

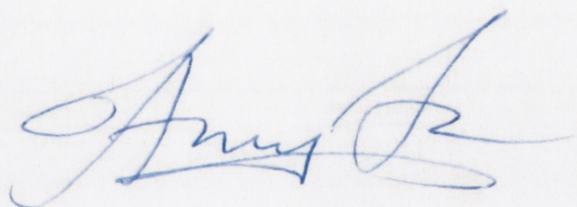
Analysis and Design of Deterministic Network Coding

Amy Fu

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

October 2014

Except where otherwise indicated, this thesis is my own original work.

A handwritten signature in blue ink, appearing to read 'Amy Fu', with a stylized, flowing script.

Amy Fu

4 - NOV - 2014

To my dearest friends and family whose love, patience, insight, humour, and the occasional bit of prodding helped me get across the finish line.

Acknowledgments

I am extremely grateful to my supervisor Dr. Parastoo Sadeghi for her guidance and support throughout my PhD candidature, and the many hours she spent patiently proofreading and providing feedback on my writing. I'd like to thank her for giving me both direction and the freedom to pursue my research interests independently, and sharing her perspective and experience in helping me grow as a researcher.

I would also like to thank Professor Muriel Médard for taking the time to read and provide feedback on a number of publications, and Professor Rod Kennedy for our discussions on research and thesis writing.

Abstract

In this thesis, we are concerned with the analysis and design of deterministic network coding schemes in a single hop wireless broadcast network. We analyse the throughput delay performance achieved under existing transmission schemes in the literature, and obtain insights into the characteristics of network coding schemes that allow them to achieve good performance. We use the results of our analysis to design a new transmission scheme that dynamically balances throughput and delay requirements to achieve improved performance, and derive a low feedback adaptation of this scheme, for use in practical applications with limited feedback.

We introduce the notion of intermediate packet delivery, where packets may be decoded in the correct order by receivers who have received incomplete information from the sender. We show that assumptions used to determine the capacity bounds in the literature do not always apply under intermediate packet delivery, and it is possible for the rate to exceed the min-cut bound. We will demonstrate how intermediate packet delivery can be used to achieve simultaneous multi-rate delivery under non-uniform receiver channel rates, and transmission optimal asynchronous packet delivery under uniform channel rates. We design a fairness algorithm that allocates the sender's resources so all receivers experience fair delivery rate performance.

Contents

Acknowledgments	vii
Abstract	ix
1 Introduction	1
1.1 Throughput and delay	3
1.1.1 Rate control	3
1.1.2 Coding	4
1.2 Fairness	6
1.3 Thesis contributions and outline	7
1.3.1 Throughput delay analysis of coding strategies	7
1.3.2 Dynamic rate control	7
1.3.3 Intermediate packet delivery	8
1.3.4 Video on demand using intermediate packet delivery	8
2 Background	11
2.1 Network coding	11
2.1.1 Butterfly network example	12
2.2 Performance measures	13
2.2.1 Decoding and delivery delay	13
2.2.2 Throughput and delivery rate	15
2.2.3 Throughput delay performance	16
2.3 System model	16
2.4 Transmission scheme model	17
2.4.1 Packet backlog	18
2.4.2 Rate control block	18
2.4.3 Transmission queue	18
2.4.4 Coding block	18
2.4.5 Feedback	19
2.5 Transmission schemes	19
2.5.1 Rate control schemes	20
2.5.1.1 Bernoulli scheme	20

2.5.1.2	Block-based scheme	21
2.5.1.3	Delay threshold scheme	21
2.5.1.4	Sliding window scheme	21
2.5.1.5	Dynamic rate control and queue threshold schemes . .	22
2.5.2	Coding schemes	22
2.5.2.1	Coding scheme A	22
2.5.2.2	Coding scheme B	23
2.5.2.3	Moded scheme B	24
2.5.2.4	RLNC scheme	26
2.5.2.5	RLNC-I scheme	26
2.5.2.6	Features of the coding schemes	26
2.6	Definitions	27
3	Throughput delay analysis	31
3.1	Introduction	31
3.1.1	Throughput delay analysis	32
3.1.1.1	Rate control analysis	33
3.1.1.2	Coding scheme analysis	33
3.1.2	Contributions and related work	34
3.2	System Model	35
3.2.1	Performance measures	35
3.2.2	Transmission schemes	36
3.2.2.1	Bernoulli rate control	37
3.2.2.2	Coding scheme A	37
3.2.2.3	Coding scheme B	37
3.2.2.4	RLNC-I scheme	37
3.2.3	Effective transmisssion queue	38
3.3	Markov State	38
3.3.1	Knowledge spaces and the Markov state	39
3.3.2	Markov chain model	39
3.3.3	Distribution of Markov states	41
3.4	Zero state delivery	41
3.4.1	Delivery cycle distributions	42
3.4.2	Delay estimate	43
3.5	Leader state delivery	44
3.5.1	Leader state distribution	46
3.5.2	Observations	46
3.5.3	Imperfect feedback	48

3.6	Coefficient-based Delivery	48
3.6.1	Effective Markov state	50
3.6.2	RLNC-I scheme	51
3.6.3	Coding scheme A	52
3.6.4	Coding scheme B	52
3.7	Conclusion	53
4	Dynamic rate control	59
4.1	Introduction	59
4.1.1	Contributions and related work	61
4.2	System Model	62
4.2.1	Performance measures	62
4.2.2	Transmission model	63
4.3	Dynamic rate control scheme	64
4.3.1	Dynamic delivery model	64
4.3.2	Decision metric	65
4.3.2.1	Throughput performance	65
4.3.2.2	Expected time to zero state delivery	66
4.3.2.3	Delivery performance	67
4.3.2.4	Benefits of adding and waiting	68
4.3.2.5	Decision metric	68
4.3.3	Feedback variables	69
4.3.3.1	Number of undelivered packets $\sum_{r=1}^R u(r)$	70
4.3.3.2	Addition rate estimate, λ_{est}	70
4.3.4	Full feedback schemes	71
4.3.4.1	Coding scheme B	72
4.3.4.2	Bernoulli rate control scheme	72
4.3.4.3	Delay threshold scheme	72
4.3.5	Performance comparison	72
4.3.5.1	Bernoulli rate control scheme	72
4.3.5.2	Delay threshold rate control scheme	74
4.3.5.3	Dynamic rate control scheme	74
4.4	Queue threshold scheme	75
4.4.1	Queue threshold rate control scheme	75
4.4.1.1	Decision metric	75
4.4.1.2	Notifications	76
4.4.2	Reduced feedback schemes	76
4.4.2.1	RLNC scheme	77

4.4.2.2	Block-based scheme	77
4.4.2.3	Sliding window scheme	77
4.4.3	Throughput delay performance	78
4.4.3.1	Block coding scheme	78
4.4.3.2	Sliding window scheme	78
4.4.3.3	Queue threshold scheme	79
4.4.4	Notification frequency	79
4.4.4.1	Block coding	80
4.4.4.2	Sliding window	80
4.4.4.3	Queue threshold	81
4.4.5	Upper bounding the delivery delay	81
4.4.5.1	Stable delivery delays	83
4.4.6	Lossy and delayed feedback	83
4.4.6.1	Feedback delay	83
4.4.6.2	Feedback erasures	87
4.5	Conclusion	89
5	Intermediate packet delivery	91
5.1	Introduction	91
5.1.1	Deterministic coding example	93
5.1.2	Contributions and related work	94
5.2	System Model	95
5.2.1	Definitions	96
5.2.2	Transmission scheme	97
5.2.3	Mode selector	98
5.2.4	Moded scheme B	98
5.3	Analytical model	100
5.3.1	Simplified encoding scheme	100
5.3.2	Buffer model	102
5.3.2.1	Regions of the buffer	104
5.3.2.2	Subregions	105
5.3.2.3	Buffer model approximations	105
5.4	Outline of delivery rate analysis	108
5.4.1	Leader transmissions	111
5.4.2	Knowledge differentials	111
5.4.2.1	Knowledge differences	112
5.4.2.2	Codings	113
5.4.3	Delivery rate	113

5.5	Leader transmissions	113
5.5.1	Mode 1 and 2 leader transmissions	114
5.5.2	Buffer position and seen packets	115
5.5.2.1	Seen packets	115
5.5.3	Calculating $L_l^*(k)$	116
5.5.4	Calculating $L_l(k)$	117
5.5.4.1	Calculating U_l	118
5.5.4.2	Calculating $T_l(k)$	118
5.5.5	Comparison with simulation	119
5.6	Knowledge differentials	120
5.6.1	Knowledge differences	120
5.6.2	Probability of a coding	121
5.6.3	Knowledge differentials	122
5.6.4	Simulation results	123
5.7	Delivery Rates	124
5.7.1	Buffer density	124
5.7.2	Delivery rate	125
5.7.3	Rate calculation algorithm	126
5.7.4	Simulation results	126
5.8	Fairness	129
5.8.1	The rate scaling property	129
5.8.2	Fairness algorithm	130
5.8.2.1	Notation	131
5.8.2.2	Calculating the mode vector β	131
5.8.3	Simulation results	132
5.9	Conclusion	134
6	Video on demand using intermediate packet delivery	137
6.1	Introduction	137
6.1.1	Contributions and related work	138
6.2	System Model	139
6.2.1	Definitions	139
6.2.2	Transmission scheme	140
6.2.3	Buffer model	142
6.2.4	Organisation of the chapter	142
6.3	Knowledge differential transmissions	143
6.3.1	Approximations	143
6.3.2	Leader transmissions	144

6.3.3	Probability that a packet is seen	145
6.3.4	Codings	146
6.3.5	Probability of a knowledge differential transmission	147
6.4	Delivery Rate Analysis	148
6.4.1	Delivery phase analysis	148
6.5	Undelivered packets	151
6.5.1	Expected number of undelivered packets	151
6.5.2	Startup phase analysis	153
6.6	Future directions	154
6.6.1	Use of redundant transmissions	156
6.6.2	Hybridisation with multiple server channel algorithms	156
6.7	Conclusion	157
7	Conclusion	159
7.1	Summary of results	159
7.1.1	Throughput delay analysis	159
7.1.2	Dynamic rate control	160
7.1.3	Intermediate packet delivery	160
7.2	Future work	161

List of Figures

1.1	Single hop broadcast erasure network.	2
1.2	An example illustration of the permissible network coded packet combinations under a block-based coding strategy. Network coded transmissions may only be constructed from the grey shaded packets.	4
1.3	An example illustration of the permissible network coded packet combinations under a window-based coding strategy. Network coded transmissions may only be constructed from the grey shaded packets.	5
2.1	The butterfly network using (a) traditional routing, and (b) network coding.	14
2.2	Single hop broadcast erasure network.	17
2.3	The transmission scheme model, which illustrates the components of a network coded transmission scheme.	20
3.1	A block diagram of the components of a transmission scheme.	36
3.2	Under coding schemes A and B, the coding scheme can further be divided into effective transmission queue and coefficient selection components.	38
3.3	A Markov chain describing transitions in the Markov state of a receiver r	40
3.4	The probability from (3.9) of a receiver taking $\leq T$ time steps to return to the zero state under the Bernoulli rate control scheme.	43
3.5	The zero state delivery delay under the Bernoulli rate control scheme, as a function of the addition rate λ . The delay estimates of (3.12) (dotted lines) are compared against simulation (solid lines).	45
3.6	Two-receiver state transition probabilities (a) in practice, and (b) under the independent receiver model. The horizontal and vertical axes correspond to the Markov states of receivers 1 and 2 respectively, where $s_1(t) = i$ and $s_2(t) = j$. Since the sender is common to both receivers, it is not possible that $s_1(t+1) = i \pm 1$ while $s_2(t+1) = j \mp 1$	47

3.7	Under the Bernoulli rate control scheme with addition rate λ , the proportion of time the leader is in each state, under the independent receiver model (3.14) and in practice. Addition rate $\lambda = 0.7$, channel rate $\mu = 0.8$	47
3.8	Simulated zero state and leader state delivery delays under the Bernoulli rate control scheme with addition rate λ , for different numbers of receivers R . The channel rate $\mu = 0.8$	49
3.9	Simulated probability of coefficient-based delivery as a function of Markov state, for Bernoulli transmission schemes with addition rate $\lambda = 0.7$. Probabilities are normalised over coefficient-based deliverable timeslots. The field size $M = R$ (the number of receivers), and channel rate $\mu = 0.8$	54
3.10	The average delivery delay of coding schemes A, B and RLNC-I under the Bernoulli rate control scheme with addition rate λ . Zero and leader state delivery delays are included for comparison. $R = 4$ receivers, channel rate $\mu = 0.8$	55
3.11	For the Bernoulli rate control scheme with coding scheme B, the probability that N packets are coded into a sender transmission, where $N = 1$ corresponds to an uncoded transmission. $R = 4$ and 8 receivers, addition rate $\lambda = 0.7$, and channel rate $\mu = 0.8$. Note that, if the transmission queue is empty, no packets will be coded.	56
4.1	A block diagram of the components of a transmission scheme.	63
4.2	A Markov chain describing transitions in the Markov state of a receiver r . $p = (1 - \mu)\lambda_{\text{est}}$, and $q = \mu(1 - \lambda_{\text{est}})$, where λ_{est} is the estimated addition rate, and μ is the channel rate.	66
4.3	The sender's estimated addition rate λ_{est} as a function of time, for the weighting factors f listed in the figure. The simulation was implemented with $R = 4$ and $\mu = 0.8$	71
4.4	Throughput-delay performance for the coding scheme B transmission schemes. From left to right, the parameters for each transmission scheme are: Bernoulli: $\lambda = 0.5, \dots, 0.75$. Delay threshold: $T_D = 2, \dots, 100$. Dynamic: $f = 2, \dots, 500$. For all transmission schemes $R = 4$ or 8, as indicated in the legend, and $\mu = 0.8$	73

4.5	Throughput delay tradeoff for the block coding, sliding window and queue threshold schemes. Feedback is assumed to be nonlossy and without delay. For all rate control schemes, $R = 4$, $\mu = 0.8$. Block coding: $B = 1, \dots, 50$. Sliding window: $W = 10$, $V = 0.6, \dots, 0.76$. Queue threshold: $T_{\min} = 1$, $H = 1, \dots, 50$	79
4.6	The notification frequency of the block coding, sliding window and queue threshold schemes. For all rate control schemes, $R = 4$, $\mu = 0.8$. Block coding: $B = 1, \dots, 50$. Sliding window: $W = 10$, $V = 0.6, \dots, 0.78$. Queue threshold: $H = 1, \dots, 50$; values of $T_{\min}$ are given in the figure.	80
4.7	Throughput delay tradeoff for the queue threshold scheme, for the values of $T_{\min}$ listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, $H = 1, \dots, 50$	82
4.8	Throughput delay tradeoff for different values of H compared with the upper bound of (4.22), for channel rates $\mu = 0.4, \dots, 0.9$. $R = 4$ receivers, $T_{\min} = 1$	84
4.9	Throughput delay tradeoff for different values of H compared with the upper bound of (4.22), for channel rates $\mu = 0.4, \dots, 0.9$. $R = 4$ receivers, $T_{\min} = 20$	85
4.10	Throughput delay tradeoff under the queue threshold and block-based coding schemes, for the notification delays listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, the minimum time between notifications $T_{\min} = 1$, threshold value $H = 1, \dots, 50$	86
4.11	Throughput delay tradeoff for the values of μ_f listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, the minimum time between notifications $T_{\min} = 10$, $H = 1, \dots, 50$	87
4.12	Throughput delay tradeoff for the values of μ_f listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, the minimum time between notifications $T_{\min} = 1$, $H = 1, \dots, 50$	88
5.1	A block diagram of the components of the transmission scheme.	98
5.2	Our model of receiver Rx_i 's buffer. Dark coloured packets have been all received and delivered; some fraction of the shaded packets have been received; white packets have not been received. Packets are ordered left to right by ascending packet index.	103

5.3	Subregions within the packet density region of receiver Rx_4 's buffer. Darker colouring corresponds to a higher packet density; white indicates a packet has not been received. Packets are ordered left to right by ascending index.	106
5.4	Dependency graph for quantities calculated in Sections 5.5 to 5.7. See Table 5.4 for a list of definitions. Labeled arrows indicate which values are required to calculate the next block, while unlabeled arrows indicate the following block depends directly on the value from the previous block. Curved arrows indicate dependencies on previously calculated values.	109
5.5	Probability of k leader transmissions of a packet, for $R = 4$ receivers, channel rates $\mu = [0.8, 0.6, 0.4, 0.2]$ and mode vector $\beta = [0.85, 0.05, 0.05, 0.05]$.	120
5.6	Probability K_r^m of a knowledge differential for a receiver Rx_r in mode m , for $R = 4$ receivers, channel rates $\mu = [0.8, 0.6, 0.4, 0.2]$ and mode vector $\beta = [0.85, 0.05, 0.05, 0.05]$	124
5.7	Probability of a knowledge differential for a receiver Rx_r in mode m , for the simulation settings given in Table 5.6.	128
5.8	Calculated delivery rates achieved after each iteration r of Algorithm 11, under Setting A from Table 5.7. These are compared against μ , the channel rate for each receiver.	133
5.9	Calculated and simulated delivery rates under the settings of Table 5.7, using the values of mode vector β derived in Algorithm 11.	134
6.1	A block diagram of the components of the transmission scheme. . . .	141
6.2	Example illustration of the contents of a receiver's buffer. Each packet $p_1, p_2, \dots$ is allocated a position in the buffer, from left to right. Dark coloured packets have been delivered, and grey packets have been seen but not delivered. White packets have not yet been seen by the receiver. ¹	142
6.3	The contents of a non-leading receiver's buffer, after it joins the network. Some fraction of the shaded rectangles correspond to packets which have been seen. Initially the buffer is empty, but as the transmission progresses, Rx_r is able to both deliver new packets from the start of the buffer (dark packets), and see some of the leader Rx_l 's next needed packets (shaded packets).	149

-
- 6.4 Simulated number of packets delivered at each receiver, as a function of time. $R = 4$ receivers, channel rate $\mu = 0.8$, receivers join at times $t_1 = 0, t_2 = 80, t_3 = 160, t_4 = 240$. Vertical dotted lines indicate the starting times of new receivers, and number of packets delivered to the leader Rx_l at these times. Double arrows indicate the duration of the startup phase for receivers. 150
- 6.5 Packets delivered at the receivers as a function of time. $R = 4, \mu = 0.8, t_1 = 0, t_2 = 10, t_3 = 20, t_4 = 30$. We can observe that fluctuations in the receivers' delivery rates can cause changes in the receivers' ordering. 152
- 6.6 Number of undelivered packets as a function of Markov state. $R = 4, \mu = 0.8, t_1 = 0, t_2 = 10, t_3 = 20, t_4 = 30$. Averaged over 5 simulations of 5000 time slots. This is compared against the prediction made in (6.12). 153
- 6.7 Delivery time for receivers, averaged over 10 simulations. $R = 4, \mu = 0.8, t_1 = 0$, and the difference between receiver starting times $t_{i+1} - t_i$ is given by the intervals listed in figure. This is compared against the lower bound given in (6.15). 155

List of Tables

2.1	An example of coding scheme A transmission. Ticks and crosses represent receivers' channel states (successful and unsuccessful reception, respectively) at each transmission.	24
2.2	An example of coding scheme B transmission. Ticks and crosses represent receivers' channel states (successful and unsuccessful reception, respectively) at each transmission.	24
3.1	The probability of transitions between Markov states for $s_r(t) > 0$	40
5.1	An example of the transmission scheme for two receivers, showing which packets have been decoded by each receiver. A ✓ means the transmitted packet $s(t)$ was received; a ✕ indicates a packet erasure. . . .	94
5.2	An example of mode 1 transmission, and the effect on receivers' buffers. Ticks and crosses represent receivers' channel states (successful and unsuccessful reception, respectively) at each transmission.	100
5.3	Percentage of undecoded packets in each receiver's buffer for $R = 4, 8$, and 8, under the indicated values of μ and β	107
5.4	Definition of the values used in Fig. 5.4.	110
5.5	A schematic of the three-receiver probability that the sender will transmit k mode l leader transmissions of each packet $\mathbf{p}_n$	115
5.6	Simulation conditions used in Fig. 5.7, and the corresponding min-cut values.	129
5.7	The settings used in fairness algorithm simulations, and their corresponding min-cuts.	133

Introduction

Network coding is a technique in which a number of different information flows, usually in the form of *packets*, are mixed, or *coded*, together for transmission [2, 42]. Network coding can offer improved performance under a number of measures. Throughput performance can be improved by reducing the number of redundant transmissions [2, 40, 45, 53, 55], while higher energy efficiency can be achieved by reducing the number of transmissions required to broadcast a unit of information [21, 79, 81]. Randomisation can be used to provide information theoretically secure communications [7], while redundancy can be used to allow resilient communications in the presence of malicious interference [37].

The improvements offered by network coding can potentially be used in a large number of applications. It can reduce the download completion time for peer-to-peer file transmissions [29], and improve the quality of a video stream [48, 78, 85]. It can overcome transmission bottlenecks [2, 45], and facilitate more efficient node repairs in a distributed data storage system [15, 16]. Much higher throughputs may even be achieved over wireless communications [40, 53, 55] and TCP connections [12, 41, 70].

Taking advantage of network coding capabilities is an often complex problem, and the design of network coding strategies for different applications is an area of active research [12, 16, 32, 48, 55, 70, 82]. Designs must address multiple performance criteria, which vary depending on the application. The literature mainly focuses on two questions:

1. How can we design network coding strategies for improved performance?
2. How can we determine the performance of a network coding strategy?

