits for its potential repeaters for each flow is however the same as MORE. Since a coded packet in iCORE may be made up of coded packets from different flows, the resulting coded packet will have the new list of repeaters as the union of the repeaters currently serving those flows.

A new TX-credit for each new repeater is then the sum of the TX-credits previously assigned to those flows. Generally speaking, if a repeater is chosen to relay traffic for more flows, it should be given more TX-credits for doing so.

4.5.3 Node State

Several parameters must be maintained to keep track of the state of a node. CentralBatchQ stores the received innovative packets. The number of innovative packets is bounded by the sum of the batch sizes of the flows coded together (N). FlowBatchID maintains a 2-by-X variable array identifying the most recent batch IDs and their corresponding flow IDs. CreditCounter maintains a transmit credit for each repeater. Finally, RepeaterList keeps track of the list of repeaters and their corresponding transmission credits.

4.5.4 Interface Cooperation

Interface cooperation is handled by our channel and interface coordinator module shown in Fig. 4.5. This module communicates with both iCORE and the bonding device. Interface bonding allows the aggregation of multiple network interfaces into a single logical one such that user applications are oblivious to multiple physical interfaces or IP addresses. Interface bonding provides *features* for throughput aggregation, load balancing and fault-tolerance. Based on [143], we implement a new bonding feature aimed specifically for interface cooperation. It sits on top of the link layer, thus making it independent from the underlying device drivers. Using the channel and interface coordinator module, the node learns about the channels/interfaces used by its neighbors through the iCORE header of a packet and updates the entries of the lookup table through IOCTL calls to the bonding

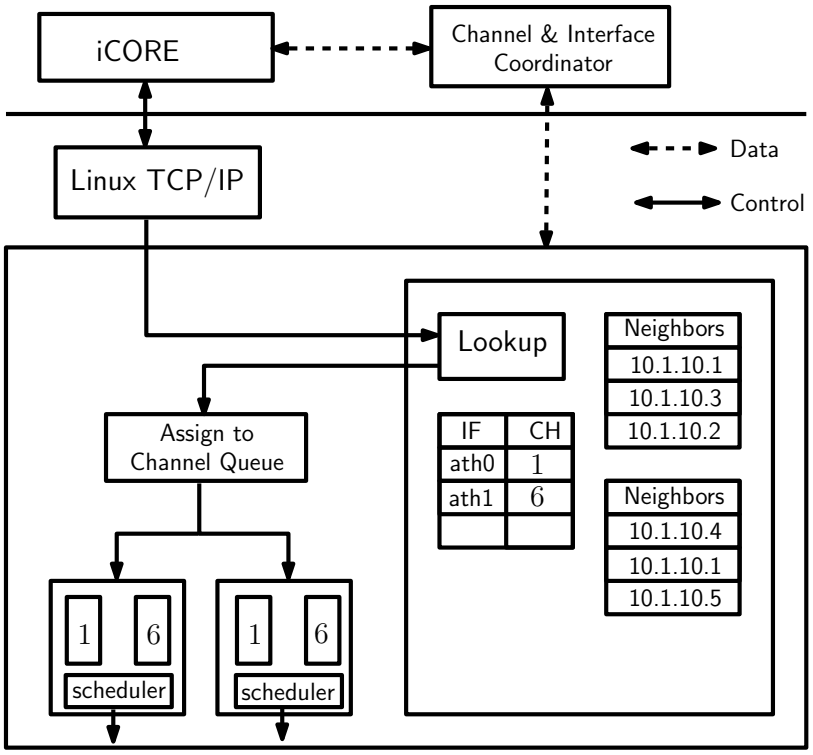


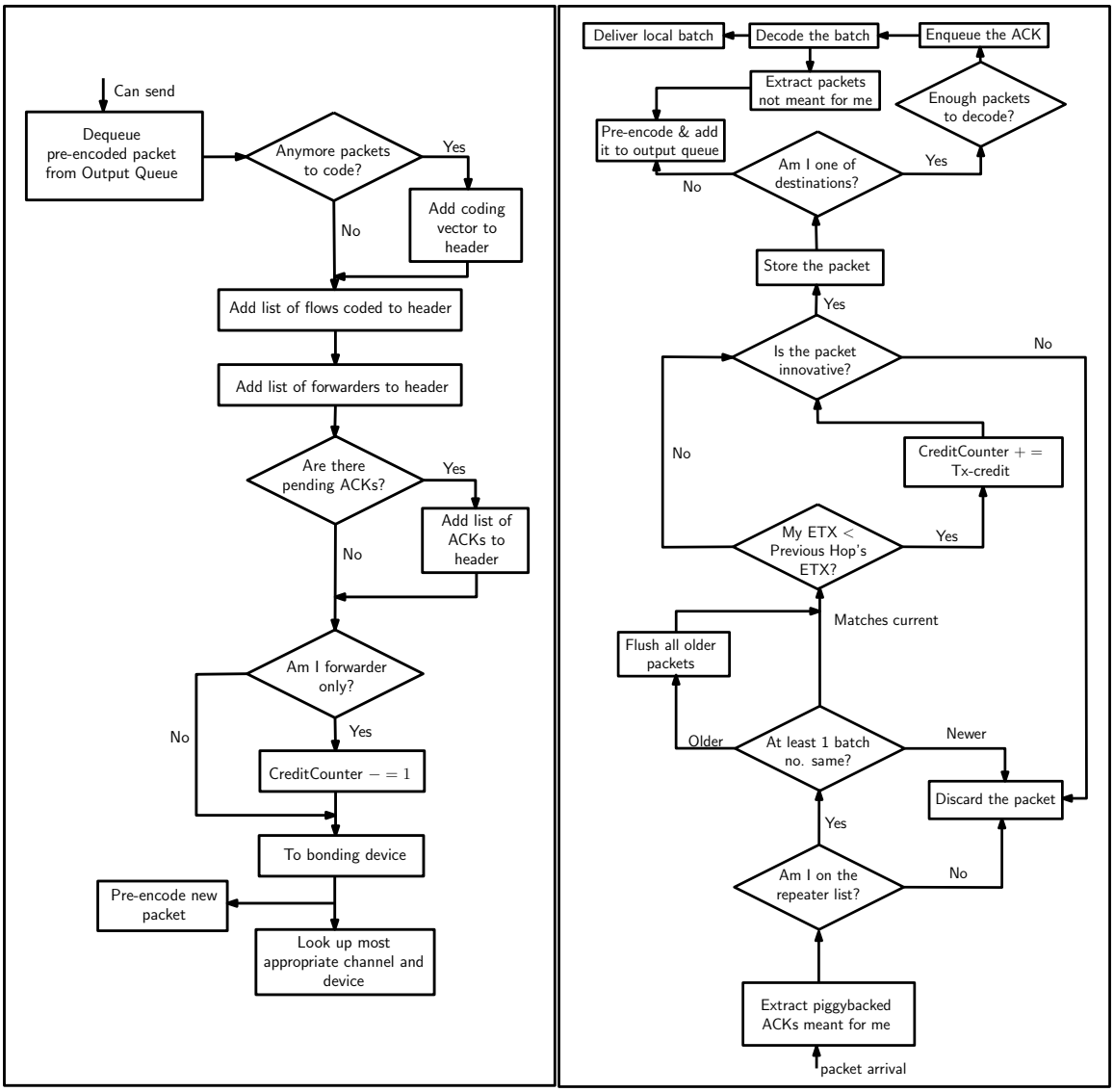
Figure 4.5: Interface cooperation in iCORE is done via our channel and interface coordinator.

device. The role of the lookup table is to allocate the most appropriate channel and physical wireless device to the packet. The most appropriate channel here means the channel to which most repeaters in the list of repeaters are listening. And the most appropriate device is the wireless physical interface currently attached to that channel.

4.5.5 Control Flow

We explain how a node operates as a sender and a receiver in this section.

Sending Process: A sending process shown in Fig. 4.6(a) starts when the bonding device driver signals a transmit opportunity at which time the sending node dequeues a pre-encoded



(a) Sending side (b) Receiving side

Figure 4.6: Control flow of iCORE.

packet (packets are pre-encoded while waiting for transmit opportunities.) The node first checks if it has any source packets to code together with the current pre-coded packet. If yes, a coding vector field is updated accordingly in the header. The list of flows and the list of repeaters as described in Section 4.5.1 are next added to the header. The node then checks if there are any pending ACKs to be added to the header. It also checks if it is acting as a repeater only or not. Afterward, the packet is sent to the bonding device (as shown in Fig. 4.5) in which the lookup module resides. The role of the lookup module is to allocate the most appropriate channel and physical wireless device to the packet. The most appropriate channel means the channel to which most repeaters (in the list of repeaters) are listening. And the most appropriate device is the wireless physical device currently attached to that channel. The entry of the lookup table can be updated through IOCTL calls from our channel and interface coordinator component running as a userspace daemon as shown in Fig. 4.5.

Receiving Process: A receiving process is shown in Fig. 4.6(b). When a packet arrives, the receiving node extracts any ACK reports meant for it. It then checks if it is included in the list of repeaters. If it is, the node checks if at least one of the batch IDs of the corresponding flows in the header matches any of its currently serving batches and flows. If the batch IDs of the corresponding flows are all older, the stored packets belonging to those batch/flow IDs are flushed out of the output queue. The node next checks if it is closer to the majority of the destinations of the flows coded together. If it is, more credits are given to the node. Linear independence check is performed afterward to screen out a non-innovative packet. An innovative packet is stored for future use. The node then checks

if it is one of the destinations in the list of flows coded together. If yes, the node checks if it has enough coded packets to start decoding. Decoding starts after an ACK is put in a temporary queue. After decoding, foreign packets are extracted from the decoded batch and put in the output queue. Finally, the local batch is delivered to upper layers.

4.5.6 Piggybacking Acknowledgments

In contrast to MORE, iCORE does not explicitly acknowledge each flow. However, iCORE piggybacks any pending ACKs on a coded packet and transmits the coded packet via normal broadcast transmissions. Upon receiving the packet, the node extracts any ACK reports destined to it and updates its transmission and batch ID accordingly. Repeaters that hear the packet with ACK reports that match all of its currently serving batches flush the packets belonging to those batches.

4.6 Performance Evaluation of iCORE

4.6.1 Implementation Testbed

We implement a four-node 802.11b/g testbed to evaluate iCORE. Each node in the testbed is a net4521 Soekris board [148] equipped with two 802.11b/g wireless interfaces (Dlink-WNA2330 [150]). Each interface is connected to an omni-directional antenna and set to ad-hoc mode without RTS/CTS capability. We set the transmit power of all the wireless interfaces to 10 dBm. The bitrate of each interface is set at 11 Mb/s. All transmissions are in broadcast mode. Debian-based Linux is used as the operating system for each Soekris board. Using the userlevel Click framework [126], we implement iCORE as a userspace

Figure 4.7: Our testbed node: A net4521 Soekris board [148] with two IEEE 802.11b/g Dlink-WNA2330 interfaces [150].

daemon. iCORE sends and receives raw 802.11 frames from the bonding interface using a libcap-like interface [151]. For a source node, UDP traffic is generated by a Click utility program udpgen.

4.6.2 Performance Metrics

We evaluate the following performance metrics:

- **Throughput** is referred to as a per-flow throughput as seen by a user application.
- **Throughput Gain** is the ratio of iCORE's throughput and MORE's throughput.
- **Packet delivery ratio** is defined as the ratio between the number of successfully delivered packets and the number of packets transmitted in a flow.

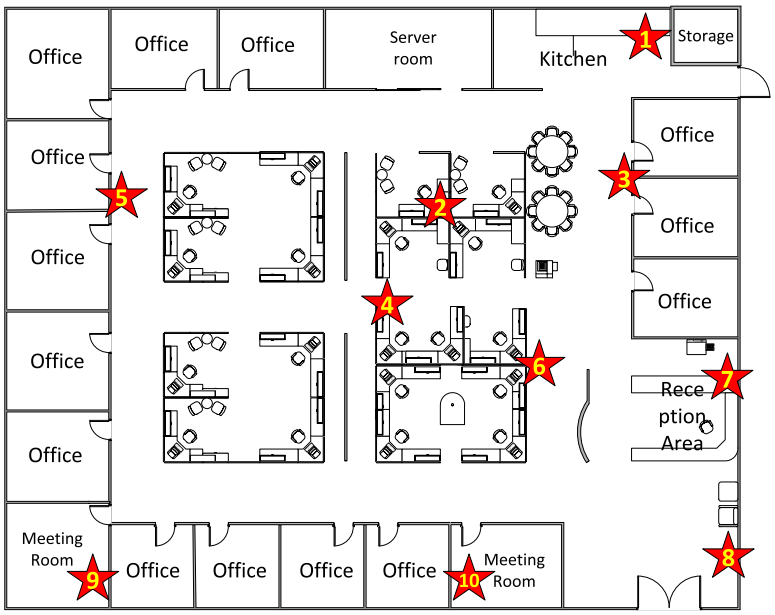


Figure 4.8: Possible node locations on our lab floor plan.

4.6.3 Experimental Setup

The possible locations where we place the four nodes are distributed across our 40m×40m office floor, as shown in Fig. 4.8. These locations are constrained by office privacy and the availability of the LAN outlets through which we send control commands (e.g., ssh and netcat) and collect experimental data. We place the testbed nodes on four of these possible locations such that they form a chain-like topology. The paths between nodes are therefore between 1 to 3 hops with the link loss rates varying between 0 and 10%. Each topology is represented by a sequence of 4 numbers indicating the 4 locations labeled in Fig. 4.8. Ten topologies are created, i.e, (1,3,4,9), (1,3,6,9), (1,3,6,10), (1,2,6,10), (1,2,4,9), (1,3,7,8), (8,6,4,5), (8,6,2,5), (7,6,4,5) and (1,3,6,8). The order in which a node appears in each topology does not designate it as a transmitter, receiver or repeater. A node can have more than one role depending on data flows to be set up.

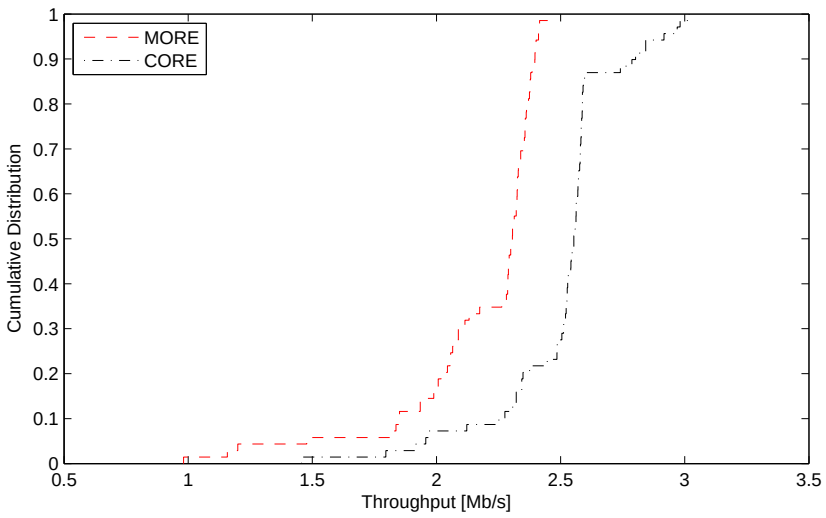


Figure 4.9: CDF of throughput for iCORE and MORE.

Five experiment runs per topology are carried out so that the number of samples used to plot the cumulative distribution functions (CDFs) is 50. Each run sets up six concurrent flows between nodes. The direction of each flow is selected randomly. In each run, MORE is executed after iCORE, and the throughput is calculated by averaging over all the flows. Each flow transfers a data file whose size varies between 4 to 12 MB. A packet size is 1500 bytes. Otherwise specified, we fix the batch size at $K = 8$ for all flows and use the ETX implementation provided by the Roofnet software [128] to measure the packet delivery probability for each link. The measurements for packet delivery probability are used by both iCORE and MORE to calculate TX-credits.

4.6.4 Experimental Results

Fig. 4.9 shows the CDFs of the throughputs for iCORE and MORE. The figure shows that more than 80% of the time the throughput of iCORE is higher than 2 Mb/s

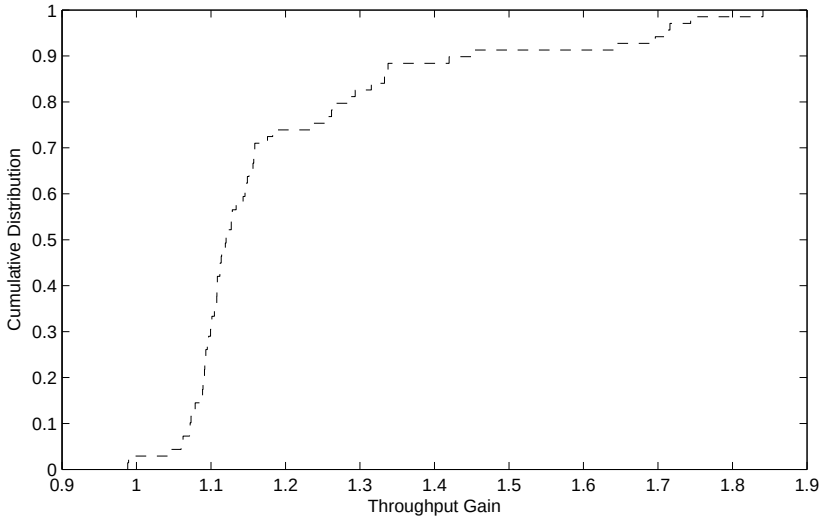


Figure 4.10: Throughput gain with iCORE.

while the throughput of MORE is higher than 1.75 Mb/s. We plot this throughput gains in Fig. 4.10 where the throughput gains of up to 80% are observed. Most gains however fall in the range of 10-20%. The increased gains result from the fact that packets captured from opportunistic listening in iCORE are now shared among the interfaces. These interfaces thus see a central pool of the packets, in which they select and code only those packets that will potentially benefit as many nodes as possible when transmitted on a particular channel. This eventually contributes to the system-wide throughput benefits.

We next compare iCORE and MORE when offered load is gradually increased. Fig. 4.11 shows the scatter plot of the offered load versus the delivered throughputs for both MORE and iCORE. Points on the 45-degree line represent perfect throughput deliveries. Interestingly, we see that while the MORE throughputs fluctuate as offered load increases, iCORE is able to sustain its throughputs in an almost linear fashion as the load ramps up. The reason is that while MORE operates on each flow independently, iCORE lets the flows

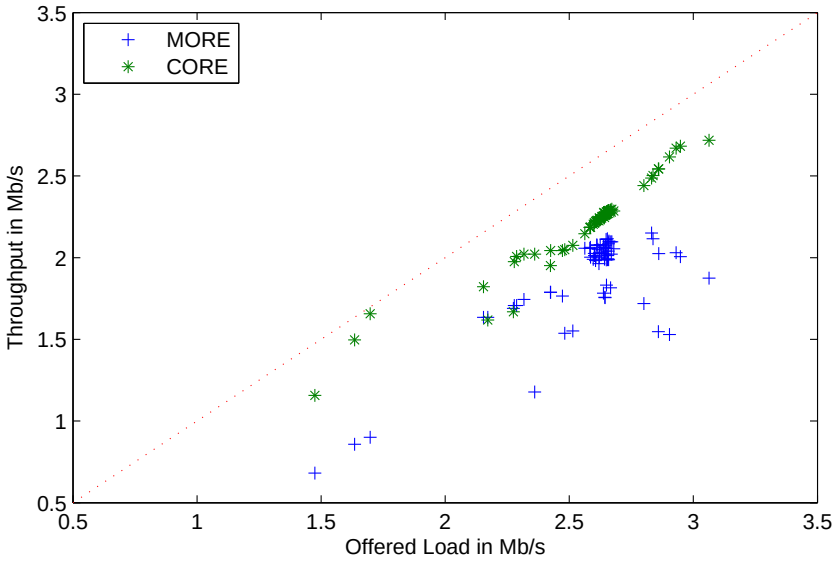


Figure 4.11: Sustaining throughputs: iCORE is able to sustain the throughputs as offered load increases.

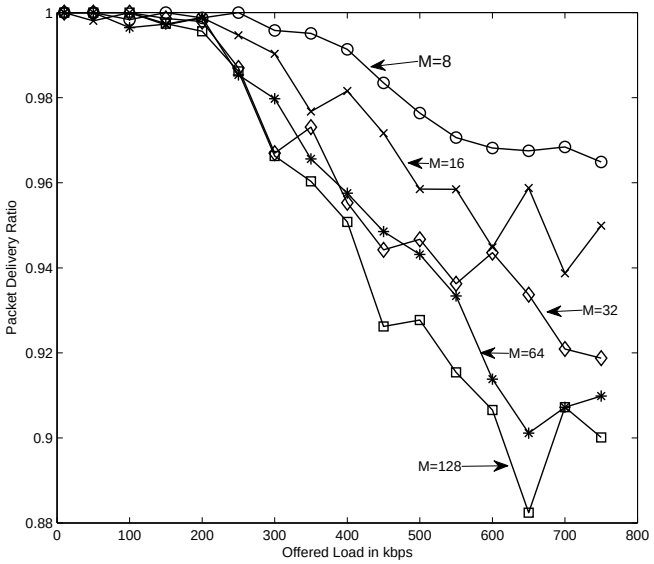
know the presence of the others and tries to exploit any idle interface/channel and help others transmit. MORE however treats all flows independently. That is, as shorter flows finish and their interfaces become idle, slower or longer flows cannot sneak in and utilize those interfaces.

In Fig. 4.12 we show the packet delivery ratios for both iCORE and MORE as a batch size K is varied in each scheme. Each data point is averaged over 5 experiment runs per topology. Each run consists of six concurrent unicast flows whose averaged offered load is shown on the x-axis. Only five topologies are considered. One can see that as the batch size increases, the packet delivery ratio of iCORE drops significantly, whereas that of MORE slightly drops. The reason for this is that as packets are coded across flows the resulting batch size (i.e., parameter N described in Section 4.4.3) becomes larger, the

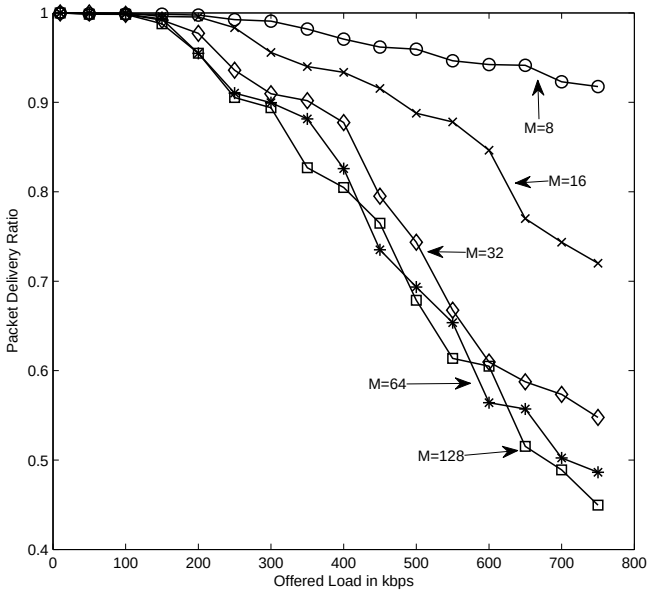
destination will need to collect more packets before it can start decoding. This results in a significant drop of the packet delivery ratio in iCORE. On the other hand, since MORE treats each flow independently, the batch size is always fixed at a given value. This is the price iCORE needs to pay in combining both inter- and intra-flow network coding.

4.7 Conclusion

To reveal the synergy between network coding, opportunistic routing and channel/interface management, we have designed and implemented iCORE – the interface COoperation Repeater aided network coding Engine. iCORE allows packets to move across interfaces and be coded across flows whenever transmit opportunities arise. We have evaluated iCORE on a 4-node chain-like topology testbed and compared it to MORE – the state-of-art intra-flow network encoding implementation. Our experimental results have revealed a promising avenue for applying iCORE to more general topologies. Scalability is the key aspect to look at. We believe that making this application a reality is a path worth exploring.



(a) Packet delivery ratio of MORE.



(b) Packet delivery ratio of iCORE.

Figure 4.12: Packet delivery ratio as a batch size is varied. One can see that iCORE is very sensitive to a batch size.

Part II

AP/Gateway-oriented Solutions:

Resource Allocation/Error Control

Chapter 5

Dynamic Buffer Allocation and Opportunistic Network Coding for Unicast Exchange

In this chapter, we will look at the impact of network coding on the link-level queueing performance of unicast exchange at an AP/gateway of the underlying IWMN. Queueing arises naturally in this context due to random packet arrival and departure at intermediate nodes where packets are coded. In the literature, various approaches have been taken to quantify the benefits of network coding from the queueing perspective. In [27], the throughput benefit was experimentally quantified for wireless mesh networks where intermediate nodes opportunistically code the packets based on the buffer contents of next-hop receivers. A special case of such opportunistic network coding (ONC) was further studied in the context of a three-node relaying network (similar to an IEEE 802.16j system)

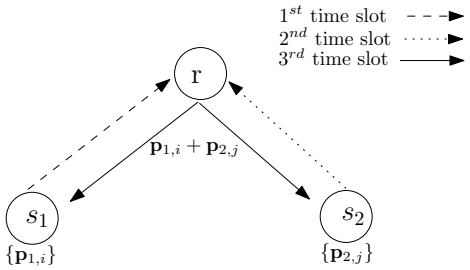


Figure 5.1: Unicast exchange at an AP/Gateway node facilitating by network coding.

in [23]. Using continuous time queueing analysis, [23] quantified the delay improvement of inter-session network coding in the presence of random packet arrival and departure at the relay. The effect of queueing at sources on network coding gain was however not considered. This effect was later studied in [102] for the same network topology and network coding strategy with the "infinite buffer" assumption. In [103], a three-node relaying topology in ad-hoc mode was also studied from an energy-delay trade-off perspective. Albeit with the "finite buffer" assumption, factors such as transmission delay and wireless channel impairment were not taken into account. Such channel impairment was earlier considered in [104] for a two-node communication using intra-session network coding, where the delay improvement of network coding was shown to be minimal. In [105], dynamic scheduling based on buffer-state information was studied in a lossy network, and the benefit of joint intra- and inter-session network coding was quantified from the throughput perspective. In [24], a wireline butterfly network – which can be thought of as a general case of a three-node relaying network – was first studied with the fluid flow network coding assumption and infinite buffer capacities. Achievable throughput regions were identified.

We are particularly interested in a wireless unicast exchange scenario shown in Fig. 5.1 as this scenario arises naturally in applications such as WiFi telephony where

two wireless nodes exchange their delay-sensitive packets through an AP/gateway. In this chapter, we propose a discrete-time Markov queueing model for such a network where an ONC scheduling protocol [27] [23] [28] [29] is employed at the AP/gateway. Unlike the previous studies, which implicitly assume two or more *static* AP/gateway buffers, one for each flow, we propose to use dynamic buffer allocation at the AP/gateway. Such dynamic buffer allocation in our context operates much like two water tanks positioned side by side and jointed at the bottom by a static valve (Fig. 5.2a). Buffer capacity is thus dynamically shared by two packet flows as packets move in and out of their respective buffers (tanks). Furthermore, the proposed model also captures the effect of (i) source queueing, (ii) wireless channel impairment and (iii) random packetized traffic arrival patterns. Motivated by an increasing demand for real-time applications, we are interested in the delay benefit of network coding in a unicast exchange scenario. To this end, the delay performance of the system is derived and compared with that of classical first-in first-out (FIFO) and round-robin (RRB) scheduling. Based on one observation made during such comparison, we propose a simple scheduling algorithm that employs a technique, called *buffer equalization*, and joint intra- and inter-session network coding. Our proposed scheme is shown to perform better than the original ONC model in terms of delay.

We organize the rest of this chapter as follows. Section 5.1 describes the system model and states the primary assumptions. Section 5.2 gives the queueing model which is formulated based on a discrete time Markov chain. From the stationary probability vector obtained from solving the Markov chain, we derive in Section 5.3 an average packet delivery delay which shall be used to quantify the benefit of network coding in Section 5.4. Section

5.5 describes the proposed scheduling scheme. The conclusion is stated in Section 5.6.

5.1 System Model and Assumptions

We consider a WLAN comprising two independent wireless sources s_1 and s_2, and an AP/gateway r. The goal is to have s_1 and s_2 exchange their randomly generated, fixed-size packets via the AP/gateway, where transmissions are subject to packet erasure. As such, the AP/gateway becomes an unavoidable transmission bottleneck. Using classical packet scheduling strategies (e.g., FIFO or RRB) at the AP/gateway, the above unicast exchange communication of two packets – one from each source – can be achieved in four transmissions: $1^{st} : s_1 \stackrel{p_1}{\rightarrow} r$, $2^{nd} : r \stackrel{p_1}{\rightarrow} s_2$, $3^{rd} : s_2 \stackrel{p_2}{\rightarrow} r$, and $4^{th} : r \stackrel{p_2}{\rightarrow} s_1$. Here, we denote the n^{th} transmission of packet p from node x to node y by $n^{th} : x \stackrel{p}{\rightarrow} y$. With network coding, however, the number of transmissions reduce to three as follows: $1^{st} : s_1 \stackrel{p_1}{\rightarrow} r$, $2^{nd} : s_2 \stackrel{p_2}{\rightarrow} r$, and $3^{rd} : r \stackrel{p_1 \oplus p2}{\rightarrow} \{s_1, s_2\}$; where $\oplus$ denotes an exclusive OR (XOR) operator, and the n^{th} broadcast transmission of packet p from node x to nodes y and z is denoted by $n^{th} : x \stackrel{p}{\rightarrow} \{y, z\}$. Packet p_2 (resp. p_1) is recovered at s_1 (resp. s_2) by $p_2 = p \oplus p_1$ (resp. $p_1 = p \oplus p_2$).

This is an ideal scenario. In presence of random packet arrival and departure, such coding at the AP/gateway might not be applicable. Based on [23,27], we let the AP/gateway opportunistically code or simply forward packets to their destinations. In particular, assume that two virtual[1] buffers are maintained by the AP/gateway – one holding packets coming from each source. The AP/gateway makes a forwarding decision opportunistically based

[1]The reason we use the word "virtual" will be discussed later.

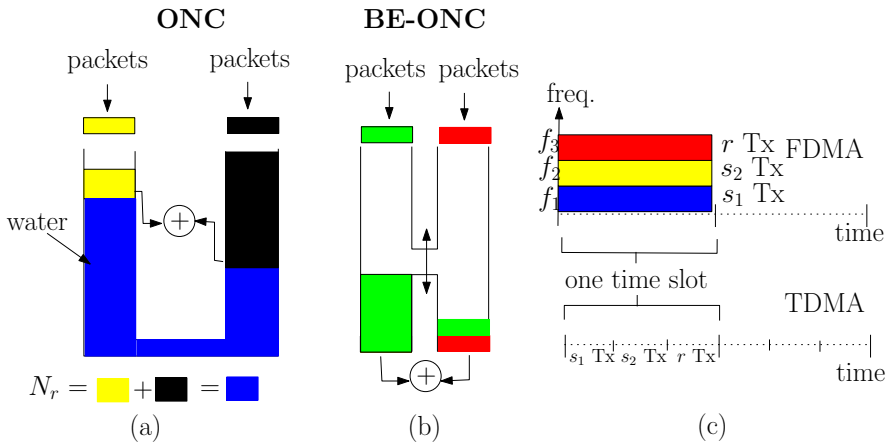


Figure 5.2: (a) Buffer dynamics of our ONC model can be viewed as two water tanks with a static open value at the bottom. When a packet arrives, it pushes the water to the other buffer through the valve. The water however must not ever spill from the two tanks. (b) Buffer dynamics of BE-ONC is viewed on the other hand as two empty tanks, where the valve now is allowed to slide up and down to move liquid (packets) between two tanks. (c) Our notion of "time slot."

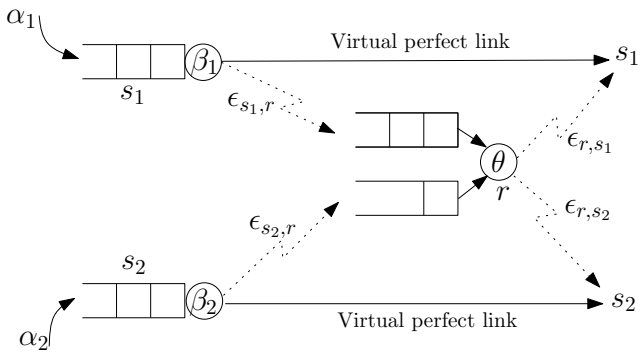


Figure 5.3: Expanded view of the queueing model of a wireless unicast exchange communication with opportunistic network coding and dynamic buffer allocation. While the dashed lines represent lossy wireless links with certain packet loss probabilities, the solid lines stand for virtual communication links with no packet error (i.e., each source is viewed as connected to itself).

on the contents of its two buffers. That is, (i) if both buffers are empty, the AP/gateway stays idle; (ii) if both buffers are not empty, it codes two packets into a single packet and transmits it whenever a transmission opportunity arises; or (iii) if only one of the buffers is empty, it transmits a head-of-line (HOL) packet from the non-empty buffer. We assume that coding is performed in a binary field, i.e., using XOR.

For comparison purposes, we also consider classical scheduling schemes such as FIFO and RRB in which the AP/gateway selects a packet for transmission in a predetermined manner, one at a time. In FIFO, we merge two virtual buffers into one and let the AP/gateway always transmit a HOL packet. In RRB, we still maintain two virtual buffers and let the AP/gateway sequence through these two buffers. In the following, we state the assumptions of our queueing model.

5.1.1 Queueing Assumptions

The queueing model of a wireless unicast exchange network is shown in Fig. 5.3. A finite buffer of capacity N_1 (resp. N_2) is maintained by wireless source s_1 (resp. s_2) to store backlogged packets, and two virtual, parallel, finite buffers are maintained by the AP/gateway – each virtual buffer accommodating packets received from each source. By "virtual" we mean that the combined capacity of the two virtual buffers at the AP/gateway is exactly N_r. Both packet flows from s_1 and s_2 share the AP/gateway buffer accommodation such that the total number of backlogged packets at the AP/gateway never exceeds N_r. This is the unique feature of our model which lets the AP/gateway to dynamically accommodate two incoming packet flows which could possibly have different arrival rates to the AP/gateway. We give a "water tank" analogy of our AP/gateway buffer model in Fig. 5.2(a) where each tank corresponds to each virtual buffer, and the amount of water corresponds to the combined buffer capacity N_r. When a packet comes into, say, tank 1, it pushes the water to tank 2 through a static valve, hence, reducing the capacity that can be used to accommodate packets coming into tank 2. Two packets – one from each flow – closest to the water shall be coded and transmitted.

We assume that packet arrivals and departures occur randomly at each node. In particular, packets enter sources s_1 and s_2 according to two independent Bernoulli processes with arrival probabilities α_1 and α_2, respectively. Each node is modeled as having one server which holds a packet currently being transmitted. We assume that packets are erased independently across receivers and time slots. Receivers are assumed to be able to perfectly detect erasures and feed this information back to corresponding transmitters via perfect

zero-delay feedback channels. A packet is retransmitted if it fails to reach its receiver. In this chapter we assume that each transmitter has an infinitely persistent automatic repeat request (ARQ) protocol implemented. A packet is released from its server only after it is successfully received by its receiver(s).

We define service time as the time a packet spends on its server. We characterize service time of each packet as follows. Let time be divided into equally spaced slots, and let packet arrival and departure events occur at the end of each time slot (Our notion of "time slot" is purely from a queueing perspective, not the medium access control (MAC) one. We show in Fig. 5.2c how this notion can be interpreted for some representative multiple access classes.) Without loss of generality, consider a packet transmission at AP/gateway r with outgoing links $l_1 = (r, s_1)$ and $l_2 = (r, s_2)$ to receivers s_1 and s_2, respectively. Service time of a packet at r then can be modeled as a discrete random variable $t = \max(t_1, t_2)$, where t_1 (resp. t_2), is geometrically distributed with parameter $(1 - \epsilon_{r,s_1})$ (resp. $(1 - \epsilon_{r,s_2})$) which is a probability that a packet is successfully transmitted on link l_1 (resp. l_2) in a time slot. Assuming t_1 and t_2 are independent, it follows from the extreme value theory [106] that

$$\Pr\{t \leq k\} = \Pr\{t_1 \leq k\} \Pr\{t_2 \leq k\} = (1 - \epsilon_{r,s_1}^k)(1 - \epsilon_{r,s_2}^k) \tag{5.1}$$

for $k \in \{1, 2, \dots\}$. Therefore, the probability that a packet is received without error by both s_1 and s_2 in a time slot is

$$\theta = 1 - \Pr\{t > 1\} = (1 - \epsilon_{r,s_1})(1 - \epsilon_{r,s_2}). \tag{5.2}$$

Service parameters β_1 and β_2 for sources s_1 and s_2, respectively, can be obtained in a similar manner.

Define a tuple $\psi = (\epsilon_{s_1,r}, \epsilon_{s_2,r}, \epsilon_{r,s_1}, \epsilon_{r,s_2})$ as an instance of channel realizations of a wireless unicast exchange network. Based on a discrete-time Markov chain, in the following section, we present a queueing model of our system for any instance of channel realizations.

5.2 Discrete Time Markov Chain Model

In this section, we formulate a discrete-time Markov chain (DTMC) to model the queueing behavior of a wireless unicast exchange network with opportunistic network coding and dynamic buffer allocation capabilities (Fig. 5.3).

Let time be divided into equally spaced time slots. Define the state space of the DTMC as $\{(n_1(j), n_2(j), v_1(j),\ v_2(j))\}$, where $n_i(j)$, $i \in \{1, 2\}$, represents the number of packets waiting in source s_i's buffer, plus the one in service, and $v_i(j), i \in \{1, 2\}$, represents the number of packets waiting in the ith virtual buffer of the AP/gateway, plus the one in service – all observed at the end of time slot j. A transition of the system state occurs due to the packet arrival or departure (i.e., successfully transmitted). We show the transition probability matrix (TPM) P of the DTMC in (5.3). Each inner square matrix $\mathbf{i} \in \{\mathbf{B}, \mathbf{C}, \mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_{1b}, \mathbf{A}_2\}$ represents the transition probabilities of going from one level to another, where *level* refers to the value of n_1. Similarly, transitions within each inner matrix $\mathbf{i}$ are characterized by the value of n_2. The total number of states is $(N_1 + 2)(N_2 + 2)(\sum_{i=0}^{N_r+1} N_r + 2 - i)$. The sum arises here because of the dynamic buffer allocation captured in our model. The dimension of each inner matrix $\mathbf{i}$ is given by $(N_2 + 2)(\sum_{i=0}^{N_r+1} N_r + 2 - i)$. Note that the boundary conditions are captured in $\mathbf{A}_{1b}$. We show the details of each inner matrix in Appendix A. The idea on how such a DTMC can

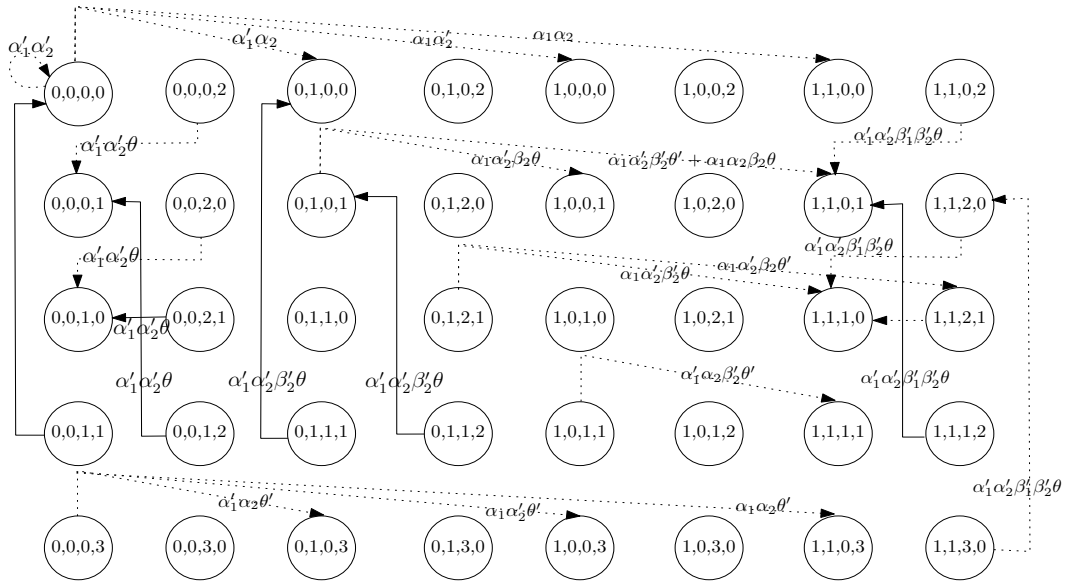


Figure 5.4: Example of a discrete time Markov chain queueing model of a wireless unicast exchange network with opportunistic network coding and dynamic buffer allocation. The entire state space and *only some of all possible transitions* are shown for a case of $N_1 = N_2 = 0$ and $N_r = 2$. While the dotted lines represents normal transitions, the solid lines emphasize those transitions that occur after network coding is performed. Transition probabilities are determined by arrival parameters α_1 and α_2 and service parameters β_1 and β_2, corresponding to sources s_1 and s_2, respectively. θ is a service parameter for AP/gateway r.

be constructed for a special case of $N_1 = N_2 = 0$ and $N_r = 2$ is shown as an example in Fig. 5.4. Note that *only a subset* of all possible transitions is shown.

$$\mathsf{P} = \begin{bmatrix} \mathbf{B} & \mathbf{C} & & & \\ \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 \\ & & & \mathbf{A}_2 & \mathbf{A}_{1b} \end{bmatrix}, \mathbf{i} = \begin{bmatrix} \mathbf{B}^{(\mathbf{i})} & \mathbf{C}^{(\mathbf{i})} & & & \\ \mathbf{A}_2^{(\mathbf{i})} & \mathbf{A}_1^{(\mathbf{i})} & \mathbf{A}_0^{(\mathbf{i})} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{A}_2^{(\mathbf{i})} & \mathbf{A}_1^{(\mathbf{i})} & \mathbf{A}_0^{(\mathbf{i})} \\ & & & \mathbf{A}_2^{(\mathbf{i})} & \mathbf{A}_{1b}^{(\mathbf{i})} \end{bmatrix}. \tag{5.3}$$

After P is obtained, the stationary probability distribution vector $\boldsymbol{\pi}$ can be com-

puted by solving $\boldsymbol{\pi} = \boldsymbol{\pi}\mathsf{P}, \quad \boldsymbol{\pi}\mathbf{1} = 1$, where $\boldsymbol{\pi}$ is a row vector with dimension $(N_1+2)(N_2+2)(\sum_{i=0}^{N_r+1} N_r + 2 - i)$, and $\mathbf{1}$ is a column vector of ones, with the same dimension. Alternatively, a more computationally efficient routine such as Power Method [107] can be used to approximate $\boldsymbol{\pi}$. The method starts with an initial vector $\boldsymbol{\pi}^{(0)}$, for $\boldsymbol{\pi}^{(0)}\mathbf{1} = 1$, and the stationary probability distribution vector is updated iteratively as $\boldsymbol{\pi}^{(j+1)} = \boldsymbol{\pi}^{(j)}\mathsf{P}, j \geq 0$, until $\|\boldsymbol{\pi}^{(j+1)} - \boldsymbol{\pi}^{(j)}\|_i < \delta$, $\forall i$. In this chapter, we set $\delta = 10^{-12}$. The resulting stationary probability distribution vector $\boldsymbol{\pi}$ is partitioned as follows:

$$\begin{aligned} \boldsymbol{\pi} &= [\boldsymbol{\pi}_0, \ldots, \boldsymbol{\pi}_m, \ldots \boldsymbol{\pi}_{N_1+1}] \\ \boldsymbol{\pi}_m &= [\boldsymbol{\pi}_m^{(0)}, \ldots, \boldsymbol{\pi}_m^{(n)}, \ldots, \boldsymbol{\pi}_m^{(N_2+1)}] \\ \boldsymbol{\pi}_m^{(n)} &= [\boldsymbol{\pi}_m^{(n,0)}, \ldots, \boldsymbol{\pi}_m^{(n,k)}, \ldots, \boldsymbol{\pi}_m^{(n,N_r+1)}] \end{aligned}$$

where $m \in \{0, 1, ..., N_1 + 1\}$ and $n \in \{0, 1, ..., N_2 + 1\}$ indicates the number of packets at s_1 and s_2, respectively; and $k \in \{0, 1, ..., N_r + 1\}$ indicates the number of packets at the AP/gateway r. The cardinality of $\boldsymbol{\pi}_m^{(n,k)}$ is completely determined by k, i.e., $|\boldsymbol{\pi}_m^{(n,k)}| = N_r + 2 - k$. By partitioning $\boldsymbol{\pi}$ in this manner, each element of $\boldsymbol{\pi}$ can be mapped to each state in the state space.

5.3 Performance Measures

Using the stationary probability vector $\boldsymbol{\pi}$ obtained in Section 5.2, we derive two performance measures of interest in this section. For convenience, let $N_1' = N_1 + 1, N_2' = N_2 + 1$ and $N_r' = N_r + 1$.

5.3.1 Coding Opportunity

We define coding opportunity as the normalized portion of time the AP/gateway has an opportunity to code two packets, one from each of its virtual buffers. In steady state, coding opportunity is given by

$$\Omega = \sum_{m=0}^{N_1'} \sum_{n=0}^{N_2'} \sum_{k\neq 0}^{N_r'} \sum_{i\neq 0}^{N_r'-k} \pi_m^{(n,k)}(i) \tag{5.4}$$

where $\pi_m^{(n,k)}(i)$ is the $(i+1)$th element of $\boldsymbol{\pi}_m^{(n,k)}$. Stated differently, coding opportunity is the probability that both virtual buffers of the AP/gateway are not empty.

5.3.2 Average Packet Delivery Delay

Packet delivery delay (PDD) is defined as the number of time slots used to successfully deliver a packet from a particular source to its respective destination. In the following, we derive average PDD as seen by a typical packet arriving to the system. To this end, we think of the system as having two packet classes 1 and 2 – corresponding to packets from s_1 and s_2, respectively – and apply Little's theorem to each class. Two individuals PDDs are then normalized.

Define

$$\begin{aligned} f_1(x) &= \sum_{m=0}^{N_1'} \sum_{n=0}^{N_2'} \sum_{k=0}^{N_r'} \sum_{i=0}^{N_r'-k} \pi_m^{(n,k)}(i)|_{x=m+k} \\ f_2(x) &= \sum_{m=0}^{N_1'} \sum_{n=0}^{N_2'} \sum_{k=0}^{N_r'} \sum_{i=0}^{N_r'-k} \pi_m^{(n,k)}(i)|_{x=n+i} \end{aligned} \tag{5.5}$$

where $f_1(x)$ (resp. $f_2(x)$) is the probability of having x packets – including the one in service – in the system as seen by a typical arriving class-1 (resp. class-2) packet regardless of whether it is to be coded with another packet from class 2 (resp. class 1) at the AP/gateway. To see this, consider a scenario where there are 3 and 7 packets in class-1 and class-2 buffers

of a AP/gateway, respectively, and 2 packets at s_1's buffer. Then, any class-1 packet arriving into the system in steady state will see the system as having (2+3=5) packets in front of it no matter how many class-2 packets there are. As a matter of fact, it will see 3 class-1 packets at the AP/gateway being coded with other 3 class-2 packets.

Using (5.5), the average number of packets from each class present in the system is $\overline{X}_j = \sum_{x=0}^{\mathcal{Y}_j} x f_j(x), \quad j = \{1, 2\}$, where $\mathcal{Y}_1 = N_1 + N_r + 2$ and $\mathcal{Y}_2 = N_2 + N_r + 2$. Now define $\lambda_e^{(j)} = (1 - P_j^L)\alpha_j, j = \{1, 2\}$, as the effective rate of class-j packets entering the system, where P_j^L is the probability that a class-j packet is dropped due to source buffer overflow. That is, $P_1^L = \sum_{n,k} \sum_{i=0}^{N_r'-k} \boldsymbol{\pi}_{N_1'}^{(n,k)}(i)$ and $\quad P_2^L = \sum_{m,k} \sum_{i=0}^{N_r'-k} \boldsymbol{\pi}_{m}^{(N_2',k)}(i)$.

Applying Little's theorem [107] to each class j, we have $PDD_j = \overline{X}_j / \lambda_e^{(j)}$. Normalizing both PDDs gives

$$PDD = \sum_{j=1}^{2} \frac{\lambda_e^{(j)} PDD_j}{\lambda_e^{(1)} + \lambda_e^{(2)}} = \frac{\overline{X}_1 + \overline{X}_2}{\lambda_e^{(1)} + \lambda_e^{(2)}}. \tag{5.6}$$

5.4 Numerical and Simulation Results

We validate our queueing model and compare its delay performance with that of FIFO and RRB scheduling schemes.

5.4.1 Simulation Scenarios and Model Validation

The wireless unicast exchange network is assumed to be stationary. We set the buffer capacities $\{N_1, N_2, N_r\}$ equal to 4 for all simulation scenarios and let Bernoulli arrival parameters α_1 and α_2 vary between zero and $\max\{\beta_1, \beta_2, \theta\}$ to avoid traffic overload. Service parameters β_1, β_2, and θ are random across nodes and time, and depend on the

conditions of the channels (i.e., $\epsilon_{s_1,r}, \epsilon_{s_2,r}, \epsilon_{r,s_1}, \epsilon_{r,s_2}$). Statistics are taken over a specified period during which channel conditions for all nodes are assumed to be constant. Each period consists of a multiple number of time slots and defines a particular channel realization $\psi = (\epsilon_{s_1,r}, \epsilon_{s_2,r}, \epsilon_{r,s_1}, \epsilon_{r,s_2})$. Each element of the tuple (realization) is varied within $[\epsilon_{min}, \epsilon_{max}]$. For each realization, the total of 1000 packets – 500 from each source – are transmitted. All channel realizations, whose tuple elements vary in a range $[\epsilon_{min}^{(i)}, \epsilon_{max}^{(i)}]$, are said to be in a channel realization set Ψ_i. All simulations are carried out in Matlab.

We validate the queueing model by comparing the theoretical PDD, derived in Section 5.3, with the PDD obtained using simulation. For both cases, PDD is plotted as a function of normalized effective packet arrival rate, $\hat{\lambda}_e = (\lambda_e^{(1)} + \lambda_e^{(2)})/2$, defined as the average number of packets entering the system in a time slot. In Fig. 5.5 the results for both cases are shown for three channel realization sets. As expected, PDD increases as arrival rates increases and as channel quality becomes worse. In all the cases, PDD is at least one queueing time slot, where one queueing time slot may be interpreted as a period comprising three equally spaced medium access time slots (Fig. 5.2c).

5.4.2 Comparison with Classical Scheduling

We are interested in how much we gain – in terms of delay – from using ONC plus dynamic buffer allocation when compared with those classical simple-to-implement scheduling schemes, i.e., FIFO and RRB. We simulate the network used for ONC. However, the AP/gateway is modified to have a FIFO or RRB scheduler. Simulations are averaged over two channel realization sets $\Psi_1 = (0, 0.1]$ and $\Psi_2 = (0, 0.2]$. The results are shown in Fig. 5.6.

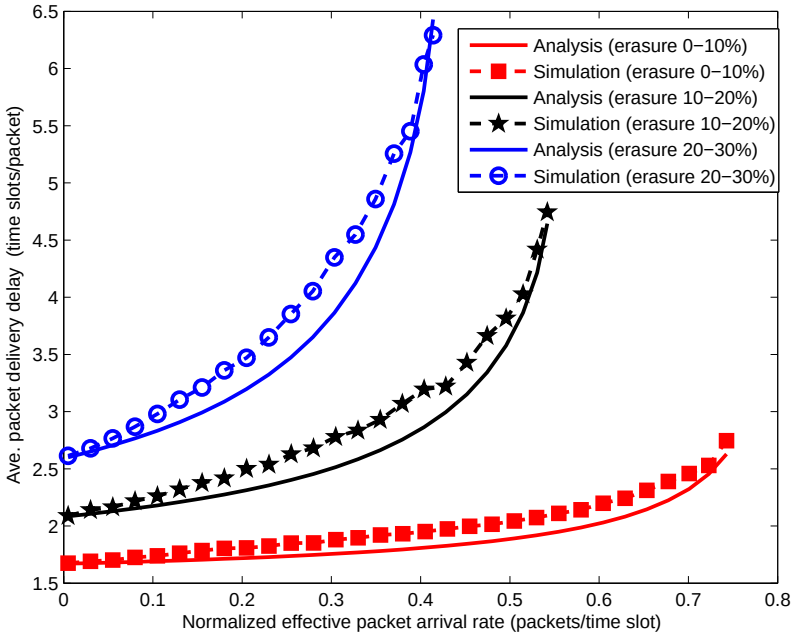


Figure 5.5: Model validation for three sets of channel realizations.

Table 5.1: Coding Opportunity versus Normalized Arrival Rates.

$\hat{\lambda_e}$	0.08	0.205	0.3046	0.4019	0.4876
Ω	0.84%	5.4%	11.79%	20.49%	30.99%

It is evident that ONC outperforms FIFO and RRB in terms of average PDD. In particular, it outperforms FIFO and RRB by approximately 47% and 36%, respectively, in low traffic load condition, and by approximately 396% and 326%, respectively, in high traffic load condition. For this specific scenario, we designate traffic as low or high based on a normalized packet arrival rate ($\hat{\lambda}_e$), i.e., it is low (resp. high) if $\hat{\lambda}_e \in (0, 0.3]$ (resp. $\hat{\lambda}_e \in (0.3, \sim 0.7)$).

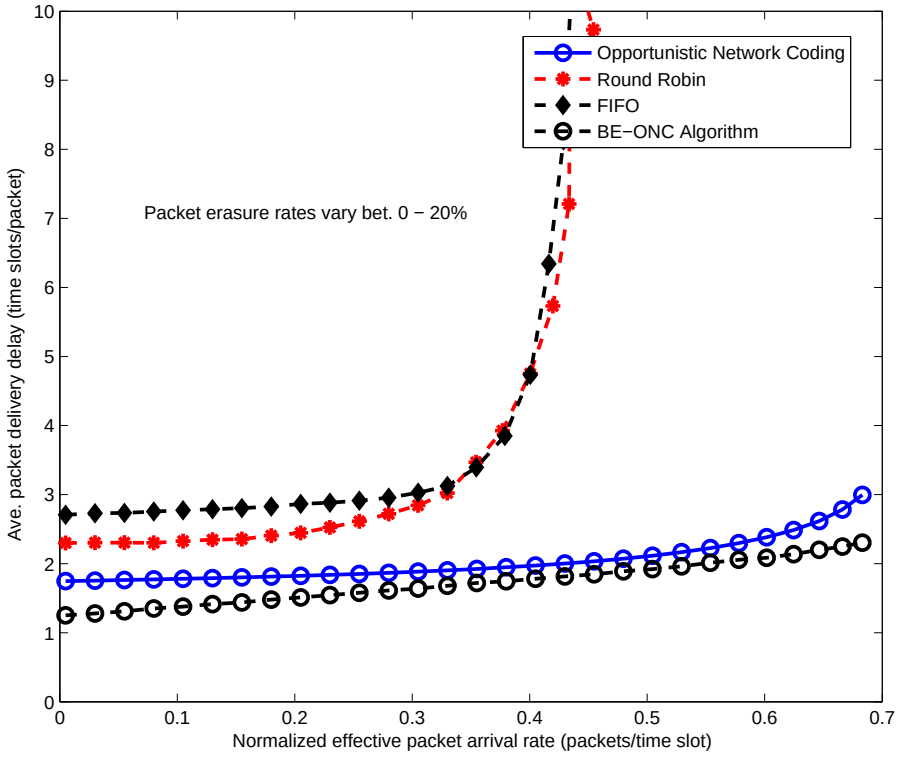


Figure 5.6: ONC in comparison with classical scheduling and BE-ONC.

One questionable observation however is that – is a performance gain in low traffic really worth the effort of network coding? From Fig. 5.6, it is unlikely so. The reason is intuitive. When the aggregate rate of packets arriving to the system is low, the likelihood that two packets – one from each source – will "meet" at the AP/gateway is low. That is, the chance that one of the AP/gateway buffers will be empty is high. In such cases, ONC operates just like RRB. Table 5.1 shows the coding opportunity Ω expressed here in percentage – as a function of normalized $\hat{\lambda}_e$. This observation leads us to the introduction of a simple scheduling algorithm that yields better results. We discuss this algorithm in the following section.

5.5 Buffer-Equalized ONC (BE-ONC)

In our ONC model, we let the AP/gateway dynamically allocate individual buffer spaces to packets coming from s_1 and s_2 – one individual buffer for each source (Fig. 5.3). As we have pointed out in Section 5.4.2, this strategy is not very effective in low traffic load conditions. Furthermore, in unbalanced traffic, where the arrival rates of both flows from s_1 and s_2 are highly different, one of the buffers at the AP/gateway might build up faster than the other. Thus, by letting the buffer occupancy at the AP/gateway be so dynamic, the lower-rate flow might starve and be blocked so frequently and unfairly. In this section, we propose a new scheduling technique that remedies this shortcoming. The idea is simple and only to lower the degree of such "dynamic" occupancy in a controllable manner like a sliding valve shown for analogy in Fig. 5.2b, where packets now are interpreted as liquid. Our technique allows packets from different flows to move across buffers (by sliding the value up/down) so that in the long run any packet entering the system will see the lining ahead of it as shortest as possible. We term this strategy as *buffer equalization* (BE). In allowing packets to move across buffers, there are two possible situations once the packets reach the head of the line: (i) They may pair up with packets from the same flow or (ii) with packets from the different flow. In the first (resp. second) case, therefore, we propose that intra-session network coding (resp. inter-session network coding) be used. Our technique works as follows.

When a packet arrives, the AP/gateway decides – based on a probability measure to be defined – which buffer the packet should go in and wait for transmission so as to minimize average PDD. Once assigned to a buffer, the packet is tagged as either *foreign*

or *native* to indicate its relationship with the assigned buffer. It is native if the buffer is in direct tandem with its source buffer, and foreign otherwise. Once the packet reaches the head of the line, it pairs up with the other HOL packet residing in the other buffer. If they both are foreign or native – meaning that they come from different flows – the AP/gateway XORs the two packets into a single packet and transmits it just like ONC does. In contrast, if their tags are different, i.e., one native and the other foreign or vice versa, the AP/gateway performs intra-session network coding. In particular, the AP/gateway XORs[2] the two packets and keeps one of the uncoded packets. This uncoded packet is necessary and must be transmitted in order for the destination to recover the information embedded in the coded packet sent previously. One question arises here – which packet to keep? One quick and sensible solution is to keep the native one so as to make way for other native packets in the buffer that the pairing foreign packet resided. The algorithm that implements this scheduling strategy is shown in Algorithm 5.1. The AP/gateway uses the probability metric ΔQ to make a buffer allocation decision against threshold Q^T. This threshold Q^T is defined as the probability that the steady-state difference in buffer occupancy between the two virtual buffers is greater than the instantaneous difference q measured at the moment the packet arrives. The probability ΔQ can be estimated from our queueing model as follows:

$$\Delta Q = \Pr\{|L_1 - L_2| > q\} \approx \sum_{m=0}^{N_1'} \sum_{n=0}^{N_2'} \sum_{k=0}^{N_r'} \sum_{i=0}^{N_r'-k} \pi_m^{(n,k)}(i)|_{|k-i|>q}. \tag{5.7}$$

Parameter μ is an urgency level of an incoming packet, ranging from 0 to 1. This urgency level may be thought of as a normalized metric derived from a media playback deadline. We

[2]Another option is to form two random linear combinations of these two packets. Both coded packets then must be transmitted in subsequent slots.

leave Q^T as a design parameter. In our simulation, we simply set Q^T to a conservative 0.5 – meaning that if the buffer difference ($|L_1 - L_2|$) has been larger than the instantaneous buffer difference q most of the time on average, then the AP/gateway will assign an arriving packet to the shorter foreign buffer. The average PDD curve resulting from applying our algorithm is shown in Fig. 5.6. Our proposed scheme clearly shows some improvement in low traffic and approaches ONC in relatively high traffic, as can be expected.

Algorithm 5.1: Buffer Equalized ONC (BE-ONC)

input : ψ, N_1, N_2, N_r. $0 \leq Q^T \leq 1$

```
1 Initialization:
2 Let p_i^(k) be packet i of source k ∈ {1, 2}
3 while p_i^(k) arrives at AP/gateway do
4     if Buffers not empty then
5         if (Buffer k longer) & (μΔQ ≥ Q^T) then
6             Enqueue p_i^(k) in buffer (k mod 2) + 1
7         else
8             Enqueue p_i^(k) in buffer k
```

5.6 Conclusion

We have presented a discrete-time Markov chain queueing model for a lossy wireless unicast exchange network which employs opportunistic network coding and dynamic buffer allocation capabilities at the bottlenecked AP/gateway. From the end-to-end delay performance perspective, we have show that such dynamic buffer allocation in synergy with

opportunistic network coding can be a promising scheduling candidate for this particular communication scenario. By allowing the AP/gateway buffer allocation to be too dynamic, however, a lower-rate packet flow may starve and never get its buffer share. We have proposed a solution, called buffer-equalized opportunistic network coding, to this problem. The proposed solution has been shown to improve in terms of delay over our original model which uses opportunistic network coding and dynamic buffer allocation.

Chapter 6

Wireless Fountain Coding with IEEE 802.11e Block ACK for Uplink Streaming

As IEEE 802.11 continues to be a primary mode of wireless access in residential buildings and offices, demands for high-bandwidth media contents also grow. Being able to stream media data from a wireless media server (e.g., a D-Link© wireless media player) to a wired client (e.g., TV, Printer, or Presentation Screen) sitting in another room is one of such demands, as shown in Fig. 6.1. This chapter addresses the problem of streaming packetized media data in such environments where media packets, subject to errors on both wireless and wireline channels, are streamed from a dedicated wireless media server to a wireline client via an AP/gateway node of the underlying IWMN.

In a wireless media streaming system, a wireless media server pre-stores encoded

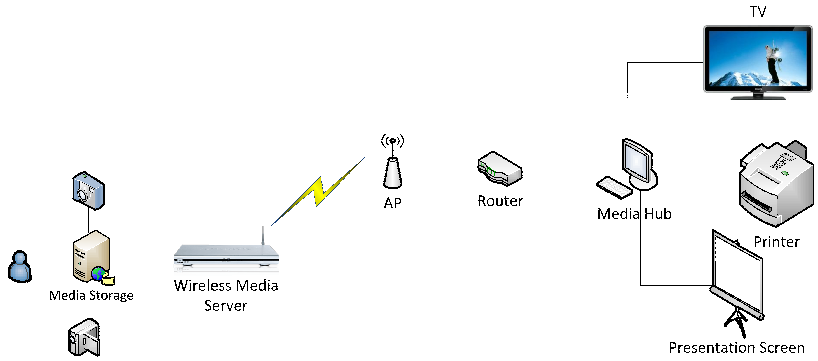


Figure 6.1: Example scenario of in-home uplink streaming.

media data and transmits it on demand to a client for playback in real-time. The client buffers the data and begins playback after a fixed delay. Once the playback begins, the client expects no interruption during the entire presentation of the media. Interruption is however inevitable due to packet errors or late arrivals of packets, which can occur in both the wireless channel and the wireline channel.