The typical approach is to design a network coding strategy, then use analysis to determine its performance [46, 51, 84]. However, the two questions are potentially interrelated, since understanding the factors that influence the performance of a network coding strategy can give us clues on how to improve our design. In this thesis we investigate both of these questions, but we take the reverse approach, analysing

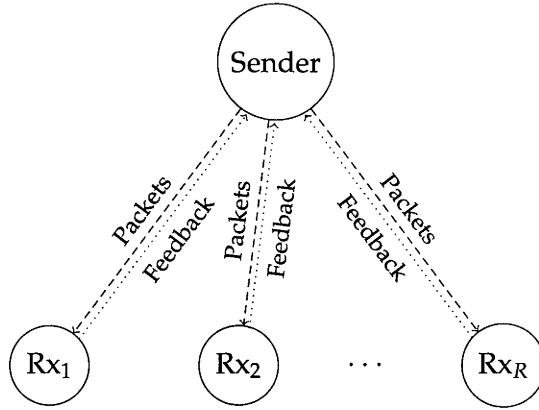


Figure 1.1: Single hop broadcast erasure network.

the performance of existing network coding strategies, then using the observations gained from our analysis to design new and improved network coding strategies.

The system model we will consider is a single hop broadcast erasure network, depicted in Fig. 1.1. This consists of a single sender and a set of receivers, connected by a *broadcast erasure channel* where every receiver is capable of overhearing every transmission made by the sender. We consider the broadcast problem, where the sender is given the task of transmitting the same set of information to all receivers. The sender broadcasts packets to the receivers, who in return transmit *feedback*, which the sender can use to determine which packets they have received. It should be noted that even this simple system model poses challenges in network coding design and analysis, because different packets may be missing across receivers. Nonetheless, it is anticipated that insights gained from our analysis of this network model will potentially be useful in the study of more complex networks.

Although there are many potential choices of performance measures, in this thesis we will primarily focus on *throughput*, *delay* and *fairness*, which are crucial for a number of applications such as video broadcast, online gaming and wireless communications. The meaning of these performance measures, as well as the open questions relating to their measurement and the provision of good performance, will be discussed in the following sections. We focus on applications which have an *in-order delivery requirement*, namely where packets must be recovered in the correct order and without loss. This is relevant in applications such as video broadcast and multi-player gaming, where packet ordering is critical and high reliability transmission is required.

1.1 Throughput and delay

The *throughput* of a system measures the rate at which new packets are transmitted by the sender, while the *delay* measures how long receivers must wait before they can recover the transmitted packets. There is often a complicated and nontrivial relationship between throughput and delay performance [4, 19, 46, 55]. It is well known that network coding can be used to achieve improved [27, 46, 55] or even optimal [2, 45] throughput in a wide variety of networks, however network coding also comes at the cost of increased delay, since packets cannot be used until they have been recovered in their original form. This typically involves waiting until enough packets have been received to decode the original content.

The mathematical relationship between throughput and delay is often difficult to characterise, and highly dependent on the network coding strategy being used. In particular, the effects of feedback often cause difficulties in analysis, due to the potential for a network coding strategy to have recursive dependencies on its own past performance. As a result, more rigorous throughput delay performance estimates only exist for network coding strategies that are affected by feedback in relatively predictable ways. For example, accurate throughput delay analysis exists for random linear network coding [33, 34] performed on a block of packets, where feedback is only transmitted after the completion of a block [50, 51, 75]. On the other hand, fewer results exist for network coding strategies with more complicated feedback dependencies. For example, many instantly decodable network coding (IDNC) algorithms, where transmissions are designed to be immediately decodable by all or a subset of receivers, exist in the literature [47, 59, 60, 65, 66], but finding analytical, closed formed results on their throughput and delay performance remains a challenge. In this thesis, we will take on the challenge of characterising more complex feedback dependencies, making some inroads into the analysis of throughput delay performance under selected network coding strategies.

Generally speaking, a network coding strategy may be divided into two key parts, the *rate control* and *coding* components, each with its own effect on the throughput and delay performance. We now discuss how rate control and coding strategies affect the throughput delay performance, as well as some of the open questions related to their study.

1.1.1 Rate control

The rate control component is responsible for regulating which subset of packets may be coded into a transmission. This is illustrated for block-based and window-based strategies in Figs. 1.2 and 1.3.

Block-based strategies, such as [20, 48, 78, 83]¹, only permit network coding within a fixed set of N packets corresponding to a block, as shown in Fig. 1.2. As the block size increases, so does the throughput [19], but decoding takes longer, since receivers must generally wait for the entire block to be received before they can decode. This waiting time can be detrimental to the throughput under larger numbers of receivers, since *all* receivers must receive the block before transmission of a new block can begin.

Window-based strategies [4, 71, 74, 80], on the other hand, allow packets to be coded from within a moving window, as shown in Fig. 1.3. This allows more flexible decoding times, which can result in better throughputs compared with block coding. However the same tradeoff between throughput and delay is observed. The faster the window advances, the greater the throughput (conditional on the packets eventually being decoded by the receivers), but this comes at the expense of fewer opportunities to decode.

While throughput and delay analysis exists for some block-based rate control strategies [19, 75, 77], to date there has been relatively little work [80] studying window-based network coded strategies. Although there has been some study of the throughput delay performance of specific coding strategies from both categories, for example [75, 80, 84], it is not generally understood how rate control can be used to achieve good throughput delay performance. Understanding the impact of rate control on the throughput and delay performance is relevant to the design of applications such as network coded TCP [70].

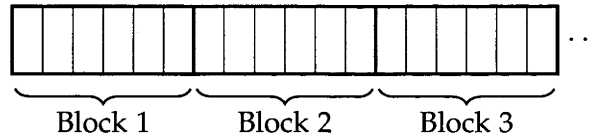


Figure 1.2: An example illustration of the permissible network coded packet combinations under a block-based coding strategy. Network coded transmissions may only be constructed from the grey shaded packets.

1.1.2 Coding

The coding component determines the network coded transmission that should be sent, within the limitations set by the rate control strategy. Strategies may focus on improving delay performance, throughput performance, or provide some compromise between the two [4, 46, 82]. We illustrate how the coding component can

¹Fountain codes [49, 63] might also be included in this category.

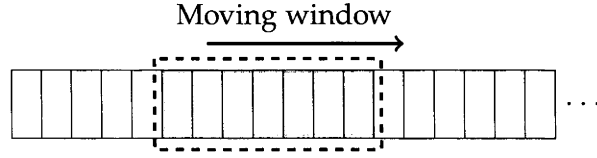


Figure 1.3: An example illustration of the permissible network coded packet combinations under a window-based coding strategy. Network coded transmissions may only be constructed from the grey shaded packets.

influence both the delay and throughput performance with the following examples.

Instantly decodable network coding (IDNC) strategies are designed with the intention to provide good decoding delay performance [47, 59, 60, 65, 66]. For instance transmissions can be chosen to maximise the number of receivers who can immediately decode a new packet. However this comes at the cost of reduced throughput since, due to the IDNC constraint, the transmission selected is not guaranteed to offer new information to all receivers.

Throughput optimal coding strategies, on the other hand, are designed to maximise the throughput performance by choosing innovative transmissions for all receivers, wherever possible. The most well known example of throughput optimal coding is *random linear network coding* (RLNC), where the transmission is a random combination of all packets in the block or coding window chosen by the rate control component. Surprisingly, this strategy is sufficient to achieve throughput optimality with high probability [34], but it does not offer any improvements to the delay performance. Other *deterministic* throughput optimal strategies, such as [18, 71, 74], choose transmissions according to feedback information. Unlike RLNC, these deterministic strategies can potentially offer some improvements to the delay performance, however, the complexity of their feedback dependencies means that there has been very little analysis of their delay performance to date. Two- and three-receiver cases have been studied in [4, 47, 72], but the extra complexity associated with larger numbers of receivers has meant that more general analysis has not been attempted in the literature. As a result, the potential delay improvements offered by throughput optimal coding strategies have been largely ignored in the literature [73, 77]. In this thesis we will demonstrate that, perhaps unexpectedly, the proper choice of coding strategy can potentially contribute a significant improvement to the delay performance, and investigate the features of coding strategies that allow this result.

1.2 Fairness

Fairness, the third performance measure we will consider in this thesis, is a measure of how effectively the available resources have been allocated to provide equitable performance across receivers. In situations where resources must be shared, fairness is important to ensure that all receivers experience reasonable performance. The provision of fair performance under various measures of fairness has been studied in work such as [31, 38, 43, 44].

An interesting question is, if the receivers experience non-uniform packet reception, how can resources be allocated so that their performance reflects their situation? For example, if receivers have different channel erasure probabilities, it would make sense for them to experience throughputs that are proportional to their *channel rates*, the rate at which packets are successfully received.

Implementing broadcast fairness to a non-uniform set of receivers is a nontrivial task. For example, under the system model of our thesis (recall Fig. 1.1), capacity results dictated by the min-cut max flow theorem indicate that the achievable broadcast rate region is bounded by the worst performing receiver [14, 52]. This suggests that achieving throughput fairness in any meaningful way is essentially impossible, since the throughput is always limited by the channel rate of the slowest receiver. Generally speaking, for the purpose of in-order packet delivery on a broadcast network, current strategies can tolerate very little non-uniformity in the amount of information stored across receivers. As a result, broadcast fairness has not yet been achieved for in-order packet delivery, under non-uniform receiver conditions.

The limitations imposed on fairness by existing strategies can be explained as follows. Capacity results in the literature (as well as most network coding strategies) generally rely on a property we call the *complete information property*:

If n packets have been included in the sender's transmissions so far, a receiver is guaranteed to decode all of these packets once it has received n innovative transmissions.

The prevailing assumption is that the complete information property is necessary to ensure that all packets are decoded. Under a single hop network, this generally means restricting all receivers to the same set of n packets to ensure decoding. This is the approach taken by the rate control strategies we discussed in Section 1.1.1.

However, the complete information property is not always necessary, since *intermediate decoding* is also possible, where the precise combination of packets known by a receiver allows certain packets to be decoded prior to the complete information property being satisfied (in other words, before all of the sender's packets have been decoded) [5, 54, 61, 71, 74]. It is the contribution of intermediate decoding that allows

the coding strategies of Section 1.1.2 to offer improvements to the delay performance. Therefore, a compelling question that might be asked is: do network coding strategies exist such that intermediate decoding is sufficient to ensure continued decoding? In this thesis, we will show for the first time that this is possible, and demonstrate how such a network coding strategy can be exploited to achieve performance fairness under non-uniform receiver conditions.

1.3 Thesis contributions and outline

In this thesis, we analyse the performance of existing networking strategies, using the results to design new network coding strategies for practical applications. Focusing on the performance measures of throughput, delay and fairness, we provide insight into the factors that influence the performance of a network coded system, and develop novel transmission strategies that improve on the performance of existing strategies in the literature. We outline the structure and contributions of this thesis.

1.3.1 Throughput delay analysis of coding strategies

In Chapter 3, we provide in-depth analysis of the throughput and delay performance of selected network coding strategies. We quantify the performance of an existing rate control scheme [71, 74], and identify factors that affect the throughput and delay.

Although the complexity introduced through the feedback component of these coding strategies means that it is not always possible to obtain exact results, we obtain a number of key simplifications and observations that will inform the remainder of the thesis. Among them, we find that, surprisingly, under some circumstances, the coding component has a large impact on the delay performance. The effect of the coding component on the delay has been largely ignored in the study of throughput optimal strategies in the literature, but we demonstrate that under the right circumstances, it can have a major impact on the delay performance. The results are published in [25, 26].

1.3.2 Dynamic rate control

In Chapter 4, we use the insights gained from our rate control analysis to design network coding strategies with improved throughput and delay performance, compared with existing strategies in the literature. We take the novel approach of using feedback information combined with a predictive model to dynamically weigh throughput and delay performance requirements against each other.

In Section 4.3 we introduce a dynamic rate control strategy that, under the approximations made using our throughput delay analysis, provides the best known tradeoff between throughput and delay performance. Surprisingly, our analysis reveals that this strategy reduces to simply limiting the number of undelivered packets stored across receivers. Despite its simplicity, the dynamic rate control scheme demonstrates better throughput delay performance in comparison with existing strategies in the literature. This work is published in [25].

In Section 4.4 we design a variant of the dynamic rate control strategy that performs well with relatively little feedback from the receivers. This rate control strategy utilises RLNC to create a low complexity, low feedback transmission strategy, which demonstrates good throughput delay performance – even under lossy and delayed feedback. This transmission strategy is potentially useful for TCP-compatible network coding, as it maintains the reliability and in-order delivery requirements of TCP, and performs reasonably well under the imperfect feedback conditions that are observed in practical situations. The results are published in [23].

1.3.3 Intermediate packet delivery

In Chapter 5 we demonstrate for the first time that *intermediate packet delivery* is possible, where continued in-order packet delivery is achieved using only intermediate decoding, and without the aid of the complete information property. We show that, by relying on intermediate packet delivery, the transmission scheme we outline is able to exceed the capacity limitations set by the min-cut bound. An exciting result of our work is that this transmission scheme can use a single sender to simultaneously support multi-rate packet delivery to different receivers.

We use the increased flexibility afforded by intermediate packet delivery to address the problem of throughput fairness. This is a task that has not been successfully attempted in the literature. We analyse the throughput performance of the transmission scheme under non-uniform receiver channel rates, and use the results to introduce a fairness algorithm that achieves throughput fairness for the receivers, relative to their channel rates. By manipulating how the sender's transmissions are coded (using what we refer to as transmission modes), the fairness algorithm can achieve approximately fair throughputs for all receivers. This work has been submitted in [24].

1.3.4 Video on demand using intermediate packet delivery

In Chapter 6, we show that the intermediate packet delivery can also be useful if receivers experience the same channel rate. We outline a *transmission optimal* network

coding scheme, adapted from [71], where every transmission is guaranteed to bring new information to every receiver in the network. This contrasts with existing transmission strategies such as [4, 71, 74] that rely on redundant transmissions to ensure decoding will take place. Curiously, using intermediate packet delivery allows all receivers to deliver packets at their channel rate, but in an often asynchronous manner. Even when receivers join the broadcast at different times, after a sufficient amount of time has passed, receivers are able to deliver all packets at their channel rate.

A potential application of this work is in video on demand broadcast applications, where any number of receivers may request to view a video at any time. One of the challenges associated with video on demand is minimising the number of *server channels* required to deliver the video to the receivers. To date, proposed video on demand methods require at least two, but often more, server channels to provide video on demand services. Under our network coding transmission strategy, multiple receivers can be served using only a single server channel. The results are published in [22].

Background

In this chapter, we provide a technical background for the work done in this thesis. An overview of the relevant details of network coding is given in Section 2.1, and the main performance measures we will consider are introduced in Section 2.2. Section 2.3 contains a detailed system model, while Section 2.4 provides a breakdown of the network coding system. Section 2.5 summarises the network coding components that will be studied, and for quick reference, Section 2.6 contains a glossary of common terminology that will be used throughout this thesis.

2.1 Network coding

Linear network coding [42, 45] (which we will refer to as *network coding* for short) was first introduced in [2]. Unlike traditional packet routing, where one original data packet is transmitted at a time across a link, network coding allows the information from multiple packets to be coded into a single transmission. This *network coded* packet contains one packet's worth of information, but provides coded information from a number of packets.

In practice, the information content of each packet is broken down into a vector of field elements from the Galois field $\mathbb{F}_M$, so that a given packet $\mathbf{p}_n$ can be expressed as

$$\mathbf{p}_n = [f_{n,1}, f_{n,2}, f_{n,3}, \dots, f_{n,L}], \quad (2.1)$$

where L is the length of the packet vector.¹

Network coding is performed by combining multiple packets using Galois field arithmetic [42]. A network coded transmission $s(t)$ is created by summing different

¹We assume in this thesis that packets are all of the same length L .

multiples of the packets' information together, so that

$$s(t) = \sum_{n=1}^{A(t)} \alpha_n \mathbf{p}_n, \quad (2.2)$$

where $\alpha_n \in \mathbb{F}_M$ is a *coding coefficient* chosen for packet $\mathbf{p}_n$, and $A(t)$ is the number of packets available to be coded into the transmission at time t . We can observe that identical operations are applied on each element of this vector, so for simplicity of explanation, in this thesis we treat each packet $\mathbf{p}_n$ as a single field element in $\mathbb{F}_M$. The algorithms we describe can be implemented in practice by applying the same encoding and decoding operations on each field element in the packet.

Along with the network coded information itself, it is also necessary for a network coded packet to include a *coefficient vector* v_s , the vector of coding coefficients used to create the network coded information,

$$v_s = [\alpha_1, \alpha_2, \dots, \alpha_{A(t)}]. \quad (2.3)$$

Each receiver stores the network coded packets it receives in a buffer, which we assume is sufficiently large to store all packets that it receives.

Each receiver must then decode the information from the network coded packets stored in its buffer. This is achieved by applying Gaussian elimination [42] to the set of stored coefficient vectors, treating each coefficient vector as a row in the matrix. If the same operations are applied to the corresponding network coded information, then once a coefficient vector has been reduced to a basis vector, the corresponding packet information has been decoded.

Depending on what information a receiver already has stored in its buffer, a network coded transmission may or may not provide new information to the receiver.

DEFINITION 2.1: A network coded packet is *innovative* to a receiver Rx_r if its coefficient vector is linearly independent of the set of coefficient vectors corresponding to the packets stored in the receiver's buffer. Otherwise, the packet is said to be *non-innovative*.

Only innovative packets can provide new information to the receiver. Applying Gaussian elimination to a non-innovative packet reduces the coefficient vector to the zero vector, at which point the receiver can discard the packet.

2.1.1 Butterfly network example

The throughput benefits of network coding can be demonstrated in the canonical *butterfly network* example from [2], shown in Fig. 2.1. It consists of two senders S_1

and S_2 , holding packets $\mathbf{p}_1$ and $\mathbf{p}_2$ respectively, intermediate nodes A and B, and two receivers Rx_1 and Rx_2 . Links are lossless and delay free, but only capable of transmitting one packet at a time. In this simple example, S_1 and S_2 simultaneously attempt to transmit packets $\mathbf{p}_1$ and $\mathbf{p}_2$ to Rx_1 and Rx_2 .

In Fig. 2.1 (a), the best possible routing solution is illustrated. S_1 and S_2 transmit their respective packets down their connecting links. Intermediate node A selects one of these packets to forward to intermediate node B, in this case $\mathbf{p}_1$, which is then passed on to both receivers Rx_1 and Rx_2 . This results in the reception of $\mathbf{p}_1$ by Rx_1 , and $\mathbf{p}_1$ and $\mathbf{p}_2$ by Rx_2 . On average, 1.5 packets can be received by each receiver at a time.

In contrast, Fig. 2.1 (b) illustrates the best solution using network coding. Here the field $\mathbb{F}_2$ is used for coding, which is equivalent to the XOR operation. Again S_1 and S_2 forward $\mathbf{p}_1$ and $\mathbf{p}_2$ down their connecting links, but instead of choosing a single packet to forward, intermediate node A transmits the XOR combination $\mathbf{p}_1 \oplus \mathbf{p}_2$, to intermediate node B. Receiver Rx_1 obtains $\mathbf{p}_1$ directly from S_1 , and is able to decode $\mathbf{p}_2$ by XORing $\mathbf{p}_1$ with $\mathbf{p}_1 \oplus \mathbf{p}_2$. Similarly, Rx_2 obtains $\mathbf{p}_2$ from S_2 , and is able to decode $\mathbf{p}_1$ from $\mathbf{p}_1 \oplus \mathbf{p}_2$. Therefore, by using network coding, intermediate node A is able to transmit a packet that allows *both* receivers to obtain $\mathbf{p}_1$ and $\mathbf{p}_2$, allowing two packets to be received by each receiver at a time. This is a 33% improvement over packet routing.

2.2 Performance measures

As we have just demonstrated in Section 2.1.1, network coding can offer improvements to the performance of a system. However, in practice, there are also performance penalties associated with its use. We now provide an overview of the key network coding performance measures we will consider in this thesis: *delivery delay*, *throughput*, *delivery rate* and *throughput delay performance*. Technical definitions and discussion of these terms are provided.

2.2.1 Decoding and delivery delay

We begin with a technical definition of what it means for a packet to be decoded.

DEFINITION 2.2: A packet $\mathbf{p}_n$ is *decoded* by a receiver if $\mathbf{p}_n$ can be recovered from all received transmissions $s(t)$ using Gaussian elimination.

Example: In Fig. 2.1 (b), upon receiving $\mathbf{p}_1$ and $\mathbf{p}_1 \oplus \mathbf{p}_2$, receiver Rx_1 is able to use Gaussian elimination to decode both packets $\mathbf{p}_1$ and $\mathbf{p}_2$.