Late packets are useless and usually caused by the varying wireless link capacity. Normally unmatched to the wireline capacity, this varying wireless link capacity typically results from the unreliable and time-varying nature of the wireless channel itself and poor channel utilization caused by excessive MAC-layer overhead. Packets must be dropped if packet injection rates on the wireless link do not match the varying link capacity. One way to improve the utilization of the wireless channel is to employ a burst transmission scheme such as Block Acknowledgment (B-ACK) as specified in the IEEE 802.11e standard [94]. The IEEE 802.11e B-ACK scheme allows a receiving station to acknowledge the reception of multiple packets transmitted in a block using a single BlockAck (BA) control packet which must be requested by a BlockAckReq (BAR) control packet from a sending station. The bitmap field in the BA control packet will indicate the reception status of all packets

within the block. Those packets not received successfully will be retransmitted selectively. Improved channel utilization then must be traded with increased protocol complexity. In the literature, several research efforts on analytical modeling of the B-ACK scheme can be found in [56–58, 95]. In [56], a simple analytical model is developed to estimate the upper bound of the throughput in an 802.11 network deploying the B-ACK scheme. Error-free channels are assumed. In [57], a model is developed to derive the relationship between the number of stations and a collision probability when B-ACK is applied to both uplink and downlink channels, assuming error-free channels. B-ACK schemes under more realistic assumptions are studied in [58, 95]. Li et al. [58] consider both channel error and collisions and derive the saturation throughput accordingly. A selective retransmission mechanism is however not modeled. Chen et al. [95] go one step further by modeling the selective retransmission mechanism for the immediate B-ACK scheme aimed specifically for high bandwidth, low latency applications. Packet errors however are assumed only in the data channel. All the control signals such as BAR and BA are assumed to be perfect as in [58].

When packets are lost in the wireline network, typically due to congestion, error control mechanisms are needed at higher layers to ensure satisfying streaming experiences. To combat such errors on an end-to-end basis, two prominent mechanisms – namely, automatic repeat request (ARQ) [61, 62] and channel coding with forward error correction (FEC) [63–66] – are normally proposed for media streaming. Although less complex than FEC, employment of ARQ at the application-layer requires a reliable feedback channel. ARQ thus inevitably incurs a variable network delay. FEC, on the other hand, can overcome these drawbacks but only at the expense of transmitting a large amount of redundant

data. Many attempts thus follow naturally, aiming at extracting the best flavors of FEC and ARQ, in a "hybrid" fashion [67–70, 72]. In [68], the authors combine source coding, FEC and application-layer ARQ to optimize video transmission over packet lossy networks. This scheme is however not suitable for media streaming because in the streaming system media data must be source-encoded and prestored offline without knowing the state of the channel during transmission. In [69], FEC and ARQ are combined to control packet errors in wireless video transmission. In making error control decisions, this scheme still resorts to the information, i.e., priority of video packets, offered by hierarchical video (source) coding. In [72], application-layer FEC and link-layer ARQ are jointly considered for one-hop video transmission over 802.11a networks. A per-packet 802.11a acknowledgement scheme is employed. FEC and ARQ are also combined to combat packet errors in wireless networks. Another study that jointly considers FEC and ARQ for wireless video transmission is [70] in which error control decisions are made according to media and channel characteristics. In [67], the author proposed an error control framework combining FEC and application-layer ARQ for media streaming over a combined wireline/802.11 network (similar to the one in Fig. 6.1) where a wireless media server and AP/gateway use a simple per-packet acknowledgement scheme.

In this chapter, we propose a new version of B-ACK in which a version of fountain coding [59] – termed here as *wireless fountain coding (WFC)* – is embedded to efficiently utilize the wireless channel. In WFC, each packet transmitted over the wireless link is a linear combination of a particular block of, say, K media packets. A wireless media server, analogous to a fountain, keeps transmitting such combined packets until AP/gateway, anal-

ogous to a water bucket, collects at least K linearly independent packets (at which stage an acknowledgement is signaled to stop the sender from sending anymore packets). WFC thus rules out the traditional selective acknowledgement mechanism. By using this version of B-ACK, protocol complexity and a wireless link-layer delay can potentially be reduced. We analytically quantify this wireless link-layer delay and study how it translates into end-to-end reliability benefits when ARQ and FEC are jointly employed at the application-layer to control errors for delay-sensitive media uplink streaming over a combined wireline/802.11 network. To quantify the reliability benefits, we derive a probability that a media packet is lost or late for presentation for three major cases: (1) Traditional B-ACK with joint employment of FEC and ARQ, (2) Modified B-ACK with employment of FEC only, and (3) Modified B-ACK with joint employment of FEC and ARQ.

We develop an integrated ns-3/EvalVid simulator to validate and evaluate our performance models. Comparison with the traditional 802.11e B-ACK scheme is performed. For fair comparison, we extend the analysis of the traditional 802.11e B-ACK scheme in [95]. Unlike [95], we further assume that BAR and BA control signals are both subject to errors while also taking application-layer FEC and ARQ into account. The accuracy of our extended model is verified by numerical results. Via extensive simulations of video streaming, we observe that the modified B-ACK scheme does not always yield end-to-end reliability benefits under certain conditions of the wireless channel. Therefore, we propose a hybrid scheme that switches between the modified and traditional B-ACK schemes according to the conditions of the wireless channel and the number of packets to transmit in a block. We show via simulations that this hybrid scheme is able to keep the end-to-end packet loss/late

probability as low as possible given the conditions of the wireless channel and the number of packets transmitted in a block. Unlike [68–70], our work does not assume any specific form of source coding as this is usually the case for a streaming system in which media data are encoded and prestored offline without the benefit of knowing the state of the channel during transmission [73].

The rest of this chapter is organized as follows. In Section 6.1, through a motivating example we illustrate how a version of fountain coding can be used to extract a delay benefit over the unreliable wireless link. Section 6.2 describes the proposed uplink streaming system. Section 6.3 is devoted to the analysis of the streaming system with the modified B-ACK scheme. Section 6.4 gives the analysis of the streaming system with the traditional B-ACK scheme. In Section 6.5, we describe the framework we develop to evaluate the performance of the proposed system. In Section 6.6, we validate and evaluate our performance models in various aspects via simulations. Section 6.7 concludes the chapter.

6.1 Motivating Example

We have mentioned in the introduction that applying wireless fountain coding in the wireless link can potentially reduce transmission delay and protocol complexity. To illustrate this, let us consider the example in Fig. 6.2(a) where a wireless media server attempts to send a file of K packets, $\mathbf{p}_1, ..., \mathbf{p}_K$, to the AP/gateway. For simplicity, suppose the wireless links in both directions are 50% reliable. According to the current 802.11 standard, four transmissions will be required to successfully deliver each packet to AP/gateway, i.e., two transmissions for the packet itself and the other two for acknowledgement. The

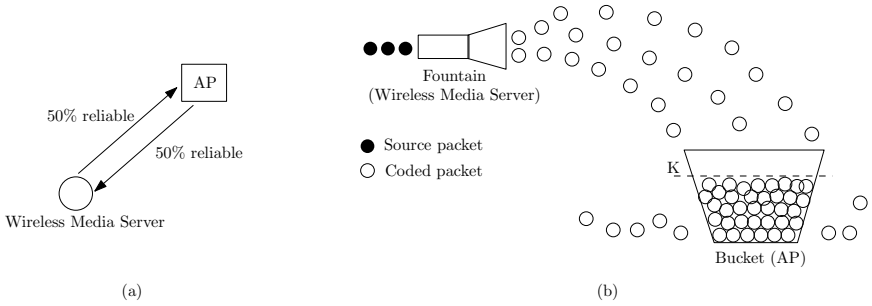


Figure 6.2: Wireless fountain coding potentially offers delay benefits in streaming in the presence of unreliable wireless channels: (a) Actual system (b) Analogy.

total of $4K$ transmissions then will be required for the delivery of the entire file. We can achieve a little improvement by using the traditional 802.11e B-ACK scheme instead. That is, after K consecutive transmissions, the AP/gateway tells the wireless media server which packets it has not received yet. The wireless media server then retransmits as many packets as indicated. This process repeats until all K packets have been received successfully. This more complex scheme requires at least $2K + 2\log_2 K$ transmissions to complete sending the entire file [10].

However, with wireless fountain coding and a relatively less complex acknowledgment protocol, a fewer number of transmissions will be required. In this case, the wireless media server will keep transmitting a random linear combination of the K packets. Being randomly mixed, all coded packets are created equal. There is no need to distinguish between transmitted packets as long as we have a mechanism to recover them at the AP/gateway. The AP/gateway starts decoding once it has collected at least K linearly independent coded packets. When this happens the AP/gateway only needs to send a single positive acknowledgement to stop the wireless media server. These operations are

shown for analogy in Fig. 6.2(b) where the wireless media server acts as a fountain and the AP/gateway as a bucket. The goal is to fill the bucket with coded packets to a level at which decoding can start. Clearly, only $2K + 2$ transmissions – i.e., $2K$ transmissions for delivering K packets and 2 transmissions for the positive acknowledgement – are required in this case.

In this chapter, we aim to analytically quantify this transmission delay induced by wireless fountain coding when applied to the IEEE 802.11e B-ACK scheme. Then we will study how this potential delay benefit, as compared to that induced by the traditional B-ACK, will translate into end-to-end reliability benefits when application-layer FEC and ARQ are jointly employed to control errors at higher layers.

Before we proceed, it should be noted that this *wireless fountain coding* scheme is neither exactly random linear fountain coding as studied in [59] nor intra-session random linear network coding as discussed in [4, 5, 7]. It is something in between. It resembles fountain coding only in the sense that, roughly speaking, it does not matter what (coded) packets are received or lost, but it only matters that enough packets are received. Wireless fountain coding however does not code data at a symbol level. Rather, data are coded at a packet level in the same manner as done in intra-session random linear network coding. This implies that coding operations in a finite field do not need to be binary operations (random XORs) as in random linear fountain coding. This is advantageous because coding in a bigger finite field increases the probability of successful decoding [7], albeit with increased complexity. Nevertheless, we cannot simply regard wireless fountain coding as pure network coding because its coding/decoding operations are not network-wide and occur only at

endpoints of a one-hop wireless link.

6.2 System Description

We consider a media streaming system as shown in Fig. 6.1 where a wireless media server in a WLAN network streams prestored media packets to a client in a wireline IP network. The wireline IP network may consist of desktop computers, routers, or gateways and so on. Although our analysis is applicable to a backbone wireline network like Internet, we restrict our attention only to a private local area network (LAN). The WLAN network, which is based on the the 802.11e standard, has a dedicated AP/gateway that will work with the wireless media server. The goal of the system is to stream delay-sensitive packets from the wireless media server to the wireline destination client such that a media stream is played out at the destination in a timely and reliable manner. To aid such timely and reliable delivery, we propose that wireless fountain coding be used in conjunction with application-layer ARQ and FEC. We modify the traditional 802.11e B-ACK scheme to work with wireless fountain coding. Wireless fountain coding arises here to potentially deliver packets across the wireless link as quickly and reliably as possible while keeping the complexity of doing so at a low level compared to the traditional 802.11e B-ACK protocol. While wireless fountain coding will provide the delay benefit in the wireless link, FEC and an application-layer ARQ will help counter packet loss at a higher layer. We describe the source model and FEC, wireless fountain coding, the modified B-ACK model and application-layer ARQ in Sections 6.2.1, 6.2.2, 6.2.3 and 6.2.4, respectively. Assumptions on the channel model and wireline delay model are presented in Section 6.2.5 and Section 6.2.6, respectively.

6.2.1 Source Model and FEC

As shown in Fig. 6.3, systematic Reed-Solomon (RS) FEC codes are applied across k packets to produce $n(> k)$ transport packets at the application layer. Each of the $n(> k)$ transport packets vertically generated thus contains both data payload and FEC protection. Since k source packets need to be buffered before the RS coding starts, we propose that the RS coding is done over a block of k source packets having the same playback deadlines, or equivalently, the same maximum allowed delay (MAD). This MAD will be tagged to each transport packet. Once generated, these n packets are buffered in an application-layer buffer and copied to a retransmission buffer for possible application-layer retransmission.

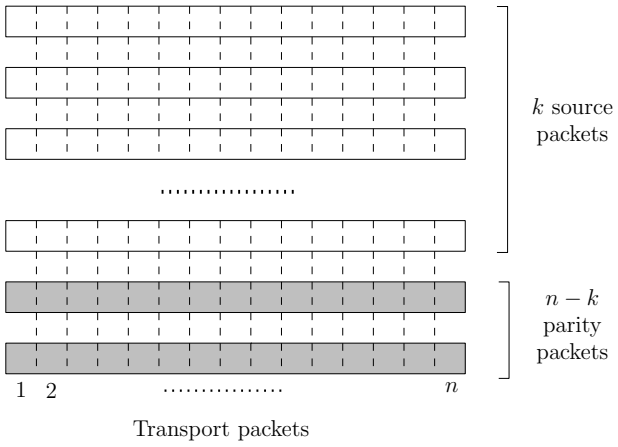


Figure 6.3: Packetization of media data and forward error correction.

6.2.2 Wireless Fountain Coding

After a UDP/IP header is added, the n transport packets described in Section 6.2.1 are buffered just above the IEEE 802.11e MAC layer, where wireless fountain coding is applied. Wireless fountain coding is performed at a wireless media server over a block of

$K(\leq n)$ packets, $\{\mathbf{p}_1, ..., \mathbf{p}_K\}$. Each coded packet transmitted by the wireless media server is a random linear combination $\mathbf{x}_i$ of K packets, i.e.,

$$\mathbf{x}_i = \sum_{j=1}^{K} c_{i,j}\mathbf{p}_j \tag{6.1}$$

where the addition and multiplication are performed over a finite field $\mathbf{GF}(2^q)$, and coefficients $c_{i,j}$ are uniformly drawn at random from $\mathbf{GF}(2^q)$. Each coded packet $\mathbf{x}_i$ is prepended with an encoding vector $\mathbf{c}_i = [c_{i,1} \ldots, c_{i,K}]$. Denote such a packet by a tuple $(\mathbf{c}_i, \mathbf{x}_i)$.

A regular MAC header/trailer is next added to each coded packet to form a MAC Protocol Data Unit (MPDU). In this chapter, we limit our analysis to the performance of a single user (wireless media server) where packet losses are caused solely by wireless link errors and not by collisions. This assumption is valid since the 802.11e MAC also provides the Point Coordination Function (PCF) for contention-free channel access. For each block of K packets, the wireless media server will keep transmitting the coded packets to the AP/gateway until at least K linearly independent coded packets are successfully received. There is no need for a selective B-ACK mechanism as specified in the 802.11e standard.

$$\left.\begin{array}{rcl} \mathbf{x}_1 &=& c_{1,1}\mathbf{p}_1 + ... + c_{1,K}\mathbf{p}_K \\ \mathbf{x}_2 &=& c_{2,1}\mathbf{p}_1 + ... + c_{2,K}\mathbf{p}_K \\ \cdots & & \\ \mathbf{x}_K &=& c_{K,1}\mathbf{p}_1 + ... + c_{K,K}\mathbf{p}_K \end{array}\right\} \Rightarrow \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,K} \\ c_{2,1} & \cdots & c_{2,K} \\ \vdots & \ddots & \vdots \\ c_{K,1} & \cdots & c_{K,K} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_K \end{pmatrix} \quad \Rightarrow \quad \mathbf{x} = \mathbf{C}\mathbf{p}. \tag{6.2}$$

Once successfully received by the AP/gateway, each MPDU will be forwarded up and buffered just above the IEEE 802.11e MAC layer of the AP/gateway waiting for a

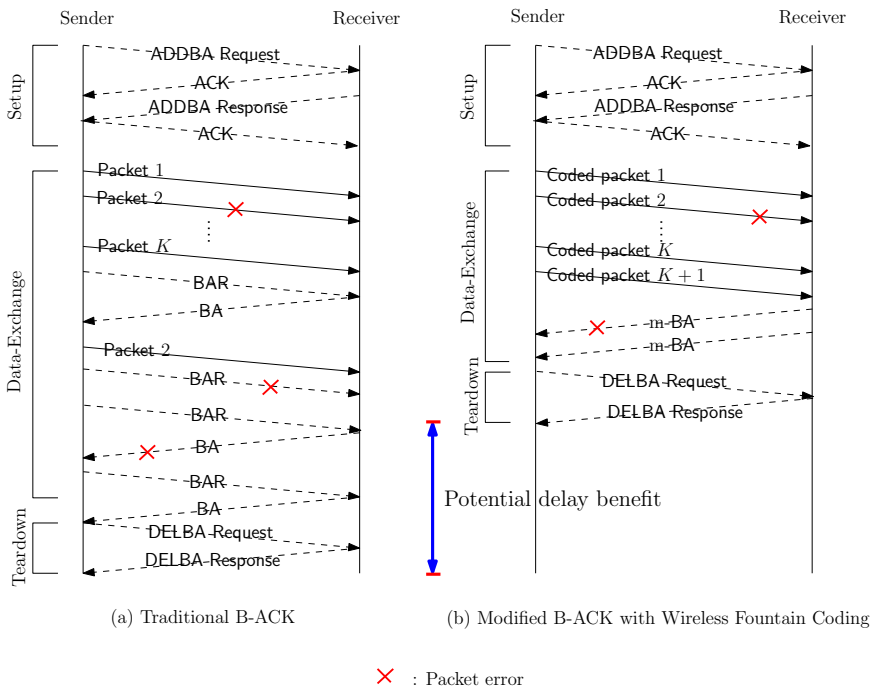
decoding process. The decoding process is triggered once K linearly independent coded packets, $(\mathbf{c}_1, \mathbf{x}_1), \ldots, (\mathbf{c}_K, \mathbf{x}_K)$, have been received. Since there is no guarantee that these K coded packets will be linearly independent, the AP/gateway first checks for linear independence by performing Gaussian Elimination on the coefficient matrix $\mathbf{C}$ shown in (6.2). If the linear independence checking is successful, the AP/gateway sends out a BA-like signal to stop the wireless media server from transmitting. The AP/gateway decodes by solving (6.2) for $\{\mathbf{p}_1, ..., \mathbf{p}_K\}$.

6.2.3 Modified B-ACK Model

We modify the traditional 802.11e immediate B-ACK scheme to accommodate wireless fountain coding. The typical packet flow of the modified version is shown in Fig. 6.4(b) in comparison with that of the traditional B-ACK shown in Fig. 6.4(a).

A complete cycle for both B-ACK scheme consists of three phases: Setup, Data-Exchange, and Teardown. The Setup phase involves a wireless media server and the AP/gateway exchanging a pair of ADDBA request and response packets to initialize a B-ACK cycle. Once the two parties agree, the operational cycle enters the Data-Exchange phase in which the difference between the traditional B-ACK and the modified one is highlighted. We briefly explain both versions of B-ACK as follows:

Traditional B-ACK: In the traditional B-ACK scheme, the wireless media server issues the first B-ACK request (BAR) to the AP/gateway upon finishing sending the first K packets. The AP/gateway then responds with a B-ACK response packet (BA) detailing, through its bitmap field, *which* packets are still missing. Knowing this information, the wireless media server starts re-transmitting the missing packets selectively. After re-



(a) Traditional B-ACK

(b) Modified B-ACK with Wireless Fountain Coding

Figure 6.4: Immediate block acknowledgement schemes: (a) Traditional version without wireless fountain coding, and (b) Modified version with wireless fountain coding. The differences between both versions lie in Data-Exchange phase.

transmitting as many packets as indicated, another BAR is sent to the AP/gateway who responds with the current status of the delivery, detailing which packets are still missing. The Data-Exchange phase repeats until all K packets are successfully received or the maximum number of transmissions per block, N_{max}, is reached.

Modified B-ACK: In the modified B-ACK scheme, the wireless media server will keep transmitting coded packets, each separated by a Short Inter-Frame Spacing (SIFS) period, until at least K linearly independent coded packets are successfully received by the AP/gateway. The AP/gateway then responds with a modified BA (m-BA) control packet to cease further transmission. No BAR control packets are needed.

For both versions of B-ACK, we assume that transmissions of data/coded packets and BAR/BA/m-BA control packets during Data-Exchange phase are subject to error whereas transmissions in Setup and Teardown phases are assumed to be perfect.

6.2.4 Application-layer ARQ

Once successfully decoded by the AP/gateway, the K IP packets are forwarded to the wireline destination client using existing routing techniques. In case one of these IP packets is lost, possibly due to congestion in the wireline network, an application-layer negative acknowledgement is sent to the wireless media server. The requested packet stored in the application-layer retransmission buffer will have priority over those packets in the normal application-layer buffer. It will be re-transmitted immediately as part of the next block of K packets. Each packet in our system will also be tagged with the maximum allowable number of application-layer retransmissions, N_{APP}.

6.2.5 Channel Model

To capture packet loss in the wireless and wireline channels, we view each channel as a packet erasure channel and model it as a two-state Markov chain. We denote ϵ_u and ϵ_f as the packet loss rates for the wireless $u$plink channel and the wireline $f$orward channel, respectively. Similarly, we denote ϵ_d and ϵ_b as the packet loss rates for the wireless $d$ownlink channel and the wireline $b$ackward channel, respectively. Without retransmission mechanisms the end-to-end packet loss rate in the uplink/forward direction is $\epsilon_{uf} = 1 - (1-\epsilon_u)(1-\epsilon_f)$ and that in the backward/downlink direction is $\epsilon_{bd} = 1 - (1-\epsilon_d)(1-\epsilon_b)$.

6.2.6 Wireline Delay Model

Another element to model is a one-way delay a packet takes to traverse in the wireline IP network. According to [73] [90], a one-way forward trip time (FTT) D_f that a packet takes to travel without loss from one wireline station to another can be modeled as a random variable having a shifted Gamma distribution with rightward shift Δ_f and parameters α_f and λ_f, i.e.,

$$f_{D_f}(t|\text{ not lost}) = \frac{\lambda_f}{\Gamma(\alpha_f)}(\lambda_f(t-\Delta_f))^{\alpha_f-1}\exp\left(-\lambda_f(t-\Delta_f)\right), \quad t \geq \Delta_f. \tag{6.3}$$

The random variable D_f can be interpreted as the time a packet takes to go through a series of α_f routers, bridges, or gateways, each of which is modeled as an M/M/1 queue with parameter λ_f and a constant processing time Δ_f/α_f plus waiting time in steady state. The mean and variance of D_f are $\mu_f = \alpha_f/\lambda_f + \Delta_f$ and $\sigma_f^2 = \alpha_f/\lambda_f^2$, respectively. The parameters λ_f and α_f of the distribution can be determined from μ_f and σ_f^2, both of which can be periodically estimated, as: $\lambda_f = (\mu_f - \Delta_f)/\sigma_f^2$ and $\alpha_f = \lambda_f(\mu_f - \Delta_f)$.

Similarly, a backward trip time (BTT) D_b can also be modeled as having a shifted Gamma distribution $f_{D_b}(t)$ with rightward shift Δ_b and corresponding parameters α_b, λ_b, μ_b and σ_b^2.

In the following, we start with the analysis of the the modified B-ACK scheme and the derivations of its corresponding end-to-end performance measures.

6.3 System Analysis I: Streaming with Modified B-ACK

The goal of this section is to express the probabilities that a packet is lost or late for presentation, in terms of parameters controllable through FEC, Modified B-ACK and application-layer ARQ. We analytically quantify the wireless link-layer delay and then use it to derive the probabilities that a media packet is lost or late for presentation. Thus, we first ask in Section 6.3.1: How long will it take to successfully deliver a block of K packets, including the tagged packet, from the wireless media server to the AP/gateway? We use the delay derived in Section 6.3.1 to obtain the packet loss/late probabilities for the cases with and without application-layer ARQ in Sections 6.3.2 and 6.3.3, respectively. That is, while FEC will be used as a primary means for error control/recovery, we keep an application-layer ARQ as an option.

6.3.1 Analysis of Wireless Delay

We are interested in how long it takes to deliver a block of K packets uplink when the modified B-ACK is used (i.e., when wireless fountain coding is employed). From this delay measure we can calculate the remaining time a tagged packet has for traveling in the wireline IP network given its MAD. For the analysis, we assume that time is divided into equally spaced slots and transmission of each packet occupies exactly one time slot. This assumption implies the following: First, the MAC layer will not change its data rate during the transmission of a block of K packets. Second, packets in our system will have the same size in order to undergo the wireless fountain coding process. Smaller packets will always be padded with zeros to meet this requirement.

Expected number of time slots required

Let t be the number of time slots required to successfully deliver K linearly independent coded packets to the AP/gateway. Intuitively, the probability that t is less than K is equal to zero, i.e., $\Pr\{t < K\} = 0$. Moreover, since each coded packet transmitted can be either physically lost or, if not so, linearly *dependent* with previously received packets, transmission of a coded packet is regarded as a *failure* with probability $(\epsilon_u + \frac{1-\epsilon_u}{q})$ (Recall that q is the size of the finite field $\mathbf{GF}(q)$ under consideration), where the second term corresponds to the event that the packet is successfully received (with probability $(1-\epsilon_u)$) but unsuccessfully decoded (with probability $1/q$) [7]. It follows that $\Pr\{t = K\} = \left(1 - (\epsilon_u + \frac{1-\epsilon_u}{q})\right)^K$. Now, if $K+1$ time slots are required to successfully deliver K packets to the AP/gateway, one of the first K coded packets transmitted must have failed. No matter which packet has failed, a new coded packet must be retransmitted successfully in the $(K+1)$th time slot. The probability of this event is $\Pr\{t = K+1\} = \binom{K}{1}\left(\epsilon_u + \frac{1-\epsilon_u}{q}\right)\left(1 - (\epsilon_u + \frac{1-\epsilon_u}{q})\right)^{K-1}\left(1 - (\epsilon_u + \frac{1-\epsilon_u}{q})\right)$. In general, the probability that $K+i$ time slots are required to successfully deliver all K coded packets to the AP/gateway is given by

$$\Pr\{t = K+i\} = \binom{K+i-1}{i}(\epsilon_u + \frac{1-\epsilon_u}{q})^i(1 - (\epsilon_u + \frac{1-\epsilon_u}{q}))^K. \tag{6.4}$$

The expected number of time slots required follows naturally as

$$\mathsf{E}\{t\} = \sum_{i=0}^{\infty}(K+i)\Pr\{t = K+i\}. \tag{6.5}$$

If the maximum number of transmissions for a block of K coded packets is limited to N_{max}, the upper range for i in (6.5) should be replaced by $N_{max} - K$ [1].

Expected wireless link-layer delay

Having received K linearly independent coded packets, the AP/gateway responds within an SIFS period by transmitting an m-BA packet (of length L_{BA} bits) to stop the wireless media server from transmitting. Due to packet erasure in the downlink channel, a sequence of m-BAs may be required until a DELBA request is returned. This DELBA request is interpreted as an acknowledgment for a successful m-BA transmission (see Fig. 6.4(b)). Each m-BA transmission thus takes $T_{m-BA} = (\mathsf{SIFS} + \frac{L_{BA}}{R_1})$ time units to complete, where R_1 is the basic transmission rate as specified in the IEEE 802.11e standard [94]. Let N_{m-BA} be the maximum allowed number of m-BA retransmissions, and $\pi_{m-BA}^{(i)}$ be the probability that an m-BA packet is retransmitted i times before it is successfully received. It follows that the expected time spent in transmitting an m-BA until it is successfully received by the wireless media server is

$$\mathsf{E}\{T_{m-BA}\} = \sum_{i=1}^{N_{m-BA}} (iT_{m-BA})\pi_{m-BA}^{(i)} \tag{6.7}$$

where

$$\pi_{m-BA}^{(i)} = \frac{\epsilon_d^i(1-\epsilon_d)}{1-\epsilon_d^{N_{m-BA}+1}}. \tag{6.8}$$

[1] Surprisingly, the result in (6.5) closely approximates that obtained by the Double Dixie Cup approach [92, 93] as shown below. Our result however requires less computation.

$$\mathsf{E}\{t\} = \int_0^\infty \sum_{i=0}^{K-1} \frac{[(1-\epsilon_u-\frac{1-\epsilon_u}{2^q})\tau]^i}{i!} e^{-(1-\epsilon_u-\frac{1-\epsilon_u}{2^q})\tau} d\tau. \tag{6.6}$$

Hence, given an IP packet of length L, a MAC overhead of length L_o, an encoding vector of length L_e and the data rate R' for the selected 802.11e PHY transmission mode [94], the expected time used to transmit K linearly independent coded packets is

$$\hat{d}_u = T_{set} + \mathsf{E}\{t\}\left(\mathsf{SIFS} + \frac{L_o + L_e + L}{R'}\right) + \mathsf{E}\{T_{m-BA}\} + T_{tear} \tag{6.9}$$

where T_{set} and T_{tear} are fixed time periods spent setting up and tearing down a modified B-ACK cycle, respectively.

We now use the result in (6.9) to derive the end-to-end packet loss/late probabilities. In Section 6.3.2, we first look at the case where only FEC is employed. Later, in Section 6.3.3 we look at the case where both FEC and application-layer ARQ are used.

6.3.2 Packet loss/late probability with FEC only

Our goal in this section is to derive the end-to-end packet lost/late probability using the probability distribution of wireline delay in (6.3) and the average wireless delay in (6.9). As in [73], we combine the packet loss probabilities and the packet delay densities into a single probability measure. That is, while a lost packet is regarded as having an infinite delay, a packet that is not lost is weighted by the probability distribution of its wireline delay. Such a combined probability when FTT D_f is greater than some maximum remaining[2] allowed delay τ of the tagged packet is

$$\begin{aligned} \hat{P}_e \triangleq \Pr\{\text{packet lost/late}\} &= \Pr\{D_f > d_{max} - \hat{d}_u\} = \Pr\{D_f > \tau\} \\ &= \epsilon^*_{uf} + (1 - \epsilon^*_{uf}) \int_{\tau}^{\infty} f_{D_f}(t|\text{ not lost})dt \end{aligned} \tag{6.10}$$

[2]Here *remaining* refers to the maximum delay that the packet is allowed to spend in the wireline IP network prior to its presentation deadline.

where d_{max} is the maximum allowed delay (MAD) of the packet and $\hat{d}_u$, defined in (6.9), is the expected delay incurred in the wireless portion of the network. Variable $\epsilon^*_{uf} = 1 - \left(1 - \epsilon_u - \frac{1-\epsilon_u}{q}\right)(1 - \epsilon_f)$ is the end-to-end packet loss probability prior to error recovery at the destination. Now with systematic (n, k)-RS FEC, the end-to-end packet loss/late probability is

$$\hat{P}_{e,FEC} = 1 - \sum_{j=0}^{n-k} \binom{n}{j} \left(\hat{P}_e\right)^j \left(1 - \hat{P}_e\right)^{n-j}. \tag{6.11}$$

6.3.3 Packet loss/late probability with FEC and application-layer ARQ

In this section we derive the end-to-end packet loss/late probability when application-layer ARQ is enabled in addition to FEC. Define one round trip time (RTT) as the sum of one FTT for data transmission and one BTT for negative acknowledgement. For m application-layer retransmission attempts, the end-to-end packet loss/late probability is the combined probability that the RTT D_{fb} of the tagged packet exceeds its maximum remaining allowed delay $\tau' = d_{max} - (j+1)\hat{d}_u$, in each round j of retransmission. That is, the end-to-end packet loss/late probability is

$$\hat{P}_e^{(m)} \triangleq \Pr\{\text{packet lost/late}\} = \Pr\{D_f > d_{max} - \hat{d}_u\} \prod_{j=1}^{m} \Pr\{D_{fb} > d_{max} - (j+1)\hat{d}_u\} \tag{6.12}$$

where, for each $j = 0, 1, 2, ..., N_{APP}$, the probability $\Pr\{D_{fb} > \tau'\} = \Pr\{D_{fb} > d_{max} - (j+1)\hat{d}_u\}$ is defined as the probability that the RTT D_{fb} is greater than the remaining time allowed for the tagged packet to traverse in the wireline network. That is,

$$\Pr\{D_{fb} > \tau'\} = \epsilon^*_{uf} + (1-\epsilon^*_{uf})\epsilon_{bd} + (1-\epsilon^*_{uf})(1-\epsilon_{bd}) \int_{\tau'}^{\infty} f_{D_f}(t|\text{ not lost}) * f_{D_b}(t|\text{ not lost}) dt \tag{6.13}$$

where "$*$" in the integrator denotes a convolution operator. Now, let $\hat{\pi}_{APP}^{(m)}$ be the probability that m application-layer retransmissions are required to successfully deliver a packet to the destination wireline client, i.e.,

$$\hat{\pi}_{APP}^{(m)} = \frac{(1-\epsilon_1)\epsilon_1^m}{1-\epsilon_1^{N_{APP}+1}} \tag{6.14}$$

where $\epsilon_1 = 1-(1-\epsilon_{uf}^*)(1-\epsilon_{bd})$ and N_{APP} is the maximum number of retransmissions allowed per packet. Then, the probability that a coded packet is lost or late for any possible number of application-layer retransmissions is

$$\hat{P}_{e,APP} = \sum_{m=0}^{N_{APP}} \hat{\pi}_{APP}^{(m)} \hat{P}_e^{(m)}. \tag{6.15}$$

Taking FEC into account, the end-to-end packet loss/late probability in this case is

$$\hat{P}_{e,APP/FEC} = 1 - \sum_{l=0}^{n-k} \binom{n}{l} \left(\hat{P}_{e,APP}\right)^l \left(1-\hat{P}_{e,APP}\right)^{n-l}. \tag{6.16}$$

The result in (6.16) will be the main performance measure we use to evaluate our system in Section 6.6. But before we get to the evaluation, we continue, in the following section, with the analysis of the traditional B-ACK scheme that we will extend from the existing work [95]. Derivations of its corresponding end-to-end performance measures as well as the accuracy of our extended model will be presented.

6.4 System Analysis II: Streaming with Traditional B-ACK

We extend the work in [95] in this section. The goal is to obtain based on [95] an accurate model for an end-to-end packet loss/late probability, that we can use to compare with those in the modified B-ACK case we propose. Using the same approach as that in

Section 6.3, we first quantify the delay incurred from wireless transmissions in Section 6.4.1, and later use it in Section 6.4.3 to derive the packet loss/late probability for the case when FEC and application-layer ARQ are both enabled in Section 6.4.2. We also verify the accuracy of our extended model.

6.4.1 Analysis of Wireless Delay

The expected wireless delay when the traditional B-ACK scheme is employed is derived in this section. Without wireless fountain coding, a communicating AP-wireless media server pair must resort to a selective acknowledgement scheme as described in Section 6.2.3.

Expected number of BAR/BA exchanges

According to [95], the number of time slots t required to successfully deliver K packets to the AP/gateway, assuming perfect BAR/BA exchanges, is given by

$$\Pr\{t = K + i\} = \sum_{j=1}^{\min(i,K)} \binom{K}{j} \binom{i-1}{j-1} \epsilon_u^i (1-\epsilon_u)^K \tag{6.17}$$

where i is the number of extra time slots required. Depending on the number of time slots $t = K + i$ required, the number of BAR/BA exchanges may vary. We show for example in Fig. 6.5 for the case when $K = 3$ and $i = 0, 1, 2, 3$.

Assuming error-free BAR/BA exchanges, we first find the expected number of such BAR/BA exchanges required for delivering K packets successfully to the AP/gateway as a function of i. We denote this function by $\bar{f}(i)$. The derivation of $\bar{f}(i)$ is straightforward

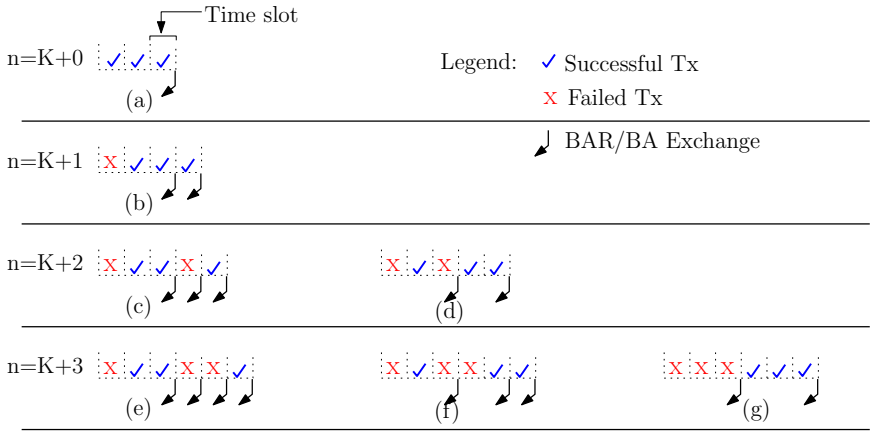


Figure 6.5: Depending on K and i, the number of BAR/BA exchanges as well as the patterns they occur may vary. This example shows all the possible patterns of BAR/BA exchanges for $K = 3$ and $i = 0, 1, 2, 3$.

but rather tedious. We state only the result as follows:

$$\bar{f}(i) = \begin{cases} 1, & \text{for} \quad i = 0 \\ \sum_{v=1}^{\min(i,K)} \dfrac{\sum_{l=1}^{|\mathcal{G}_{i,v}|} g_i^{(l)}(K,v)\langle g_i^{(l)}(K,v)\rangle}{\binom{K+i-1}{i}}, & \text{for} \quad 1 \le i \le N_{max} - K \end{cases} \tag{6.18}$$

where the function $g_i^{(l)}(x_1, y_1) = \prod_{j=1}^{M} \binom{x_j}{y_j}$ represents a BAR/BA-exchange pattern. By "pattern" we mean *how many* and *when* BAR/BA exchanges have occurred over a B-ACK cycle. Each pattern occurs with a probability equal to the value of the function $g_i^{(l)}(x_1, y_1)$ itself. For a BAR/BA-exchange pattern to be valid, the following conditions must hold: (1) $\sum_j y_j = i$; (2) $\sum_j x_j = K + i$; (3) $y_j \le x_j$; (4) $x_{j+1} = y_j$, $1 \le j \le M - 1$; (5) $y_j \le y_{j+1}, 1 \le j \le M - 1$; and (6) $y_M = 0$, $M = \langle g_i^{(l)}(x_1, y_1)\rangle$. We define $\langle g_i^{(l)}(x_1, y_1)\rangle$ as the number of BAR/BA exchanges required when additional i time slots are needed to deliver x_1 packets to the AP/gateway, given that y_1 packets have failed in the first K time

slots. For each i and v in (6.18), we also define a set

$$\mathcal{G}_{i,v} = \left\{ g_i^{(l)}(K,v) : \sum_{l}^{|\mathcal{G}_{i,v}|} g_i^{(l)}(K,v) = \binom{K}{v}\binom{i-1}{v-1} \right\}$$

to describe a collection of BAR/BA-exchange patterns, and $|\mathcal{G}_{i,v}|$ to represent the cardinality of that set.

Expected delay of BAR/BA exchanges

Once we obtain $\bar{f}(i)$, we still need to determine how long it takes for the $\bar{f}(i)$ BAR/BA exchanges to complete when both BAR and BA packets themselves are subject to error. Consider first the case when a BAR is lost. Every BAR expects a BA from the AP/gateway within an SIFS period. If a BA is not received within this period, the wireless media server retransmits a new BAR immediately. The time it takes for a BAR of length L_{BAR} bits to be transmitted, lost, retransmitted once and successfully acknowledged by the AP/gateway through a BA, is

$$T_{BAR} = 2\frac{L_{BAR}}{R_1} + 2 \cdot \mathsf{SIFS} + \frac{L_{BA}}{R_1}. \tag{6.19}$$

On the other hand, a BA is considered to be *lost* if the wireless media server fails to receive it within SIFS. In such a case, the wireless media server reacts by retransmitting a BAR until the BAR is successfully acknowledged or the maximum number of BAR retransmissions, N_{BAR}, is reached. The time it takes for a single exchange of BAR/BA to complete with exactly one BAR retransmission is

$$T_{BA} = 2\frac{L_{BAR}}{R_1} + 3 \cdot \mathsf{SIFS} + 2\frac{L_{BA}}{R_1}. \tag{6.20}$$

The combined delay of these two events is $T_{BACK} = \epsilon_u T_{BAR} + \epsilon_d T_{BA}$. If we let $\pi_{BAR}^{(j)}$ be the probability that a BAR is retransmitted j times before an exchange of BAR/BA is successful, then it follows that on average the combined delay is

$$\mathsf{E}\{T_{BACK}\} = \sum_{j=1}^{N_{BAR}} (jT_{BACK})\pi_{BAR}^{(j)} \tag{6.21}$$

where

$$\pi_{BAR}^{(j)} = \frac{(\epsilon_{ud})^i(1-\epsilon_{ud})}{1-(\epsilon_{ud})^{N_{BAR}+1}}. \tag{6.22}$$

Expected wireless link-layer delay

Finally, the average time required for a successful delivery of K packets, each of length L bits, to the AP/gateway is

$$d_u = T_{set} + \mathsf{E}\{t\}\left(\mathsf{SIFS} + \frac{L_o + L}{R'}\right) + \bar{f}\left(\lceil \mathsf{E}\{t\} \rceil - K\right)\mathsf{E}\{T_{BACK}\} + T_{tear} \tag{6.23}$$

where $\mathsf{E}\{t\}$ is obtained in a similar manner to (6.5) with $\Pr\{t = K + i\}$ replaced by (6.17). All other parameters preserve their original meanings as defined in (6.9).

6.4.2 Packet Loss/Late Probability

Using the result from (6.23), we now derive the end-to-end packet loss/late probability. We assume that both application-layer retransmission and FEC are used as a means for error control/recovery in this case. A negative acknowledgement (N-ACK) is used to inform the wireless media server when FEC fails to recover an original packet. The lost packet will assume highest priority among those packets in the normal application-layer transmission buffer and must be transmitted first.

Using the same approach as that in Section 6.3.3 and Section 6.3.2, we combine the packet loss probabilities and the packet delay densities into a single probability measure. Such a combined probability when RTT D_{fb} is greater than some maximum remaining allowed delay τ of a tagged packet is

$$\Pr\{D_{fb} > \tau\} = \epsilon_{uf} + (1-\epsilon_{uf})\epsilon_{bd} + (1-\epsilon_{uf})(1-\epsilon_{bd}) \int_{\tau}^{\infty} f_{D_f}(t|\text{ not lost}) * f_{D_b}(t|\text{ not lost})dt \tag{6.24}$$

where ϵ_{uf} and ϵ_{bd} are defined in Section 6.2.5. For m application-layer retransmission attempts, the probability that the tagged packet is lost or late for presentation is

$$P_e^{(m)} \triangleq \Pr\{\text{packet lost/late}\} = \Pr\{D_f > d_{max} - \frac{d_u}{K}\} \prod_{j=1}^{m} \Pr\{D_{fb} > d_{max} - (j+1)\frac{d_u}{K}\} \tag{6.25}$$

where

$$\Pr\{D_f > d_{max} - \frac{d_u}{K}\} = \epsilon_{uf} + (1-\epsilon_{uf}) \int_{d_{max}-\frac{d_u}{K}}^{\infty} f_{D_f}(t|\text{ not lost})dt \tag{6.26}$$

is the probability that the tagged packet is lost or late when no application-layer retransmission ($j = 0$) is used. The term d_u/K can be interpreted as the average delay each of K packets in the block experiences in the wireless channel. Now let $\pi_{APP}^{(m)}$ be the probability that m application-layer retransmissions are required to deliver a packet successfully to the destination wireline client, i.e.,

$$\pi_{APP}^{(m)} = \frac{(1-\epsilon_0)\epsilon_0^m}{1-\epsilon_0^{N_{APP}+1}} \tag{6.27}$$

where $\epsilon_0 = 1-(1-\epsilon_{uf})(1-\epsilon_{bd})$ and N_{APP} is the maximum allowed number of application-layer retransmissions. Then, the probability that a packet is lost or late for any possible number of application-layer retransmissions is given as

$$P_{e,APP} = \sum_{m=0}^{N_{APP}} \pi_{APP}^{(m)} P_e^{(m)}. \tag{6.28}$$

Finally, taking FEC into account, the end-to-end packet loss/late probability is

$$P_{e,APP/FEC} = 1 - \sum_{l=0}^{n-k} \binom{n}{l} P_{e,APP}^{l} (1 - P_{e,APP})^{n-l} \tag{6.29}$$

where n and k are the usual parameters of the RS-FEC code defined in Section 6.2.1.

6.4.3 Accuracy of Extended Model

We have derived the end-to-end packet loss/late probabilities for three major cases: (1) Modified B-ACK with application-layer FEC, (2) Modified B-ACK with application-layer FEC and ARQ, and (3) Traditional B-ACK with application-layer FEC and ARQ. In case (3), we have extended the analysis in [95] to account for the situations when BA and BAR are subject to errors. This extension is necessary so that in the evaluation we can make fair comparison between the three cases.

To verify the accuracy of our extended model, we ask how our extended models are close, in terms of end-to-end packet loss/late probability, to the original model and when? We plot in Fig. 6.6 these end-to-end packet loss/late probabilities against the fixed packet loss rates of the wireless link for all the three cases. The first extended model, labeled as **Extended Model + APP-Layer ARQ**, corresponds to (6.28), while the second extended model, labeled as **Extended Model + FEC + APP-Layer ARQ**, corresponds to (6.29). For the original model [95], the end-to-end packet loss/late probability is obtained in the same manner as (6.29) *but* with the expected wireless delay d_u given by

$$d_u = T_{set} + \mathsf{E}\{t\} \left(\mathsf{SIFS} + \frac{L_o + L}{R'} \right) + T_{tear} \tag{6.30}$$

where $\mathsf{E}\{t\}$ is calculated in a way similar to that for (6.5) with $\Pr\{t = K + i\}$ replaced by (6.17). Compared to (6.23), the term $\bar{f}(\cdot)$, which accounts for erroneous BAR/BA

exchanges, is missing from (6.30) because the BAR/BA exchanges in the original model are perfect. We label the end-to-end packet loss/late probability for this case by **Chen et al. Model [95] + FEC + APP-Layer ARQ** which is also plotted in Fig. 6.6. We use the 802.11e parameters in Table 6.1 to plot the above three results in Matlab. The packet loss rates on both downlink and uplink wireless channels are set equal for each data point.

We see from Fig. 6.6 that the model **Extended Model +FEC + APP-Layer ARQ** closely follows the original model [95] (**Chen et al. Model [95] + FEC + APP-Layer ARQ**) when the packet loss rates on the wireless link are relatively low. That is, with low packet loss rates, BAR/BA exchanges are also less likely to fail. The two models then almost resemble in terms of end-to-end packet lost/late probability. However, when the packet loss rates on the wireless links are relatively high, the end-to-end packet loss/late probability of our extended model diverts substantially from that of the original model [95]. This behavior can be expected since in our extended model we assume that both BA and BAR are subject to errors. But such assumptions were not made in the original model. The increased end-to-end packet loss/late probability in our extended model (**Extended Model +FEC + APP-Layer ARQ**) thus accounts for increased delay caused by erroneous exchanges of BA and BAR as quantified in Section 6.4.1. The comparison between the two extended models is clear: without FEC, the first extended model is inferior to the second.

In the following we will restrict our attention to the extended model and the models we have developed for the modified B-ACK case.

Table 6.1: Parameters for validation of the extended model.

IEEE 802.11e parameters	
Basic rate for control signal R_1	6 Mbps
Data rate R'	11 Mbps
Length of block acknowledgment (BA) packet L_{BA}	152 bytes
Max. no. BA (modified BA) packets per block $N_{BA}(N_{m-BA})$	4
Length of BA response (BAR) packet L_{BA}	42 bytes
Max. no. BAR packets per block N_{BAR}	4
Max. no. app-layer retransmissions N_{APP}	1
Slot time	20μs
SIFS	10μs
Block size K	7μs
Channel parameters	
$\Delta_f = \Delta_b$	25 ms
$\lambda_f = \lambda_b$	$(12.25 \text{ ms})^{-1}$
$\alpha_f = \alpha_b$	4
Wireless packet error rates ϵ_u and ϵ_d	varies in $(0, 0.35]$
Wireline packet error rates ϵ_f and ϵ_b	0.01

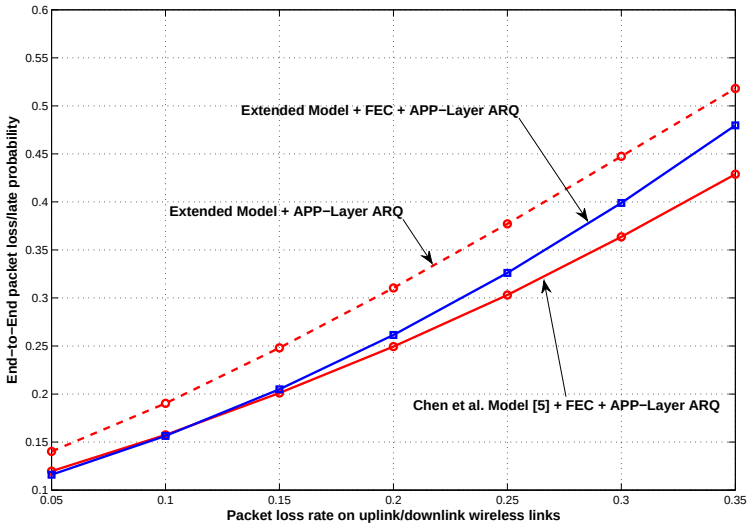


Figure 6.6: Comparison between the extended model and the original model developed in [95].

6.5 Performance Evaluation Framework

In this section we describe the necessary components we use to evaluate our system. We first explain in Section 6.5.1 how video data is packetized and FEC-protected. Later in Section 6.5.2 we briefly describe how we map a packet loss/late probability to video distortion. Section 6.5.3 gives the details of our simulation platform.

6.5.1 Packetization of Video Data

Fig. 6.7(a) shows how a video sequence is packetized in our video streaming system. A video sequence consists of multiple groups of pictures (GOP). Assuming that only intra-coded (I-) and predictively coded (P-) frames are used, each GOP comprises of one I-frame

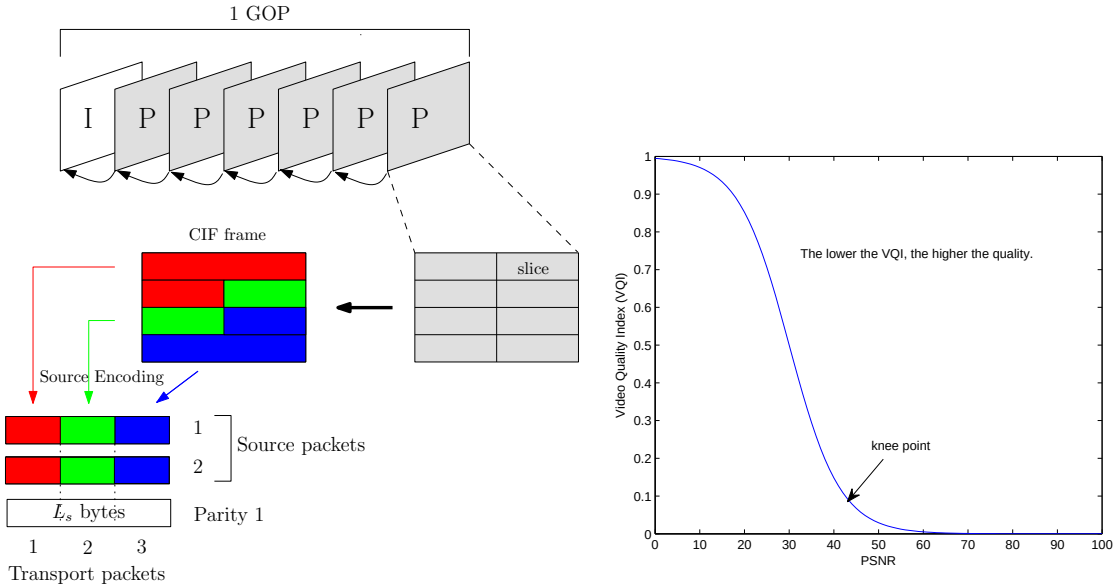


Figure 6.7: (a) Packetization and FEC: The variable number of slices may be mapped into each transport packet because each slice contains the varying number of data bits that need to be encoded. We show a particular instance where a "green" slice contains more data bits than any other slices. Systematic RS code is applied to a group of $k = 2$ source packets to produce $n = 3$ transport packets in this example. (b) Mapping PSNR to VQI as suggested in [89]: A case of the *Foreman* video sequence.

and multiple P-frames. Each frame is divided into multiple slices, each of which is defined as a block of independently coded pixels. Each video frame is encoded into k source packets, all of which are to be transmitted using n transport packets by applying the (n, k) systematic RS code.

6.5.2 Loss-Distortion Model

Video quality is jointly affected by both network-dependent and application-specific parameters. While packet losses and delay jitters are main causes from the network perspective, video codecs, source coding rate, packetization schemes, and content characteristics (e.g., fast/slow moving pictures) are some major, representative causes from the application perspective. Recently, Tao et al. [88] have proposed the loss-distortion model that captures a wide range of these application-specific and network-dependent factors. We adopt this model however only to capture the network-dependent factors. The term *distortion* in our context thus refers to video distortion resulting from the transmission impairments in the network only.

The main idea behind this model is to map packet losses to the quality of predictively encoded video sequences reconstructed by the receiver. According to [88], the overall distortion caused by $\overline{\eta}$ consecutive packet lost, on average, in a single loss event[3] is given by

$$\overline{\mathcal{D}} = \mathcal{SL} \cdot P_e \overline{\eta} \cdot \mathcal{D}_1 \tag{6.31}$$

where $\mathcal{S}$ is the number of slices in each packet, $\mathcal{L}$ is the number of packets per frame, and $\mathcal{D}_1$ is the total average distortion incurred when a single slice is lost. The probability P_e in (6.31) is the packet loss probability which can be approximated as derived in (6.11), (6.16), and (6.29). The distortion of video can be mapped to the conventional measure such

[3]We assume throughout this chapter that a single loss event results in only one packet being lost or late, i.e., $\overline{\eta} = 1$. Although this assumption seems oversimplified at first, the experiments performed in [88] confirm that the parameter $\overline{\eta}$ fluctuates narrowly in the range $[1.02, 2.08]$.

as peak signal-to-noise ratio (PSNR) as

$$\mathsf{PSNR} = 10 \log_{10} \frac{255^2}{\overline{\mathcal{D}}}. \tag{6.32}$$

To evaluate the subjective quality of a particular video sequence perceived by an average human eye, we can translate PSNR into subjective video quality index (VQI) as [89]

$$\text{VQI} = \frac{b_1}{1 + \exp(b_2(\mathsf{PSNR} - b_3))} \tag{6.33}$$

where parameters b_1, b_2, and b_3 depend on the video characteristics of a video sequence under consideration [89]. As an example, we show in Fig.7.3(b) how the PSNR of a *Foreman* video sequence ($b_1 = 1, b_2 = 0.175$, and $b_3 = 30$) is mapped to its VQI which varies from 0 (best quality) to 1 (worst quality). Different video sequences will have different VQI mappings. This implies that even video sequences with the same average PSNR are not necessarily perceived equally by an average human eye. Moreover, this mapping also suggests that PSNR only reflects subjective VQI in a certain range. Further increases in PSNR beyond the "knee point" do not translate into subjective quality improvements perceivable by an average human eye.

6.5.3 Simulation Framework

As shown in Fig. 6.8, our simulation framework consists of tools from the Video Quality Evaluation Tool-set (EvalVid) [97] and the ns-3 network simulator [98]. While the EvalVid toolset takes care of the video coding and decoding at the source and the destination, respectively, the ns-3 simulator involves several network-related components. We explain our simulation framework according to the tasks performed by the source, the network, and the destination as follows:

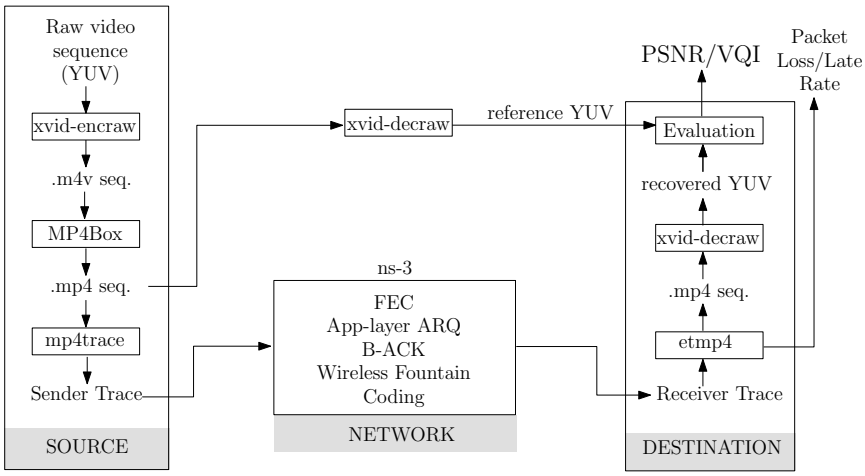


Figure 6.8: Simulation framework.

Source

At the source, a raw YUV video sequence is encoded using the **xvid-encraw** encoder [99]. The encoded video is next encapsulated in an mp4 video sequence using **MP4Box** [99]. Using this mp4 video sequence, an application-level sender trace containing frame ID, frame sizes, playout times, frame types and so on, is next generated (using **mp4trace** [97]) and sent as input to the ns-3 simulator for network simulations.

Network

In the ns-3 simulator, we implement a module that translates the application-level sender trace received from the **mp4trace** tool into a network-level trace containing packet ID, packet sizes, packet interarrival times, playout times from which we derive the MAD. Each packet is also assigned the maximum number of application-layer retransmissions, N_{APP}. After tagging each packet with MAD and N_{APP}, an FEC module [101] performs RS coding across a block of k packets to produce n transport packets. These FEC-protected

packets are passed down to the network-layer at which UDP/IP headers are added. Just above the IEEE 802.11e MAC layer we implement a wireless fountain coding function based on the ns-3 *wifi-devices* module. After coding, each coded packet is prepended with a wireless fountain coding header which is a vector of coding coefficients $c_{i,j}$ as described in Section 6.2.2. We integrate this wireless fountain coding function with the existing IEEE 802.11e B-ACK module available in the *ns-3-blockack* repository. On the receiving side, the AP/gateway equipped with a decoding function uses the coefficients in the wireless fountain coding header to decode the coded packets pertaining to a current block of $K(\leq n)$ packets before forwarding the decoded packets to the wireline destination. Packets are routed in the wireline network by the *ns-3 global routing* function which performs pre-simulation static route computation on a layer-3 IPv4 topology. Once a topology has been constructed and addresses assigned, the routing database will be initialized, and the static unicast forwarding tables will be created for each wireline node.

Destination

At the destination, we modify the **etmp4** tool to translate a receiver trace to a probably distorted mp4 video sequence. At this stage, a corresponding packet loss/late rate is calculated. The **xvid-decraw** decoder [99] is then used to recover the raw YUV video sequence from the probably distorted mp4 video sequence. At the same time, the mp4 video sequence is also sent from the source to another **xvid-decraw** decoder for creating a reference YUV raw video sequence. This step of re-creating the reference YUV raw sequence is necessary to ensure that the video distortion caused by the decoder itself is not taken into account in the evaluation step as the received mp4 video sequence must also go through

the same process. The evaluation takes place offline and essentially involves calculating the corresponding PSNR and VQI. The offline video evaluation is performed by comparing the received raw YUV sequence with the reference raw YUV sequence.

For the traditional B-ACK transmission, no wireless fountain coding is implemented.

6.6 Simulation Results

In this section we evaluate the performance of uplink video streaming in a typical combined wireline/802.11 network. We first validate our analytical models in Section 6.6.1. We then study, in Section 6.6.2, how packet loss/late probabilities and corresponding PSNRs are affected by block sizes K and FEC protection strength n/k. Later, this study will lead us to see how the conditions of the end-to-end forward and end-to-end backward channels can either boost or deteriorate the performance of video streaming. Using this observation, we propose in Section 6.6.3 a simple algorithm that lets the wireless media server and the receiving AP/gateway switch between the modified B-ACK strategy and the traditional B-ACK strategy to improve the performance of video streaming.

6.6.1 Model Validation

We first validate our performance models by simulating the uplink network shown in Fig. 6.9 in which each point-to-point wireline link is modeled as a packet eraser channel with a bandwidth of 10 Mbps. In the wireless link, packets are erased across time and the link capacity is fixed at 11 Mbps. The CSMA channel capacity of the LAN is 100 Mbps.

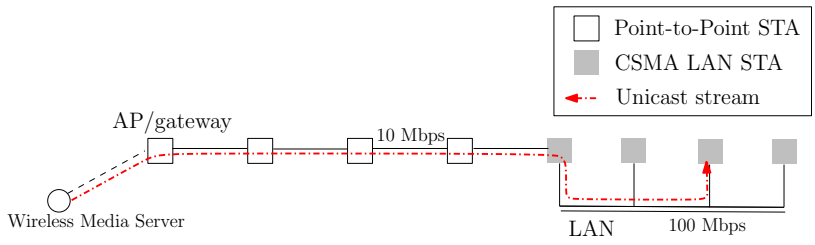


Figure 6.9: Uplink media streaming in a combined wireline/802.11 network.

The video sequence simulated is of H.264/MPEG-4 AVC Common Intermediate Format (CIF). MAD is fixed at 130 ms. Otherwise specified, the simulation parameters shown in Table 6.2 are used throughout the rest of this chapter.

The goal is to stream the *Foreman* video sequence from a stationary wireless media server to one of the wireline clients on the LAN using the following three strategies:

1. Traditional 802.11e B-ACK with both FEC and application-layer ARQ enabled.

2. Modified B-ACK with FEC enabled but application-layer ARQ disabled.

3. Modified B-ACK with both FEC and application-layer ARQ enabled.

The block size K of packets to be coded in the modified B-ACK scheme – or block-acknowledged in the traditional IEEE 802.11e B-ACK scheme – varies between 7-12 depending on the amount of information being encoded in each video frame. For wireless fountain coding operations, we fix the size of the finite field at 2^8. The packet size at the application level is 1024 bytes.

The accuracy of our models is shown in Fig. 6.10 where we plot both the simulation and analytical results for the end-to-end packet loss/late probabilities as functions of FEC protection strength n/k for the above three strategies. The simulation results are taken at

Table 6.2: Simulation parameters.

IEEE 802.11e parameters	
Basic rate for control signal R_1	6 Mbps
Data rate R'	11 Mbps
Length of block acknowledgment (BA) packet L_{BA}	152 bytes
Max. no. BA (modified BA) packets per block $N_{BA}(N_{m-BA})$	4
Length of BA response (BAR) packet L_{BA}	42 bytes
Max. no. BAR packets per block N_{BAR}	4
Max. no. app-layer retransmissions N_{APP}	1
Slot time	20μs
SIFS	10μs
Channel parameters	
$\Delta_f = \Delta_b$	25 ms
$\lambda_f = \lambda_b$	$(12.25 \text{ ms})^{-1}$
$\alpha_f = \alpha_b$	4
Wireless packet error rates ϵ_u and ϵ_d	varies in $(0, 0.3]$
Wireline packet error rates ϵ_f and ϵ_b	varies in $(0, 0.1]$
Video parameters	
Group of pictures (GOP)	30
No. frames per second	30
No. packets per frame $\mathcal{L}$	4
No. slices in each packet $\mathcal{S}$	2
No. consecutive packets lost on average in a single loss event $\bar{\eta}$	1

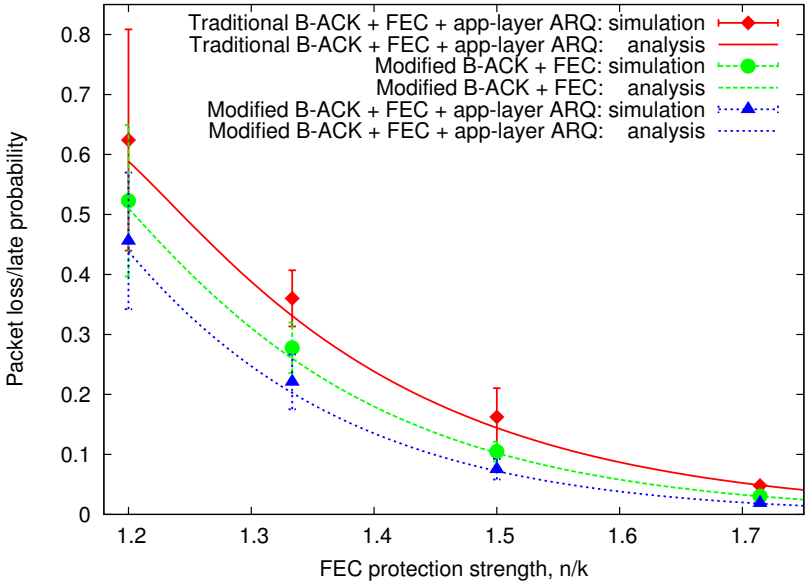


Figure 6.10: Model validation: Loss/late probability as a function of FEC protection strength, n/k.

discrete values of n/k, i.e., at $12/10, 12/9, 12/8$, and $12/7$. The higher the value of n/k, the better the FEC protection strength. For each value of n/k, the packet loss/late probability is averaged across 30 independent simulation runs each of which streams the *Foreman* video from the wireless media server to one of the wireline clients in the LAN. For each averaged point, a 95% confidence interval is asserted.