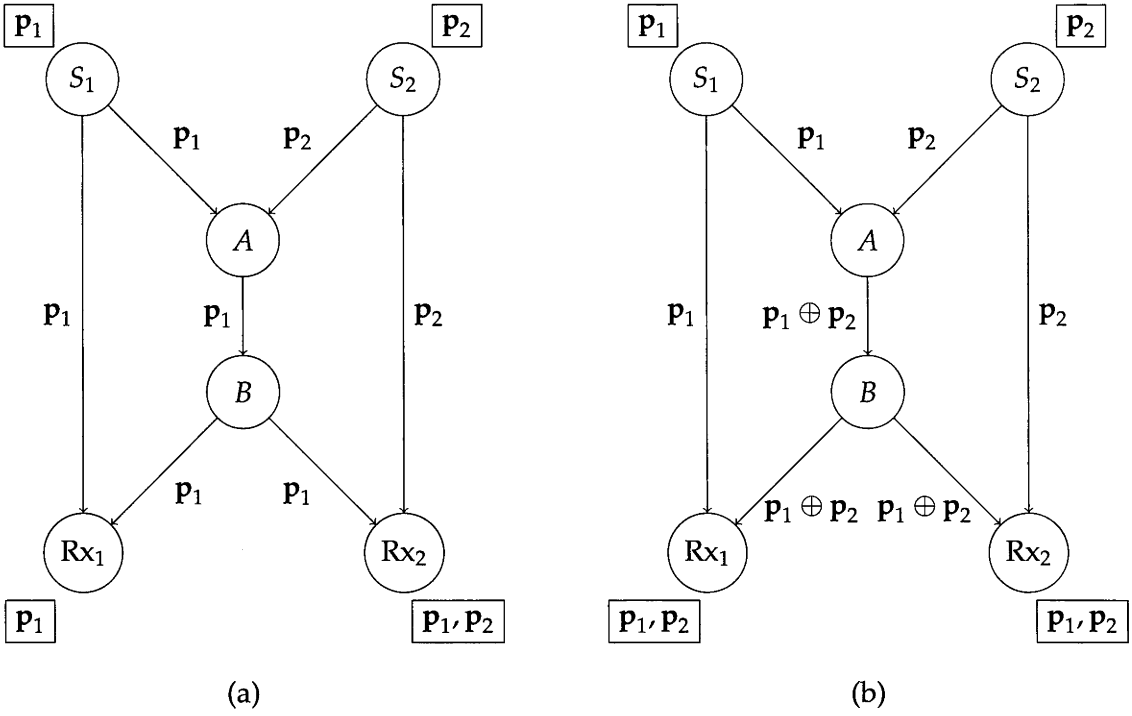


Figure 2.1: The butterfly network using (a) traditional routing, and (b) network coding.

Recall from Section 1.1 that, while network coding can improve the throughput of a system, it also typically comes at the expense of greater delays, since receivers must wait until information is decoded before they can use it. In this thesis, we assume that sender transmissions are both transmitted and received within discrete time slots, so we can measure this decoding delay as follows.

DEFINITION 2.3: The *decoding delay* of a packet at a receiver Rx_i is defined as the number of time slots between the first transmission of a packet by the sender, and its successful recovery at that receiver. A receiver's decoding delay is defined as the average decoding delay across the packets it has received.

Example: If a packet p_3 was first transmitted at time $t = 5$, and decoded by Rx_i at $t = 8$, then the decoding delay at Rx_i is $8 - 5 = 3$ time slots.

Under network coding it is possible for packets to be decoded at different times, in some cases even altering the ordering of the original packets. In this thesis, we study a more stringent form of delay known as the *delivery delay*, which is measured for applications where packets may only be used once they are decoded in the correct order $p_1, p_2, p_3, \dots$, namely from the *oldest* packet stored at the sender, p_1 , to the *newest*.

DEFINITION 2.4: A packet p_n is *delivered* to a receiver Rx_i once it and all older packets $p_1, \dots, p_n$ have been decoded by Rx_i .²

Example: If packets p_1 , p_2 , and p_3 are decoded by a receiver Rx_i at times $t = 2$, $t = 10$, and $t = 8$ respectively, then p_3 is delivered to Rx_i at time $t = 10$.

DEFINITION 2.5: The *delivery delay* of a system is the average time between the first transmission of a packet, and the first time that it has been delivered by each of the receivers.

Example: If a system consists of two receivers Rx_1 and Rx_2 who experience delivery delays of 1, 5, 8 and 2, 10, 10 time slots for packets p_1 to p_3 , respectively, then the delivery delay of the system is $\frac{1+5+8+2+10+10}{2 \times 3} = 6$ time slots.

Note that the delivery delay of a system must be greater or equal to the decoding delay, since packets cannot be delivered until they are decoded.

2.2.2 Throughput and delivery rate

The *throughput* of a system measures the rate at which packets of information can be communicated from the sender to the receivers. This is an important performance measure in Chapters 3 and 4 of this thesis.

DEFINITION 2.6: The *throughput* of a system is the average rate at which packets are delivered across all receivers.

Example: If two receivers Rx_1 and Rx_2 have delivered 8 and 6 packets respectively over the last 10 time slots, then the throughput is $\frac{8+6}{2 \times 10} = 0.7$ packets per time slot.

A related measure is the *delivery rate*, which forms the primary performance measure in Chapters 5 and 6.

DEFINITION 2.7: The *delivery rate* of a receiver Rx_i is the average rate at which packets are delivered to that particular receiver.

Example: If two receivers Rx_1 and Rx_2 have delivered 8 and 6 packets respectively over the last 10 time slots, then the delivery rate of Rx_1 is $\frac{8}{10} = 0.8$ packets per time slot, and the delivery rate of Rx_2 is $\frac{6}{10} = 0.6$ packets per time slot.

Note that while the throughput is the average number of packets delivered across all receivers, the delivery rate is specific to a particular receiver. We make this distinction because as we will see in Chapters 5 and 6, we must allow for the possibility that different receivers have distinct delivery rates.

²Note that when we say that a packet is delivered to a receiver, we mean that it is delivered to the application at that receiver.

2.2.3 Throughput delay performance

Although throughput and delivery delay are independently useful performance measures, recall that network coding results in a tradeoff [4, 19, 46, 55, 80] between throughput and delay performance. Therefore, it is useful to measure the jointly achievable performance under the two measures.

DEFINITION 2.8: The *throughput delay performance* is a measure of throughput values and delivery delays simultaneously achievable by a system. Under the same network conditions, higher throughputs for a given delivery delay corresponds to better throughput delay performance.

Example: If, under the same system model, two transmission strategies X and Y both experience an average delivery delay of 2, and achieve throughputs of 0.7 and 0.75 respectively, then transmission strategy Y has better throughput delay performance.

Generally higher delivery delays correspond to higher achievable throughputs, while lower delivery delays come at the cost of reduced throughput.

2.3 System model

We now outline the system model that we will consider in this thesis. Our network consists of a single sender, connected to R receivers $Rx_1, Rx_2, \dots, Rx_R$ by a wireless broadcast channel, as was illustrated in Fig. 2.2. The sender divides the information to be sent into equal sized *packets*, which we represent by $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \dots$. The sender aims to deliver these packets in the correct order, from oldest to newest, to all receivers.

Time is slotted, and the sender broadcasts packets to the receivers at a rate of one packet per time slot. Each receiver Rx_i receives the packet with probability μ_i , which we refer to as the *channel rate*, independently at each time slot. The majority of this thesis studies the case where all channel rates are the same – that is, $\mu_1 = \mu_2 = \dots = \mu_R$. Where this is the case, μ will be used to denote the channel rate. If, on the other hand, the channel rates are distinct, the vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_R]$ will be used to represent the channel rates of the network. Packets successfully received through the packet erasure channel are assumed to be error-free.

The receivers are connected to the sender by individual feedback channels. Receivers use these channels to transmit notifications to the sender to provide an indication of which packets have been received – the exact details of the feedback algorithms used will be discussed in the relevant chapters. We assume that the sender is able to receive notifications without any cross-receiver interference. The majority of this thesis will consider the case of *perfect feedback*: that is, notifications are

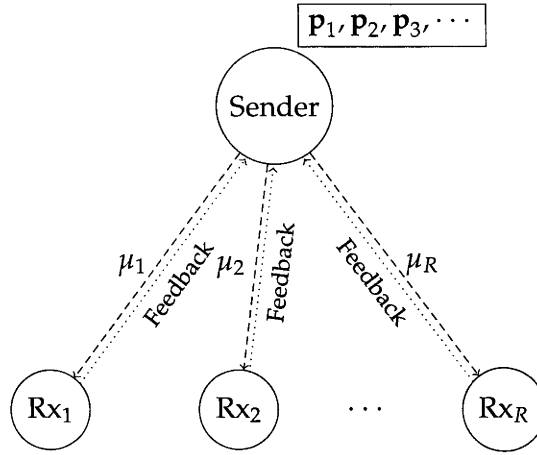


Figure 2.2: Single hop broadcast erasure network.

assumed to be received immediately after a transmission is made, with no loss or delay. However, the effects of lossy and delayed feedback are explored in Chapter 4.

We measure the performance of the network coding transmission schemes studied in this thesis using Matlab simulations. Simulations are run on the order of 10,000 time slots for each combination of parameters. This number was chosen so that the estimation error remained acceptably low (estimated to be less than 5% in most cases), while simulations did not take more than a few hours each to run.

2.4 Transmission scheme model

At each time slot t , the sender must determine what network coded packet it should transmit. Each transmission $s(t)$ is chosen according to what we call a network coding *transmission scheme*. We now provide a detailed breakdown of the procedure used by the sender to determine each network coded transmission. The *transmission scheme model*, as illustrated in Fig. 2.3, is a fairly general model that can be applied to essentially all single hop wireless broadcast network coded transmission schemes. It is used in its complete form in Chapters 3 and 4, while a reduced version is considered in Chapters 5 and 6. The transmission scheme model consists of four components: a packet backlog, rate control block, transmission queue, and coding block. Information is passed into these components in the form of feedback from the receivers. Together, these components determine $s(t)$, the network coded transmission that will be sent at each time t . We now outline the function of each component.

2.4.1 Packet backlog

The packet backlog contains all of the packets that are currently available for the sender to transmit. We assume in this thesis that the sender has an infinite backlog of packets available for transmission – in other words, the sender is permitted to code any packet it requires.

2.4.2 Rate control block

The rate control block is responsible for limiting the number of new packets that can be transmitted by the sender. It achieves this by employing a *rate control scheme* to decide when to introduce new packets from the application into the transmission queue. At each time slot, the rate control scheme determines $a(t)$, the number of packets to be *added* to the transmission queue. Depending on the transmission scheme, $a(t)$ may take any integer value greater than or equal to 0. If the rate control scheme determines that $a(t) = 0$, then it *waits* and no new packets are added.

2.4.3 Transmission queue

The transmission queue stores all packets passed in by the rate control scheme. Only packets in the transmission queue may be transmitted by the sender. At any time t , the total number of packets that have been passed into the transmission queue is

$$A(t) = \sum_{i=1}^t a(i). \quad (2.4)$$

Packets may also be dropped from the transmission queue once it is guaranteed they will not be coded again. In some transmission schemes such as [71], a packet $\mathbf{p}_n$ will only be dropped once it has been determined that all receivers have decoded it. Under other schemes such as [74], packets may be removed from the transmission queue prior to this point in time. At any time t , a total of $A'(t) \leq A(t)$ packets exist in the transmission queue. Since this thesis will focus on the delivery of packets in the transmission queue to the receivers, with a slight abuse of notation, the packets in the transmission queue will be referred to, from oldest to newest, as $\mathbf{p}_1, \dots, \mathbf{p}_{A'(t)}$.

2.4.4 Coding block

The coding block employs a *coding scheme* to determine which of the packets in the transmission queue to code into the outgoing transmission $s(t)$ at each time slot, and with what coefficients. Since the coding block may only choose packets from the

transmission queue, transmissions are of the form

$$s(t) = \sum_{i=1}^{A'(t)} \alpha_i(t) \mathbf{p}_i, \quad (2.5)$$

where the coefficients $\alpha_i(t)$ are chosen at each time slot from the field $\mathbb{F}_M$ of an appropriate size.³ This combination is transmitted along with the corresponding *coefficient vector* $v_s(t)$. Each uncoded packet $\mathbf{p}_i$ is represented by the coefficient vector equal to the standard basis vector e_i , whose i -th entry is 1. The coefficient vector is therefore given by

$$\begin{aligned} v_s(t) &= \sum_{i=1}^{A'(t)} \alpha_i(t) e_i \\ &= [\alpha_1(t), \alpha_2(t), \dots] \end{aligned} \quad (2.6)$$

so that the i -th entry of the coefficient vector $\alpha_i(t)$ corresponds to the coefficient of $\mathbf{p}_i$. Receivers use the information in the coefficient vector to recover the original packets by performing Gaussian elimination on the packets in their buffers.

2.4.5 Feedback

Network coded transmission schemes generally rely on some form of feedback from the receivers. In this thesis, a unit of feedback from a receiver is called a *notification*. The properties of the feedback required vary from scheme to scheme. Some transmission schemes require *full feedback*, where a notification is transmitted by every receiver at every time slot. Others can generally make do with *partial feedback*, where receivers only need to transmit notifications in selected timeslots, under conditions specified by the transmission scheme. In this thesis we will primarily focus on the case of lossless, delay free feedback from the receivers, as it has simpler properties for the purpose of analysis. In Chapter 4, however, we will also study the effects of lossy and delayed feedback on the throughput delay performance of the system.

2.5 Transmission schemes

We now provide a detailed summary of the transmission scheme components that will be studied. These can be divided into two categories: rate control schemes and coding schemes, which correspond to the rate control and coding blocks illustrated in

³The coding schemes of [71, 74] prove that it is always possible to find an *innovative* combination for all receivers if $M \geq R$, the number of receivers.

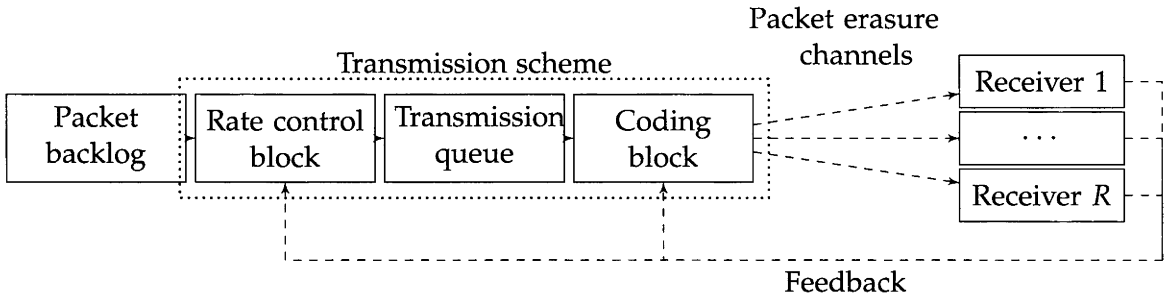


Figure 2.3: The transmission scheme model, which illustrates the components of a network coded transmission scheme.

Fig. 2.3. Generally existing analysis in the literature considers transmission schemes as a single entity, formed from the combination of a rate control and coding scheme. However, in this thesis, we treat these components separately, to better understand how the rate control and coding components individually affect the performance of a transmission scheme.

2.5.1 Rate control schemes

We describe each of the rate control schemes that we will study throughout this thesis. Recall that the rate control scheme is responsible for determining $a(t)$, the number of packets added to the transmission queue at time t .

2.5.1.1 Bernoulli scheme

The Bernoulli rate control scheme is based on the arrival process described in [71, 74].⁴ At each time slot, the probability of each value of $a(t)$ is given by

$$\begin{aligned} \Pr(a(t) = 1) &= \lambda, \\ \Pr(a(t) = 0) &= 1 - \lambda \end{aligned} \tag{2.7}$$

independently at each time slot. Since packet additions to the transmission queue are determined randomly, this rate control scheme does not require any feedback from the receivers.

⁴Note that we use Bernoulli scheme to moderate the number of packets from the packet backlog that are coded by the sender. This treatment differs slightly from [71, 74], where the arrival process fundamentally limits the transmission rate of the sender.

2.5.1.2 Block-based scheme

Under the block-based rate control scheme, packets are divided into blocks of size B for transmission.⁵ The sender begins transmission of a new block by setting $a(t) = B$. While receivers are in the process of receiving this block, $a(t) = 0$ so the sender does not add new packets to the transmission queue. Receivers only transmit a notification once they have decoded all packets in the current block. After a notification has been received from every receiver, all the packets that were in the transmission queue can be dropped, and the sender begins transmission of a new block.

2.5.1.3 Delay threshold scheme

The delay threshold rate control scheme is taken from [4]. The delay threshold scheme operates under two settings, which we call the *start* and *stop* settings. By default the sender is in the start setting, where it adds a new packet to the transmission queue whenever all packets in the transmission queue have been decoded by at least one of the receivers. However, if any packets have remained inside the transmission queue for more than the threshold T_D number of time slots, the sender switches to the stop setting. During this time the sender waits, setting $a(t) = 0$, while the coding block transmits uncoded copies of the expired packet(s). Once all packets remaining in the transmission queue are less than T_D time slots old, the sender reverts back to the start setting.

2.5.1.4 Sliding window scheme

The sliding window rate control scheme is taken from [80]. The sliding window rate control scheme only permits the coding of packets within a fixed window of size W , which by default moves at a predetermined rate V . Each time slot that the window advances, a new packet is added to the transmission queue. If the window size exceeds W , the oldest packet is dropped from the window. Therefore the transmission queue consists of a window of the last W packets.

To ensure that every packet can eventually be decoded, the sliding window scheme also ensures that for every packet $\mathbf{p}_n$ exiting the window, the receiver knows at least one combination of the form $\mathbf{p}_n + \sum_{i=n+1}^{\infty} \alpha_i \mathbf{p}_i$. This ensures that the receiver will have enough information to decode the packet at some future point in time. Whenever a packet at the end of the window is still unseen by one of the receivers, that receiver will transmit a notification to the sender. Whenever the sender receives one of these notifications, it will pause the sliding window so that no new packets

⁵In this thesis we will only consider the case where B is a fixed integer, however, there exist results such as [82] where the possibility of variable block sizes has been considered.

are added. This ensures that an appropriate combination of every packet $\mathbf{p}_n$ must be received before the sender will continue to advance the window.

The sliding window rate control scheme only requires feedback whenever a receiver is about to drop outside the sliding window. Depending on V , the window speed, and W , the size of the window, the probability of requiring feedback can become very small [80].

2.5.1.5 Dynamic rate control and queue threshold schemes

In Chapter 4 we will introduce two novel rate control schemes, the dynamic rate control and queue threshold schemes, whose design is based on the analysis performed in Chapter 3. The principle behind both rate control schemes is to regulate the transmission queue according to the number of packets waiting to be delivered. The details and derivation of these transmission schemes will be provided in Chapter 4.

2.5.2 Coding schemes

We now outline the coding schemes that will be considered in this thesis. Recall that the coding scheme is responsible for determining the network coding coefficients selected for the sender's transmissions.

2.5.2.1 Coding scheme A

Coding scheme A was originally described in [74].⁶ This coding scheme assumes that lossless, delay free feedback is available from the receivers. The complete coefficient selection algorithm is given in Algorithm 1, and an example is provided in Table 2.1. It relies on the following concept of a *witnessed* packet.⁷

DEFINITION 2.9: A receiver R_{x_r} has *witnessed* a packet $\mathbf{p}_n$ if it knows a packet combination of the form

$$\mathbf{p}_n + \mathbf{c}, \quad (2.8)$$

where $\mathbf{c}$ is some linear combination of newer packets $\mathbf{p}_{n+1}, \dots, \mathbf{p}_{A'(t)}$. Otherwise, it is *unwitnessed*. If $\mathbf{p}_n$ has been witnessed, then the *witness* for that packet is defined as the unique linear combination

$$W_r(\mathbf{p}_n) = \mathbf{p}_n + \mathbf{q}, \quad (2.9)$$

⁶In [74], coding scheme A is referred to as ARQ for network coding (ANC).

⁷In [74] this is referred to as a *seen* packet.

where $\mathbf{q}$ consists only of packets that have not been witnessed by that receiver.

Under coding scheme A, the sender organises the receivers according to its oldest unwitnessed packet. Moving from oldest to newest unwitnessed packet, it adds an appropriate multiple of each packet to the transmission $s(t)$. The coefficient chosen is the smallest value that will allow the corresponding receivers to witness their oldest unwitnessed packet. As a result, coding scheme A will allow all receivers to witness their oldest unwitnessed packet.

It should be noted that in the original transmission scheme of [74] the coefficients chosen for the receivers' oldest unwitnessed packets are not specified, other than requiring that they not be in the *veto list* (See Algorithm 1). In this thesis, we always choose the lowest value coefficient that is not in the veto list. The reasons for this choice of coefficients will become apparent in Chapter 3.

Algorithm 1 Coding scheme A.

- 1: Organise receivers $Rx_1, \dots, Rx_R$ into groups G_j , so that G_j contains all receivers Rx_r whose oldest unwitnessed packet is $\mathbf{p}_j$. Let G be the set of all groups G_j which contain at least one receiver.
 - 2: **for** $j = 1, \dots, A'(t)$ **do**
 - 3: Set $\alpha_j = 0$.
 - 4: Initialise the empty *veto list* $v_j = \{\}$.
 - 5: **for** Each receiver $Rx_r \in G_j$ **do**
 - 6: Calculate the witness $W_r(\mathbf{p}_w)$ of each packet $\mathbf{p}_w$ that has been witnessed by Rx_r .
 - 7: Find $y_r = \alpha_w \sum W_r(\mathbf{p}_w)$, the sum of all $W_r(\mathbf{p}_w)$ corresponding to witnessed packets $\mathbf{p}_w$, weighted by the coefficients α_w already chosen for these packets.
 - 8: Find the coefficient of $\mathbf{p}_j$ in y_r , and add it to the veto list v_j .
 - 9: **end for**
 - 10: Set α_j to the lowest value coefficient not listed in v_j .
 - 11: **end for**
 - 12: Set $s(t) = \sum_{i=1}^{A'(t)} \alpha_i \mathbf{p}_i$.
-

2.5.2.2 Coding scheme B

Coding scheme B, outlined in Algorithm 2, is equivalent to the coding scheme introduced in [71].⁸ An example of coding scheme B in action is given in Table 2.2. Under coding scheme B, the sender transmits a minimal combination based on the *next needed* packet of each receiver.