From Fig. 6.10, we see that the simulation results follow the same trend as the analytical ones. Intuitively, the packet loss/late probabilities decrease as more FEC protection is added to packets for all the three strategies. Even with FEC only, our modified B-ACK scheme still outperforms the traditional B-ACK scheme by up to 15%. The modified B-ACK scheme with both FEC and application-layer ARQ enabled leads to further improvements by up to 25%. As the FEC protection grows higher, however, our modified

B-ACK strategies perform only slightly better than the traditional B-ACK strategy. This is intuitive. With higher FEC protection strength, the client resorts less to retransmissions. A variable network delay introduced by the retransmission mechanisms, as opposed to a constant delay produced by FEC, is then reduced or eliminated. This in turn lessens the role of wireless fountain coding as part of the modified B-ACK scheme because the delay benefit offered by wireless fountain coding may no longer be needed to compromise the variable network delay.

At lower FEC protection strength, we also observe that the simulated end-to-end packet loss/late probabilities fluctuate in a wider range than those at higher FEC protection strength. This should come at no surprise because when FEC protection strength is weak, all the three strategies resort to application-layer retransmissions more often, resulting in a variable network delay which in turn translates into the highly fluctuating levels of the end-to-end packet loss/late probabilities.

From Fig. 6.10, we also see that the analytical results are slightly more optimistic than the simulation ones. The reason is due to the fact that in the simulations the media packets are transmitted in non-promiscuous mode once entering the CSMA LAN. A slight delay therefore occurs as all the LAN devices have to (1) peek into the media packets to determine whether the packets are addressed to them, and if not so, (2) pass the packet onto the next LAN device until the correct destination network address is reached. This delay is variable and not included in our model.

In the following, we will focus only on the first and third strategies for fair comparison.

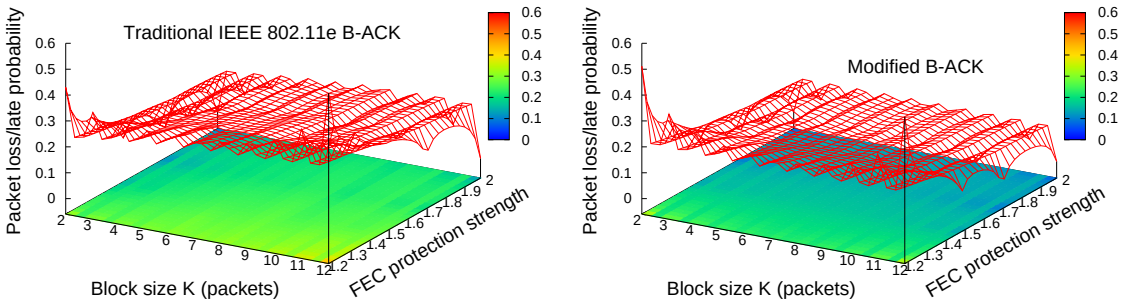
6.6.2 Effects of block size K and FEC protection strength n/k

In this section, we study how the packet loss/late probabilities and the corresponding PSNRs are affected by block size K and FEC protection strength n/k. Later, this study will lead us to propose a hybrid scheme that takes advantage of both the modified and traditional B-ACK schemes as the conditions of the end-to-end forward and backward channels change. This hybrid scheme is presented in Section 6.6.3.

We simulate uplink video streaming in the network shown in Fig. 6.9. The *Foreman* video sequence is used in each simulation run. The simulation results are averaged over 30 simulation runs. The maximum allowed delay (MAD) of each packet is fixed at 120 ms. We plot end-to-end packet loss/late probabilities and their corresponding PSNRs as 3D functions of block size K and FEC protection strength n/k in Fig. 6.11 and Fig. 6.12, respectively. The simulation results from the two transmission strategies, i.e., modified and traditional B-ACK, are plotted side by side in each figure. For both cases, the application-layer ARQ is enabled. In particular, we fix the maximum number of application-layer retransmissions per packet $N_{APP} = 1$.

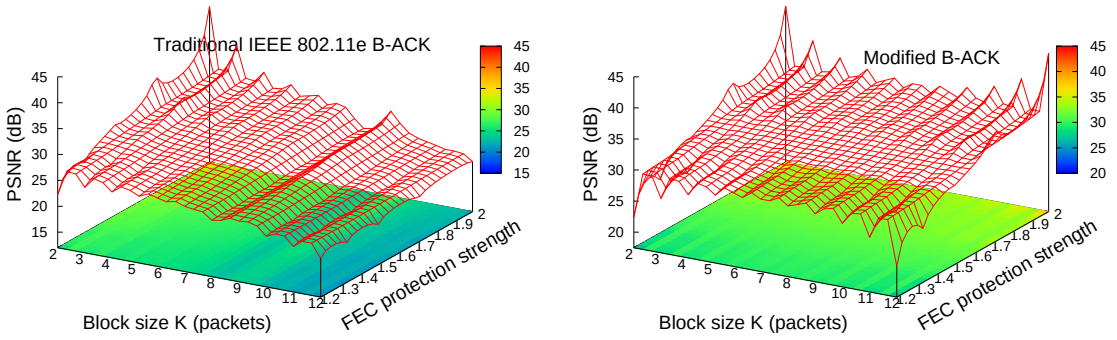
Fig. 6.11 and Fig. 6.12 suggest that media streaming with the modified B-ACK mechanism outperforms media streaming with the traditional IEEE 802.11e B-ACK in terms of end-to-end packet loss/late probability and PSNR, respectively, for most combinations of K and n/k values. For a fixed block size K, the packet loss/late probability and PSNR improve as FEC protection strength increases for both B-ACK schemes. We see this trend in Section 6.6.1.

For fixed FEC protection strength ($n/k = 1.714$), we take a snapshot of Fig. 6.11



(a) Traditional IEEE 802.11e B-ACK (b) Modified B-ACK

Figure 6.11: Simulation results: Packet loss/late probability as a function of block sizes K and FEC protection strength n/k.



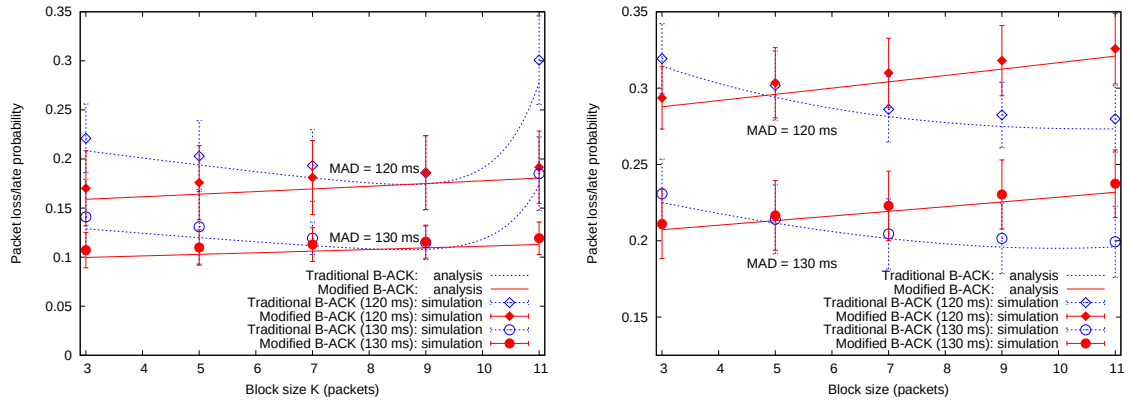
(a) Traditional IEEE 802.11e B-ACK (b) Modified B-ACK

Figure 6.12: Simulation results: PSNR as a function of block sizes and FEC protection strength n/k.

and show the comparison between the traditional and modified B-ACK schemes in Fig. 6.13(a). The results for MAD = 130 ms are also shown. For the two different values of MAD, the modified B-ACK scheme is shown to outperform the traditional B-ACK scheme for all block sizes K. From our extensive simulations, however, we observe that this is true only when the end-to-end forward channel is better than the end-to-end backward channel (i.e., when $\epsilon_{uf} < \epsilon_{bd}$). This implies that the delay benefit offered by wireless fountain coding in the modified B-ACK translates into substantial improvements in the end-to-end packet loss/late probability. This probability however increases slightly as block size K increases. This is because with a bigger K the decoding delay incurred at the AP/gateway increases, which in turn causes the end-to-end packet loss/late probability to increase.

On the other hand, if the backward channel is as good as or better than the forward channel (i.e., when $\epsilon_{bd} \leq \epsilon_{uf}$), we see in Fig. 6.13(b) that the traditional B-ACK outperforms the modified B-ACK for a certain range of block sizes K ($K = 5 - 11$). The results in Fig. 6.13(b) are intuitive. When the backward channel is as good as or better than the forward channel, it is better to take advantage of the backward channel through the link-layer retransmission strategy of the traditional IEEE 802.11e B-ACK scheme which requires frequent control signals through the downlink wireless channel. If, on the other hand, the backward channel is more severe, it is better to turn off the traditional B-ACK and resort to the Modified B-ACK which relies mostly on the forward wireless channel. Only a few control signals through the downlink wireless channel are required for the modified B-ACK.

Having observed how the backward and forward channels affect the performance



(a) (b)

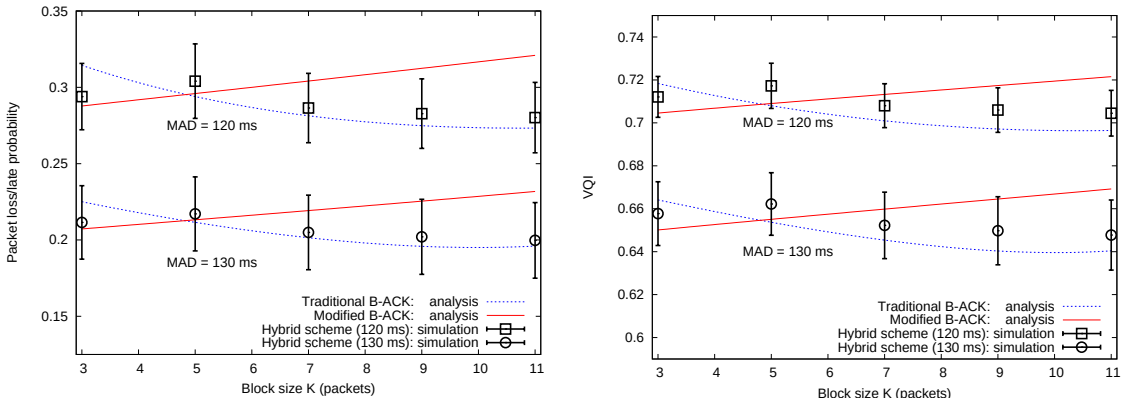
Figure 6.13: Packet loss/late probability as a function of block sizes for fixed FEC protection strength: (a) When the forward channel is better than the backward channel, Modified B-ACK outperforms Traditional B-ACK, but (b) When the backward channel is as good as or better than the forward channel, however, Modified B-ACK prevails only for a certain range of block size K.

of video streaming, our next objective is to devise a hybrid scheme that can switch between the modified and traditional B-ACK schemes whenever the condition of the forward channel is getting more severe than the backward channel. This switching is desirable so that the packet loss/late probability is kept as low as possible at all levels of block size K in use. We propose this hybrid scheme in the following section.

6.6.3 The Hybrid Scheme

Given the conditions of the wireless and wireline channels, our goal is to keep the end-to-end packet loss/late rates as low as possible over a range of block sizes K, as shown in Fig. 6.13(b). Since the AP/gateway is in connection with both the wireless and the wireline network, we propose that the AP/gateway monitors the conditions of both the wireless and wireline channels. In practice, the AP/gateway may estimate the conditions of the channel by sending out probe packets and then infer various characteristics of the channels such as loss or delay from such probes. Various versions of on-line channel estimation using a probing technique can be found in [80–82]. In particular, while the approach in [80] focuses on end-to-end loss estimation, those approaches in [81,82] primarily focus on loss estimation on individual links.

The AP/gateway will dictate when to use the modified B-ACK or the traditional B-ACK based on the conditions of the channels it is monitoring. However, since the conditions of the wireline channel are rather stable, it turns out that our hybrid scheme is called into action only when the condition of the uplink *wireless* channel is worse than that of the downlink channel. When the hybrid scheme is called into action, the AP/gateway makes a decision when to use or not to use the modified B-ACK (wireless fountain coding) based on

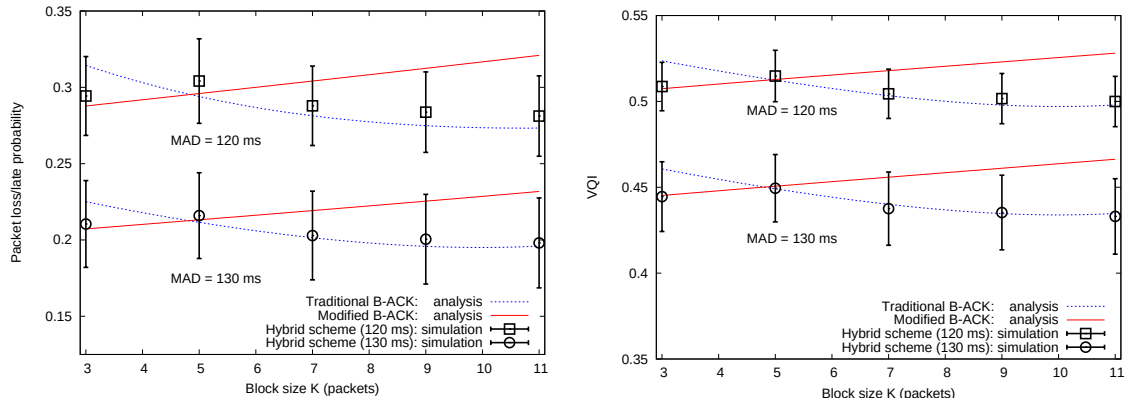


(a) (b)

Figure 6.14: Performance of the hybrid scheme when the *Foreman* video is transmitted: (a) Packet loss/late probability and (b) Video quality index (VQI).

the transmission status of a current block of packets as well as on the block size K in use. Recall that K is constrained by n, the FEC parameter. The receiving AP/gateway will instruct the wireless media server to switch from the modified B-ACK to the traditional B-ACK, or vice versa, only if the transmission of the current block of K packets is complete.

Figs. 6.14-6.15 show the simulation results of the hybrid scheme when applied to the network shown in Fig. 6.9 for the *Foreman* and *Mother & Daughter* video streams, respectively. Also shown for comparison in each plot are the analytical results corresponding to the traditional and modified B-ACK schemes. Two different MADs of 120 and 130 ms are considered. For all the cases, we fix the FEC protection strength n/k at 1.714. All the simulation results are averaged over 30 simulation runs. A 95% confidence interval is asserted. Video quality index (VQI) is used here to show the subjective quality of the two different video sequences perceived by an average human eye.



(a) (b)

Figure 6.15: Performance of the hybrid scheme when the *Mother and Daughter* video is transmitted: (a) Packet loss/late probability and (b) Video quality index (VQI).

We observe that the packet loss/late probabilities resulting from the hybrid scheme closely follow the possible lowest packet loss/late probabilities predicted by the analytical model. The simulation results however are slightly more pessimistic because the switching from the modified B-ACK to the traditional B-ACK, or vice versa, occurs only if the delivery of the current block of K packets is complete. This will incur some variable delay. A similar observation can also be made with the VQI results shown in Figs. 6.14(b) and 6.15(b). Interestingly, even though the packet loss/late rate performances of the *Foreman* and *Mother & Daughter* video sequences are very similar, their corresponding subject quality as perceived by an average human eye are substantially different. This is because the characteristics of both videos are quite different. That is, while the main object (foreman) in the *Foreman* video is fast-moving, the main object (mother and daughter) in the *Mother & Daughter* video stays still most of the time.

6.7 Conclusion

We have developed two major performance models for uplink media streaming in a combined wireline/WiFi network: Streaming with Modified B-ACK and streaming with Traditional B-ACK. In each model, we have derived the end-to-end packet loss/late probability that expresses the relationship between several parameters pertaining to FEC, ARQ, MAC-layer B-ACK mechanisms. The accuracy and benefits of our models have been illustrated through extensive simulations.

Chapter 7

Downlink Media Streaming with Wireless Fountain Code

In this chapter, we address the problem of streaming packetized media data in a combined wireline/802.11 network where media packets, subject to errors on both wireless and wireline channels, are streamed from wireline media servers to a set of wireless clients associated with a particular AP/gateway of the underlying IWMN, such as shown in Fig. 7.1. Similar to the uplink streaming system discussed in Chapter 6, each wireline media server prestores encoded media data and transmits it on demand to a wireless client(s) for playback in real-time. The client buffers the data and begins playback after a fixed delay. Once the playback begins, the client expects no interruption during the entire presentation of the media. Interruption is however inevitable due to several reasons one of which is insufficient bandwidth of the wireless channel (e.g., due to the time-varying propagation characteristics). Insufficiency in bandwidth normally translates into packet errors and/or late arrivals of no-

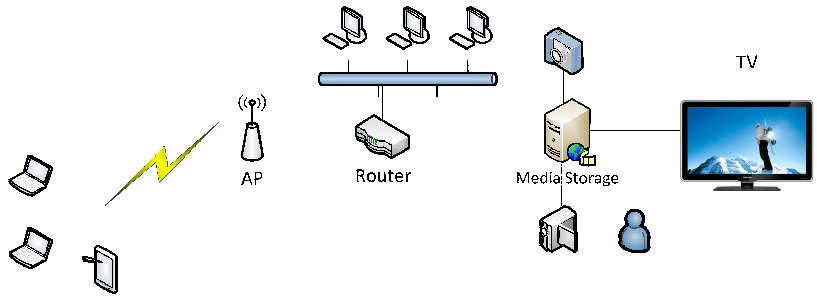


Figure 7.1: Example scenario of in-home downlink streaming.

error packets, which in turn degrades the end-to-end performance of media streaming.

In the literature, several efforts have tried to address this issue from various angles, one of which is to apply network coding [4, 5] to the last-hop wireless link in order to potentially release the bandwidth bottleneck. Network coding arises naturally here to take advantage of the broadcast nature of the wireless downlink channel. To see this let's say the AP/gateway has 2 packets to send to Alice and Bob, one packet for each. Also assume that the link from the AP/gateway to either Alice or to Bob is symmetric and 50% reliable. Using the traditional method of packet-by-packet acknowledgement, each packet would need 4 transmissions to complete (i.e., 2 for the data packet and another 2 for acknowledgement), resulting in the total of 8 transmissions required for the whole system. However, if the AP/gateway is allowed to algebraically mix packets and broadcast the mixed versions to Alice and Bob, only 4 transmissions are required to deliver 4 mixed packets to both users, i.e., 2 mixed packets for each user. Using coding coefficients embedded in the header of each mixed packet, each user can individually recover his/her packet. In the literature, several efforts have exploited this simple scenario in many directions. For example, Nguyen et al. [55] assume that the AP/gateway knows the buffer contents of all wireless receivers in order to make a coding decision through a Markov Decision Process (MDP). In [87], a coding decision is based on the importance and deadlines of video packets as well as the

network states. Like [55], the buffer contents of receiving nodes must be known prior to coding. Coding decisions based on how video data are compressed are studied in [50] for video multicast in a wireless mesh network with application-layer FEC. In [51], mixing of video packets across coding generations is proposed to potentially conceal errors between video frames (inter-frame errors).

Another solution for efficient media streaming is to jointly employ different error control techniques at various layers of the network protocol stack. Two prominent mechanisms, namely, Automatic Repeat reQuest (ARQ) and channel coding with forward error correction (FEC), are normally proposed for media streaming. Although less complex than FEC, employment of ARQ at the application-layer requires a reliable feedback channel. ARQ thus inevitably incurs a variable network delay. FEC, on the other hand, can overcome these drawbacks but only at the expense of transmitting a large amount of redundant data. Many attempts follow naturally, aiming at extracting the best flavors of FEC and ARQ, in a "hybrid" fashion [67–70, 72]. In [68], the authors combine source coding, FEC, and application-layer ARQ to optimize video transmission over lossy networks. This scheme is however not suitable for media streaming because in the streaming system media data must be pre-compressed and pre-stored offline without knowing the state of the channel during transmission.

In [69], FEC and ARQ are combined to control packet errors in wireless video transmission. In making error control decisions, this scheme still resorts to the information, i.e., priority of video packets, offered by hierarchical video (source) coding. In [72], application-layer FEC and link-layer ARQ are jointly considered for one-hop video trans-

mission over 802.11a networks. A per-packet 802.11a acknowledgement scheme is employed. FEC and ARQ are also combined to combat packet errors in wireless networks. Another study that jointly considers FEC and ARQ for wireless video transmission is [70] in which error control decisions are made according to media and channel characteristics. In [67], ARQ and FEC are jointly employed at the application layer for video streaming in a combined wireline/WiFi network. Only a simple per-packet acknowledgement scheme is used to control errors at the wireless link layer. The end-to-end delay and packet loss probability are modeled as functions of ARQ and FEC parameters and used as a basis for a delay-constrained error control algorithm.

In this chapter, we propose that a version of fountain coding [59] – termed also in this chapter as wireless fountain coding (WFC) – be employed at an AP/gateway to efficiently utilize the bandwidth of the wireless channel for delay-sensitive media downlink streaming over a combined wireline/802.11e network. This *wireless fountain coding* is neither exactly random linear fountain coding as described in [59] nor random linear network coding as studied in [7]. It is different from random linear fountain coding in that operations in a finite field need not be binary operations (random XORs) as in random linear fountain coding. Operating in a bigger finite field will increase the probability that combined packets are linearly independent [7]. On the other hand, *wireless fountain coding* is not purely network coding as its coding/decoding operations occur only at endpoints of a one-hop link.

With WFC, the AP/gateway repeatedly broadcasts combined packets to wireless receivers until all the receivers have enough packets to start decoding. For a batch of, say, K packets, how many broadcast transmissions are required to ensure that all intended N

receivers will successfully receive enough packets in order to start decoding? We provide the answer to this question by analytically obtaining the moment generating function of the wireless link-layer delay. We then study how it translates into end-to-end performance measures. In particular, we derive the probability that a media packet is lost or late for presentation and relate it to a video distortion measure. In order to illustrate the benefits of our proposed system, we develop an integrated ns-3/EvalVid simulator to evaluate our performance model in comparison with the system proposed in [67] (i.e., without WFC), which will be referred to as the *traditional scheme*. Via extensive simulations, we observe that streaming with WFC is able to support more concurrent video flows compared to the traditional scheme without WFC. When the deadlines imposed on video packets are relatively stringent, streaming with WFC also shows superior performance in terms of packet loss/late probability, video distortion and video frame delay, over the traditional scheme.

Unlike [55, 87], we assume in our system that the AP/gateway has no knowledge of the buffer contents of intended wireless receivers. Similar to [51, 67], and unlike [50, 68–70], our work also does not assume any specific form of source coding. This is a valid assumption as this is usually the case for a streaming system in which media data are encoded and prestored offline without the benefit of knowing the state of the channel during transmission [73]. No retransmission techniques are implemented in our system, i.e., FEC is the only method used for error control.

The rest of this chapter is organized as follows. Section 7.1 describes the proposed downlink streaming system. In Section 7.2 we express the end-to-end packet loss/late probability in terms of FEC, WFC, and channel parameters. In Section 7.3 we describe the

framework we develop for evaluating the performance of the proposed system. In Section 7.4 we evaluate our performance model in various aspects via simulations. Section 7.5 concludes the chapter.

7.1 System Description

7.1.1 Overview

We consider a media streaming system as shown in Fig. 7.1 where multiple wireline media sources (servers) stream prestored media packets to wireless nodes through a particular AP/gateway. The wireline IP network may consist of desktop computers, routers, or gateways, media servers and so on. Connected to the wireline IP network is a WiFi network which operates according to the IEEE 802.11e standard. The goal of the system is to concurrently deliver unicast and multicast flows from wireline media sources to wireless nodes associated with a particular AP/gateway in the downlink direction. We assume that packets of these unicast and multicast flows are source-encoded and protected by Reed-Solomon (RS) codes at the application layer (which will be explained in Section 7.1.2).

With a maximum allowed delay (MAD) tag and a UDP/IP header added, these packets are then routed across the wireline IP network to the AP/gateway of interest. Arriving at the AP/gateway, these packets are buffered on a per-flow basis just above the MAC layer waiting for a wireless fountain coding (WFC) process, to be explained in Section 7.1.3.

Having no knowledge of packets already buffered at each wireless destination node, the AP/gateway applies WFC to a group of K media packets which have the same MAD

and keeps transmitting the coded packets until it reaches a threshold. Then it proceeds to transmit the next batch of K' packets which may not necessarily equal to K. Wireless fountain coding here can potentially exploit multi-user diversity in the broadcast wireless channel since wireless receptions of the users at different locations can be highly independent depending on the link conditions. The fact that the AP/gateway resorts to a broadcast transmission mode implies that neither link-layer nor application-layer retransmission mechanisms are available. The only error recovery mechanism available is FEC. That is, while WFC will potentially reduce the delay in the wireless links, FEC will help counter packet losses at a higher layer.

At the MAC layer, an IEEE 802.11e MAC header is next added to each coded packet to form a MAC Protocol Data Unit (MPDU) before it is broadcast to those wireless nodes addressed by the current batch of K packets. Once successfully received by a wireless node, each MPDU will be forwarded up and buffered just above the MAC layer waiting for a decoding process. Once decoded, these packets will be forwarded to the upper layers for FEC and finally for multimedia presentation.

In the following, we describe the source model and FEC, and wireless fountain coding in Sections 7.1.2 and 7.1.3, respectively. Assumptions on the channel model are stated in Section 7.1.4.

7.1.2 Source Model and FEC

As shown in Fig. 7.2, systematic Reed-Solomon (RS) FEC codes are applied across k packets to produce $n(> k)$ transport packets at the application layer of each wireline source node. Each of the $n(> k)$ transport packets vertically generated thus contains both

data payload and FEC protection. Since k source packets need to be buffered before the RS coding starts, we propose that the RS coding is done over a block of k source packets having the same playback deadlines, or equivalently, the same maximum allowed delays (MAD). This MAD will be tagged to each transport packet.

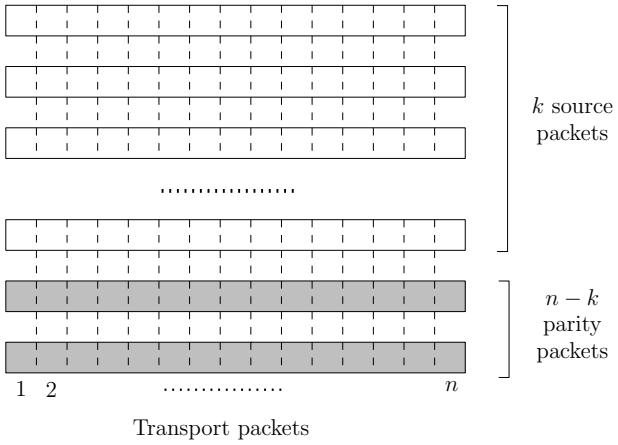


Figure 7.2: Packetization of media data and forward error protection (FEC).

7.1.3 Wireless Fountain Coding

After a UDP/IP header is added to each transport packet, the packet will be routed across the wireline network. We do not assume any particular routing protocol here. Arriving at the AP/gateway of interest, such packets are buffered on a per-flow basis just above the IEEE 802.11e MAC layer where WFC is applied.

Wireless fountain coding is performed over a block of K packets, $\{\mathbf{p}_1, ..., \mathbf{p}_K\}$ with the same MAD. Each coded packet transmitted by the AP/gateway is a random linear combination $\mathbf{x}_i$ of K packets, i.e.,

$$\mathbf{x}_i = \sum_{j=1}^{K} c_{i,j}\mathbf{p}_j \tag{7.1}$$

where the addition and multiplication are performed over a finite field $\mathbf{GF}(2^q)$, and the coefficients $c_{i,j}$ are uniformly drawn at random from $\mathbf{GF}(2^q)$. Each coded packet $\mathbf{x}_i$ is prepended with an encoding vector $\mathbf{c}_i = [c_{i,1} \ldots, c_{i,K}]$. We denote such a packet by a tuple $(\mathbf{c}_i, \mathbf{x}_i)$.

A regular MAC header/trailer is next added to each coded packet to form a MPDU. For each block of K packets, the AP/gateway will keep broadcasting coded packets to wireless nodes until it reaches a threshold (to be calculated in Section 7.2.2) in order to assure that at least K linearly independent coded packets are successfully received by each wireless destination.

$$\left.\begin{array}{rcl} \mathbf{x}_1 &=& c_{1,1}\mathbf{p}_1 + ... + c_{1,K}\mathbf{p}_K \\ \mathbf{x}_2 &=& c_{2,1}\mathbf{p}_1 + ... + c_{2,K}\mathbf{p}_K \\ & \cdots & \\ \mathbf{x}_K &=& c_{K,1}\mathbf{p}_1 + ... + c_{K,K}\mathbf{p}_K \end{array}\right\} \Rightarrow \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,K} \\ c_{2,1} & \cdots & c_{2,K} \\ \vdots & \ddots & \vdots \\ c_{K,1} & \cdots & c_{K,K} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_K \end{pmatrix} \quad \Rightarrow \quad \mathbf{x} = \mathbf{C}\mathbf{p}. \tag{7.2}$$

Once successfully received by each wireless destination, each MPDU will be forwarded up and buffered just above the IEEE 802.11e MAC layer waiting for a decoding process. The decoding process is triggered once K linearly independent coded packets, $(\mathbf{c}_1, \mathbf{x}_1), \ldots, (\mathbf{c}_K, \mathbf{x}_K)$, have been received. Since there is no guarantee that these K coded packets will be linearly independent, each wireless destination first checks for linear independence by performing Gaussian Elimination on the coefficient matrix $\mathbf{C}$ shown in (7.2). If the linear independence checking is successful, each wireless destination decodes $\{\mathbf{p}_1, ..., \mathbf{p}_K\}$ by solving (7.2).

7.1.4 Channel Model

To capture packet loss in the wireless and wireline channels, we view each channel as a packet erasure channel and model it as a two-state Markov chain. We denote by ϵ_f and ϵ_d the packet loss rates for the wireline $f$orward channel and the wireless $d$ownlink channel, respectively. Without retransmission mechanisms the end-to-end packet loss rate in the forward/downlink direction is $\epsilon_{fd} = 1 - (1 - \epsilon_f)(1 - \epsilon_d)$. We assume that the packet loss process at the wireless link to destination node j follows Bernoulli distribution with parameter $\epsilon_{d,j}$. We also assume that the AP/gateway knows this parameter for each associated wireless node.

7.2 System Analysis

In this section we derive the packet loss/late probability as seen by each wireless node in concurrent downlink unicast/multicast streaming.

7.2.1 Wireline Delay Model

The first element to model is a one-way delay a packet takes to traverse in the wireline IP network. According to [73] [90], a one-way forward trip time (FTT) D that a packet takes to travel without loss from one wireline station to another can be modeled as a random variable having a shifted Gamma distribution with rightward shift Δ_f and parameters α_f and λ_f, i.e.,

$$f_D(t|\text{ not lost}) = \frac{\lambda_f}{\Gamma(\alpha_f)}(\lambda_f(t-\Delta_f))^{\alpha_f-1}\exp\left(-\lambda_f(t-\Delta_f)\right), \quad t \geq \Delta_f. \tag{7.3}$$

The random variable D can be interpreted as the time a packet takes to go through a series of α_f routers, bridges, or gateways, each of which is modeled as an M/M/1 queue with parameter λ_f and a constant processing time Δ_f/α_f plus waiting time in steady state. The mean and variance of D are $\mu_f = \alpha_f/\lambda_f + \Delta_f$ and $\sigma_f^2 = \alpha_f/\lambda_f^2$, respectively. The parameters λ_f and α_f of the distribution can be determined from μ_f and σ_f^2, both of which can be periodically estimated, as: $\lambda_f = (\mu_f - \Delta_f)/\sigma_f^2$ and $\alpha_f = \lambda_f(\mu_f - \Delta_f)$.

7.2.2 Delay Analysis of Wireless Transmission

Suppose a certain number of unicast and multicast flows are enqueued for transmission at the AP/gateway of interest. Out of this number, assume that U unicast and M multicast packets are selected for transmission. The goal is to deliver $K = U + M$ packets to N ($\geq U + M$) wireless nodes so that all the packets meet their presentation deadlines. Further, assume that successive broadcast transmissions to a set of N wireless nodes are independent, and that each broadcast transmission takes exactly one time slot.

Lemma 7.2.1 *The number of time slots, $t_{j,K}$, required to broadcast K linearly independent coded packets successfully from the AP/gateway to the jth wireless node, for $j \in \{1, 2, \ldots, N\}$, is*

$$\Pr\{t_{j,K} = K+i\} = \binom{K+i-1}{i} \left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{2^q}\right)^i \left(1 - \left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{2^q}\right)\right)^K, i = 0, 1, 2, \ldots \tag{7.4}$$

where $\epsilon_{d,j}$ is the packet loss rate on a link between the AP/gateway and wireless node j.

Proof Let time be divided into equally spaced slots and transmission of each packet oc-

cupies exactly one time slot[1]. Intuitively, the probability that $t_{j,K}$ is less than K is equal to zero, i.e., $\Pr\{t_{j,K} < K\} = 0$. Moreover, since each coded packet transmitted can be either physically lost or, if not so, linearly *dependent* with previously received packets, a transmission of a coded packet is regarded as *failure* with probability $\left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{q}\right)$ (recall that q is the size of the finite field $\mathbf{GF}(q)$ under consideration), where the second term corresponds to the event that the packet is successfully received (with probability $(1-\epsilon_{d,j})$) but unsuccessfully decoded (with probability $1/q$) [7]. It follows that $\Pr\{t_{j,K} = K\} = \left(1 - (\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{q})\right)^K$. Now, if $K+1$ time slots are required to successfully deliver K packets to wireless node j, one of the first K coded packets transmitted must have failed. No matter which packet has failed, a new coded packet formed by a random linear combination of original K packets must be retransmitted successfully in the $(K+1)$th time slot. The probability of this event is $\Pr\{t_{j,K} = K+1\} = \binom{K}{1}\left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{q}\right)\left(1 - \left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{q}\right)\right)^{K-1}\left(1 - \left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{q}\right)\right)$. In general, the probability that $K+i$ time slots are required to successfully deliver all K coded packets to a particular wireless node j is therefore

$$\Pr\{t_{j,K} = K+i\} = \binom{K+i-1}{i}\left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{q}\right)^i\left(1 - \left(\epsilon_{d,j} + \frac{1-\epsilon_{d,j}}{q}\right)\right)^K. \quad (7.5)$$

Intuitively, the number of time slots required to ensure that all N wireless nodes have received at least K linearly independent coded packets is the maximum of all $t_{j,K}$ for $j = 1, 2, ..., N$. That is,

$$t_{(N,K)} = \max_{1 \le j \le N}\{t_{j,K}\}. \quad (7.6)$$

[1]This assumption implies the following. First, the MAC layer will not change its data rate during the transmission of a block of K packets. Second, packets in our system will have the same size in order to undergo the wireless fountain coding process. Smaller packets will always be padded with zeros to meet this requirement.

The following theorem states the moment generating function of $t_{(N,K)}$ for any N and K.

Theorem 7.2.2 *Suppose we have N wireless nodes each of which needs to receive at least K linearly independent coded packets in order to decode its packet. Also suppose that the probability of dropping a packet on each link from AP/gateway to wireless node j, $j = 1, 2, ..., N$, is $\epsilon_{d,j}$. Then the moment generating function for $t_{(N,K)}$ is*

$$F(z) = \sum_{\omega=0}^{\infty} \left(1 - \prod_{j=1}^{N} \sum_{i=0}^{\omega} z^{\omega} \Pr\{t_{j,K} = K + i\} \right) \tag{7.7}$$

where $\Pr\{t_{j,K} = K + i\}$ is as stated in Lemma 7.2.1.

Proof The proof follows from the definition of a moment generating function. That is,

$$F(z) \;=\; \sum_{\omega=0}^{\infty} z^{\omega} \Pr\{t_{(N,K)} > \omega\} \;=\; \textstyle\sum_{\omega=0}^{\infty} z^{\omega} q_{\omega} \tag{7.8}$$

where $q_{\omega} = \Pr\{t_{(N,K)} > \omega\}$ is the probability of failure in obtaining K packets at all N wireless nodes up to and including the ωth trial. Then we may rewrite (7.8) as

$$\begin{aligned} F(z) &= \sum_{\omega=0}^{\infty} \left(1 - \prod_{j=1}^{N} z^{\omega} \Pr\{t_{j,K} \leq \omega\} \right) \\ &= \sum_{\omega=0}^{\infty} \left(1 - \prod_{j=1}^{N} \sum_{i=0}^{\omega} \Pr\{t_{j,K} = K + i\} \right). \end{aligned} \tag{7.9}$$

Substituting (7.4) for $\Pr\{t_{j,K} = K+i\}$ we obtain the moment generating function for $t_{(N,K)}$.

Based on Theorem 7.2.2, we can obtain the mean and the variance of $t_{(N,K)}$ as stated in the following corollary.

Corollary 7.2.3 *The mean and the variance of $t_{(N,K)}$ can be expressed as*

$$\mathsf{E}\{t_{(N,K)}\} = F(1) \tag{7.10}$$

and

$$\sigma^2_{t_{(N,K)}} = 2F'(1) + F(1) - (F(1))^2. \tag{7.11}$$

Proof Let p_ω be the probability of successfully sending K linearly independent coded packets to all wireless nodes in ω transmissions (time slots). Then the mean of $t_{(N,K)}$ is

$$\mathsf{E}\{t_{(N,K)}\} = \sum_{\omega=0}^{\infty} \omega p_\omega. \tag{7.12}$$

Recall that $q_\omega = \Pr\{t_{(N,K)} > \omega\}$ is the probability of failure in obtaining K packets at all N wireless nodes up to and including the ωth transmission. That is, $p_\omega = q_{\omega-1} - q_\omega$. Therefore, we can rewrite (7.12) as

$$\mathsf{E}\{t_{(N,K)}\} = \sum_{\omega=1}^{\infty} \omega(q_{\omega-1} - q_\omega) = \sum_{\omega=1}^{\infty} q_\omega = F(1) \tag{7.13}$$

where $F(z)$ is as defined in (7.8).

Similarly, the second moment of $t_{(N,K)}$ is

$$\begin{aligned}
\mathsf{E}\{t^2_{(N,K)}\} &= \sum_{\omega=1}^{\infty} \omega^2(q_{\omega-1} - q_\omega) \\
&= 1^2(q_0 - q_1) + 2^2(q_1 - q_2) + 3^2(q_2 - q_3) + ... \\
&= (1^2 q_0 + 2^2 q_1 + 3^2 q_2 + ...) - (1^2 q_1 + 2^2 q_2 + 3^q_3 + ...) \\
&= \sum_{\omega=0}^{\infty} (\omega+1)^2 q_\omega - \sum_{\omega=1}^{\infty} \omega^2 q_\omega \\
&= \sum_{\omega=0}^{\infty} (2\omega+1) q_\omega = 2F'(1) + F(1)
\end{aligned} \tag{7.14}$$

from which the variance follows as $\sigma^2_{t_{(N,K)}} = 2F'(1) + F(1) - (F(1))^2$.

From now on, we will refer to $\mathsf{E}\{t_{(N,K)}\}$ as the threshold the AP/gateway uses to limit its broadcast transmissions of coded packets. That is, for a given group of K packets, the AP/gateway will not transmit more than $\mathsf{E}\{t_{(N,K)}\}$ times.

Now, given an IP packet of length L, MAC overhead of length L_o, encoding vector of length L_e, and the basic data rate R_1 as specified in the IEEE 802.11e standard, the expected time required to ensure that all N wireless nodes have successfully received at least K linearly independent coded packets is

$$\bar{d} = \mathsf{E}\{t_{(N,K)}\}\left(\mathsf{SIFS} + \frac{L_o + L_e + L}{R_1}\right) \tag{7.15}$$

where we assume that consecutive broadcast transmissions are separated by an SIFS period, and that decoding takes places instantaneously and simultaneously at each wireless node once all the K linearly independent coded packets have been received.

7.2.3 End-to-End Packet Loss/Late Probability

Using (7.15), we now express the probability that a coded packet, as seen by wireless node j, is lost or late or linearly dependent with the previously received coded packets. As in [73], we combine the packet loss probabilities and the packet delay densities into a single probability measure. That is, while a lost packet is regarded as having an infinite delay, a packet that is not lost is weighted by the probability distribution of its wireline delay. Let us define the "maximum remaining allowed delay" (τ) as the maximum delay that the tagged packet is allowed to spend in the wireline IP network prior to its presentation deadline at wireless node j. That is, $\tau = d_{max,j} - \bar{d}$, where $d_{max,j}$ is the maximum allowed delay of a packet destined for wireless node j and $\bar{d}$ is the expected delay in the wireless downlink. Then, the probability that the forward trip time D of a tagged packet is greater than its maximum remaining allowed delay is

$$P_{e,j} = \Pr\{D > d_{max,j} - \bar{d}\} = \epsilon_{fd,j} + (1 - \epsilon_{fd,j}) \int_{d_{max,j} - \hat{d}_d}^{\infty} f_D(t|\text{ not lost})dt \tag{7.16}$$

where $\epsilon_{fd,j} = 1-(1-\epsilon_f)(1-\epsilon_{d,j}-\frac{1-\epsilon_{d,j}}{2^q})$, and the wireline delay distribution $f_D(t|$ not lost$)$ is as defined in (7.3). Note that (7.16) gives the worst-case packet loss/late probability as it might be the case that some wireless nodes may have received K linearly independent coded packets with delay less than $\hat{d}_d$. Now, with (n,k)-RS FEC, the end-to-end packet loss/late probability as seen by the wireless node j is

$$P_{e,FEC,j} = 1 - \sum_{i=0}^{n-k} \binom{n}{i} (P_{e,j})^i (1-P_{e,j})^{n-i}. \tag{7.17}$$

In the following we will relate $P_{e,FEC,j}$ to video distortion using the loss-distortion model proposed in [88].

7.3 Performance Evaluation Framework

In this section we describe the necessary components we use to evaluate our system. In Section 7.3.1, we explain how video data is packetized and protected by FEC. Later in Section 6.5.2 we briefly describe how we map packet loss/late probability to video distortion. Section 7.3.2 gives the details of our simulation platform.

7.3.1 Packetization of Video Data

Fig. 7.3(a) shows how a video sequence is packetized in our video streaming system. A video sequence consists of multiple groups of pictures (GOP). Assuming that only intra-coded (I-) and predictively coded (P-) frames are used, each GOP comprises of one I-frame and multiple P-frames. Each frame is divided into multiple slices, each of which is defined as a block of independently coded pixels. Each video frame is encoded into k source packets, all of which are to be transmitted using n transport packets by applying the (n,k) systematic

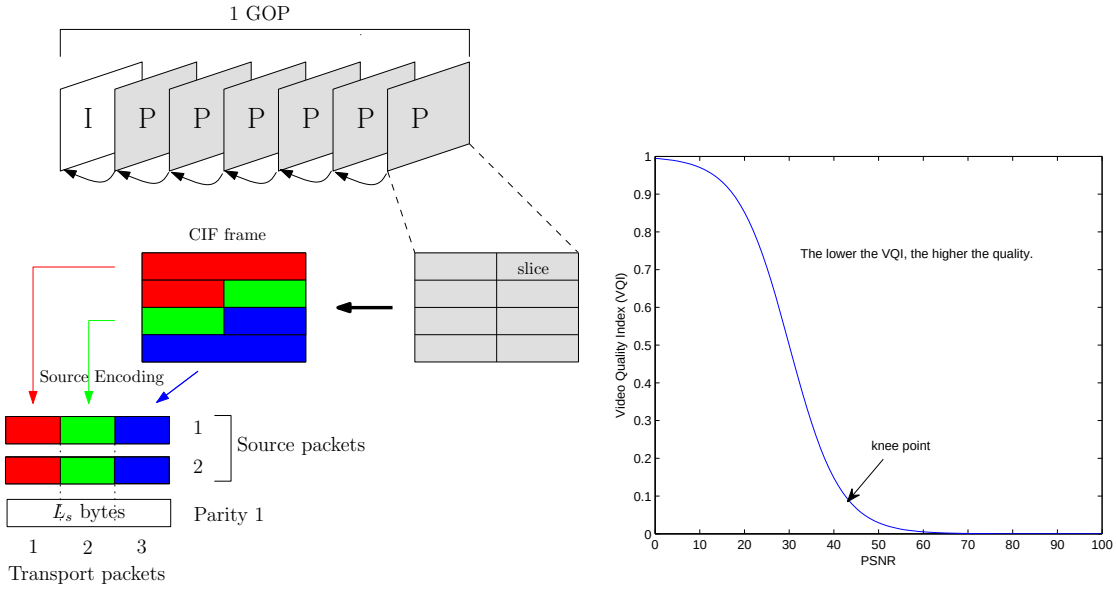


Figure 7.3: (a) Packetization and FEC: The variable number of slices may be mapped into each transport packet because each slice contains the varying number of data bits that need to be encoded. We show a particular instance where a "green" slice contains more data bits than any other slices. Systematic RS code is applied to a group of $k = 2$ source packets to produce $n = 3$ transport packets in this example. (b) Mapping PSNR to VQI as suggested in [89]: A case of the *Foreman* video sequence.

RS code. We use the same model for mapping loss to video distortion, as in the uplink streaming case (See Chapter 6, Section 6.2.5 for detail).

7.3.2 Simulation Framework

As shown in Fig. 7.4, our simulation framework consists of tools from the Video Quality Evaluation Tool-set (EvalVid) [97] and the ns-3 network simulator [98]. While the EvalVid toolset takes care of the video coding and decoding at the source and the

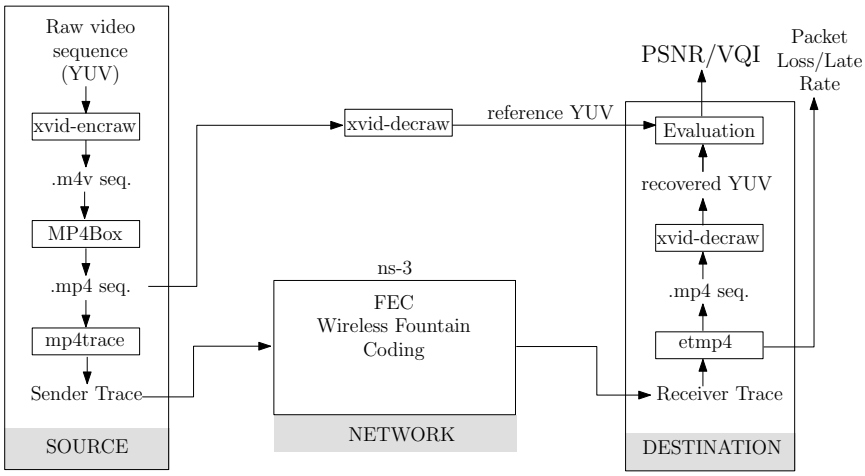


Figure 7.4: Simulation framework.

destination, respectively, the ns-3 simulator involves several network-related components. We explain our simulation framework according to the tasks performed by the source, the network, and the destination(s) as follows:

Source

At each source, a raw YUV video sequence is encoded using the **xvid-encraw** encoder [99]. The encoded video is next encapsulated in an mp4 video sequence using **MP4Box** [99]. Using this mp4 video sequence, an application-level sender trace containing frame ID, frame sizes, playout times, frame types and so on, is next generated (using **mp4trace** [97]) and sent as input to the ns-3 simulator for network simulations.

Network

In the ns-3 simulator we implement a module that translates the application-level sender trace received from the **mp4trace** tool into a network-level trace containing packet

IDs, packet sizes, packet interarrival times, and playout times from which we derive the MAD. Each packet is also assigned the maximum number of application-layer retransmissions, N_{APP}. After tagging each packet with MAD and N_{APP}, a FEC module [101] performs RS coding across a block of k packets to produce n transport packets. These FEC-protected packets are passed down to the network-layer at which UDP/IP headers are added. Packets are routed in the wireline network by the *ns-3 global routing* function which performs pre-simulation static route computation on a layer-3 IPv4 topology. Once a topology has been constructed and addresses have been assigned to nodes, the routing database will be initialized, and the static unicast forwarding tables will be created for each wireline node.

Just above the IEEE 802.11e MAC layer in the AP/gateway, we implement a WFC function based on the ns-3 *wifi-devices* module. Packets are buffered on a per-flow basis when they arrive at the AP/gateway. The AP/gateway selects, for WFC, a group of K packets with the same MAD. After coding, each coded packet is prepended with a WFC header which is a vector of coding coefficients $c_{i,j}$ as described in Section 7.1.3.

Destination

On the receiving side, each wireless node equipped with a decoding function uses the coefficients in the WFC header to decode the coded packets pertaining to a current block of K packets before forwarding them to the upper layer for forward error correction.

At each wireless node, we modify the **etmp4** tool to translate a receiver trace to a probably distorted mp4 video sequence. At this stage, a corresponding packet loss/late rate is calculated. The **xvid-decraw** decoder [99] is then used to recover the raw YUV video sequence from the probably distorted mp4 video sequence. At the same time, the mp4

video sequence is also sent from the source to another **xvid-decraw** decoder for creating a reference YUV raw video sequence. This step of re-creating the reference YUV raw sequence is necessary to ensure that the video distortion caused by the decoder itself is not taken into account in the evaluation step as the received mp4 video sequence must also go through the same process. The evaluation takes place offline and essentially involves calculating the corresponding PSNR and VQI. The offline video evaluation is performed by comparing the received raw YUV sequence with the reference raw YUV sequence.

7.4 Simulation Results

In this section, we evaluate the performance of downlink video streaming in a typical combined wireline/802.11 network shown in Fig. 7.5. Each point-to-point wireline

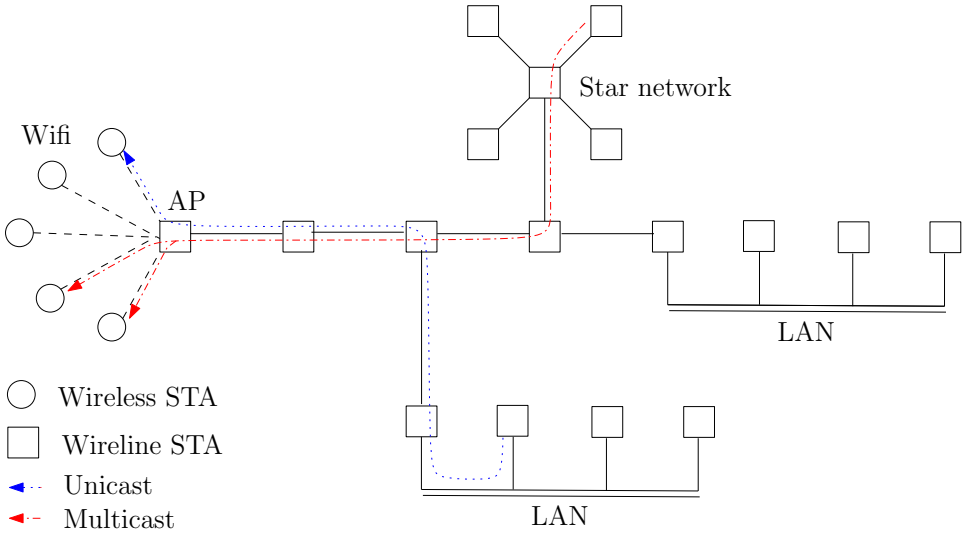


Figure 7.5: Downlink media streaming in a combined wireline/802.11 network.

link in the network is modeled as a packet eraser channel with a bandwidth of 10 Mbps. The CSMA channel capacity of the LAN is 100 Mbps. In the wireless link, packets are

erased across time and space and the link capacity is fixed at 11 Mbps. The video sequence simulated is of H.264/MPEG-4 AVC Common Intermediate Format (CIF). For wireless fountain coding operations, we fix the size of the finite field at 2^8. That is, each symbol is one byte long. The packet size at the application level is 1024 bytes. We use a default random number generator (PRNG) built into the ns-3 simulator for all the simulation runs each of which is fed with a different seed [98]. Otherwise specified, the simulation parameters shown in Table 7.1 are used throughout the rest of this chapter.

For comparison purposes, when the WiFi network does not have the WFC capability, we assume that a traditional IEEE 802.11e PCF is employed to sequence through a list of head-of-line (HOL) unicast and multicast packets according to [67]. This implies that, in addition to FEC, both link-layer and application-layer retransmission mechanisms are still available for unicast flows. These retransmission mechanisms are, however, not available for multicast flows.

7.4.1 Effects of Number of Flows

First we study how many video unicast and multicast flows the AP/gateway should code together so as to meet a target packet loss/late rate or video distortion level as seen by all the wireless nodes receiving those video flows. We simulate downlink video unicast and multicast on the network shown in Fig. 7.5. A video sequence is selected uniformly at random by each wireline source from the set {**Foreman, Mother & Daughter, Hall, Waterfall**} and transmitted from a wireline source node to its corresponding wireless destination(s). We fix the number of unicast flows to 4 and gradually increase the number of multicast flows from 1 to 7. Each multicast flow serves $2-4$ wireless nodes. The maximum

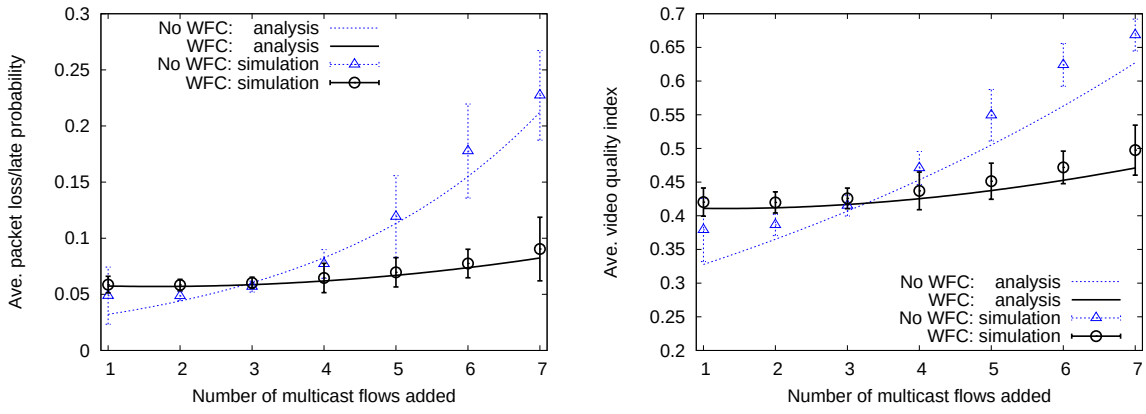
Table 7.1: Simulation parameters.

IEEE 802.11e Parameters	
Basic rate for control signal R_1	6 Mbps
Data rate R'	11 Mbps
Slot Time	20μs
SIFS	10μs
Channel parameters	
$\Delta_f = \Delta_b$	25 ms
$\lambda_f = \lambda_b$	$(12.25 \text{ ms})^{-1}$
$\alpha_f = \alpha_b$	4
Wireless packet error rate ϵ_d	varies in $(0, 0.3]$
Wireline packet error rate ϵ_f	varies in $(0, 0.1]$
Video Parameters	
Group of Pictures (GOP)	30
No. frames per second	30
No. packets per frame $\mathcal{L}$	4
No. slices in each packet $\mathcal{S}$	2
No. consecutive packets lost on average in a single loss event $\bar{\eta}$	1

allowed delay of video packets is fixed at 180 ms for all unicast and multicast flows.

In Fig. 7.6 we compare our WFC scheme with the traditional scheme in terms of end-to-end packet loss/late probability and video quality index (VQI), both of which are averaged over all wireless destination nodes. Both the simulation and analytical results are shown. In particular, for each value of the number of multicast flows added, the packet loss/late probability and VQI are averaged over 30 independent simulation runs each of which in turn is averaged over all the wireless destination nodes. For each averaged point, we assert a 95% confidence interval.

From Fig. 7.6, we observe that more multicast flows can be supported by using WFC without a significant increase in packet loss/late probability (Fig. 7.6(a)) and without a significant decrease in video distortion (Fig. 7.6(b)). On the other hand, a sharp increase in packet loss/late probability is observed for the transmission without wireless fountain coding as more multicast flows are added. However, the benefit of WFC is not realized until a certain number of multicast flows (3 in this case) is added to the existing unicast flows. The reason is that, as more multicast flows (i.e., more wireless nodes) are supported, wireless channel diversity is not fully exploited by a traditional method which polls the wireless nodes selectively. With a fewer number of wireless nodes to support, i.e., fewer channels/links, the probability that such polling will select wireless nodes with "bad" channels is reduced. On the other hand, with WFC, packets are broadcast to the wireless nodes through a collection of "good" and "bad" channels/links, all at once. This is why the traditional polling MAC outperforms the transmission with WFC in a lower range of the number of multicast flows as shown in Fig. 7.6.



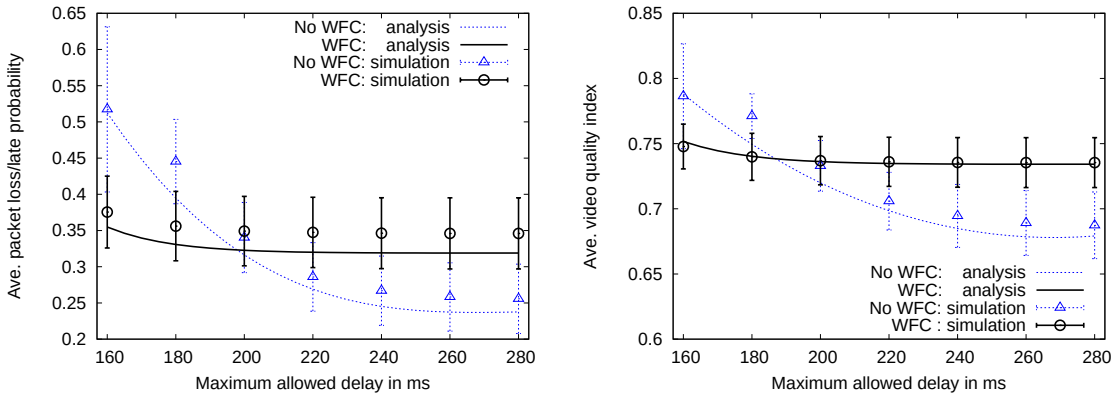
(a) (b)

Figure 7.6: Performance of the downlink video unicast/multicast as the number of multicast flows increases: (a) WFC helps accommodate more multicast flows without a significant increase in packet loss/late probability, (b) No significant increase in video distortion is seen when WFC is used.

7.4.2 Effects of Maximum Allowed Delay (MAD)

Now, given a certain number of unicast and multicast flows to code, we are also interested in how WFC performs as the MAD of video packets varies. In this case, we fix the number of unicast and multicast flows and gradually increase the maximum allowed delay of video packets for all the flows. In particular, we fix the numbers of unicast and multicast flows to 4, and let each multicast flow serve only 2 non-overlapping wireless nodes. A video sequence is again selected by each wireline source uniformly at random from the set {**Foreman, Mother & Daughter, Hall, Waterfall**}. The MAD of video packets is varied from a stringent 160 ms to a more relaxed 280 ms. For each value of MAD, the packet loss/late probability and VQI are averaged over 30 independent simulation runs each of which is averaged over all the wireless destination nodes. For each averaged point, a 95% confidence interval is asserted.

We show both the analytical and the simulation results in Fig. 7.7. We observe that streaming with WFC performs well, in terms of average end-to-end packet loss/late probability (Fig. 7.7(a)) and average VQI (Fig. 7.7(b)), only when presentation deadlines are relatively tight (i.e., small MAD). However, when the presentation deadlines are more relaxed, the application- and wireless link-layer retransmission techniques, as employed by the unicast flows in the traditional scheme, prevail. Since retransmission techniques normally incur variable network delays, when video packets are given more relaxed deadlines they enjoy the extra time in the network. Streaming with WFC however fails to exploit this extra time because retransmissions are not implemented. Error recovery in this case relies solely on FEC. In addition, since the AP/gateway determines the threshold $\mathsf{E}\{t_{(N,K)}\}$ for



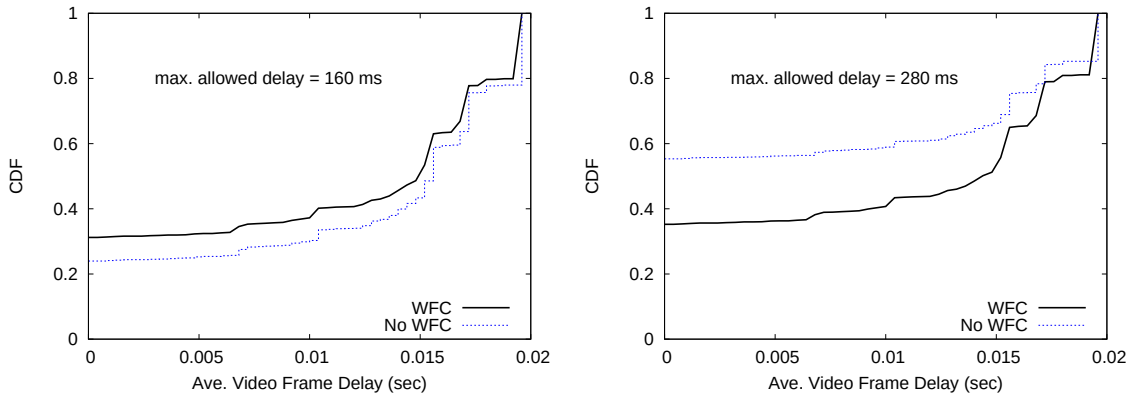
(a) (b)

Figure 7.7: Performance of the downlink video unicast/multicast as the maximum allowed delay of video packets varies: (a) There is a limit on the maximum allowed delay of video packets up to which WFC is more beneficial in terms of packet loss/late probability than streaming without WFC, (b) Mapping packet loss/late probability to video quality index averaged over all flows results in the same conclusion.

each batch of K packets based only on the conditions of wireless channels and *not* on the MAD $d_{max,j}$ for each wireless node j, $j = 1, 2, ..., N$, the AP/gateway fails to adjust this threshold according to $d_{max,j}$.

7.4.3 Average Video Frame Delay

In this section, we take the same simulation settings from Section 7.4.2 and look closely at two extreme points of Fig. 7.7, that is, when MAD is 160 ms and when it is 280 ms. In particular, we would like to confirm how packet loss/late probabilities at these two extreme points translate into video frame delay which is easily conceivable by average human eyes. In perfect streaming, video frames should appear to viewers at the rate of exactly



(a) (b)

Figure 7.8: Video frame delay for 2 extreme cases: (a) Streaming with WFC outperforms streaming without WFC (MAD = 160 ms), (b)Streaming without WFC outperforms streaming with WFC (MAD = 280 ms).