⁸Algorithm 2 is a slightly simpler but equivalent form of the coding scheme in [71]. This simplification is possible because it can be shown that, in each iteration j of the algorithm, the *veto list* concept used in [71] only impacts the selection of the coefficient α_j .

t	$s(t)$	Rx ₁	Rx ₂	Rx ₃
0		$p_1 \cdots p_{10}$	$p_1 \cdots p_3, p_4 + p_6$	p_1, p_2
1	$p_3 + p_5 + p_{11}$	✗ $p_1 \cdots p_{10}$	✓ $p_1 \cdots p_3, p_4 + p_6, p_5 + p_{11}$	✓ $p_1, p_2, p_3 + p_5 + p_{11}$
2	$p_4 + p_{11}$	✗ $p_1 \cdots p_{10}$	✗ $p_1 \cdots p_3, p_4 + p_{11}, p_5 + p_{11}, p_6 - p_{11}$	✓ $p_1, p_2, p_3 + p_5 + p_{11}, p_4 + p_{11}$
3	$p_5 + p_7 + p_{11}$	✓ $p_1 \cdots p_{11}$	✗ $p_1 \cdots p_3, p_4 + p_{11}, p_5 + p_{11}, p_6 - p_{11}$	✓ $p_1, p_2, p_3 - p_7, p_4 + p_{11}, p_5 + p_7 + p_{11}$

Table 2.1: An example of coding scheme A transmission. Ticks and crosses represent receivers' channel states (successful and unsuccessful reception, respectively) at each transmission.

t	$s(t)$	Rx ₁	Rx ₂	Rx ₃
0		$p_1 \cdots p_{10}$	$p_1 \cdots p_4, p_6$	p_1, p_2, p_7
1	p_{11}	✗ $p_1 \cdots p_{10}$	✓ $p_1 \cdots p_4, p_6, p_{11}$	✗ p_1, p_2, p_7
2	$p_{11} + p_5$	✗ $p_1 \cdots p_{10}$	✓ $p_1 \cdots p_6, p_{11}$	✓ $p_1, p_2, p_7, p_{11} + p_5$
3	$p_{11} + p_7$	✗ $p_1 \cdots p_{10}$	✗ $p_1 \cdots p_6, p_{11}$	✓ $p_1, p_2, p_5, p_7, p_{11}$
4	$p_{11} + p_7 + p_3$	✓ $p_1 \cdots p_{11}$	✓ $p_1 \cdots p_7, p_{11}$	✓ $p_1 \cdots p_3, p_7, p_{11}$

Table 2.2: An example of coding scheme B transmission. Ticks and crosses represent receivers' channel states (successful and unsuccessful reception, respectively) at each transmission.

DEFINITION 2.10: The *next needed* packet $N_i(t)$ of a receiver Rx_i is the oldest packet which has not been decoded by that receiver.

The sender organises receivers in a list from newest to oldest, according to their next needed packet. Beginning with the newest next needed packet in the list, it adds each packet only if the receiver(s) corresponding to them would not otherwise receive an innovative packet. The coefficient chosen for each packet is the smallest value that will ensure the transmission is innovative to the corresponding receivers.

2.5.2.3 Moded scheme B

In Chapter 5 we will introduce moded scheme B, given in Algorithm 3, which is a generalisation of coding scheme B. At each time slot, the sender selects a *transmission mode* m . The sender then applies the same coding algorithm as coding scheme B to the next needed packets of receivers $Rx_m, Rx_m + 1, \dots, Rx_R$. Note that since the next needed packets of $Rx_1, \dots, Rx_{m-1}$ are not coded by the sender, transmissions will quite possibly be non-innovative for these receivers. Coding scheme B can be considered a subcase of moded scheme B, where $m = 1$.

Algorithm 2 Coding scheme B.

- 1: Organise receivers $Rx_1, \dots, Rx_R$ into groups G_j , so that G_j contains all receivers Rx_i whose next needed packet $N_i(t) = \mathbf{p}_j$. Let G be the set of all groups G_j which contain at least one receiver.
 - 2: Initialise $s(t) = 0$.
 - 3: **for** each group $G_j \in G$, from high to low j , **do**
 - 4: Initialise the empty *veto list* $v_j = \{\}$.
 - 5: **for** each receiver $Rx_i \in G_j$ **do**
 - 6: Calculate r_i , the result of performing Gaussian elimination on $s(t)$ with the packets in Rx_i 's buffer.
 - 7: **if** $r_i = 0$ **then**
 - 8: Add 0 to v_j .
 - 9: **else if** $r_i = \alpha \mathbf{p}_j$ for some field element α **then**
 - 10: Add α to v_j .
 - 11: **end if**
 - 12: **end for**
 - 13: **if** $0 \in v_j$ **then**
 - 14: Set a_j to the smallest value not listed in v_j .
 - 15: Set $s(t) = s(t) + a_j \mathbf{p}_j$.
 - 16: **end if**
 - 17: **end for**
-

Algorithm 3 Moded scheme B.

- 1: For mode m , organise receivers $Rx_m, \dots, Rx_R$ into groups G_j , so that G_j contains all receivers Rx_i whose next needed packet $N_i(t) = \mathbf{p}_j$. Let G be the set of all groups G_j which contain at least one receiver.
 - 2: Initialise $s(t) = 0$.
 - 3: **for** each group $G_j \in G$, from high to low j , **do**
 - 4: Initialise the empty *veto list* $v_j = \{\}$.
 - 5: **for** each receiver $Rx_i \in G_j$ **do**
 - 6: Calculate r_i , the result of performing Gaussian elimination on $s(t)$ with the packets in Rx_i 's buffer.
 - 7: **if** $r_i = 0$ **then**
 - 8: Add 0 to v_j .
 - 9: **else if** $r_i = \alpha \mathbf{p}_j$ for some field element α **then**
 - 10: Add α to v_j .
 - 11: **end if**
 - 12: **end for**
 - 13: **if** $0 \in v_j$ **then**
 - 14: Set a_j to the smallest value not listed in v_j .
 - 15: Set $s(t) = s(t) + a_j \mathbf{p}_j$.
 - 16: **end if**
 - 17: **end for**
-

2.5.2.4 RLNC scheme

In the random linear network coding scheme, or RLNC for short, network coding coefficients for each packet are selected at random from a large field size. This procedure is outlined in Algorithm 4. By using RLNC, if the field size is sufficiently large, then the probability of transmitting innovative information to a receiver approaches 1 [34]. The RLNC scheme does not rely on feedback from the sender to determine the network coding coefficients.

Algorithm 4 RLNC scheme, with field size $\mathbb{F}_M$.

- 1: Initialise $s(t) = 0$.
 - 2: **for** $i = 1, \dots, A'(t)$ **do**
 - 3: Choose α_i randomly from the field $\mathbb{F}_M$.
 - 4: Set $s(t) = s(t) + \alpha_i \mathbf{p}_i$.
 - 5: **end for**
-

2.5.2.5 RLNC-I scheme

The innovative RLNC scheme, or RLNC-I for short, is a small field size version of RLNC. Unlike RLNC, feedback is used to ensure that every transmission is innovative to all receivers. The sender constructs a random combination of all $A'(t)$ packets in the transmission queue, by choosing packet coefficients randomly from $\mathbb{F}_M$. Feedback is used to ensure that the final packet is innovative to all receivers. If the chosen network coded packet is not innovative to all receivers, a new set of random coefficients is chosen until an appropriate network coded packet is found.

Algorithm 5 RLNC-I coding scheme, with field size $\mathbb{F}_M$.

- 1: Initialise $s(t) = 0$.
 - 2: Let G be the set of all receivers $Rx_1, \dots, Rx_R$ who have not delivered all packets in the transmission queue.
 - 3: **while** $s(t)$ is non-innovative to at least one receiver $Rx_r \in G$ **do**
 - 4: **for** $i = 1, \dots, A'(t)$ **do**
 - 5: Choose α_i randomly from the field $\mathbb{F}_M$.
 - 6: Set $s(t) = s(t) + \alpha_i \mathbf{p}_i$.
 - 7: **end for**
 - 8: **end while**
-

2.5.2.6 Features of the coding schemes

There are two features that are common to the majority of coding schemes studied in this thesis.

Firstly, with the exception of the RLNC scheme, all of the coding schemes we will study require that complete knowledge of the information stored at the receivers is available. In other words, ideal feedback conditions are necessary for these transmission schemes to function properly.

Secondly, with the exception of the coded scheme, all the coding schemes we will study have what we term the innovation guarantee property.

DEFINITION 2.11: A coding scheme has the *innovation guarantee property* if each transmission it selects is innovative to all receivers who are missing packets from the transmission queue.

In other words, if a receiver is still missing information from the transmission queue, these coding schemes select network coded combinations that are guaranteed to provide innovative information to that receiver.

2.6 Definitions

For quick reference, we provide a list of definitions for the terminology that will be used throughout this thesis. Terms are listed in alphabetical order.

Add: A packet is *added* when it is first passed into the transmission queue.

Channel rate: A receiver's *channel rate* is the average rate at which packets are successfully received.

Coded: If a sender transmission $s(t)$ includes a nonzero multiple of a packet $\mathbf{p}_n$, then that packet is said to be *coded* into $s(t)$.

Coding: A vector C that indicates which of the receivers' next needed packets have been coded into a transmission.

Complete information property: If n packets have been transmitted by the sender so far, a receiver is guaranteed to decode all of these packets once it has received n innovative transmissions.

Decoded: A packet $\mathbf{p}_n$ is *decoded* by a receiver once it can be recovered from the received transmissions using Gaussian elimination.

Delivered: A packet $\mathbf{p}_n$ is *delivered* once packets $\mathbf{p}_1, \dots, \mathbf{p}_n$ have been decoded.

Delivery delay: The *delivery delay* of a network coded system is the average time between the first transmission of a packet by the sender, and the packet's delivery to each of the receivers.

Delivery rate: The *delivery rate* of a given receiver is the average rate at which packets are delivered to it.

Fairness: A system is considered to be *fair* if the performance experienced by each

receiver is proportional to the quality of the physical conditions it experiences.

Faster receiver: A receiver Rx_i is *faster* than another receiver Rx_j if it has delivered more packets.

Innovative: A transmission $s(t)$ is *innovative* to a receiver Rx_i if receiving $s(t)$ will give the receiver new information about the packets stored at the sender. Otherwise, it is *non-innovative*.

Intermediate packet delivery: If a receiver delivers a new packet, but has not yet delivered all of the packets so far transmitted by the sender, then it is classified as an intermediate packet delivery.

Newer packet: A packet p_n is *newer* than another packet p_m if its index $n > m$.

Next needed packet: The first packet that has not yet been decoded by a particular receiver.

Non-innovative: A packet p_n is *non-innovative* to a receiver Rx_i if it is able to construct the packet using linear combinations of the information it has already received. In this case, receiving $s(t)$ will not give the receiver new information.

Non-uniform channel rates: A system has *non-uniform* channel rates if receivers experience distinct channel rates, $\mu_1 > \mu_2 > \dots > \mu_R$.

Notification: A unit of feedback transmitted by a receiver.

Older packet: A packet p_n is *older* than another packet p_m if its index $n < m$.

Packet index: The index of a packet p_n is n .

RLNC: Random linear network coding.

Seen: A packet p_n has been *seen* if the receiver knows a linear combination of p_n and older packets, $p_1, \dots, p_{n-1}$.⁹

Slower receiver: A receiver Rx_i is *slower* than another receiver Rx_j if it has delivered fewer packets.

Throughput: The average rate at which packets are delivered to all receivers.

Throughput delay performance: The set of all simultaneously achievable throughputs and delivery delays of a system. The higher the throughput for a given delivery delay, the better the throughput delay performance.

Transmission rate: The average rate at which new packets are transmitted by the sender. This can be approximated by the average rate at which packets are added to the transmission queue.

Transmission scheme: An algorithm for determining what network coded packet the sender should transmit at each time slot.

Undelivered: A packet p_n is *undelivered* to a receiver Rx_i if one or more of the packets

⁹This definition of seen is taken from [71], the coding scheme upon which much of the work in this thesis is based. Note, however, that the term "seen" has also been used in [70, 74, 80] to mean a combination of only p_n and newer packets.

$\mathbf{p}_1, \dots, \mathbf{p}_n$ has not yet been decoded.

Uniform channel rates: A system has *uniform* channel rates if the receivers all experience the same channel rate, $\mu_1 = \mu_2 = \dots = \mu_R$.

Witnessed: A packet $\mathbf{p}_n$ has been *witnessed* by a receiver Rx_i if the receiver knows a linear combination of $\mathbf{p}_n$ and newer packets, $\mathbf{p}_{n+1}, \mathbf{p}_{n+2}, \dots$.

Throughput delay analysis

In this chapter we study in-order packet delivery delay of existing coding schemes with applications in wireless broadcast. Although the analysis we will perform is specific to these coding schemes, it will provide insight into the general nature of network coding schemes that will inform the remainder of this thesis. Unlike previous works such as [72, 73], where the asymptotic behaviour of the delivery delay is studied, we provide a general analysis of the three conditions under which in-order packet delivery is possible at a receiver: 1) zero state delivery, 2) leader state delivery, and 3) coefficient-based delivery.

We use a Markov model to represent the difference between the *knowledge space* of the sender and a receiver. For zero state delivery, we calculate the expected distribution of decoding cycle lengths and resulting delivery delay under the Markov model. For leader state decoding, we make use of a simplifying independent Markov model among receivers to shed light on the factors that determine the probability of delivering packets while a leader. Finally, we compare the coefficient-based delivery probabilities of two transmission schemes and a baseline random transmission algorithm to show that surprisingly (and fortunately), the probability of coefficient-based delivery is significant in one of the transmission schemes. We verify our analysis by extensive simulations and discuss the usefulness of our study for understanding and design of better transmission algorithms.

3.1 Introduction

In recent times there have been many advances in the capabilities of wireless communication systems. High bandwidth, low delay communications are required. Network coding is one method for achieving high bandwidth transmission. While it can offer improvements to the throughput [2], it also typically comes at the cost of an additional *delivery delay*, as packets must first be decoded, then delivered in the correct order to the application.

While high throughput and low delivery delay are both desirable, it is well known that network coding introduces a tradeoff between these performance measures [4, 19, 46, 55, 84]. Generally the higher the throughput, the higher the delay, while lower delays can only be achieved at the cost of reduced throughput. As a result of this tradeoff, when evaluating different network coding strategies it is often useful to consider the *throughput delay performance*, which measures the best jointly achievable throughput and delivery delay performance.

Although the tradeoff between throughput and delay performance is well established, the throughput delay performance of a given network coded system can vary greatly across different network coding schemes. Some network coding strategies, for example instantaneously decodable network coding, are designed to minimise delay, while others such as random linear network coding aim to maximise throughput. As a result, a diverse range of results and techniques exist for quantifying the throughput delay performance of different network coding schemes.

3.1.1 Throughput delay analysis

Analysis of the throughput delay performance achieved using network coding is a nontrivial task. One of the primary reasons for this is the presence of *feedback*. Under *feedback-based* transmission schemes, the receivers transmit information about their progress back to the sender, who uses the information to determine the content of future transmissions. By adapting these transmissions to the receiver conditions, the sender can potentially achieve improved throughput delay performance.

The effects of feedback can be difficult to analyse. The sender's transmissions depend on the information passed by the receivers, but the information stored at the receivers is itself determined by the sender's past transmissions. As a result, the use of feedback in a network coded system can introduce large numbers of recursive dependencies, as the sender's actions are determined by the information stored at the receivers, and vice versa. Some analytical results exist in the literature for network coding strategies that make use of feedback in simple ways, but the throughput delay performance of more complex, feedback-dependent coding strategies is in many cases not well understood.

We now present an overview of existing throughput delay analysis in the literature, focusing on results for broadcast applications in packet erasure channels, as they are directly related to our work. Recall the network coding model introduced in Section 2.4.

3.1.1.1 Rate control analysis

The rate control scheme is responsible for determining how quickly new packets should be introduced for network coding. There are two key classes of rate control strategies: block-based and window-based rate control strategies. These were briefly explored in Section 1.1.1.

Under *block-based* strategies, packets are divided into blocks of a fixed size. Feedback from the receivers indicates to the sender when the block has been successfully received, and once a block has been successfully received by all receivers, the sender commences transmission of a new block. The relatively simple feedback dependencies of block-based rate control mean that a number of quantitative results exist for the throughput delay performance of block-based coding [19, 56, 58, 75, 82].

Window-based rate control strategies, on the other hand, allow transmissions to be coded from a moving window of packets. Throughput delay analysis of window-based rate control strategies is perhaps more complex than that of block-based strategies, because the number of packets that receivers must obtain to decode the window changes over time. Some results exist for the throughput delay performance of the simpler, feedback independent rate control schemes. The throughput delay performance of the sliding window scheme was determined in [80], while asymptotic delay analysis exists for the Bernoulli scheme.¹

3.1.1.2 Coding scheme analysis

The coding block is responsible for determining what combination of the available packets should be selected by the sender for each outgoing transmission. Analysing the throughput delay performance of coding schemes is often an extremely complicated task, as network coding and decoding behaviours and the subsequent throughput delay performance are affected by feedback in subtle and often unpredictable ways.

From an analytical perspective, the simplest coding scheme in the literature is random linear network coding (RLNC), where coding coefficients are chosen randomly and therefore without any dependency on feedback. It is shown in [34] that, for a sufficiently large field size, RLNC allows the optimal throughput to be achieved, albeit without any improvements to the delay performance. This is to our knowledge the only coding scheme whose throughput delay performance has been completely characterised.

Other network coding schemes are generally more complex, and therefore difficult to analyse in exact form. For example, although there has been much study

¹A description of these transmission schemes can be found in Section 2.5.1.

of instantaneously decodable network coding (IDNC) schemes, including [47, 55, 59, 65, 66], the throughput delay performance of such schemes is extremely difficult to quantify [47]. Throughput optimal coding schemes have been studied for the two- and three-receiver subcases in [4, 47, 72], but precise analysis becomes intractable for larger numbers of receivers [4] and has therefore not been attempted in the literature. However, some asymptotic results for the throughput delay performance have been provided in [73]. It has been speculated in [73, 80] that throughput optimal coding schemes offer little improvement to the delay performance of a network coded system compared with random linear network coding.

3.1.2 Contributions and related work

In this chapter, we provide a non-asymptotic characterisation of the throughput delay performance of selected throughput optimal feedback-based transmission schemes. Some related analysis has been performed in [80], however, their work does not consider the impact of the coding scheme on the delivery delay performance. By first understanding the mechanism by which packets are delivered in transmission schemes such as [71, 74], we gain insight into the nature of the *throughput-delay trade-off*, the set of throughput values and delivery delays simultaneously achievable by a system. This in itself is a difficult problem, owing to the complex interactions between the sender and receivers. To manage this, we categorise the methods of packet delivery into three categories: zero state, leader state and coefficient-based delivery. These distinctions are made based on the difference between the number of packets known by the sender and a receiver at each time step. This important variable is Markovian, therefore, we refer to it as a receiver's *Markov state*. By decoupling the contributions of each method of delivery, we can present an approximation that removes the effect of cross receiver interactions. In return for some loss of accuracy, we are able to transform a mathematically intractable problem into one that gives easily calculable results.

Feedback-based throughput optimal coding schemes have also been proposed to reduce the transmission queue size [74], and minimise the delivery delay [71], however, no attempt has been made to study the extent to which these schemes work. To the best of our knowledge, 1) there has been no work on characterising the non-asymptotic delivery behaviour of the Bernoulli scheme considered in [71, 74], and 2) the delay performance of the coding schemes presented in [71, 74] has not yet been analysed.

3.2 System Model

The system model we will consider is essentially that described in Section 2.3. For the reader's convenience, we restate the features of the model that are pertinent to this chapter. A single sender aims to transmit a backlogged set of packets $\mathbf{p}_1, \mathbf{p}_2, \dots$ in the correct order to a set of R receivers, as shown in Fig. 3.1. Time is slotted, denoted by $t = 1, 2, \dots$, and the sender can broadcast at the rate of one network coded packet per time slot. The receivers are connected to the sender via independent erasure channels with uniform channel rate μ , so that they successfully receive transmissions with probability μ at each time slot.²

Receivers store received packets in a buffer and send a notification after each successful packet reception, which we assume the sender detects instantaneously without error.³ The sender uses this information to record which packets receivers have stored in their buffers. Based on this information, a *transmission scheme* can be devised to determine the packet combinations the sender will transmit. In the remainder of this section, we review the performance measures that will be considered, and outline the components of the transmission schemes we will study.

3.2.1 Performance measures

In this chapter, we consider two performance measures: *throughput* and *delivery delay*. We use the following definitions.

DEFINITION 3.1: The *throughput* of a system is measured as the average number of packets delivered per time slot, across all receivers.

DEFINITION 3.2: A packet $\mathbf{p}_n$ is said to be *delivered* to a receiver if that receiver has decoded all packets $\mathbf{p}_1, \dots, \mathbf{p}_n$. Otherwise, $\mathbf{p}_n$ is said to be *undelivered* to that receiver.

DEFINITION 3.3: The *delivery delay* of a transmission scheme is measured as the average number of time slots between any packet $\mathbf{p}$ becoming available for transmission, to the time it is delivered to each of receivers.⁴

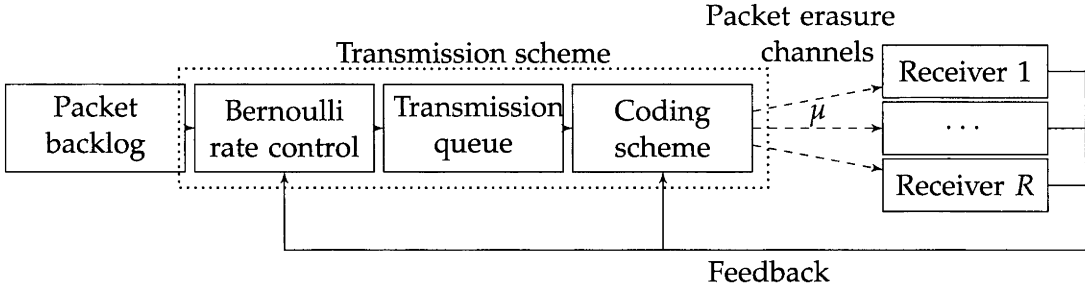


Figure 3.1: A block diagram of the components of a transmission scheme.