30 frames/second, or equivalently, 0.5 second/frame. However, with late or lost packets, each video frame once constructed will appear with some delay. We plot the cumulative distribution function (CDF) of this delay for these two extreme cases in Fig. 7.8(a) and Fig. 7.8(b), respectively.

Fig. 7.8(a) confirms that when our proposed scheme outperforms the traditional scheme in terms of packet loss/late probability (MAD = 160 ms), this improvement also translates into reduced average video frame delay. On the other hand, when streaming without WFC prevails, we see significant improvements in terms of video frame delay (e.g., in Fig. 7.8(b)). Little improvements however are observed for the streaming system with WFC. This can be expected because while the packet loss/late probability for streaming without WFC is significantly reduced at MAD = 280 ms, the packet loss/late probability

for the case with WFC experiences only a slight reduction (e.g., in Fig. 7.7(a)).

7.5 Conclusion

We have developed a performance model for downlink media streaming with wireless fountain coding (WFC) in a combined wireline/802.11 network. We have analytically obtained the moment generating function of the wireless link-layer delay and used it to derive the parameter that AP/gateway can use to make a coding-based decision. Based on the expected wireless link-layer delay, the end-to-end packet loss/late probability, that expresses the relationship between several parameters pertaining to FEC, WFC and channel conditions, has been derived and translated into a video distortion measure. The accuracy and benefits of our model have been discussed and illustrated through extensive simulations.

Chapter 8

Summary and Discussions

As wireless access has been increasingly deployed as a preferred mode of network access for anytime-anywhere communications, next generation content distribution applications also continue to demand high bandwidth. Yet, with its current form, an infrastructure wireless mesh network (IWMN) still have to meet the requirements of such highly demanding applications. With multiple radios per node, the problem can become even worse if channel assignment and routing are not properly handled. Addressing the issues related to resource allocation on a particular channel is also important for satisfying such demands. Built on the theory of network coding, this thesis advocates a set of alternative solutions to the above mentioned problems for high performance media content distribution in IWMNs. Our results show that the proposed solutions can provide promising throughput, delay and reliability gains over the traditional schemes without network coding. We conclude by summarizing the contributions of this thesis in Section 8.1 and later pointing out interesting avenues for future research in Section 8.2.

8.1 Summary of Contributions

Chapter 3: Network-wide Solution: Joint CA, Link Scheduling and Network Coding

- Using the linear optimization technique, we formulate the joint network coding, CA and link scheduling problem for throughput optimization in network-coded multi-radio IWMNs, taking into account the broadcast link interference constraints, the coding and flow constraints, the fairness constraints, and the node-radio constraints. Our formulation explicitly takes opportunistic listening and lossiness of broadcast links into account. The formulation is also general enough to encompass the non-coding case, i.e., the joint multi-path routing, CA and scheduling problem in the conventional, *routed* multi-radio IWMNs. We analyze the complexity of the resulting linear program in comparison with that of the non-coding case.

- Based on this formulation, we mathematically formulate the CA and broadcast link scheduling problem as a two-sided multi-assignment problem and subsequently convert it to a minimum cost flow problem whose dual is derived and used as a platform for devising our auction-based, CA and scheduling algorithm. The complexity of the overall algorithm is also analyzed and shown to run in polynomial time.

- We evaluate our proposed solution through extensive simulations and experiments on a 10-node 802.11b/g testbed. Comparison with the single-path and multi-path routing solutions with various CA strategies is also performed. Our results show that the proposed solution can effectively exploit the increasing number of channels and radios

and exhibits throughput gains over the traditional routing counterparts. We also show that, through proper fine-tuning of a network coding related parameter, unfairness introduced during the channel assignment phase via auctions can be effectively dealt with.

- Our solution has been evaluated experimentally via iCORE – the **i**nterface **CO**operation **R**epeater-aided network coding **E**ngine. iCORE integrates network coding with channel/interface management. It allows nodes to code packets within or across flows as well as move packets across interfaces whenever it sees more transmit opportunities. It sits on top of the MAC, thus making it independent from device drivers. We have illustrated its functionalities as a repeater-based 802.11 network. Our experimental results show promising avenues for extending iCORE to more general topologies, which we did for the network-wide solution discussed in Chapter 3.

Chapter 5: AP/Gateway-oriented Solution I: Dynamic Buffer Allocation

- We have proposed a discrete-time Markov queueing model for a wireless lossy unicast exchange network which employs opportunistic network coding and dynamic buffer allocation at a bottlenecked AP. Unlike earlier studies – which assume two or more static buffers at the AP, one for each packet flow – we have proposed that the AP dynamically allocates buffer space to incoming packets without assuming the static capacities of their respective buffers. Such dynamic buffer allocation in our context thus operates much like two water tanks positioned side by side and connected at the bottom by a *static* open valve.

- Motivated by an increasing demand for real-time applications, we show the delay benefit of network coding for a generic wireless unicast exchange network over classical scheduling schemes such as first-in first-out and round-robin schemes.

- To improve the performance further, we have also proposed a simple scheduling algorithm, called *buffer equalized opportunistic network coding*, which operates similarly to a pair of water tanks with a *sliding* open valve in the middle to allow packets (water) to move across tanks. We have shown that the proposed scheme improves in terms of delay over the *static* valve model.

Chapter 6: AP/Gateway-oriented Solution II: Wireless Fountain Coding for Uplink Streaming

- We propose a modified B-ACK scheme that integrates a version of fountain coding with the traditional 802.11e B-ACK scheme in such a way that protocol complexity and wireless link-layer delay are reduced when compared to that of the traditional 802.11e B-ACK scheme. We analytically quantify this delay for both the modified and the traditional B-ACK schemes, assuming channel uncertainty in both forward and backward directions of the combined wired/wireless network.

- Based on the analytically quantified wireless link-layer delay, we derive the probability that a media packet is lost or late for presentation end-to-end, for both the modified B-ACK scheme and the traditional 802.11e B-ACK scheme. Such probabilities are expressed in terms of FEC, ARQ, and B-ACK parameters.

- We propose a hybrid scheme that switches between the modified and traditional

802.11e B-ACK mechanisms according to wireless channel conditions. For a given range of block size of packets, this hybrid scheme tends to keep the end-to-end packet loss/late probability as low as possible.

- Through simulations, the accuracy and benefits of our proposed model compared to the traditional 802.11e B-ACK schemes are illustrated for various network settings.

Chapter 7: AP/Gateway-oriented Solution III: Wireless Fountain Coding for Downlink Streaming

- We propose a media streaming system which employs wireless fountain coding at the AP/gateway to exploit multi-user diversity in the wireless downlinks of a combined wireline/802.11 network.

- We analytically obtain the moment generating function of the wireless link-layer delay and use it to quantify the expected wireless link-layer delay which the AP can use to make a coding-based decision.

- Based on the analytically quantified wireless link-layer delay, we derive the probability that a media packet is lost or late for presentation end-to-end and relate it to video distortion through a loss-distortion model. Such a probability and distortion are expressed in terms of FEC, WFC, channel and H.264/MPEG-4 AVC video parameters.

- Through simulations, the accuracy and benefits of our proposed model compared to the traditional streaming system without WFC [67] are illustrated for various network settings.

8.2 Future Work

We have studied the joint problem of channel assignment, random network coding and broadcast link scheduling for an infrastructure multi-channel multi-radio wireless mesh network in Chapter 3. The main objective is to optimize throughputs for multiple unicast connections in a fair manner. Delay however is completely ignored. The proposed framework thus can be extended to the scenarios where delay is a major concern in the network. Development of distributed solutions for channel assignment is also an interesting avenue for future research. In addition, striking the balance between fairness and throughput is another important research issue.

One drawback identified in iCORE is that as a batch size K increases for all flows, the packet delivery ratio of iCORE drops significantly, whereas that of MORE slightly drops. This is the price iCORE needs to pay in combining both inter- and intra-flow network coding, whereas MORE focuses on combining packets within an independent flow (intra-flow network coding). As packets are routed through multiple hops, the resulting batch size of packets that a destination needs to collect grow significantly. This results in significant decoding errors. One remedy to this problem is to limit the number of flows that can be combined and prioritize the intra-flow network coding over the inter-flow counterpart. To what extent such a limit could be is an interesting topic for future investigation.

As a side note for our dynamic buffer allocation scheme proposed in Chapter 5, our models are generic enough to embrace the scenarios where packets are combined or coded across traffic classes (best effort, voice, video, and background as defined in the IEEE 802.11e/n standard) at the MAC layer instead of across traffic flows. As a result, source

s_1, or s_2, can now be thought of as a group of wireless nodes, each group transmitting the same class of traffic. Each "virtual" buffer at the AP/gateway then represents a MAC buffer holding packets belonging to a specific class of traffic. One interesting direction for future investigation would be to generalize the idea of dynamic buffer allocation (or buffer equalization) to an arbitrary number of buffers connected, or analogously, to a farm of connected tanks. The analytical model can also be extended to consider channel-adaptive transmission in the wireless links.

Finally, one of the lessons we have learnt in the proposed streaming systems (Chapters 6 and 7) is that adding intelligence (wireless fountain coding in our case) at some layer of the protocol stack does not always translate into the benefits realizable at any other layers. For certain conditions of the channels we still have to resort to a hybrid way of operations in order to extract the best flavors out of the two models we have studied for the uplink streaming case, for example.

Bibliography

[1] "IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE STD 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. C1–1184, June 12 2007.

[2] "Supplement to IEEE Standard for Information Technology Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band," *IEEE STD 802.11a-1999*, 1999.

[3] "List of WLANs Channels," [Online]. Available: *http://en.wikipedia.org/wiki/List_of_WLAN_channels*.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. on Information Theory,* vol. 46, no. 4 pp. 1204–1216, Jul 2000.

[5] S. Li and R. Cai, "Linear network coding," *IEEE Trans. on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[6] R. Koetter, "An algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[7] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE International Symposium on Information Theory 2003,* p. 442–447, 2003.

[8] R. Koetter and M. Medard, "Beyond routing: an algebraic approach to network coding," in *Proc. IEEE INFOCOM'02*, vol. 1, pp. 122–130, 2002.

[9] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, pp. 11–20, 2003.

[10] C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul, "Wireless network coding: Opportunities and challenges," in *Proc. IEEE MILCOM 2007*, pp. 1–8, Oct. 2007.

[11] C. Fragouli, J. Widmer, and J. Y. Le Boudec, "A network coding approach to energy efficient broadcasting: From theory to practice," in *Proc. IEEE INFOCOM'06*, pp. 1–11, 2006.

[12] R. W. Yeung, "Multilevel diversity coding with distortion," *IEEE Trans. on Information Theory*, vol. 41, no. 2, pp. 412–422, 1995.

[13] R. W. Yeung and Z. Zhang, "On symmetrical multilevel diversity coding," *IEEE Trans. on Information Theory*, vol. 45, no. 2, pp. 609–621, 1999.

[14] J. R. Roche and R. W. K. P. Hau, "Symmetrical multilevel diversity coding," *IEEE Trans. on Information Theory*, vol. 43, no. 3, pp. 1059–1064, 1997.

[15] R. W. Yeung and Z. Zhang, "Distributed source coding for satellite communications," *IEEE Trans. on Information Theory*, vol. 45, no. 4, pp. 1111–1120, 1999.

[16] G. B. Dantzig and D. R. Fulkerson, "On the max-flow min-cut theorem of networks," *RAND Corporation*, 2003.

[17] C. Fragouli and E. Soljanin, "Network coding fundamentals," *Foundations and Trends in Networking*, vol. 2, no. 1, pp. 1–133, 2007.

[18] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Proc. of the 15th annual ACM Symposium on Parallel Algorithms and Architectures*, New York, NY, USA, pp. 286–294, 2003.

[19] Y. Wu, P. A. Chou, and K. Jain, "A comparison of network coding and tree packing," in *Proc. of International Symposium on Information Theory, 2004.* .

[20] Y. Wu, P. A. Chou, and S. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. on Communications*, vol. 53, no. 11, pp. 1906–1918, 2005.

[21] Y. E. Sagduyu and A. Ephremides, "Cross-layer optimization of MAC and network

coding in wireless queueing tandem networks," *IEEE Trans. on Information Theory*, vol. 54, no. 2, pp. 554–571, 2008.

[22] Y. Wang, I. D. Henning, and D. K. Hunter, "Efficient information exchange in wireless sensor networks using network coding," in *Fourth Workshop on Network Coding, Theory and Applications, 2008 (NetCod'08)* , pp. 1–6, 2008.

[23] W. Chen, K. B. Letaief, and Z. Cao, "Opportunistic network coding for wireless networks," in *Proc. IEEE ICC'07*, pp. 4634–4639, 2007.

[24] P. Parag and J. F. Chamberland, "Queueing analysis of a butterfly network," in *Proc. IEEE International Symposium on Information Theory, 2008*, pp. 672–676, 2008.

[25] S. Chieochan, E. Hossain, T. Issariyakul, and D. Niyato, "Opportunistic network coding and dynamic buffer allocation in a wireless butterfly network," in *Proc. IEEE GLOBECOM'09*, Hawaii, USA, 2009.

[26] S. Bhadra and S. Shakkottai, "Looking at large networks: Coding vs. queueing," in *Proc. IEEE INFOCOM'06*, pp. 1–12, 2006.

[27] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard, "The importance of being opportunistic: Practical network coding for wireless environments," in *Proc. of the Allerton Annual Conference on Communications, Control, and Computing*, 2005.

[28] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *SIGCOMM Comput Commun. Rev.*, vol. 36, no. 4, pp. 243–254, 2006.

[29] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," *IEEE/ACM Trans. on Networking*, vol. 16, no. 3, pp. 497–510, 2008.

[30] Q. Dong, J. Wu, W. Hu, and J. Crowcroft, "Practical network coding in wireless networks," in *Proc. ACM MobiCom'07*, New York, NY, USA, pp. 306–309, 2007.

[31] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability benefit of network coding," Tech. Rep., *University of Massachusetts Amherst*, vol. 8, 2007.

[32] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability gain of network coding in lossy wireless networks," in *Proc. IEEE INFOCOM'08*, pp. 2171–2179, 2008.

[33] K. Li and X. Wang, "Cross-layer design of wireless mesh networks with network coding," *IEEE Trans. on Mobile Computing*, vol. 7, no. 11, pp. 1363–1373, 2008.

[34] C. Fragouli and E. Soljanin, "Network coding applications," *Foundations and Trends in Networking*, vol. 2, no. 2, pp. 135–269, 2007.

[35] S. Chachulski and S. Katti, "Trading structure for randomness in wireless opportunistic routing," in *Proc. the conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 169–180, New York, NY, USA, 2007.

[36] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "More: A network coding approach to opportunistic routing," *MIT Technical Report*, 2006.

[37] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard, "Symbol-level network coding

for wireless mesh networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 401–412, 2008.

[38] D. S. Lun, M. Medard, and R. Koetter, "Efficient operation of wireless packet networks using network coding," in *Proc. International Workshop on Convergent Technologies (IWCT)*, 2005.

[39] C. Fragouli, J. Widmer, and J. Y. le Boudec, "On the benefits of network coding for wireless applications," in *Proc. the 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006* , pp. 1–6, 2006.

[40] C. M. Cheng, H. T. Kung, C. K. Lin, C. Y. Su, D. Vlah, and M. Cambridge, "Rainbow: A wireless medium access control using network coding for multi-hop content distribution," in *Proc. IEEE MILCOM 2008*, pp. 1–10, 2008.

[41] Z. Li, B. Li, and L. C. Lau, "On achieving maximum multicast throughput in undirected networks," *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2467–2485, 2006.

[42] D. S. Lun and M. Medard, "On coding for reliable communication over packet networks," in *Proc. of the Allerton Annual Conference on Communications, Control, and Computing*, 2004.

[43] D. S. Lun, M. Medard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," in *Proc. International Symposium on Information Theory* , pp. 1848–1852, 2005.

[44] D. S. Lun, M. Medard, and R. Koetter, "Network coding for efficient wireless unicast," in *in IEEE International Zurich Seminar on Communications*, 2006.

[45] D. S. Lun, N. Ratnakar, M. Medard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2608–2623, 2006.

[46] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: a decentralized approach based on network coding," in *Proc. IEEE INFOCOM'05*, vol. 3, pp. 1607–1617, 2005.

[47] D. S. Lun, M. Medard, T. Ho, and R. Koetter, "Network coding with a cost criterion," in *Proc. 2004 International Symposium on Information Theory and its Applications (ISITA 2004)*, pp. 1232–1237, 2004.

[48] T. Ho, B. Leong, M. Medard, R. Koetter, Y.-H. Chang, and M. Effros, "On the utility of network coding in dynamic environments," in *Proc. International Workshop on Wireless Ad-Hoc Networks*, pp. 196–200, 2004.

[49] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Annual Allerton Conference on Communication control and Computing*, vol. 41, pp. 40–49, 2003.

[50] H. Wang, S. Xiao, and C. C. J. Kuo, "Robust and flexible wireless video multicast with network coding," in *Proc. IEEE GLOBECOM'07*, pp. 2129–2133.

[51] M. Halloush and H. Radha, "Network coding with multi-generation mixing: analysis and applications for video communication," in *Proc. ICC'08*, 2008, pp. 198–202.

[52] S. Karande, M. Wu, and H. Radha, "Network embedded FEC (NEF) for video multicast in presence of packet loss correlation," in *Proc. IEEE International Conference on Image Processing (ICIP'05)*, vol. 1, 2005.

[53] M. Wu, S. S. Karande, and H. Radha, "Network-embedded FEC for optimum throughput of multicast packet video," *Signal Processing: Image Communication*, vol. 20, no. 8, pp. 728–742, 2005.

[54] H. Seferoglu and A. Markopoulou, "Opportunistic network coding for video streaming over wireless," in *Proc. Packet Video*, EPFL, Lausanne, Nov. 2007.

[55] X. Y. Dong Nguyen, Thinh Nguyen, "Multimedia wireless transmission with network coding," in *Proc. Packet Video*, pp. 326–335, 2007.

[56] S. Simoens, P. Pellati, J. Gosteau, K. Gosse, and C. Ware, "The evolution of 5 GHz WLAN toward higher throughputs," *IEEE Wireless Communications*, vol. 10, no. 6, pp. 6–13, 2003.

[57] C. Liu and A. Stephens, "An analytic model for infrastructure WLAN capacity with bidirectional frame aggregation," in *Proc. IEEE WNCN'05*, vol. 1, 13-17 Mar. 2005, pp. 113–119.

[58] T. Li, Q. Ni, T. Turletti, and Y. Xiao, "Performance analysis of the IEEE 802.11e Block ACK scheme in a noisy channel," in *Proc. BroadNets'05*, vol. 1, 3-7 Oct. 2005, pp. 511–517.

[59] D. J. C. MacKay, "Fountain codes," *IEE Proceedings Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[60] P. A. Chou and M. van der Schaar, *Multimedia over IP and Wireless Networks: Compression, Networking, and Systems*. Academic Press, 2007.

[61] C. H. Wang, R. I. Chang, J. M. Ho, and S. C. Hsu, "Rate-sensitive ARQ for real-time video streaming," in *Proc. IEEE GLOBECOM'01*, 2001.

[62] H. C. Wei, Y. C. Tsai, and C. W. Lin, "Prioritized retransmission for error protection of video streaming over WLANs," in *Proc. IEEE ISCAS'04*, vol. 2, 2004.

[63] J. C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for Internet telephony," in *Proc. IEEE INFOCOM'99*, vol. 3, 1999.

[64] J. Rosenberg, L. Qiu, and H. Schulzrinne, "Integrating packet FEC into adaptive voice playout buffer algorithms on the internet," in *Proc. IEEE INFOCOM'00*, vol. 3, pp. 1705–1714, 2000.

[65] P. Frossard and O. Verscheure, "Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery," *IEEE Trans. on Image Processing*, vol. 10, no. 12, pp. 1815–1825, 2001.

[66] P. Frossard, "FEC performance in multimedia streaming," *IEEE Communications Letters*, vol. 5, no. 3, pp. 122–124, 2001.

[67] A. Argyriou, "Cross-layer error control for multimedia streaming in wireless/wireline packet networks," *IEEE Trans. on Multimedia*, vol. 10, no. 6, pp. 1121–1127, 2008.

[68] F. Zhai, Y. Eisenberg, T. N. Pappas, R. Berry, A. K. Katsaggelos, T. Instruments, and T. X. Dallas, "Rate-distortion optimized hybrid error control for real-time packetized video transmission," *IEEE Trans. on Image Processing*, vol. 15, no. 1, pp. 40–53, 2006.

[69] F. Hartanto and H. R. Sirisena, "Hybrid error control mechanism for video transmission

in the wireless IP networks," in *Proc. IEEE 10th Workshop on Local and Metropolitan Area Networks*, pp. 126–132, 1999.

[70] H. Seferoglu, Y. Altunbasak, O. Gurbuz, and O. Ercetin, "Rate distortion optimized joint ARQ-FEC scheme for real-time wireless multimedia," in *Proc. IEEE ICC'05*, vol. 2, 2005.

[71] Q. Li and M. Van Der Schaar, "Providing adaptive QoS to layered video over wireless local area networks through real-time retry limit adaptation," *IEEE Trans. on Multimedia*, vol. 6, no. 2, pp. 278–290, 2004.

[72] M. van der Schaar, S. Krishnamachari, S. Choi, and X. Xu, "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs," *IEEE Trans. on Selected Areas in Communications (JSAC)*, vol. 21, no. 10, pp. 1752–1763, 2003.

[73] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media", Tech. Report, Microsoft Research, Redmond, 2002.

[74] N. Thomos and P. Frossard, "Raptor network video coding," in *Proc. the international workshop on mobile video*, New York, NY, USA, pp. 19–24, 2007.

[75] ——, "Collaborative video streaming with Raptor network coding," in *Proc. IEEE International Conference on Multimedia and Expo*, pp. 497–500, 2008.

[76] A. Albanese, J. Blomer, J. Edmonds, and M. Luby, "Priority encoding transmission," *IEEE Trans. on Information Theory*, 1996.

[77] X. Liu, G. Cheung, and C. N. Chuah, "Structured network coding and cooperative local peer-to-peer repair for mbms video streaming," in *Proc. IEEE Int. Workshop on Multimedia Signal Processing*, 2008.

[78] N. Thomos, J. Chakareski, and P. Frossard, "Randomized network coding for UEP video delivery in overlay networks," in *Proc. International Conference on Multimedia and Expo*, 2009.

[79] J. M. Walsh and S. Weber, "A concatenated network coding scheme for multimedia transmission," in *Proc. Fourth Workshop on Network Coding, Theory and Applications, 2008 (NetCod'08)*, 2008, pp. 1–6.

[80] S. Tao and R. Guérin, "On-line estimation of Internet path performance: An application perspective," in *Proc. IEEE INFOCOM'04*, vol. 3, 2005, pp. 1774–1785.

[81] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley, "Inferring link loss using striped unicast probes," in *Proc. IEEE INFOCOM'01*, Anchorage, AL, USA, 2001.

[82] R. Caceres, N. G. Duffield, J. Horowitz, D. Towsley, and T. Bu, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. on Information Theory*, vol. 45, no. 7, pp. 2462–2480, 2002.

[83] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Hybrid ARQ-random network coding for wireless media streaming," in *Proc. IEEE ICCE'08*, 2008, pp. 115–120.

[84] K. Lakshminarayanan, I. Stoica, and S. Shenker, "Building a flexible and efficient routing infrastructure: Need and challenges," 2003.

[85] D. P. Pezaros, D. Hutchison, and U. K. Lancaster, "Quality of service assurance for the next generation internet," *Computing Department, Faculty of Applied Science, Lancaster, UK, LA1 4YR*, 2001.

[86] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Trans. on Vehicular Technology*, vol. 58, no. 2, Feb. 2009.

[87] H. Seferoglu and A. Markopoulou, "Opportunistic network coding for video streaming over wireless," in *Proc. 16th International Packet Video Workshop*, Lausanne, Switzerland, 2007, pp. 191–200.

[88] S. Tao, J. Apostolopoulos, and R. Guerin, "Real-time monitoring of video quality in ip networks," *IEEE/ACM Trans. on Networking*, vol. 16, no. 5, pp. 1052–1065, 2008.

[89] VQEG, "Final report on the validation of objective models of video quality assessment," Tech. Rep., Video Quality Experts Group (VQEG), 2003.

[90] A. Mukherjee, "On the dynamics and significance of low frequency components of internet load," *Internetworking: Research and Experience*, vol. 5, pp. 163–205, 1994.

[91] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[92] D. J. Newman, "The double Dixie Cup problem," *American Mathematical Monthly*, pp. 58–61, 1960.

[93] M. Sharif and B. Hassibi, "A delay analysis for opportunistic transmission in fading broadcast channels," in *Proc. IEEE INFOCOM'05*, vol. 4, 2005.

[94] IEEE, "IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 8: Medium access control (MAC) quality of service enhancements," p. 1, 2005.

[95] Y. Chen, S. Emeott, and R. R. Choudhury, "An analytical model of block acknowledgement and selective retransmission in an 802.11e WLAN network," in *Proc. IEEE GLOBECOM'06*, 2006, pp. 1–5.

[96] Y. J. Liang, J. G. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: Does burst-length matter?" in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, vol. 5, 2003.

[97] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid-a framework for video transmission and quality evaluation," *Lecture notes in Computer Science*, pp. 255–272, 2003.

[98] NS3, "The ns-3 network simulator." [Online]. Available: http://www.nsnam.org/

[99] "Xvid codec." [Online]. Available: http://www.tkn.tu-berlin.de

[100] Y. H. Lin, I. C. Jan, P. C.-I. Ko, Y. Y. Chen, J. M. Wong, and G. J. Jan, "A wireless PDA-based physiological monitoring system for patient transport," *IEEE Trans. Information Technology Biomed.*, vol. 8, no. 4, pp. 439-447, Dec. 2004.

[101] "Schifra Reed-Solomon error correcting code library." [Online]. Available: `http://www.schifra.com/`

[102] X. He and A. Yener, "On the energy-delay trade-off of a two-way relay network," in *Proc. Conference on Information Sciences and Systems (CISS'08)*, pp. 865-870, March 2008.

[103] X. Liu, G. Cheung, and C.-N. Chuah, "Rate-distortion optimized network coding for cooperative video stream repair in wireless peer-to-peer networks," in *Proc. International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM'08)*, June 2008.

[104] B. Shrader and A. Ephremides, "A queueing model for random linear coding," in *Proc. IEEE Military Communications Conference (MILCOM'07)*, October 2007.

[105] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," in *Proc. Workshop on Network Coding, Theory & Applications*, 2007.

[106] E. Castillo, *Extreme value theory in engineering*, Academic Press, August 1988.

[107] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*, Wiley-Interscience, April 2006.

[108] I. F. Akyildiz and X. Wang, "Wireless mesh networks: A survey," *Computer Networks*, vol. 47, pp. 445–487, 2005.

[109] M. Alicherry, R. Bhatia, and L. E. Li, "Joint channel assignment and routing for

throughput optimization in multiradio wireless mesh networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 42, no.11, pp. 1960–1971, 2006.

[110] A. Raniwala, K. Gopalan, and T.-C. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *ACM Mobile Comput. Commun. Rev. (MC2R)*, vol. 8, no. 2, pp. 50–65, 2004.

[111] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *Proc. ACM MobiCom'05*, pp. 73–87, 2005.

[112] A. P. Subramanian, H. Gupta, S. R. Das, and J. Cao, "Minimum interference channel assignment in multiradio wireless mesh networks," *IEEE Trans. on Mobile Computing*, pp. 1459–1473, 2008.

[113] S. Avallone and I. F. Akyildiz, "A channel assignment algorithm for multi-radio wireless mesh networks," *Computer Communications*, vol. 31, no. 7, pp. 1343–1353, 2008.

[114] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *Proc. IEEE INFOCOM'05*, pp. 2223–2234, 2005.

[115] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian, "Practical, distributed channel assignment and routing in dual-radio mesh networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 99–110, 2009.

[116] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks," in *Proc. IEEE INFOCOM'06*, vol. 6, 2006.

[117] S. Avallone, I. F. Akyildiz, and G. Ventre, "A channel and rate assignment algorithm and a layer-2.5 forwarding paradigm for multi-radio wireless mesh networks," *IEEE/ACM Trans. on Networking*, vol. 17, no. 1, pp. 267–280, 2009.

[118] P. Kyasanur and N. H. Vaidya, "Routing and interface assignment in multi-channel multi-interface wireless networks," in *Proc. IEEE WNCN'05*, vol. 4, 2005.

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

[120] B. J. Ko, V. Misra, J. Padhye, and D. Rubenstein, "Distributed channel assignment in multi-radio 802.11 mesh networks," in *Proc. IEEE WCNC'07*, pp. 3978–3983, 2007.

[121] X. Lin and S. Rasool, "A distributed joint channel-assignment, scheduling and routing algorithm for multi-channel ad-hoc wireless networks," in *Proc. IEEE INFOCOM'07*, pp. 1118–1126, 2007.

[122] K. Xing, X. Cheng, L. Ma, and Q. Liang, "Superimposed code based channel assignment in multi-radio multi-channel wireless mesh networks," in *Proc. ACM MobiCom'07*, pp. 15–26, 2007.

[123] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Information Theory*, vol. 51, no. 6, pp. 1973–1982, 2005.

[124] S. J. Wright, Primal-Dual Interior Point Methods, Society of Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[125] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, The MIT Press, 1991.

[126] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Trans. Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000.

[127] S. Chieochan and E. Hossain, "iCORE: Implementation of multi-channel multi-radio repeater-aided wireless coded network," in *Proc. 13th International Symposium on Wireless Personal Multimedia Communications (WPMC'10)*, Recife, Brazil, Oct. 11-14, 2010.

[128] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11 b mesh network," in *Proc. the 11th Annual International Conference on Mobile Computing and Networking*, p. 42, 2005.

[129] X. Zhang and H. Su, "Network coding based scheduling and routing schemes for service-oriented wireless mesh networks," *IEEE Wireless Communications*, vol. 16, no. 4, pp. 40–46, August 2009.

[130] H. Su and X. Zhang,"Modeling throughput gain of network coding in multi-channel multi-radio wireless ad hoc networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 27, no. 5, pp. 593–605, June 2009.

[131] X. Zhang and B. Li, "On the benefits of network coding in multi-channel wireless networks," in *Proc. IEEE SECON'08*, San Francisco, California, June 17-20, 2008.

[132] S. C. Kwon, F. Hendessi, and F. Fekri, "Cooperative network coding and coding-

aware channel assignment in multi-channel, multi-interface wireless networks," in *Proc. IEEE SECON'09*, pp. 1–9, 2009.

[133] P. Kyasanur and N. H. Vaidya, "Capacity of multi-channel wireless networks: Impact of number of channels and interfaces," in *Proc. the 11th Annual International Conference on Mobile Computing and Networking*, pp. 43–57, 2005.

[134] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 69–74, 2004.

[135] H. Skalli, S. Ghosh, S. K. Das, L. Lenzini, and M. Conti, "Channel assignment strategies for multiradio wireless mesh networks: Issues and solutions," *IEEE Communications Magazine*, vol. 45, no.11, pp. 86–95, 2007.

[136] W. Si, S. Selvakennedy, and A. Y. Zomaya, "An overview of Channel Assignment methods for multi-radio multi-channel wireless mesh networks," *Journal of Parallel Distrib. Comput.*, vol. 70, no. 5, pp. 505–524, May 2010.

[137] M. K. Marina, S. R. Das and A. P. Subramanian "A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks," *Computer Networks*, vol. 54, no. 2, pp. 241–256, Feb. 2010.

[138] J. Tang, G. Xue, and W. Zhang, "Interference-aware topology control and QoS routing in multi-channel wireless mesh networks," in *Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, p. 68–77, 2005.