3.2.2 Transmission schemes

Our transmission model, shown in Fig. 3.1, is based on the model outlined in Section 2.4.⁵ Recall that a network coded transmission scheme can be divided into three components. From left to right in the diagram, these are: a *rate control block*, which passes new packets from the packet backlog into the *transmission queue*, which contains the set of packets from which the *coding block* determines $s(t)$, the network coded transmission to be sent at time t .

In this chapter we will consider three transmission schemes. As shown in Fig. 3.1, we substitute a Bernoulli rate control scheme into the rate control block, and three different throughput optimal coding schemes are used for the coding block. To study the effects of the coding scheme on the performance, we will focus on two existing schemes, the *drop-when-seen* coding scheme of [73, 74] and the asymptotically optimal delivery scheme of [71], which we call coding schemes A and B respectively. As a means of comparison, we also consider the innovative random linear network coding (RLNC-I) scheme. Recall that throughput optimal coding schemes have the *innovation guarantee property*, meaning that, at each time slot, the transmitted packet $s(t)$ will be innovative for all receivers who are still missing packets from the transmission queue. For quick reference, the essential details of the rate control and coding schemes we will study in this chapter are summarised below: complete details can be found in

²Our analysis can also be applied to systems with non-uniform channel rates, but for clarity of explanation we only consider uniform channel rates.

³Although this can be difficult to achieve in practice, it greatly simplifies analysis. We will make some comments on the effect of imperfect feedback later in this chapter.

⁴In this chapter, the delivery delay is measured from the time a packet first enters the transmission queue to the time it is delivered at the receiver. This more convenient definition is a reasonably close upper bound on the delivery delay given in Definition 2.5, because it is not guaranteed that a packet added to the transmission queue will be immediately coded into a transmission.

⁵The reader may wish to revisit Section 2.4 for a summary of the function of each block.

Section 2.5.

3.2.2.1 Bernoulli rate control

Under the Bernoulli rate control scheme, at each time slot the sender chooses to introduce either one new packet, or none into the transmission queue. The *add decision* $a(t)$ is determined by a Bernoulli process with *addition rate* λ , so that the sender will add a new packet with probability

$$\Pr(a(t) = 1) = \lambda. \quad (3.1)$$

If the Bernoulli rate control scheme chooses to *add*, then we set the add decision $a(t) = 1$; otherwise it *waits*, and we set $a(t) = 0$.

3.2.2.2 Coding scheme A

Coding scheme A relies on the concept of witnessed packets. A packet $\mathbf{p}_i$ is *witnessed* by a receiver if it can use the packets in its buffer to create a combination of the form $\mathbf{p}_i + f(\mathbf{p}_{>i})$, where $f(\mathbf{p}_{>i})$ is some linear combination of newer packets $\mathbf{p}_{i+1}, \mathbf{p}_{i+2}, \dots$. If this is not possible, then $\mathbf{p}_i$ is *unwitnessed*.

Coding scheme A ensures that with each successful packet reception, a receiver witnesses its next unwitnessed packet. To determine what coded packet to transmit next, the sender lists the oldest unseen packet from each receiver. Moving from oldest to newest unseen packet, it adds an appropriate multiple of each packet, so that the resulting packet is innovative to all the corresponding receivers.

3.2.2.3 Coding scheme B

Under coding scheme B, the sender transmits a minimal combination based on the oldest undecoded packet of each receiver. The sender lists these oldest undecoded packets and their corresponding receivers, then beginning with the newest packet in the list, it adds in older packets only if the receiver(s) that correspond to them would not otherwise receive an innovative packet.

3.2.2.4 RLNC-I scheme

The sender transmits a random combination of all $A'(t)$ packets in the transmission queue. Coefficients are selected randomly from $\mathbb{F}_M$, where $M \geq R$. To ensure throughput optimality, feedback is used to ensure that the final packet satisfies the innovation guarantee property. If the current set of coefficients are not innovative to all

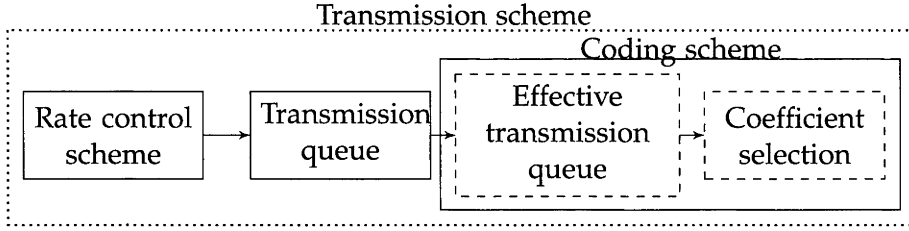


Figure 3.2: Under coding schemes A and B, the coding scheme can further be divided into effective transmission queue and coefficient selection components.

receivers still missing packets from the transmission queue, new random coefficients are chosen until an appropriate combination is found.

3.2.3 Effective transmisssion queue

It should be noted that, unlike the RLNC-I scheme, coding schemes A and B are more selective about the packets coded into each transmission.

Property 3.1. *Under coding schemes A and B, the sender will only code a new packet $\mathbf{p}_n$ if one of the receivers has decoded $\{\mathbf{p}_1, \dots, \mathbf{p}_{n-1}\}$.*

This property holds under coding schemes A and B since the sender is only permitted to code the oldest unseen or undecoded packet of each user. As a result, under these coding schemes the sender codes packets from an *effective transmission queue* which is limited to the next needed packet $\mathbf{p}_n$ of the receiver with the most delivered packets. Although still a part of the coding scheme, we treat the effective transmission queue as a conceptual intermediate block, as shown in Fig. 3.2. The role of the effective transmission queue will be further discussed in Section 3.6.1.

3.3 Markov State

In this section, we define the Markov state, which will be an important analytical tool throughout this thesis. By using the Markov state to distinguish between different methods of packet delivery we are able to provide insight into the delivery behaviour of throughput optimal coding schemes that has so far been missing from the literature. The Markov state will provide also an important tool for estimating the receivers' delivery delays. The Markov state is based on the concept of *virtual queue length* introduced in [73].

3.3.1 Knowledge spaces and the Markov state

At time t , the *transmission list* $V_s(t)$ is defined as the set of standard basis vectors

$$V_s(t) = \{e_1, e_2, \dots, e_{A'(t)}\} \quad (3.2)$$

corresponding to the uncoded packets $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{A'(t)}$ which are currently in the transmission queue. The sender chooses packets for transmission from the *transmission knowledge space*

$$K_s(t) = \text{span}(V_s(t)), \quad (3.3)$$

which is the set of all linear combinations the sender can compute using packets from the transmission queue. The size of the transmission knowledge space is given by

$$|K_s(t)| = M^{|V_s(t)|}, \quad (3.4)$$

where M is the field size and the notation $|X|$ represents the cardinality of the set X . The *reception list* $V_r(t)$ of a receiver r is defined as the set of received coefficient vectors which, after Gaussian elimination, correspond to vectors from the current transmission knowledge space $K_s(t)$. The *receiver knowledge space* is similarly defined as the set of all linear combinations that can be calculated from its reception list, $K_r(t) = \text{span}(V_r(t))$. These concepts will be used in our analysis of coefficient-based delivery in Section 3.6.

DEFINITION 3.4: The *Markov state* of a receiver r is defined as the difference between the size of the transmission list and reception list,

$$s_r(t) = |V_s(t)| - |V_r(t)|. \quad (3.5)$$

It should be noted that the removal of packets from the transmission queue does not affect the Markov state, since each packet removal decrements both $|V_s(t)|$ and $|V_r(t)|$.

3.3.2 Markov chain model

Under the Bernoulli rate control scheme, if $\lambda < \mu$ then changes to a receiver's Markov state $s_r(t)$ over time can be modelled as a traversal through a Markov chain. This is illustrated in Fig. 3.3, where the states $0, 1, 2, \dots$ correspond to the values of $s_r(t)$. Whether $s_r(t)$ increases, decreases or remains the same between time slots depends on both the add decision $a(t)$ and the receiver's channel conditions. For example, by (3.5), if a receiver experiences an erasure and a new packet is added to the transmis-

State transition	Probability	Shorthand notation
$s_r(t+1) = s_r(t) + 1$	$\lambda\bar{\mu}$	p
$s_r(t+1) = s_r(t) - 1$	$\bar{\lambda}\mu$	q
$s_r(t+1) = s_r(t)$	$\lambda\mu + \bar{\lambda}\bar{\mu}$	$1 - p - q$

Table 3.1: The probability of transitions between Markov states for $s_r(t) > 0$.

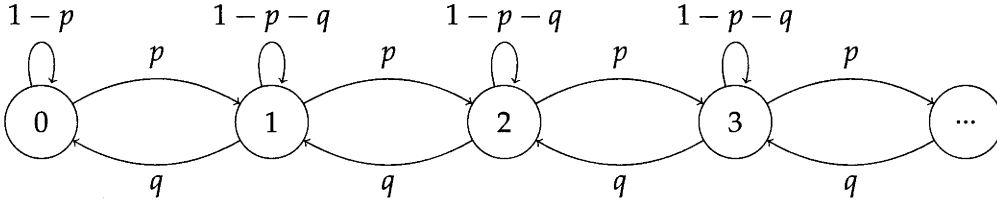


Figure 3.3: A Markov chain describing transitions in the Markov state of a receiver r .

sion queue, then the receiver's Markov state will increase by 1. The allowable state transitions for states greater than zero and their probabilities are listed in Table 3.1. Note that as long as $\lambda < \mu$, the Markov chain is positive recurrent. Although the Markov chain model is perfectly accurate for any receiver considered on its own, the fact that the sender is shared means that receivers' Markov states can exhibit a significant amount of correlation with one another. Nevertheless this model still provides valuable insight into the delivery delay characteristics of the transmission schemes we will study.

Using the concept of Markov state we can categorise the ways in which the next packet p_n of a receiver r can be delivered⁶ to a receiver as follows.

1. Zero state delivery

Zero state delivery occurs when a receiver r is in the *zero state*, i.e. its Markov state $s_r(t) = 0$. At this point, the size of the reception list equals the size of the transmission list. Since coding schemes A, B and RLNC-I all satisfy the innovation guarantee property, any time that $s_r(t) = 0$, all packets in the transmission queue have been delivered by the complete information property (see Section 1.2).

2. Leader state delivery

Under coding schemes A and B, a receiver r is called a *leader* if it has the minimum Markov state, i.e. $s_r(t) = \min \{s_i(t)\}$. Leader state delivery occurs when new packets are delivered by the current leader, although we require

⁶This categorisation is also applicable to decoding without the in-order delivery constraint. Zero and leader state delivery both result in the decoding of all packets stored at the receiver, while *coefficient-based decoding* results in the decoding of some subset of packets stored at the receiver.

that $s_i(t) > 0$ to differentiate this from zero state delivery. In Section 3.2.3 we noted that the effective transmission queue is limited to the receiver with most packets in its buffer. Therefore as shown in [73], the complete information property implies that receiving a transmission while a leader results in the delivery of all packets in the effective transmission queue to that receiver.

3. Coefficient-based delivery

Under all of the coding schemes, coefficient-based delivery accounts for any packets delivered to a receiver while it is neither leading nor in the zero state. Coefficient-based delivery occurs when the inclusion of the coefficient vector $v_s(t)$ into the reception list of a receiver r results in the decoding of the next needed packet $\mathbf{p}_r$. In this case, some fraction of the packets stored at the receiver are delivered.

3.3.3 Distribution of Markov states

Since the Markov state will form the basis of our analysis, the first step is to find the probability $S_r(k)$ that at a randomly selected time, the receiver r is in Markov state k . This is equivalent to finding the stationary distribution of the Markov chain corresponding to that receiver. For the Markov chain of Fig. 3.3, if the addition rate λ is less than the channel rate μ , a stationary distribution exists such that

$$pS_r(k) = qS_r(k+1). \quad (3.6)$$

Solving for $\sum_{k=0}^{\infty} S_r(k) = 1$, we obtain

$$S_r(k) = \left(1 - \frac{p}{q}\right) \left(\frac{p}{q}\right)^k. \quad (3.7)$$

In the following sections, we shall analyse the effect of the Markov state on the receivers' delivery delay.

3.4 Zero state delivery

In this section we will estimate the *zero state delivery delay* expected under the transmission schemes we study.

DEFINITION 3.5: The *zero state delivery delay* is defined as the delivery delay experienced if only zero state delivery is permitted.

Our estimation of the zero state delivery delay can be used to upper bound on the delivery delay for the transmission schemes that we study. It is important to

observe that, as long as the innovation guarantee property holds, the distribution of a receiver's Markov state depends only on its channel rate μ and the addition rate λ . Therefore zero state delivery is not affected by the coding scheme, the presence of other receivers, or even the quality of feedback. This independence makes zero state delivery analysis a valuable tool, as delay performance estimates for this component of the transmission scheme can be made without the intractable complications that have hindered the study of many network coded transmission schemes to date.

To find the zero state delivery delay, it is not sufficient to know the proportion of time a receiver spends in the zero state, calculated in (3.7). The zero state delivery delay depends on the distribution of times between returns to the zero state, which we call *delivery cycles*, and the distribution of transmission queue additions within each cycle. Therefore, we shall use random walk analysis techniques to calculate the distribution of delivery cycle lengths, and based on this work, find an accurate approximation for the zero state delivery delay of Bernoulli transmission schemes.

3.4.1 Delivery cycle distributions

DEFINITION 3.6: A receiver starting in Markov state 0 goes through a *delivery cycle* of length T if its first return to the zero state in the Markov chain occurs after exactly T time slots.

We begin by calculating $P_{0,0}(T)$, the probability that a delivery cycle will be of length T . This problem will be solved in two steps. First, we characterise a path through the Markov chain that consists of only *moving steps* where $s_r(t+1) = s_r(t) \pm 1$. Then we factor in the effect of *pause steps*, where $s_r(t+1) = s_r(t)$.

In the first time step there are two possibilities. The receiver can remain at state 0 with probability $1 - p$, which gives us $P_{0,0}(1) = 1 - p$. If it instead moves up to state 1, it must return to 0 in $T > 1$ time steps. For a path of fixed length T to start at and return to 0, it must consist of $2k$ moving steps, k up and k down, and $T - 2k$ pause steps, where $1 \leq k \leq \lfloor T/2 \rfloor$. If no other encounters with the zero state are permitted, the first and last time steps must be up and down steps respectively. Then the number of paths that first return to the zero state in exactly $2k$ steps without pauses is given by the $(k-1)$ -th Catalan number [6]

$$C_{k-1} = \frac{1}{k} \binom{2k-2}{k-1}. \quad (3.8)$$

Now we factor in the $T - 2k$ pauses. These pauses cannot occur in the first or last time step, otherwise the delivery cycle length would not be T . For a given path of $2k$ moving steps, there are $\binom{T-2}{2k-2}$ choices for pause locations. Therefore the probability

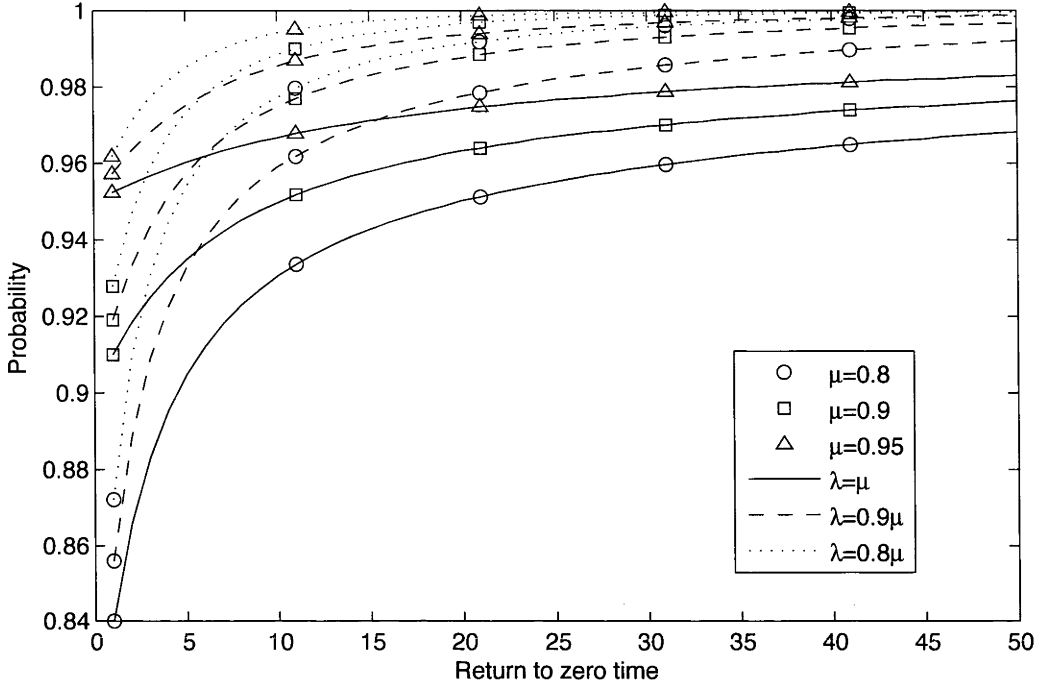


Figure 3.4: The probability from (3.9) of a receiver taking $\leq T$ time steps to return to the zero state under the Bernoulli rate control scheme.

of taking exactly $T > 1$ timeslots to return to the zero state is given by

$$P_{0,0}(T) = \sum_{k=1}^{\lfloor T/2 \rfloor} \frac{1}{k} \binom{2k-2}{k-1} \binom{T-2}{2k-2} p^k q^k (1-p-q)^{T-2k}. \quad (3.9)$$

The cumulative delivery cycle length probabilities are given for a number of values of λ and μ in Fig. 3.4. We can observe that the greater the load factor $\rho = \frac{\lambda}{\mu}$, the less likely it is that the delivery cycle length will be less than or equal to some value T . Similarly, low channel rates μ also make long delivery cycles more likely. Longer delivery cycles equate to larger zero state delivery delays, since receivers are able to deliver new packets less frequently.

3.4.2 Delay estimate

We now use the delivery cycle distributions we have calculated to accurately estimate the zero state delivery delay. This task is divided into two parts. Firstly, for a delivery cycle of length T , we estimate the number of packets added to the transmission queue and their expected delivery delay. This result is then combined with (3.9) to obtain an accurate estimate of the zero state delivery delay.

Where the delivery cycle is of length $T = 1$, the probability that one packet is added and then immediately delivered is simply $\lambda\mu$. Since the packet is immediately delivered, it incurs no delivery delay.

In Section 3.4 we established that, for all other coding cycles with length $T \geq 2$, the Markov state must increase in the first time slot, and decrease in the last time slot. Therefore, we must have $a(t) = 1$ in the first time slot, and $a(t) = 0$ in the last time slot.

We assume that in the remaining $T - 2$ time slots additions occur uniformly with probability λ . Then the average number of packets delivered over a delivery cycle of length T is estimated to be

$$1 + \lambda(T - 2) \quad (3.10)$$

and the average zero state delivery delay for each of these packets is $T/2$. The total delay incurred by these packets would then be

$$T + 0.5\lambda T(T - 2). \quad (3.11)$$

Therefore the zero state delivery delay, including the $T = 1$ delivery cycle, can be estimated as

$$\frac{\sum_{T=2}^{\infty} P_{0,0}(T)(T + 0.5\lambda T(T - 2))}{\lambda\mu + 1 + \sum_{T=2}^{\infty} P_{0,0}(T)\lambda(T - 2)}. \quad (3.12)$$

In Fig. 3.5 we show how our calculated estimate, truncated at $T = 1000$, matches well with the average delivery delays obtained from simulation.

3.5 Leader state delivery

In this section we study the *leader state delivery delay* under coding schemes A and B. While leader state delivery was shown to be an intractable complication in previous analysis [4, 73], we demonstrate that the Markov model can be used to make useful observations about the impact of leader state delivery on the delivery delay.

DEFINITION 3.7: The *leader state delivery delay* is defined as the delivery delay experienced if only zero and leader state delivery are allowed.

It should be emphasised that the leader state delivery delay measures the delay achieved under the combined effects of zero and leader state delivery. Note that as commented in Section 3.2.3, leader state delivery has an equal impact on coding schemes A and B, but does not apply to the RLNC-I scheme. We investigate the amount of time receivers spend leading and its impact on the leader state delivery delay, compared with zero state delivery on its own.

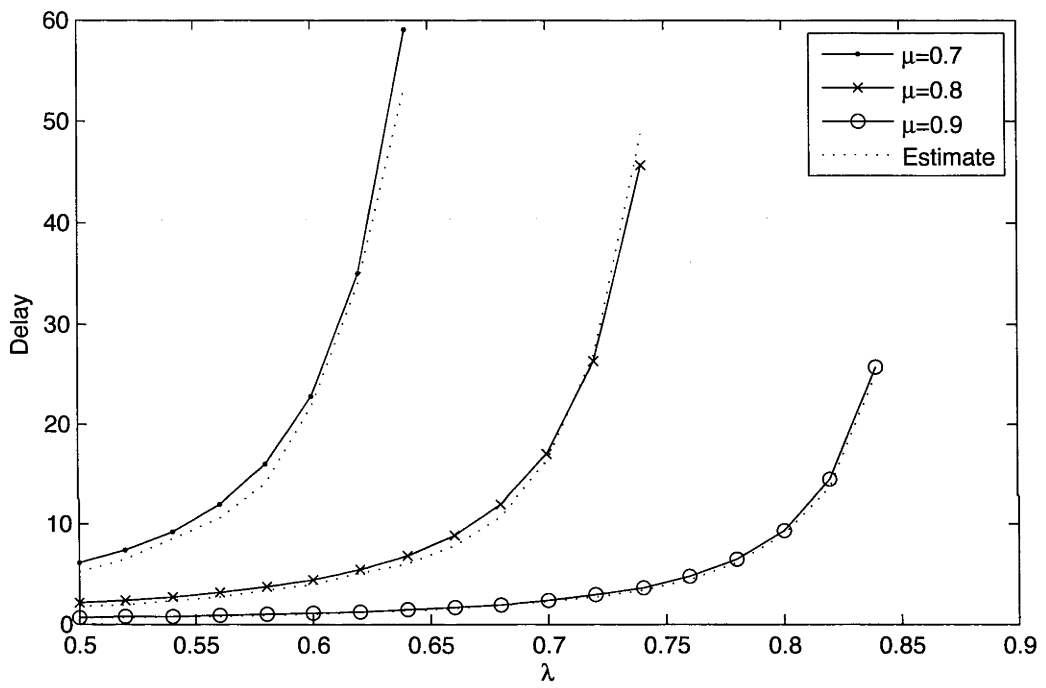


Figure 3.5: The zero state delivery delay under the Bernoulli rate control scheme, as a function of the addition rate λ . The delay estimates of (3.12) (dotted lines) are compared against simulation (solid lines).

The leader state delivery delay corresponds to decoding by the complete information assumption, on the restricted set of packets passed in by the effective transmission queue block in Fig. 3.2.

3.5.1 Leader state distribution

Based on the Markov state distribution calculated in 3.3.3, we can bound the average time that the leader(s) spend in each Markov state. By (3.7), the probability of a receiver r being in a state $\geq k$ is

$$S_r(\geq k) = \sum_{i=k}^{\infty} S_r(i) = \left(\frac{p}{q}\right)^k. \quad (3.13)$$

So if receivers' Markov states were independent of one another, the probability of having a leader in state k would be given by

$$L(k) = \left(1 - \left(\frac{p}{q}\right)^R\right) \left(\frac{p}{q}\right)^{Rk}. \quad (3.14)$$

However, since the sender is common to all receivers, there is a noticeable amount of correlation between receivers' Markov states. This is illustrated in Fig. 3.6, which compares the joint Markov state transition probabilities for two receivers under each model. In practice the correlated transition probabilities result in the receivers' Markov states being more closely grouped together than predicted by the independent receiver model. Fig. 3.7 shows that the probability of leading from states $k > 0$ is higher in practice than under the independent receiver model in (3.14).

3.5.2 Observations

The correlation between the receivers' Markov states means that in practice, the probability of the leader being in state k is bounded between the values in the single receiver case and under the independent receiver model. Therefore, although the independent receiver model is not entirely accurate, it can still be used to make the following observations about the leader state.

1. The probability that a receiver r is leading is $\geq 1/R$, since receivers have identical statistical properties, and at least one receiver must lead at each time slot.
2. The leader will most likely be in Markov state $k = 0$. The larger the number of receivers R , the more likely this is the case.

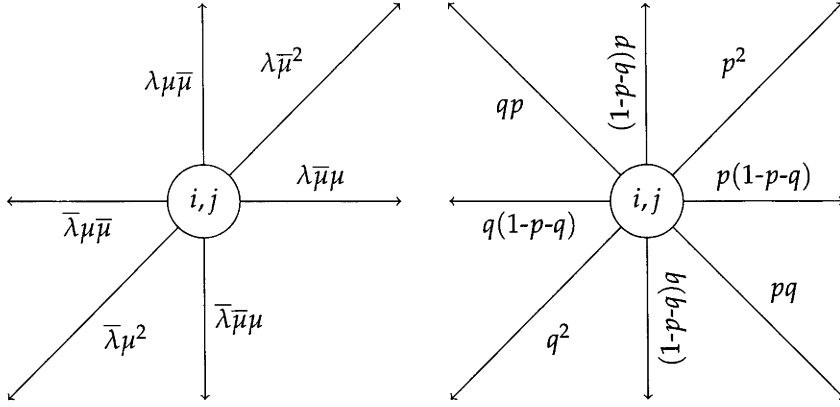


Figure 3.6: Two-receiver state transition probabilities (a) in practice, and (b) under the independent receiver model. The horizontal and vertical axes correspond to the Markov states of receivers 1 and 2 respectively, where $s_1(t) = i$ and $s_2(t) = j$. Since the sender is common to both receivers, it is not possible that $s_1(t+1) = i \pm 1$ while $s_2(t+1) = j \mp 1$.

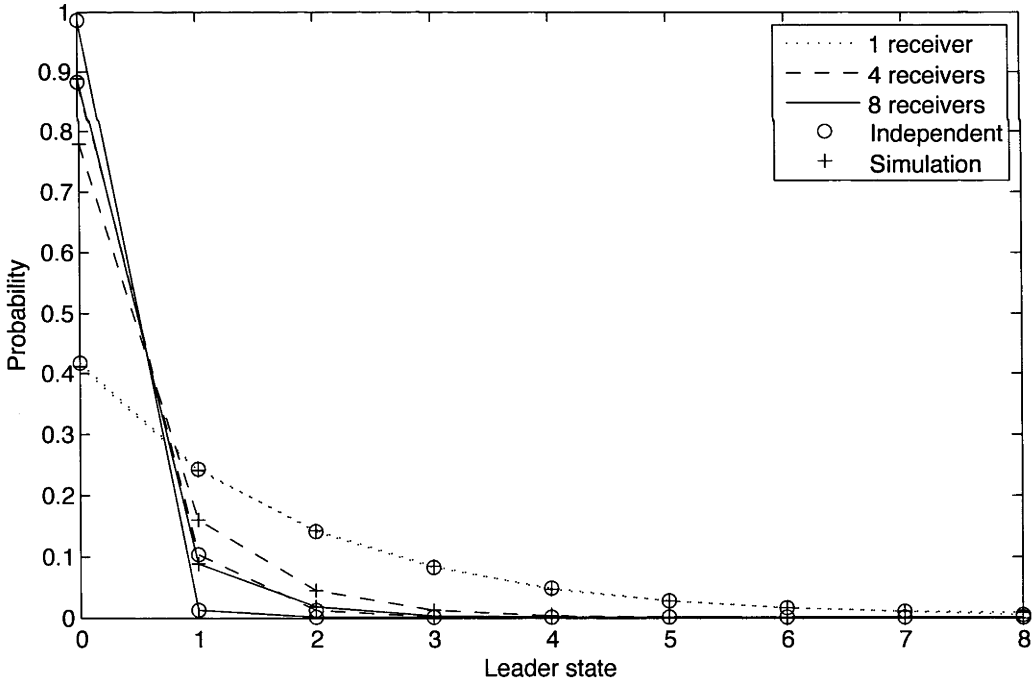


Figure 3.7: Under the Bernoulli rate control scheme with addition rate λ , the proportion of time the leader is in each state, under the independent receiver model (3.14) and in practice. Addition rate $\lambda = 0.7$, channel rate $\mu = 0.8$.

3. When $k = 0$, leader state delivery does not occur since this is classified as zero state delivery. Therefore, from observation 2, leader state delivery makes a diminishing contribution to the delivery delay performance as the number of receivers increases.
4. The higher a receiver's Markov state k , the lower its likelihood of leading.
5. By (3.7) and (3.14) as the load factor $\rho \rightarrow 1^-$, or equivalently $\lambda \rightarrow \mu^-$, the state probability distribution $S_r(\geq k)$ converges on 1 more slowly. This increases the probability that the leader will be in a Markov state $k > 0$, and therefore the impact leader state delivery has on the delivery delay.

We can observe some of these effects in Fig. 3.8. $R = 1$ represents the extreme case where there is only one receiver who is always leading, and so results in extremely low delivery delays. As R increases, however, the leader state delivery delay quickly converges towards the zero state delivery delay, confirming observation 3. Even at moderate values of R , for example $R = 10$, the difference between the zero and leader state delivery delay is negligibly small. By contrast, as the load factor ρ increases, so does the impact of leader state delivery, consistent with observation 5.

3.5.3 Imperfect feedback

Under imperfect feedback conditions, it is expected that the contribution of leader state delivery would be further diminished, since the sender would not always know which packets the leader has received. This leads to two possible choices for the sender. The sender could assume the transmission has been lost, most likely resulting in the transmission of redundant information and worsened throughput performance. Alternatively, it could assume that the transmission has been received, potentially overestimating the leader's channel rate. In order to maintain the innovation guarantee property, the sender would need to account for the possibility that the leader has received all packets for which the outcome has not yet been determined. This overestimation would result in an effective transmission queue closer in size to that of the actual transmission queue, reducing the delivery delay improvement that can be offered by leader state decoding.

3.6 Coefficient-based Delivery

Coefficient-based delivery accounts for any remaining packets delivered while a receiver is neither leading nor in the zero state. The delivery delay is therefore the delay

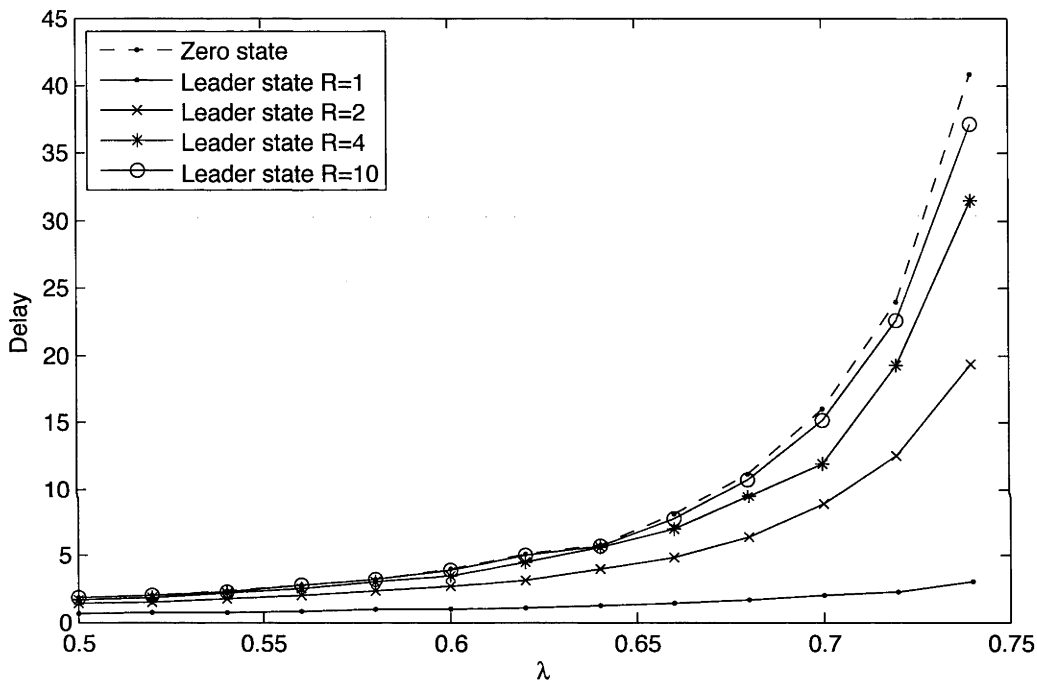


Figure 3.8: Simulated zero state and leader state delivery delays under the Bernoulli rate control scheme with addition rate λ , for different numbers of receivers R . The channel rate $\mu = 0.8$.

experienced under the combined effects of zero state, leader state and coefficient-based delivery. To date, the impact of coefficient-based delivery has not been well understood because of the difficulty of analysing its effects. In the literature it is generally speculated to contribute a small, if not negligible, improvement on the delivery delay. However, through simulation we demonstrate two important principles for improving the likelihood of coefficient-based delivery: minimising the coding field size M , and maintaining sparse codes (i.e. minimising the number of nonzero coefficients $\alpha_i(t)$ in (2.5)). When these conditions are met, coefficient-based delivery can reduce the delivery delay significantly.

Say that at time t the sender transmits a packet $s(t)$ with coefficient vector $v_s(t)$. Then using the concepts from Section 3.3.1, the next needed packet will be delivered if and only if the following condition holds.

LEMMA 3.1: At time t , a receiver can deliver its next needed packet $\mathbf{p}_n$ iff it receives a packet with coefficient vector $v_s(t) \in \text{span}(K_r(t-1) \cup e_n) \setminus K_r(t-1)$.

Proof. A packet $\mathbf{p}_n$ is decoded iff $e_n \in K_r(t)$. Say that $e_n \notin K_r(t-1)$. Then for $\mathbf{p}_n$ to be delivered at time t , $e_n \in \text{span}(K_r(t-1) \cup v_s(t))$. To satisfy the innovation guarantee property, $v_s(t) \notin K_r(t-1)$. Therefore to deliver packet $\mathbf{p}_n$ at time t , $v_s(t) \in \text{span}(K_r(t-1) \cup e_n) \setminus K_r(t-1)$. $\square$

As we shall show, the probability of coefficient-based delivery depends on both the coding scheme used and the *effective Markov state*, which we now define.

3.6.1 Effective Markov state

The *effective transmission list* $V_s^*(t)$ is defined as the set of basis vectors corresponding to packets in the effective transmission queue. In the RLNC-I scheme, typically $V_s^*(t) = V_s(t)$, unless all coefficients selected for the newest packet happen to be 0. In contrast, under coding schemes A and B the effective transmission queue is limited by the the number of packets known by the leading receiver(s), so that

$$|V_s^*(t)| = \min(\max_{r \in 1, \dots, R} (|V_r(t-1)| + 1), |V_s(t)|) \quad (3.15)$$

and $V_s^*(t) = \{e_1, e_2, \dots, e_{|V_s^*(t)|}\}$. Similarly to (3.3), the *effective transmission knowledge space* is $K_s^*(t) = \text{span}(V_s^*(t))$. The *effective Markov state* of a receiver r can then be defined as

$$s_r^*(t) = |V_s^*(t)| - |V_r(t-1)|. \quad (3.16)$$

This differs from (3.5) in that, in order to calculate the probability that the *current* transmission $s(t)$ will deliver $\mathbf{p}_n$, it compares the effective transmission list to the

reception list *prior* to packet receptions in the current time slot.

LEMMA 3.2: A receiver r can only coefficient-based deliver its next needed packet $\mathbf{p}_n$ when its effective Markov state decreases, i.e. $s_r^*(t) = s_r^*(t-1) - 1$.

Proof. It is always true that $K_r(t-1) \subset K_s^*(t-1)$. If the receiver is not a leader, then it has not decoded all packets in $K_s^*(t-1)$ and $e_n \in K_s^*(t-1)$. Therefore by Lemma 3.1, coefficient-based delivery can only occur if $v_s(t) \in K_s^*(t-1)$, so that $V_s^*(t) = V_s^*(t-1)$. Receiving an innovative packet means that $|V_r(t)| = |V_r(t-1)| + 1$, so by (3.16) $s_r^*(t) = s_r^*(t-1) - 1$. $\square$

So in order for coefficient-based delivery to be possible, three conditions must first be satisfied:

- The receiver is neither a leader nor in the zero state
- No new packets are encoded by the sender
- The receiver successfully receives the transmitted packet.

Therefore, of the time slots a receiver is neither in the zero state or a leader, approximately $\bar{\lambda}\mu$ of these provide an opportunity for coefficient-based delivery to occur. We now investigate the effectiveness of coding schemes A and B and RLNC-I in utilising this fraction of *coefficient-based deliverable* time slots to minimise delay.

3.6.2 RLNC-I scheme

To gain some insight into the probability of coefficient-based delivery, we first study the RLNC-I scheme. Here we will illustrate how the effective Markov state affects the probability of coefficient-based delivery.

We first calculate for a single receiver the probability that with receiver knowledge space $K_r(t)$, the next needed packet $\mathbf{p}_n$ will be delivered. The total number of possible transmissions is given by the size of the transmission knowledge space $K_s(t)$ minus the receiver knowledge space. Therefore by Lemma 3.1, the probability of selecting a packet under the RLNC-I scheme which allows $\mathbf{p}_n$ to be delivered is

$$\begin{aligned} & \frac{|\text{span}(K_r(t-1) \cup e_n) \setminus K_r(t-1)|}{|K_s^*(t) \setminus K_r(t-1)|} \\ &= \frac{M-1}{M^{s_r^*(t)}-1} \end{aligned} \tag{3.17}$$

Therefore, the probability of coefficient-based delivery depends only on the effective Markov state of the receiver and the field size M . The exponential dependence on

these factors means that the coefficient-based delivery probability will be very small for high effective Markov states and large field sizes.

For the multiple receiver case, simulations show that there is a fairly negligible difference between the RLNC-I coefficient-based delivery probabilities for the single and multiple receiver cases, provided they are coded using the same field size M . Some of these probabilities, normalised over coefficient-based deliverable time slots, are shown in Fig. 3.9, and the resulting delay performance for $M = 4$ is given in Fig. 3.10. As expected, the small coefficient-based delivery probability results in only a slight improvement over zero state delivery.

3.6.3 Coding scheme A

Under coding scheme A, the sender codes only the first unseen packet of each receiver. Furthermore the coefficients chosen are the smallest that will satisfy the innovation guarantee property.⁷ Although a field size $M \geq R$ is necessary to guarantee innovation, the majority of the time coefficients from a much smaller field size $\mathbb{F}_2$ are sufficient.

In Fig. 3.9 the coefficient-based delivery probability of coding scheme A is compared against the RLNC-I scheme. Under coding scheme A with four receivers and $\mathbb{F}_4$, the probability of coefficient-based delivery in fact lies between the probabilities for the $\mathbb{F}_2$ and $\mathbb{F}_4$ RLNC-I scheme. This occurs since in practice the sender usually selects binary field coefficients, effectively coding from the field $\mathbb{F}_2$. With 8 receivers and a field $\mathbb{F}_8$, packet coefficients are nearly always selected from the field $\mathbb{F}_4$. This results in coefficient-based delivery probabilities close to the RLNC-I $\mathbb{F}_4$ case. In Fig. 3.10 we can observe that coding scheme A has significantly better delay performance compared with the RLNC-I scheme. This is primarily due to the role of leader state delivery, with a slight contribution from coefficient-based delivery.

3.6.4 Coding scheme B

Coding scheme B, by contrast, attempts to closely mimic a systematic, uncoded scheme by coding additional packets into each transmission only if it is necessary to maintain the innovation guarantee property. Each of these extra packets has the additional property that, if received, the coded transmission will allow the corresponding receiver to deliver its next needed packet.

We can expect that at least λ of the sender's transmissions, corresponding to the first transmission of each new packet, will be uncoded. Fig. 3.11 agrees with

⁷In [74] it is suggested that any coefficient satisfying the innovation guarantee is suitable, but in our implementation the smallest allowable coefficient is chosen.

this prediction, and we can observe that the four-receiver case has a slightly higher proportion of uncoded packets compared with the eight receiver case. This can be attributed to the fact that the smaller the number of receivers, the lower the probability that additional packets need to be included in each transmission.

Coding scheme B is shown in Fig. 3.9 to have a coefficient-based delivery probability that is significantly higher than coding scheme A and decays far more slowly as a function of the effective Markov state. The coefficient-based delivery probability for the eight-receiver case, which has a smaller fraction of uncoded packets, is somewhat less than its four-receiver counterpart. From Fig. 3.10 we can observe that the higher coefficient-based delivery probabilities of coding scheme B result in significantly better delivery delay compared with both coding scheme A and the RLNC-I scheme. The improvements are especially notable at high addition rates, with an almost threefold improvement in the delivery delay compared with the leader state delivery delay.

We can give an intuitive explanation for the link between coding sparsity and a higher probability of coefficient-based delivery. If a large fraction of undelivered packets are already decoded, this effectively reduces the size of the system of equations corresponding to unknowns in the receiver's buffer. If the transmitted combination is itself sparse, then there is a good probability that its few nonzero elements are those previously decoded by the receiver. Where the elements corresponding to other receivers are already known, the sender will ensure that the transmitted combination allows the delivery of the receiver's next needed packet.

3.7 Conclusion

In this chapter, we analysed the throughput delay performance of coding schemes A, B and RLNC-I under the Bernoulli rate control scheme. Using the receivers' Markov states, we were able to distinguish three categories of packet delivery: zero state delivery, leader state delivery and coefficient-based decoding.

The three transmission schemes we studied were determined to have the same zero state delivery characteristics, and a Markov chain model was used to obtain an accurate estimate of the throughput delay performance under zero state delivery. The Markov chain model was then used to analyse the leader state delivery delay. Although cross receiver dependencies meant that an exact characterisation was not possible, we were able to highlight the dependency of leader state delivery on the number of receivers, the receivers' channel rates and the load factor ρ . Leader state delivery was found to contribute minimal benefits to the delivery delay performance if either larger number of receivers were present, or the load factor was not too close

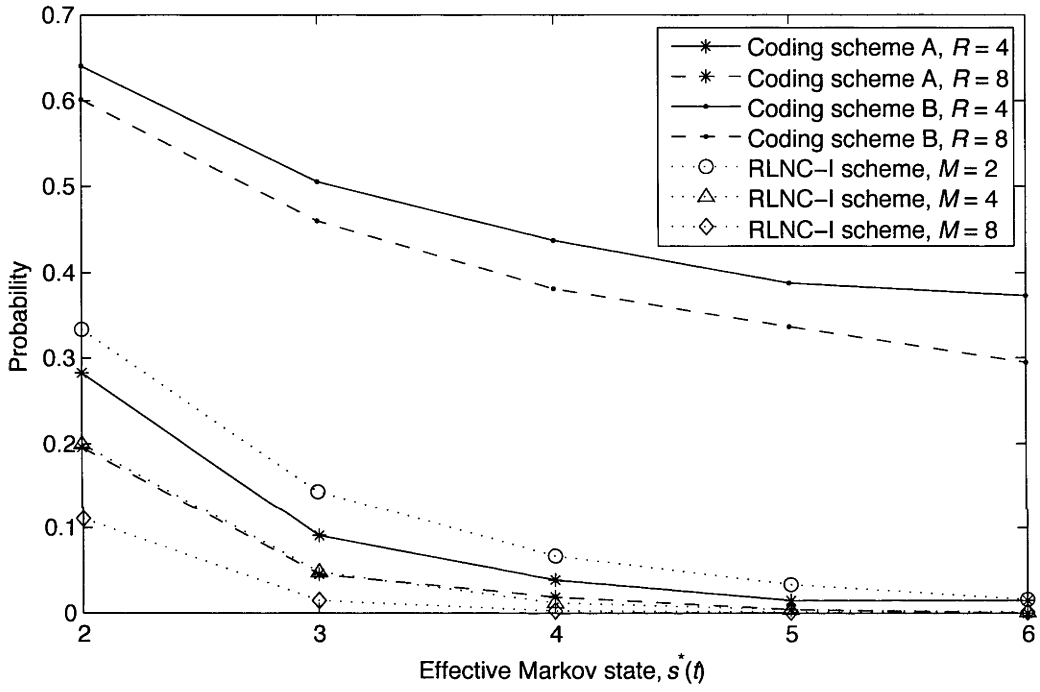


Figure 3.9: Simulated probability of coefficient-based delivery as a function of Markov state, for Bernoulli transmission schemes with addition rate $\lambda = 0.7$. Probabilities are normalised over coefficient-based deliverable timeslots. The field size $M = R$ (the number of receivers), and channel rate $\mu = 0.8$.