[139] S. Merlin, N. Vaidya, and M. Zorzi, "Resource allocation in multi-radio multi-channel multi-hop wireless networks," in *Proc. IEEE INFOCOM'08*, pp. 610–618, 2008.

[140] A. H. M. Rad and V. W. S. Wong, "Joint optimal channel assignment and congestion control for multi-channel wireless mesh networks," in *Proc. IEEE ICC'06*, pp. 1984–1989, 2006.

[141] A. H. M. Rad and V. W. S. Wong, "Joint channel allocation, interface assignment and MAC design for multi-channel wireless mesh networks," in *Proc. IEEE INFOCOM'07*, pp. 1469–1477, 2007.

[142] J. Luo, C. Rosenberg, and A. Girard, "Engineering wireless mesh networks: Joint scheduling, routing, power control and rate adaptation," *IEEE/ACM Transaction on Networking*, to appear.

[143] C. Chereddi, P. Kyasanur, and N. H. Vaidya, "Net-X: A multichannel multi-interface wireless mesh implementation," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 3, pp. 84–95, 2007.

[144] T. Shen and N. Vaidya, "Experiments on a multichannel multi-interface wireless mesh network," Tech. Rep., Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, USA, May 2008.

[145] Y. Zhang, "Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment," Tech. Rep. (TR96-01), Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD, July, 1995.

[146] P. Bahl, R. Chandra, P. P. C. Lee, V. Misra, J. Padhye, D. Rubenstein, and Y. Yu, "Opportunistic use of client repeaters to improve performance of WLANs," *IEEE/ACM Trans. on Networking (TON)*, vol. 17, no. 4, pp. 1160–1171, 2009.

[147] MATLAB, "The Matrix Laboratory." [Online]. Available:
http://www.mathworks.com/

[148] Soekirs Engineering, Inc., "net451 Soekris borads and cases" [Online]. Available:
http://www.soekris.com/products/net4521.html

[149] Voyage Linux, "x86 Embedded Linux," [Online]. Available:
http://www.voyage.linux.hk/

[150] DLink, "DLink-WNA2330 wireless interfaces" [Online]. Available:
http://www.dlink.ca/

[151] libcap, "libcap – a system-independent interface for user-level packet capture. libpcap provides a portable framework for low-level network monitoring. " [Online]. Available:
http://sourceforge.net/projects/libpcap/

Appendix A

Inner Matrices of DTMC for a Lossy Wireless Unicast Exchange Network

In this appendix, we give the details of each inner matrix $\mathbf{i} \in \{\mathbf{B}, \mathbf{C}, \mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_{1b}, \mathbf{A}_2\}$ as defined in Chapter 3 for a wireless unicast exchange network. First, let $N = N_r + 2$, where N_r is the total buffer size of the AP. Also let $\alpha_i' = 1 - \alpha_i$, $\beta_i' = 1 - \beta_i, i \in \{1, 2\}$, and $\theta' = 1 - \theta$. We now describe each inner matrix as follows.

A.0.1 Descriptions of Matrix B

The elements of matrix **B** are given in Table A.1, along with their appropriate dimensions. Notice a common multiplier α_1', which represents a transition from level 0 to

itself. We define all elementary matrices appeared Table A.1 as follows:

$$\Phi_{v\times v}=\begin{bmatrix}1 & & & & \\ \theta & \theta' & & & \\ & \ddots & \ddots & & \\ & & & \theta & \theta'\end{bmatrix},\Omega_{v\times v}=\begin{bmatrix}\theta & & & \\ & \ddots & & \\ & & & \theta\end{bmatrix},\Gamma_{v\times w}=\begin{bmatrix}1 & & & 0 & 0\\ \theta & & & \vdots & \vdots\\ & \ddots & & & \\ & & \theta & 0 & 0\end{bmatrix}$$

$$\Lambda_{v\times w}=\begin{bmatrix}\alpha_2'\beta_2'\theta & \alpha_2\beta_2\theta & & & 0\\ \alpha_2'\beta_2'\theta & \alpha_2\beta_2\theta & & & \vdots\\ & \ddots & \ddots & & \\ & & \alpha_2'\beta_2'\theta & \alpha_2\beta_2\theta & 0\end{bmatrix},\Delta_{v\times v}=\begin{bmatrix}\alpha_2'\beta_2'\theta' & \alpha_2\beta_2\theta' & & \\ & \ddots & \ddots & \\ & & \alpha_2'\beta_2'\theta' & \alpha_2\beta_2\theta'\\ & & & \alpha_2'\theta'\end{bmatrix}$$

$$\Psi_{v\times v}=\begin{bmatrix}\alpha_2'\beta_2' & \alpha_2\beta_2 & & \\ \alpha_2'\beta_2'\theta & \alpha_2'\beta_2'\theta'+\alpha_2\beta_2\theta & \alpha_2\beta_2\theta' & \\ & \ddots & \ddots & \ddots\\ & & \alpha_2'\beta_2'\theta & \alpha_2'\theta'+\alpha_2\beta_2\theta\end{bmatrix},$$

$$\widehat{\Phi}_{v\times v}=\begin{bmatrix}0 & 1 & & & \\ \vdots & \theta & \theta' & & \\ & & \ddots & \ddots & \\ \vdots & & & \theta & \theta'\\ 0 & & & & \theta\end{bmatrix},\widehat{\Omega}_{v\times v}=\begin{bmatrix}0 & \theta & & \\ & \ddots & \ddots & \\ & & 0 & \theta\end{bmatrix},\widehat{\Gamma}_{v\times w}=\begin{bmatrix}0 & 1 & & & 0\\ \vdots & \theta & & & \vdots\\ & & \ddots & & \\ 0 & & & \theta & 0\end{bmatrix}.$$

For matrix $\mathbf{A}_{1b,B}$, the only difference from $\mathbf{A}_{1,B}$ is the absence of parameter α_2', which indicates that s_2's buffer is already full. Therefore packets will be discarded anyway

Table A.1: Elements of Matrix B

Matrix	$\mathbf{B}^{(0)}$	$\mathbf{A}_2^{(i)}$	$\mathbf{A}_1^{(i)}$
$\mathbf{B}_B$	$\alpha_1'\alpha_2' \cdot \Phi_{N\times N}$	$\alpha_1'\alpha_2' \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1'\alpha_2' \cdot \Omega_{(N-i)\times(N-i)}$
$\mathbf{C}_B$	$\alpha_1'\alpha_2 \cdot \Phi_{N\times N}$	$\alpha_1'\alpha_2 \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1'\alpha_2 \cdot \Omega_{(N-i)\times(N-i)}$
$\mathbf{A}_{2,B}$	$\alpha_1'\alpha_2\beta_2 \cdot \widehat{\Phi}_{N\times N}$	$\alpha_1'\alpha_2\beta_2 \cdot \widehat{\Gamma}_{(N-i)\times(N-i+1)}$	$\alpha_1'\alpha_2\beta_2 \cdot \widehat{\Omega}_{(N-i)\times(N-i)}$
$\mathbf{A}_{1,B}$	$\alpha_1' \cdot \Psi_{N\times N}$	$\alpha_1' \cdot \Lambda_{(N-i)\times(N-i+1)}$	$\alpha_1' \cdot \Delta_{(N-i)\times(N-i)}$
$\mathbf{A}_{0,B}$	$\alpha_1'\alpha_2\beta_2' \cdot \Phi_{N\times N}$	$\alpha_1'\alpha_2\beta_2' \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1'\alpha_2\beta_2' \cdot \Omega_{(N-i)\times(N-i)}$

should they arrive. The subscript $v \times w$, or $v \times v$, here specifies the dimension of a matrix defined. In particular, it should be noted that $v \times v$ denotes a dimension of a square matrix, whereas $v \times w$ indicates a dimension of an arbitrary matrix. These notations will be used throughout the rest of this appendix.

A.0.2 Descriptions of Matrix C

All the elementary matrices defined above for matrix **B** are applicable to matrix **C** as well. We show the elements of matrix **C**, along with their appropriate dimensions, in Table A.2. Notice a common multiplier α_1 here. This should come as no surprise because matrix **C** represents a transition from level 0 to level 1, which requires one packet arrival into s_1.

Again, matrix $\mathbf{A}_{1b,C}$ differs from $\mathbf{A}_{1,C}$ only in the absence of parameter α_2'. The

Table A.2: Elements of Matrix C

Matrix	$\mathbf{B}^{(0)}$	$\mathbf{A}_2^{(i)}$	$\mathbf{A}_1^{(i)}$
$\mathbf{B}_C$	$\alpha_1\alpha_2' \cdot \Phi_{N\times N}$	$\alpha_1\alpha_2' \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2' \cdot \Omega_{(N-i)\times(N-i)}$
$\mathbf{C}_C$	$\alpha_1\alpha_2 \cdot \Phi_{N\times N}$	$\alpha_1\alpha_2 \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2 \cdot \Omega_{(N-i)\times(N-i)}$
$\mathbf{A}_{2,C}$	$\alpha_1\alpha_2'\beta_2 \cdot \widehat{\Phi}_{N\times N}$	$\alpha_1\alpha_2'\beta_2 \cdot \widehat{\Gamma}_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2'\beta_2 \cdot \widehat{\Omega}_{(N-i)\times(N-i)}$
$\mathbf{A}_{1,C}$	$\alpha_1 \cdot \Psi_{N\times N}$	$\alpha_1 \cdot \Lambda_{(N-i)\times(N-i+1)}$	$\alpha_1 \cdot \Delta_{(N-i)\times(N-i)}$
$\mathbf{A}_{0,C}$	$\alpha_1\alpha_2\beta_2' \cdot \Phi_{N\times N}$	$\alpha_1\alpha_2\beta_2' \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2\beta_2' \cdot \Omega_{(N-i)\times(N-i)}$

same reasoning as given above also applies here.

A.0.3 Descriptions of Matrix $\mathbf{A}_0$

The contents of matrix $\mathbf{A}_0$ are slightly different from matrix $\mathbf{C}$ because they both represent a transition from a lower level to the one just above it. We additionally define new elementary matrices below, and show the elements of matrix $\mathbf{A}_0$, along with their appropriate dimensions, in Table A.3. Notice a common multiplier α_1, which indicates an packet arrival into s_1, triggering a transition from level i to level $i+1$, $i \in \{0, 1, \cdots, N_1+1\}$.

Define:

$$\widetilde{\Phi}_{v\times v}=\begin{bmatrix}1 & & & & \\ \theta & \theta' & & & \\ \ddots & \ddots & & & \\ & & \theta & \theta' & \\ & & & \theta & \frac{\theta'}{\beta_1'}\end{bmatrix},\widetilde{\Phi}^{\triangleright}{}_{v\times v}=\begin{bmatrix}1 & & & & \\ \theta & \theta' & & & \\ \ddots & \ddots & & & \\ & & \theta & \theta' & \\ & & & \theta & \frac{\theta'}{\beta_1'\beta_2'}\end{bmatrix},$$

$$\widetilde{\Omega}_{v\times v}=\begin{bmatrix}\theta & & & \\ & \ddots & & \\ & & \theta & \\ & & & \frac{\theta}{\beta_1'}\end{bmatrix},\widetilde{\Omega}^{\triangleright}{}_{v\times v}=\begin{bmatrix}\theta & & & \\ & \ddots & & \\ & & \theta & \\ & & & \frac{\theta}{\beta_1'\beta_2'}\end{bmatrix},$$

$$\widetilde{\Delta}_{v\times v}=\begin{bmatrix}\alpha_2'\beta_2'\theta' & \alpha_2\beta_2\theta' & & & \\ & \ddots & \ddots & & \\ & & \alpha_2'\beta_2'\theta' & \alpha_2\beta_2\theta' & \\ & & & \alpha_2'\beta_2'\theta' & \frac{\alpha_2\beta_2\theta'}{\beta_1'} \\ & & & & \frac{\alpha_2'\theta'}{\beta_1'}\end{bmatrix},$$

$$\widetilde{\Psi}_{v\times v}=\begin{bmatrix}\alpha_2'\beta_2' & \alpha_2\beta_2 & & & \\ \alpha_2'\beta_2'\theta & \alpha_2'\beta_2'\theta'+\alpha_2\beta_2\theta & \alpha_2\beta_2\theta' & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_2'\beta_2'\theta & \alpha_2'\beta_2'\theta'+\alpha_2\beta_2\theta & \frac{\alpha_2\beta_2\theta'}{\beta_1'} \\ & & & \alpha_2'\beta_2'\theta & \frac{\alpha_2'\theta'+\alpha_2\beta_2\theta}{\beta_1'}\end{bmatrix},$$

$$\widehat{\Phi}^{\triangleright}{}_{v\times v} = \begin{bmatrix} 0 & 1 & & & & & \\ \vdots & \theta & \theta' & & & & \\ & & \ddots & \ddots & & & \\ & & & & \theta & \theta' & \\ & & & & & \theta & \frac{\theta'}{\beta_1'} \\ 0 & & & & & & \frac{\theta}{\beta_1'} \end{bmatrix}, \widehat{\Omega}^{\triangleright}{}_{v\times v} = \begin{bmatrix} 0 & \theta & & & \\ & \ddots & \ddots & & \\ & & 0 & \theta & \\ & & & 0 & \frac{\theta}{\beta_1'} \end{bmatrix}.$$

Table A.3: Elements of Matrix $\mathbf{A}_0$

Matrix	$\mathbf{B}^{(0)}$	$\mathbf{A}_2^{(i)}$	$\mathbf{A}_1^{(i)}$
$\mathbf{B_C}$	$\alpha_1\alpha_2'\beta_1' \cdot \widetilde{\Phi}_{N\times N}$	$\alpha_1\alpha_2'\beta_1' \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2' \cdot \widetilde{\Omega}_{(N-i)\times(N-i)}$
$\mathbf{C_C}$	$\alpha_1\alpha_2\beta_1 \cdot \widetilde{\Phi}_{N\times N}$	$\alpha_1\alpha_2\beta_1' \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2\beta_1' \cdot \widetilde{\Omega}_{(N-i)\times(N-i)}$
$\mathbf{A_{2,C}}$	$\alpha_1\alpha_2'\beta_1'\beta_2 \cdot \widehat{\Phi}^{\triangleright}{}_{N\times N}$	$\alpha_1\alpha_2'\beta_1'\beta_2 \cdot \widehat{\Gamma}_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2'\beta_1'\beta_2 \cdot \widehat{\Omega}^{\triangleright}{}_{(N-i)\times(N-i)}$
$\mathbf{A_{1,C}}$	$\alpha_1\beta_1' \cdot \widetilde{\Psi}_{N\times N}$	$\alpha_1\beta_1' \cdot \Lambda_{(N-i)\times(N-i+1)}$	$\alpha_1\beta_1' \cdot \widetilde{\Delta}_{(N-i)\times(N-i)}$
$\mathbf{A_{0,C}}$	$\alpha_1\alpha_2\beta_1'\beta_2' \cdot \widetilde{\Phi}^{\triangleright}{}_{N\times N}$	$\alpha_1\alpha_2\beta_1'\beta_2' \cdot \Gamma_{(N-i)\times(N-i+1)}$	$\alpha_1\alpha_2\beta_1'\beta_2' \cdot \widetilde{\Omega}^{\triangleright}{}_{(N-i)\times(N-i)}$

For matrix $\mathbf{A}_{1b,A_0}$, the only difference from $\mathbf{A}_{1,A_0}$ is the absence of parameter α_2'. The same reasoning as given above again applies.

A.0.4 Descriptions of Matrix $\mathbf{A}_2$

We show the elements of matrix $\mathbf{A}_2$ in Table A.4. Since matrix $\mathbf{A}_2$ represents a transition from level $i+1$ to level i, $i \in \{0, 1, \cdots, N_1+1\}$, a service completion at s_1 is

therefore required. A common multiplier in this case is then β_1, as shown in Table A.4. We define additional elementary matrices shown in Table A.4 as follows:

$$\Phi^{+}{}_{v\times w} = \begin{bmatrix} 1 & & & & \\ \theta & \theta' & & & \\ & \ddots & \ddots & & \\ & & \theta & \theta' & \\ & & & \theta & \end{bmatrix}, \Omega^{+}{}_{v\times w} = \begin{bmatrix} \theta' & & \\ & \ddots & \\ & & \theta' \\ & & 0 \end{bmatrix}, \Gamma^{+}{}_{v\times v} = \begin{bmatrix} \theta & & & 0 \\ \theta & & & \\ & \ddots & & \\ & & \theta & 0 \end{bmatrix}$$

$$\Phi^{\circ}{}_{v\times w} = \begin{bmatrix} 0 & 1 & & & \\ \vdots & \theta & \theta' & & \\ & & \ddots & \ddots & \\ & & & \theta & \theta' \\ \vdots & & & & \theta \\ 0 & & \cdots & & 0 \end{bmatrix}, \Gamma^{\circ}{}_{v\times v} = \begin{bmatrix} 0 & \theta & & \\ \vdots & \theta & & \\ & & \ddots & \\ 0 & & & \theta \end{bmatrix} \Omega^{\circ}{}_{v\times w} = \begin{bmatrix} 0 & \theta' & & \\ \vdots & & \ddots & \\ & & & \theta' \\ 0 & & & 0 \\ 0 & \cdots & & 0 \end{bmatrix},$$

$$\Psi^{+}{}_{v\times w} = \begin{bmatrix} \alpha_2'\beta_2' & \alpha_2\beta_2 & & & \\ \alpha_2'\beta_2'\theta & \alpha_2'\beta_2'\theta' + \alpha_2\beta_2\theta & \alpha_2\beta_2\theta' & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_2'\beta_2'\theta & \alpha_2'\beta_2'\theta' + \alpha_2\beta_2\theta & \\ & & & \alpha_2'\beta_2'\theta & \end{bmatrix},$$

$$\Delta^{+}{}_{v\times w} = \begin{bmatrix} \alpha_2'\beta_2'\theta' & \alpha_2\beta_2\theta' & & \\ & \ddots & \ddots & \\ & & \alpha_2'\beta_2'\theta' & \alpha_2\beta_2\theta' \\ & & & \alpha_2'\beta_2'\theta' \\ & & & 0 \end{bmatrix}, \Lambda^{+}{}_{v\times v} = \begin{bmatrix} \alpha_2'\beta_2'\theta & \alpha_2\beta_2\theta & & \\ \alpha_2'\beta_2'\theta & \alpha_2\beta_2\theta & & \\ & \ddots & \ddots & \\ & & \alpha_2'\beta_2'\theta & \alpha_2\beta_2\theta \end{bmatrix}.$$

Table A.4: Elements of Matrix $\mathbf{A}_2$

Matrix	$\mathbf{B}^{(0)}$	$\mathbf{A}_2^{(i)}$	$\mathbf{A}_1^{(i)}$
$\mathbf{B}_{A_2}$	$\kappa\alpha_2' \cdot \Phi^{+}{}_{N\times(N-1)}$	$\kappa\alpha_2' \cdot \Gamma^{+}{}_{(N-i)\times(N-i)}$	$\kappa\alpha_2' \cdot \Omega^{+}{}_{(N-i)\times(N-i-1)}$
$\mathbf{C}_{A_2}$	$\kappa\alpha_2 \cdot \Phi^{+}{}_{N\times(N-1)}$	$\kappa\alpha_2 \cdot \Gamma^{+}{}_{(N-i)\times(N-i)}$	$\kappa\alpha_2 \cdot \Omega^{+}{}_{(N-i)\times(N-i-1)}$
$\mathbf{A}_{2,A_2}$	$\kappa\alpha_2'\beta_2 \cdot \Phi^{\circ}{}_{N\times(N-1)}$	$\kappa\alpha_2'\beta_2 \cdot \Gamma^{\circ}{}_{(N-i)\times(N-i)}$	$\kappa\alpha_2'\beta_2 \cdot \Omega^{\circ}{}_{(N-i)\times(N-i-1)}$
$\mathbf{A}_{1,A_2}$	$\kappa\alpha_1' \cdot \Psi^{+}{}_{N\times(N-1)}$	$\alpha_1' \cdot \Lambda^{+}{}_{(N-i)\times(N-i)}$	$\kappa\alpha_1' \cdot \Delta_{(N-i)\times(N-i-1)}$
$\mathbf{A}_{0,A_2}$	$\kappa\alpha_2\beta_2' \cdot \Phi^{+}{}_{N\times(N-1)}$	$\kappa\alpha_2\beta_2' \cdot \Gamma^{+}{}_{(N-i)\times(N-i)}$	$\kappa\alpha_2\beta_2' \cdot \Omega^{+}{}_{(N-i)\times(N-i-1)}$
Notation: $\kappa = \alpha_1'\beta_1$			

To satisfy the boundary conditions, the lowest right corner element of $\mathbf{A}_{1,A_2}$ must not contain β_2' because this element represents a transition from one buffer-overflow state of the relay to another overflow state, which at the same time requires a packet from s_1 to finish and enter the buffer. Therefore, no matter a packet from s_2 finishes being served or not, it will be blocked anyway. Hence, β_2' can be eliminated from the boundary element.

This is also true for the lowest right corner element of $\mathbf{A}_{0,A_2}$. β_2' must also be eliminated. Finally, for matrix $\mathbf{A}_{1b,A_2}$, the only difference from $\mathbf{A}_{1,A_2}$ is the absence of parameter α_2'. The same reasoning as given above again applies.

A.0.5 Descriptions of Matrices $\mathbf{A}_1$ and $\mathbf{A}_{1b}$

The elements of matrix $\mathbf{A}_1$ are shown in Table A.5. The dimensions of these elements are shown in the second row of the table. We define additional elementary matrices used in Table A.5 as follows:

$$\Phi^{\star}{}_{v\times v} = \begin{bmatrix} \alpha_1'\beta_1'\theta' + \alpha_1\beta_1\theta & & & & \\ \alpha_1\beta_1\theta & \alpha_1'\beta_1'\theta' & & & \\ & \ddots & \ddots & & \\ & & & \alpha_1\beta_1\theta & \alpha_1'\beta_1'\theta' \end{bmatrix},$$

$$\widehat{\Phi}^{\star}{}_{v\times v} = \begin{bmatrix} 0 & \alpha_1'\beta_1'\theta' + \alpha_1\beta_1\theta & & & \\ \vdots & \alpha_1\beta_1\theta & \alpha_1'\beta_1'\theta' & & \\ & & \ddots & \ddots & \\ 0 & & & \alpha_1\beta_1\theta & \alpha_1'\theta' \\ & & & & \alpha_1\beta_1\theta \end{bmatrix},$$

$$\Psi^{\star}{}_{v\times v} = \begin{bmatrix} \alpha_2'\beta_2'(\alpha_1'\beta_1'\theta' + \alpha_1\beta_1\theta) & \alpha_2\beta_2(\alpha_1'\beta_1'\theta' + \alpha_1\beta_1\theta) & \\ \alpha_1\alpha_2'\beta_1\beta_2'\theta & \alpha_1'\alpha_2'\beta_1'\beta_2'\theta' + \alpha_1\alpha_2\beta_1\beta_2\theta & \alpha_1'\alpha_2\beta_1'\beta_2\theta' \\ \ddots & \ddots & \ddots \\ & & \alpha_1'\alpha_2\beta_2\theta' \\ & \alpha_1\alpha_2'\beta_1\beta_2'\theta & \alpha_1'\alpha_2'\theta' + \alpha_1\alpha_2\beta_1\beta_2\theta \end{bmatrix}.$$

Table A.5: Elements of Matrix $\mathbf{A}_1$

Matrix	$\mathbf{B}^{(0)}$	$\mathbf{C}^{(0)}$	$\mathbf{A}_2^{(i)}$	$\mathbf{A}_1^{(i)}$	$\mathbf{A}_0^{(i)}$
Dim.	$N\times N$	$N\times(N-1)$	$(N-i)\times(N-i+1)$	$(N-i)\times(N-i)$	$(N-i)\times(N-i-1)$
$\mathbf{B}_{A_1}$	$\kappa_3\beta_1'\cdot\widetilde{\Phi}$	$\kappa_2\beta_1\cdot\Phi^{+}$	$\kappa_3\beta_1'\cdot\Gamma$	$\alpha_2'\cdot\Phi^{\star}$	$\kappa_2\beta_1\cdot\Omega^{+}$
$\mathbf{C}_{A_1}$	$\kappa_1\beta_1'\cdot\widetilde{\Phi}$	$\kappa_4\beta_1\cdot\Phi^{+}$	$\kappa_1\beta_1'\cdot\Gamma$	$\alpha_2\cdot\Phi^{\star}$	$\kappa_4\beta_1\cdot\Omega^{+}$
$\mathbf{A}_{2,A_1}$	$\kappa_3\beta_1'\beta_2\cdot\widehat{\Phi}^{\triangleright}$	$\kappa_2\beta_1\beta_2\cdot\Phi^{\circ}$	$\kappa_3\beta_1'\beta_2\cdot\widehat{\Gamma}$	$\kappa_2\beta_1\beta_2\cdot\widehat{\Phi}^{\star}$	$\kappa_2\beta_1\beta_2\cdot\Omega^{\circ}$
$\mathbf{A}_{1,A_1}$	$\alpha_1'\beta_1'\cdot\widetilde{\Psi}$	$\alpha_1\beta_1\cdot\Psi^{+}$	$\alpha_1'\beta_1'\cdot\Lambda$	$\Psi^{\star}$	$\alpha_1\beta_1\cdot\Delta^{+}$
$\mathbf{A}_{0,A_1}$	$\kappa_1\beta_1'\beta_2'\cdot\widetilde{\Phi}$	$\kappa_4\beta_1\beta_2'\cdot\Phi^{+}$	$\kappa_1\beta_1'\cdot\Gamma$	$\alpha_2\beta_2'\cdot\Phi^{\star}$	$\kappa_4\beta_2'\cdot\Omega^{+}$
Notations: $\kappa_1=\alpha_1'\alpha_2, \kappa_2=\alpha_1\alpha_2', \kappa_3=\alpha_1'\alpha_2', \kappa_4=\alpha_1\alpha_2$					

We note several boundary conditions within matrix $\mathbf{A}_1$ as follows. The lowest right corner element of matrices $\mathbf{B}_{A_1}$ and $\mathbf{A}_{1,A_1}$ is given by $\alpha_2'(\alpha_1'\theta' + \alpha_1\beta_1\theta)$. For $\mathbf{C}_{A_1}$, it is $\alpha_2(\alpha_1'\theta' + \alpha_1\beta_1\theta)$. For $\mathbf{A}_{0,A_1}$, it is $\alpha_2\beta_2'(\alpha_1'\theta' + \alpha_1\beta_1\theta)$. These conditions represent a

transition from a state, in which overflow occurs at the relay's buffer, to another, in which overflow also occurs at the relay. Notice the absence of β_1' and β_1' here. Buffer overflow at s_2 is captured by matrix $\mathbf{A}_{1b,A_1}$, which is not shown here. The only difference of this boundary matrix from $\mathbf{A}_{1,A_1}$ is the absence of parameter α_2'. The same reasoning as given above again applies. Finally, buffer overflow at s_1 is captured by $\mathbf{A}_{1b}$, which differs from $\mathbf{A}_1$ only in the absence of α_1'. This is intuitive because if the buffer of s_1 is full, further arrivals into this source will be blocked anyway. Parameter α_1' thus can be eliminated from all the elements in $\mathbf{A}_1$.

Appendix B

Forward Auction Algorithm

The pseudo code of a standard forward auction algorithm is shown in this appendix. The key idea of this algorithm is to assign a time slot to the hyperarc with the highest bid, i.e., the hyperarc that will utilize this time slot most efficiently in terms of packet injection rate.

Algorithm B.1: Forward auction algorithm.

input : $b_{lk}, p, \pi, C(l), \epsilon, \Phi^{\circ}$

output:

1. Assignment $\Phi = \{(l,k) \mid$ each hyperarc l is assigned to a single distinct time slot$\}$
2. Price vector $p = (p_1, \ldots, p_\tau)$ and Profit vector $\pi = (\pi_1, \ldots, \pi_L)$

Initialization:

unassignedHyperarcs $= \{l \mid l$ unassigned$\}$ and $\Phi = \Phi^{\circ}$

while unassignedHyperarcs *NOT empty* **do**

  for $l \in$ unassignedHyperarcs **do**

    Find time slot $k^* = \arg\max_{k \in C(l)} \{b_{lk} - p_k\}$ Set

    $\eta_l = \max_{k \in C(l)} \{b_{lk} - p_k\}$, and $w_l = \max_{k \in C(l), k \neq k^*} \{b_{lk} - p_k\}$

    if k^* *is the only member of* $C(l)$ **then**

      Set $w_l = -\infty$;

    Set $p_{k^*} := \max\{\rho, b_{lk^*} - w_l + \epsilon\}$ and $\pi_l := w_l - \epsilon$

    if $b_{lk^*} - w_l + \epsilon > \rho$ **then**

      Add (l, k^*) to Φ

    if k^* *was assigned to hyperarc* $l' \neq l$ *at start of iteration* **then**

      Remove (l', k^*) from Φ

Vitae

Surname: Chieochan ***Given Name***: Surachai

Place of Birth: Thailand ***Date of Birth***: February 13, 1978

Educational Institutes Attended

M.Eng. (Telecommunications), Asian Institute of Technology 2004 to 2006

BS.EE., Washington University in St.Louis 1998 to 2002

1. **S. Chieochan** and E. Hossain, "Adaptive Radio Resource Allocation in OFDMA Systems: A Survey of the state-of-the-art Approaches," *Wiley Wireless Communications and Mobile Computing*, no.4, vol.9, 2009 (Invited Paper).

2. **S. Chieochan** and E. Hossain, "Channel Assignment Schemes for Infrastructure-Based 802.11 WLANs: A Survey," in *IEEE Communications Surveys and Tutorials*, vol.12 no.1, pp. 124136, 2010.

3. **S. Chieochan** and E. Hossain, "Wireless fountain coding with IEEE 802.11e block ACK for media streaming in combined wireline/802.11 networks: A performance study," *IEEE Transactions on Mobile Computing*, vol.10, no.10, pp. 14161433, Oct., 2011.

4. **S. Chieochan** and E. Hossain, "Downlink media streaming with wireless fountain coding in combined wireline/802.11 networks," *Wireless communications and Mobile Computing (Wiley)*, to appear (published online on 18 January 2011, DOI: 10.1002/wcm.1092).

5. **S. Chieochan** and E. Hossain, "Channel Assignment for Throughput Optimization in Multi-Channel Multi-Radio Wireless Mesh Networks Using Network Coding," submitted to *IEEE Transactions on Mobile Computing*.

6. **S. Chieochan** and E. Hossain, "Network Coding for Unicast in a WiFi Hotspot: Promises, Challenges, and Testbed Implementation," submitted to *Computer Networks (Elsevier)*.

7. **S. Chieochan**, E. Hossain, T. Issariyakul, and D. Niyato "Opportunistic Network Coding and Dynamic Buffer Allocation in a Wireless Butterfly Network," in *Proc. IEEE GLOBECOM'09*, Nov. 2009.

8. **S. Chieochan** and E. Hossain, "iCORE: Implementation of multi-channel multi-radio repeater-aided network coding for WiFi," in *Proc. of the 13th International Symposium on Wireless Personal Multimedia Communications (WPMC'10)*, Recife, Brazil, 10-14 October 2010.

9. **S. Chieochan** and E. Hossain, "Cooperative Relaying in Wi-Fi Networks with Network Coding," submitted to IEEE Wireless Communications.

10. **S. Chieochan** and E. Hossain, " Dynamic Buffer Allocation in Coded Wireless Butterfly Networks for UDP Delivery," submitted to *IEEE Transactions on Mobile Computing*.

Title of Thesis: Network Coded Media Distribution in Infrastructure Wireless Mesh Networks

Author: _____

SURACHAI CHIEOCHAN

September 2011

THESIS WITHHOLDING FORM

At our request, the commencement of the period for which the partial license shall operate shall be delayed from September 2011 for a period of at least six months.

_____

(Supervisor)

_____

(Department Chairman)

_____

(Dean of Graduate Studies)

_____

(Signature of Author)

_____

(Date)