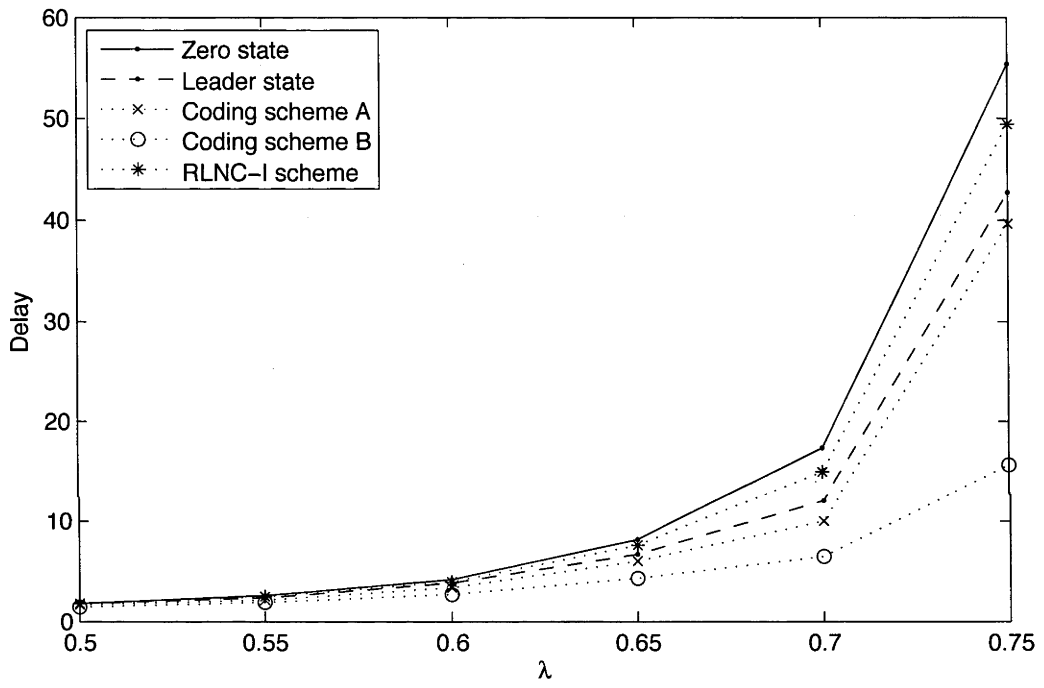


Figure 3.10: The average delivery delay of coding schemes A, B and RLNC-I under the Bernoulli rate control scheme with addition rate λ . Zero and leader state delivery delays are included for comparison. $R = 4$ receivers, channel rate $\mu = 0.8$.

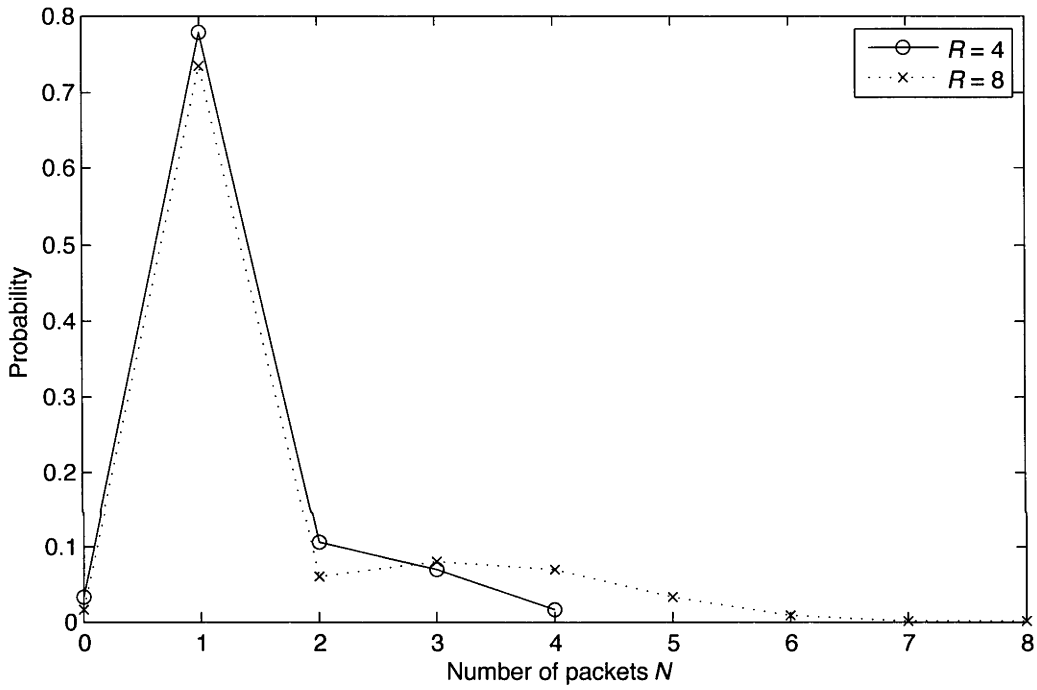


Figure 3.11: For the Bernoulli rate control scheme with coding scheme B, the probability that N packets are coded into a sender transmission, where $N = 1$ corresponds to an uncoded transmission. $R = 4$ and 8 receivers, addition rate $\lambda = 0.7$, and channel rate $\mu = 0.8$. Note that, if the transmission queue is empty, no packets will be coded.

to one.

The occurrence of coefficient-based delivery was found to be a matter of chance for coding schemes A and RLNC-I, depending on both the effective field size the sender was using to code and the current Markov state. As a result, in both these schemes, coefficient-based delivery was shown to contribute a relatively mild improvement to the delivery delay. Interestingly, the use of coding scheme B significantly improved the impact of coefficient-based delivery on the delivery delay performance, particularly at high throughputs. In contrast to coding schemes A and RLNC-I, it exhibited only a mild dependency on the Markov state. The reasons for the notable improvement to the throughput delay performance are not immediately clear, however further investigation of coding scheme B conducted in Chapter 5 will illuminate some of the mechanisms behind this behaviour.

The insights gained from this chapter open up two interesting avenues of questioning. Firstly, now that the relationship between the Markov state and the delivery delay is understood, *is it possible to manipulate the receivers' Markov states to elicit improved delivery delay performance?* The design of such schemes will be studied in Chapter 4. Secondly, it was observed that, at high transmission rates coding scheme B was able to maintain reasonably good throughput delay performance – despite obtaining very little assistance from zero or leader state delivery. With this being the case, one might then ask the question: *is it possible to implement a transmission scheme that does not rely on the complete information property at all?* We will devise two such transmission schemes, and analyse their performance in Chapters 5 and 6.

Dynamic rate control

In this chapter, we build on the insights gained from our Chapter 3 transmission scheme analysis to take a first step in realising a dynamic tradeoff between throughput and delivery delay in a wireless network coded broadcast system. We introduce two new rate control schemes. The *dynamic rate control scheme* improves on the throughput delay performance achieved by proposed transmission schemes in the literature. The *queue threshold* scheme, a variant of the dynamic rate control scheme, is designed to maintain good throughput delay performance, while reducing the amount and quality of the feedback required.

Using a predictive model based on the receivers' Markov states studied Chapter 3, we derive a decision metric that dynamically adjusts the sender's transmission rate to effectively balance the competing needs of throughput and delay performance. Surprisingly, taking this approach leads us to the simple conclusion that the sender should regulate its addition rate based on the total number of undelivered packets stored at the receivers. We show that despite its simplicity, the dynamic rate control scheme results in noticeably improved throughput delay performance compared with existing schemes in the literature.

We then introduce the queue threshold scheme, which is an RLNC adaptation of the dynamic rate control scheme designed to function with smaller volume of feedback. We demonstrate that the queue threshold scheme results in better throughput delay performance when compared with other RLNC transmission schemes in the literature, requires less feedback for the equivalent throughput delay performance compared with the well studied block coding approach, and is able to tolerate both lossy and delayed feedback from the receivers.

4.1 Introduction

High throughput, reliability and low delay are three key performance measures for a number of applications. One such example is video broadcasting where the receivers

not only wish to watch live video, but also want to keep a high quality copy for later use. This might include the broadcast of a lecture or conference recording, or perhaps simultaneous download and viewing of a purchased movie. Other applications include the broadcast of common information in multiplayer gaming, where users' actions must be logged in order, as well as certain scientific or mission-critical applications with low delay requirements.

One of the issues associated with achieving good throughput and delay performance for such applications is the problem of managing the *throughput delay tradeoff*. Recall that higher throughputs generally come at the cost of higher the delivery delay, since at higher throughput receivers are presented with less opportunities to decode. Much work [4, 59, 66, 71, 74, 82, 84] has been done exploring the problem of obtaining good performance within the limitations imposed by the throughput delay tradeoff.

The most common approach to obtaining good performance has been to bound one of these two parameters, then focus on improving the performance of the other. For example, *instantaneously decodable* network coding schemes such as [59, 66] select transmissions to guarantee the immediate decodability of any new information, and attempt to maximise the number of receivers who obtain innovative information. *Hard deadline* schemes such as [46, 82] set a fixed deadline by which packets must be decoded, and attempt to maximise the number of packets decoded at the receivers by the deadline. *Throughput optimal* schemes such as [71, 74] work with a predetermined throughput, and attempt to achieve good delay performance within this constraint.

While there has been much work studying throughput and delay performance in these contexts, to date there has been little work on the middle ground, designing transmission schemes that are neither throughput nor delay constrained. Therefore the question remains:

Is there any systematic way to balance the competing goals of good throughput and delay performance?

Another aspect of performance that must be considered in practical applications is the use of feedback. Feedback provides the sender with valuable information about the status of the receivers, and is generally required to achieve good throughput delay performance. However, in reality, there may be limitations on both the quantity and quality of feedback available. Feedback may be limited by the available bandwidth, and can be lost over the wireless medium, or through collisions between feedback originating from different receivers. In applications such as satellite communications, where large round trip times may exist, large feedback delays may be experienced.

A drawback of a number existing network coded schemes in the literature such

as [4, 47, 59, 60, 65, 66, 71, 74] is that they do not consider the practical limitations of feedback. In many of these proposed network coding schemes, it is assumed that feedback is both lossless and delay free, and often a significant volume of feedback is required by the sender to make the appropriate choice of transmission. The design of network coding schemes that are performant under degraded or reduced volume of feedback is an active area of research [17, 35, 51, 58, 67, 68, 77, 80].

4.1.1 Contributions and related work

In this chapter, we design a high throughput, low delay transmission scheme for wireless broadcast transmission. Our work differs from [8, 10, 62] in that we assume that applications do not tolerate packet losses. We introduce the novel concept of *dynamic rate adaptation*, where the sender adapts its transmission rate to dynamically achieve a balance between the conflicting requirements of throughput and delay performance. A related idea is considered in [82], where the block size is chosen to maximise the number of packets delivered to all receivers by a hard deadline. Our dynamic rate control scheme is shown to have noticeably improved throughput delay performance compared with existing transmission schemes in the literature [4, 71, 74].

To meet the low feedback requirements of many practical applications, we design a low feedback variant of the dynamic rate control scheme, which we call the queue threshold scheme. We demonstrate that the queue threshold scheme can be adapted to reduce the feedback volume to an amount comparable with block coding, with only a minimal effect on the throughput delay performance. We determine a simple upper bound on the delay that is a reasonable estimate of the throughput delay performance, and demonstrate that for a given parameter setting, the delay performance of the system remains relatively steady under changes to the receivers' channel rates. We will also demonstrate that this transmission scheme performs reasonably well under both lossy and delayed feedback.

The robustness of the queue threshold scheme to both lossy and delayed feedback makes it suitable for practical implementation. It has already been demonstrated in [70] that network coding schemes can be adapted to fulfil the requirements of TCP. Although in this chapter we will restrict our discussion to a single-hop wireless broadcast network, the rate control principles of our proposed approach also have the potential to be applied to TCP over multi-hop networks.

4.2 System Model

The system model of this chapter is identical to that studied in Section 3.2. For the sake of completeness, it is repeated here in full. A single sender aims to transmit a backlogged set of data packets $\mathbf{p}_1, \mathbf{p}_2, \dots$ in the correct order to a set of R receivers. Time is slotted, denoted by $t = 1, 2, \dots$, and the sender can broadcast at the rate of one network coded packet per time slot. The receivers are connected to the sender via independent packet erasure channels, where erasures occur independently at each time slot. These packet erasure channels have a *channel rate* of μ , so that on average each receiver obtains μ of the broadcast transmissions.

Receivers store the packets they receive in a buffer, and use Gaussian elimination to decode the original packets. The receivers transmit feedback in the form of *notifications* to the sender, updating it on their progress in receiving the packets. The sender uses this feedback to dynamically adjust its transmissions, with the aim of allowing receivers to achieve better throughput delay performance. In Section 4.3 we assume that all notifications are received instantaneously and without loss, while in Section 4.4 we will also consider the effects of notification erasures and delay on the receivers' performance, as well as methods to reduce the volume of notifications transmitted by the receivers.

4.2.1 Performance measures

Three performance measures will be used in this chapter to compare the transmission schemes studied in this chapter: the throughput, delivery delay and notification frequency. For quick reference, their definitions are listed here. The throughput delay performance will be considered throughout the chapter, while the notification frequency will be studied in Section 4.4.

DEFINITION 4.1: The *throughput* is the average number of packets delivered per time slot, across receivers.

DEFINITION 4.2: A packet $\mathbf{p}_n$ is said to be *delivered* to a receiver if that receiver has already decoded all packets $\mathbf{p}_1, \dots, \mathbf{p}_n$. Otherwise, $\mathbf{p}_n$ is said to be *undelivered* to that receiver.

DEFINITION 4.3: The *delivery delay* of a transmission scheme is measured as the average number of time slots between any packet $\mathbf{p}$ becoming available for transmission, to the time it is delivered to each of receivers.¹

¹Note that, like Chapter 3, the delivery delay is measured from the time a packet first enters the transmission queue to the time it is delivered at the receiver.

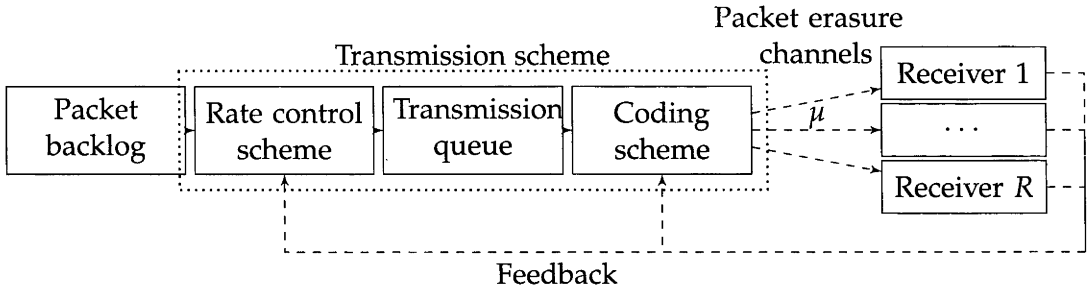


Figure 4.1: A block diagram of the components of a transmission scheme.

DEFINITION 4.4: The *notification frequency* is the proportion of time slots in which a receiver transmits a notification to the sender, averaged across receivers.

4.2.2 Transmission model

In this chapter we will use the general transmission model illustrated in Fig. 4.1, the details for which can be found in Section 2.4. Recall that we divide a network coded system into a number of components:

- A *packet backlog* that contains an infinite backlog of all packets $p_1, p_2, \dots$ waiting to be transmitted;
- A *rate control scheme* that is responsible for determining $a(t)$, the number of packets that should be *added* into the transmission queue from the backlog at each time slot;
- A *transmission queue* that holds all packets available for transmission;
- A *coding scheme* that is responsible for determining which network coded combination of the packets in the transmission queue should be transmitted;
- *Feedback*, transmitted by the receivers in the form of notifications, that is used to inform the rate control and coding schemes.

In this chapter we will compare the performance of different transmission schemes by substituting different rate control schemes, coding schemes and feedback mechanisms into this transmission model. The exact schemes we will use will be discussed in the relevant sections.

4.3 Dynamic rate control scheme

We now aim to address the question: *how can we use rate control to improve the throughput delay performance of a network coded system?* We begin by using the Markov state model that we introduced in Chapter 3 to estimate the short term throughput and delay performance of the system. A predictive model is formulated to determine how the packets added into the transmission queue affect the throughput and delay performance. This model is used to dynamically determine the best opportunities for new packets to be added. Using this decision metric, we will demonstrate that improved throughput delay performance can be achieved, compared with existing rate control schemes from the literature.

4.3.1 Dynamic delivery model

To design a dynamic rate control scheme, we need a model to predict the effect that the rate control scheme's decisions will have on the short term throughput and delay performance of the system. Since the dynamic nature of the rate control scheme means that there is no straightforward way to predict when the sender will choose to add or wait, we instead turn to the Bernoulli rate control scheme that was analysed in Chapter 3. By modeling the dynamic rate control decisions as a Bernoulli process, we are essentially determining the expected behaviour of the system as if we knew nothing about its future actions, other than the average throughput achieved by the system.

Using this idea, we introduce the *dynamic delivery model*, which we will use to estimate the receivers' short term throughput and delivery delay performance. Since the sender only transmits at a rate of one packet per time slot, we similarly restrict our dynamic rate control to add at most one packet to the transmission queue at each time slot. If a packet is added to the transmission queue, $a(t) = 1$ and we say that the sender made an *add decision*. If not, $a(t) = 0$, and the sender made a *wait decision*. We model the add and wait decisions of the sender by the random Bernoulli process,

$$\begin{aligned}\Pr(a(t) = 1) &= \lambda_{\text{est}}, \\ \Pr(a(t) = 0) &= 1 - \lambda_{\text{est}},\end{aligned}\tag{4.1}$$

where λ_{est} is the *estimated addition rate* of the system. We will discuss how this value is determined in Section 4.3.2.5, but for now we treat λ_{est} as a fixed parameter in our model.

In our dynamic delivery model, we assume that zero state delivery is the only method by which packets can be delivered to the receivers. In other words, a receiver

Rx_r delivers the packets in the transmission queue iff its Markov state $s_r(t) = 0$.² This is a somewhat pessimistic performance estimate, as leader state and coefficient-based delivery, discussed in Chapter 3, sometimes allow packets to be delivered prior to meeting this condition. However, since it is difficult to predict how rate control will affect leader state and coefficient-based delivery, the zero state delivery delay provides a tractable upper bound on the expected delivery delay.

In many situations, zero state delivery provides a reasonable estimate of the delivery delay. For moderate to large values of R , leader state delivery has a relatively minor impact, and under coding schemes A and RLNC, the effects of coefficient-based delivery on the delay performance were relatively insignificant. Under imperfect feedback conditions, the contributions from leader and coefficient-based delivery are further diminished. However, the zero state delivery model is not always accurate. If coding scheme B is used, the addition rate $\lambda \rightarrow \mu$, or R the number of receivers is small, then leader state and coefficient-based delivery can still have a significant impact on the actual delivery performance.³

4.3.2 Decision metric

Now that the dynamic delivery model has been established, we can outline how add and wait decisions are determined in the dynamic rate control scheme. Unlike the network coding schemes of Chapter 3, where packets were added to the transmission queue with probability $\lambda < \mu$, in the dynamic rate control scheme we are about to introduce, a *decision metric* M is used to determine whether to add or wait. This allows the sender to dynamically adjust its addition rate based on the delivery performance of the receivers, resulting in better throughput-delay performance. We now describe how the sender will calculate M , by weighing the impacts of add and wait decisions on both the throughput and delivery delay performance.

4.3.2.1 Throughput performance

Our first task is to measure the effect of an add or wait decision on a receiver's throughput performance. For this purpose we define two variables, $P_T^A(r)$ and $P_T^W(r)$, which measure changes to a receiver r 's throughput performance under add and wait decisions respectively. Every time the sender waits, there is no increase in the total

²Recall from Chapter 3 that a receiver's Markov state is given by the difference between the number of packets that have been added to the transmission queue, and number of innovative packets that have been received by a receiver.

³Although in this section we shall use the zero state delivery model to make predictions about the receivers' throughput and delay performance, the simulation results of Section 4.3.5 measure the observed performance from the combination of all three delivery methods.

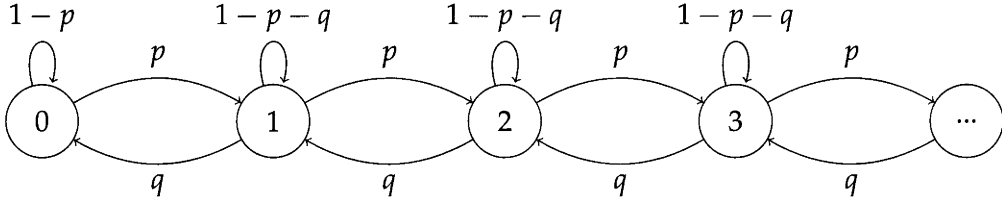


Figure 4.2: A Markov chain describing transitions in the Markov state of a receiver r . $p = (1 - \mu)\lambda_{\text{est}}$, and $q = \mu(1 - \lambda_{\text{est}})$, where λ_{est} is the estimated addition rate, and μ is the channel rate.

throughput. Therefore,

$$P_T^W(r) = 0. \quad (4.2)$$

On the other hand, adding increases the total throughput by one packet, giving

$$P_T^A(r) = 1. \quad (4.3)$$

4.3.2.2 Expected time to zero state delivery

To measure the delivery delay performance, we compare the expected delivery delay for a receiver in Markov state k , under add and wait decisions. Our first task is to calculate the average time it takes for a receiver in Markov state k to zero state deliver. This is equivalent to finding the average time to move from state k to 0 for the first time, under the Markov chain in Fig. 4.2.⁴

Let E_k be the expected *time to zero state*, i.e. expected number of time steps it takes for a receiver starting at state k to reach 0 for the first time. Since our Markov chain is positive recurrent, we know that E_k exists.

The receiver's journey from state k to 0 can be considered as a series of traversals through the Markov chain from k to $k - 1$ for the first time, $k - 1$ to $k - 2$ for the first time, and so on. Because the transition probabilities between adjacent states in the Markov chain are the same for all $k > 0$, the average time required for each traversal is the same. Therefore, the expected time to zero state starting from k can be expressed as

$$E_k = kE_1. \quad (4.4)$$

Studying the Markov chain of Fig. 3.3, there are three possible transitions at each time step. Starting at state k , the receiver's Markov state may increment, decrement or remain the same, with the probabilities listed in Table 3.1 adapted to λ_{est} , as calculated in (4.13). Each state transition corresponds to a one-unit increase in delay.

⁴This is essentially the same Markov chain that was studied in Chapter 3.

From this information, we can establish the relationship between the time to the zero state for different values of k :

$$E_k = 1 + qE_{k-1} + pE_{k+1} + (1 - p - q)E_k. \quad (4.5)$$

Substituting $k = 1$, we obtain

$$E_1 = 1 + qE_0 + pE_2 + (1 - p - q)E_1. \quad (4.6)$$

As a starting condition, $E_0 = 0$, since in this case the receiver is already at $k = 0$. Additionally from (4.4), $E_2 = 2E_1$. Substituting in these values, we obtain

$$E_1 = \frac{1}{q - p}. \quad (4.7)$$

Therefore, using (4.4) and the values in Table 3.1, we obtain the expected time for a receiver to move from Markov state k to the zero state,

$$E_k = \frac{k}{\mu - \lambda_{\text{est}}}. \quad (4.8)$$

4.3.2.3 Delivery performance

We now introduce two values $P_D^A(r)$ and $P_D^W(r)$, which quantify changes to the delivery delay performance under add and wait decisions respectively. We define this to be the expected time to zero state under add and wait decisions. For a receiver r starting in state k_r , adding will result in one of two possible Markov states, $k_r + 1$ and k_r , depending on whether the receiver experiences an erasure or not. Therefore, the expected time to zero state under adding is given by

$$\begin{aligned} P_D^A(r) &= \bar{\mu}E_{k_r+1} + \mu E_{k_r} \\ &= \frac{k_r + \bar{\mu}}{\mu - \lambda_{\text{est}}}. \end{aligned} \quad (4.9)$$

Similarly, waiting will result in receiver r moving to state k_r or $k_r - 1$. Therefore the expected time to zero state under waiting is given by

$$\begin{aligned} P_D^W(r) &= \bar{\mu}E_{k_r} + \mu E_{k_r-1} \\ &= \frac{k_r - \mu}{\mu - \lambda_{\text{est}}}. \end{aligned} \quad (4.10)$$

4.3.2.4 Benefits of adding and waiting

For a given receiver r , we now calculate the performance benefits of adding and waiting.

Lower times to zero state are desirable, therefore we multiply values $P_D^A(r)$ and $P_D^W(r)$, which measure the time to zero state, by a factor of -1. We also observe that when a receiver moves to the zero state, it will deliver *all* the packets in its buffer. Therefore, we also scale $P_D^A(r)$ and $P_D^W(r)$ by the number of undelivered packets $u(r)$ currently stored in its buffer. The information provided by $u(r)$ is particularly important because it tells us how many packets will have their delivery delays affected by the sender's decision, and therefore how great an impact the sender's decision will have on the delay performance.

On the other hand, throughput and delay may not be of equal importance. Therefore we scale $P_T^A(r)$ and $P_T^W(r)$, which measure throughput, by a weighting factor f , which determines the relative importance of one unit of throughput, compared with one unit of delay. This single free parameter f is important because it allows us to study the throughput delay sensitivity of the system. This idea is used in other work such as [84], where the parameter p effectively dictates the balance between the throughput and delivery delay.

Combining our results, we obtain two values $B_A(r)$ and $B_W(r)$, which we define as the measured benefits of adding and waiting respectively,

$$\begin{aligned} B_A(r) &= fP_T^A(r) - u(r)P_D^A(r) \\ &= f - u(r) \frac{k_r + \bar{\mu}}{\mu - \lambda_{\text{est}}} \end{aligned} \quad (4.11)$$

$$\begin{aligned} B_W(r) &= fP_T^W(r) - u(r)P_D^W(r) \\ &= -u(r) \frac{k_r - \mu}{\mu - \lambda_{\text{est}}}. \end{aligned} \quad (4.12)$$

4.3.2.5 Decision metric

We now derive the decision metric which will be used to determine whether the sender makes add or wait decisions.

We first need to establish the value of λ_{est} . We take this to be the average observed addition rate, upper bounded by μ , the maximum possible delivery rate,

$$\lambda_{\text{est}} = \min(A(t)/t, \mu - \epsilon), \quad (4.13)$$

where

$$A(t) = \sum_{i=1}^t a(i) \quad (4.14)$$

and ϵ is a small, positive value, taken to be $\epsilon = 0.0001$ in our simulations.⁵ This is a practical way to choose λ_{est} for our dynamic delivery model, because it allows the sender to dynamically adjust its transmission rate to accurately reflect its throughput and delay performance priorities. The performance of this system under inaccurate values of λ_{est} will be discussed in Section 4.3.3.2.

For a single receiver r , the difference in performance between adding and waiting is given by

$$M(r) = B_A(r) - B_W(r) \quad (4.15)$$

$$= f - \frac{u(r)}{\mu - \lambda_{\text{est}}}. \quad (4.16)$$

If $M(r) > 0$, it indicates that adding is more beneficial to receiver r 's performance than waiting. It is interesting to note that $M(r)$ does not depend on the actual Markov state k_r of the receiver. This comes about because in (4.4), the expected time to zero E_k is linearly dependent on k . Adding a new packet merely increments the Markov state by 1, compared with waiting. Therefore the difference in performance between adding and waiting does not depend on the receiver's Markov state k .

Our decision metric is therefore given by the sum of the receivers' performance differences,

$$\begin{aligned} M &= \sum_{r=1}^R M(r) \\ &= Rf - \frac{\sum_{r=1}^R u(r)}{\mu - \lambda_{\text{est}}}. \end{aligned} \quad (4.17)$$

If $M > 0$ then the sender decides to add a new packet to the transmission queue. Otherwise, it waits. It is important to note that the $\lambda_{\text{est}} < \mu$ constraint set by (4.13) ensures that the weighting of $u(r)$ will always be negative.

4.3.3 Feedback variables

There are two feedback variables that affect the decision metric, M : the total number of undelivered packets stored at the receivers, $\sum_{r=1}^R u(r)$, and the delivery rate estimate λ_{est} . We briefly discuss the impact of each variable on the sender's decision.

⁵It is of course reasonable to assume that a receiver's delivery rate λ_{est} will be less than the channel rate μ , but the reason for this explicit upper bound will become apparent in (4.17).

4.3.3.1 Number of undelivered packets $\sum_{r=1}^R u(r)$

Let us assume for now that λ_{est} is constant. When this is the case, $u(r)$ becomes the only variable required to determine M . From (4.17), we can observe that there is a threshold T_U for the number of undelivered packets stored among the receivers above which, $M < 0$ and the sender will wait, but below which the sender will add. The threshold value can be found by solving the equation

$$Rf - \frac{T_U}{\mu - \lambda_{\text{est}}} = 0, \quad (4.18)$$

yielding the solution

$$T_U = Rf(\mu - \lambda_{\text{est}}). \quad (4.19)$$

Therefore the sender's decision strategy can be equivalently phrased as follows:

If the total number of undelivered packets $\sum_{r=1}^R u(r)$ stored at the receivers is greater than or equal to T_U , given in (4.19), then wait. Otherwise, add a new packet to the transmission queue.

The greater the value of T_U , the longer the sender will spend adding new packets to the transmission queue, before waiting to reduce the number of undelivered packets at the receivers. This means that greater values of T_U will generally result in higher throughput, but also higher delivery delays.

4.3.3.2 Addition rate estimate, λ_{est}

Here we investigate the consequences of an inaccurate addition rate estimate, λ_{est} . Let us assume that there is some value λ , which is the correct addition rate for the system, and let $\lambda_{\text{est}} = \lambda + \Delta$ be the (possibly inaccurate) estimate based on the observed addition rate so far. The effect of the discrepancy Δ on the undelivered packet threshold of (4.19) is

$$T'_U = Rf(\mu - \lambda - \Delta).$$

Notice that this only differs from (4.19) by a Δ term. The result is that the greater λ_{est} is compared with λ , the smaller T_U will be, and vice versa.

Interestingly, this results in a negative feedback loop where if λ_{est} is too high, the addition rate will drop below λ_{est} , in turn reducing the measurement λ_{est} . On the other hand, a low value of λ_{est} will have the opposite effect, raising the addition rate, and therefore the estimate λ_{est} . The end result is a steady addition rate, where

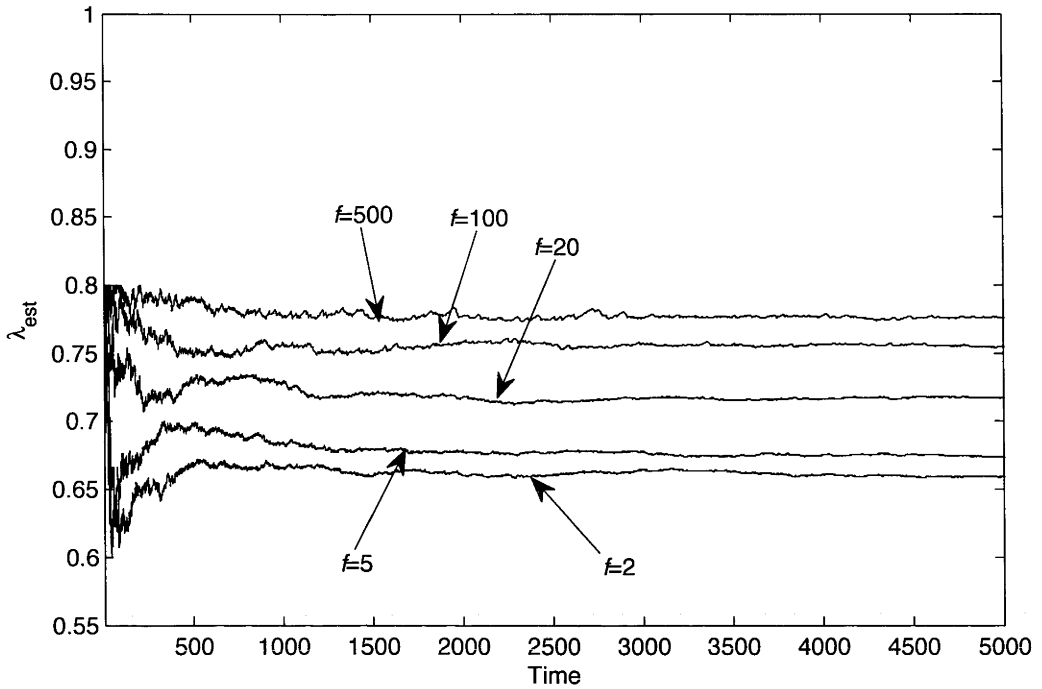


Figure 4.3: The sender's estimated addition rate λ_{est} as a function of time, for the weighting factors f listed in the figure. The simulation was implemented with $R = 4$ and $\mu = 0.8$.

fluctuations will to some degree be corrected by the system. It can be observed in Fig. 4.3 that, in line with our analysis, λ_{est} quickly converges upon a steady state value. (It is worth noting, however, that at large values of t , this can in part be attributed the convergence of $\frac{A(t)}{t}$.)

4.3.4 Full feedback schemes

We now summarise the *full feedback schemes* we will use to compare the performance of the dynamic rate control scheme. Under full feedback schemes, the sender requires that feedback be transmitted by the receivers at every time slot, which is assumed to be received without error or delay. We make use of coding scheme B for the coding component, since of the three coding schemes we studied in Chapter 3 it had by far the best delay performance. Coding scheme B is paired with the Bernoulli and delay threshold rate control schemes, as well as the dynamic rate control scheme we derived in Section 4.3.2. More detail on these components can be found in Chapter 2.5.

4.3.4.1 Coding scheme B

Under coding scheme B, the sender transmits a minimal combination based on the oldest undecoded packet of each receiver. The sender lists these oldest undecoded packets and their corresponding receivers, then beginning with the newest packet in the list, it adds in older packets only if the receiver(s) that correspond to them would not otherwise receive an innovative packet. Coding scheme B relies on full feedback from the receivers for accurate information about the packet combinations stored at the receivers.

4.3.4.2 Bernoulli rate control scheme

Under this rate control scheme, the add decision $a(t)$ is determined by a Bernoulli process with *addition rate* λ , so that the sender will add with probability

$$\Pr(a(t) = 1) = \lambda. \quad (4.20)$$

4.3.4.3 Delay threshold scheme

The delay threshold rate control scheme is taken from [4]. This scheme operates under two modes, which we call *start* and *stop*. By default the sender is set to start mode, where it adds whenever one of the receivers has decoded all packets in the transmission queue. However, if any of the packets inside the transmission queue have been present for more than some threshold T_D number of time slots, the sender switches to stop mode. In this case the sender waits, and the coding block transmits uncoded copies of the expired packet(s). Once all packets remaining in the transmission queue are less than T_D time slots old, the sender reverts back to start mode.

4.3.5 Performance comparison

We now compare the throughput delay performance of our dynamic rate control scheme against the Bernoulli and delay threshold rate control schemes from the literature. The throughput delay performance of the three rate control schemes paired with coding scheme B is given in Fig. 4.4. The results, shown for 4 and 8 receivers, are discussed here.

4.3.5.1 Bernoulli rate control scheme

As expected, the Bernoulli rate control scheme exhibits the worst throughput-delay performance of the three rate control schemes. As the throughput increases, so does

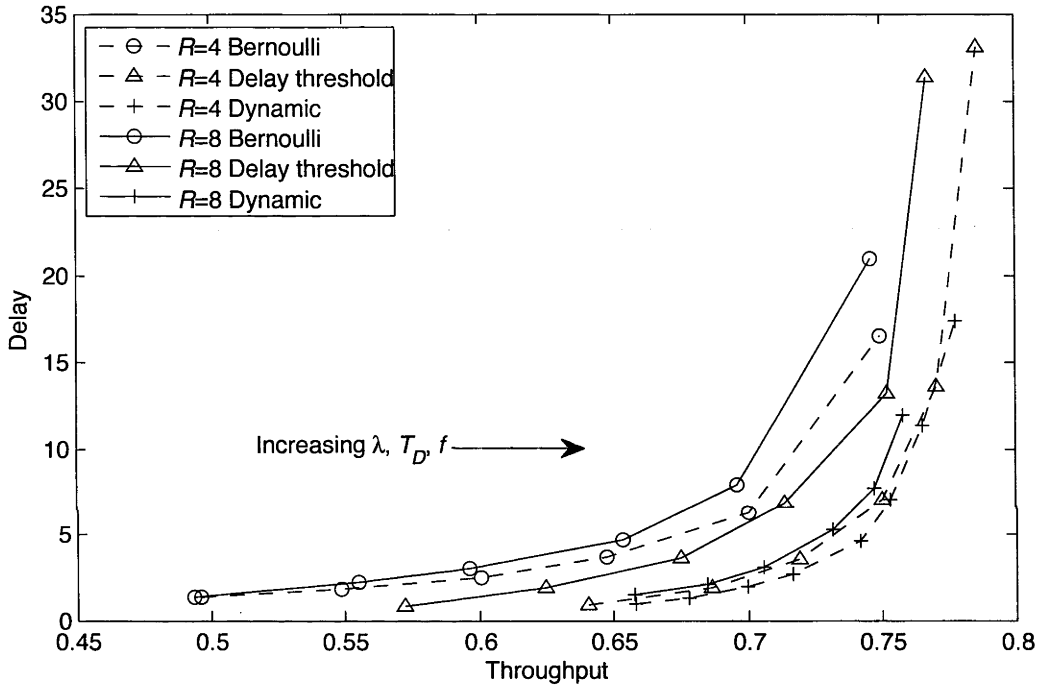


Figure 4.4: Throughput-delay performance for the coding scheme B transmission schemes. From left to right, the parameters for each transmission scheme are: Bernoulli: $\lambda = 0.5, \dots, 0.75$. Delay threshold: $T_D = 2, \dots, 100$. Dynamic: $f = 2, \dots, 500$. For all transmission schemes $R = 4$ or 8 , as indicated in the legend, and $\mu = 0.8$.

the average delivery delay, with the 4-receiver case performing marginally better than the 8-receiver case.

4.3.5.2 Delay threshold rate control scheme

The delay threshold rate control scheme performs significantly better than the Bernoulli rate control scheme. The strategy of reducing the rate when one or more receivers is experiencing significant delays greatly improves the delivery delay. However, the delay performance for the 8-receiver case is significantly worse than the 4-receiver case. This is most likely caused by the transmission inefficiency of the stop mode as the number of receivers increases, as we now explain.

In stop mode, the sender transmits an uncoded packet to allow the worst performing receiver to deliver its next needed packet(s), thus improving its delivery delay. However, in doing so every other receiver will incur a one-unit throughput penalty, since the uncoded transmission will not provide them with any innovative information. With larger numbers of receivers, this penalty can become quite significant.

4.3.5.3 Dynamic rate control scheme

The dynamic rate control scheme further improves upon the delay threshold scheme. We can observe in Fig. 4.4 that, as intended, increasing the weighting factor f results in higher throughput, but also higher delays. The real improvement over the delay threshold rate control scheme can be seen in the 8-receiver case, where for the same throughput, the dynamic rate control scheme experiences approximately half the delivery delay of the delay threshold scheme. This can be attributed to the fact it is a fairer, better informed rate control scheme. Unlike the delay threshold scheme, the dynamic rate control scheme does not disproportionately weight the needs of the worst performing receivers. When determining whether to add or wait, the sender weighs the requirements of *all* receivers, as illustrated in (4.17), instead of only the receiver(s) with the worst delay performance.

Furthermore, the dynamic rate control scheme considers both throughput and delay performance when determining whether to add or wait. This is in stark contrast to the Bernoulli rate control scheme, which only attempts to control the throughput, and the delay threshold scheme, for which the addition rate is controlled purely on the basis of the delay performance of the worst receiver. By recognising that the add/wait decision is a tradeoff between throughput and delay performance, our dynamic rate control scheme is able to determine at each time slot which performance measure can be most improved upon under the current circumstances.

4.4 Queue threshold scheme

Although the dynamic rate control scheme can achieve good throughput delay performance, it is unlikely that the assumptions made about lossless, delay free feedback will hold in practice. Additionally, bandwidth is a valuable resource, so it would be advantageous to design a transmission scheme that can reduce the number of time slots in which feedback is required from the receivers. In this section we consider how the dynamic rate control scheme we derived in Section 4.3 might be adapted to perform well under reduced, lossy and delayed feedback.

4.4.1 Queue threshold rate control scheme

We now design the queue threshold rate control scheme by making some modifications to the decision metric of the dynamic rate control scheme, and determine a way to reduce the notification frequency required.

4.4.1.1 Decision metric

The queue threshold scheme follows a similar decision process to (4.17) of the dynamic rate control scheme. Recall from Section 4.3.3.1 that the decisions made by the dynamic rate control scheme could be equivalently determined by testing if the total number of undelivered packets $\sum_{r=1}^R u(r)$ exceeded a threshold value T_U . Instead of depending on $u(r)$, the number of undelivered packets stored at each receiver, the decision process is based on the *queue size* $Q(r)$, the number of packets in the *transmission queue* that have not yet been delivered by the receiver Rx_r . We use the queue size $Q(r)$ for our decision metric as it is simpler to determine from limited feedback than $u(r)$.

We observed in Section 4.3.3.2 that λ_{est} is more or less a constant value. Therefore, we take H to be a constant threshold value determined by the sender. If

$$\sum_{r=1}^R Q(r) > RH, \quad (4.21)$$

then $a(t) = 0$ and the sender waits. Otherwise, $a(t) = 1$ and the sender adds a new packet to the transmission queue. We will discuss the effect of H on the throughput and delay performance later in this chapter.

4.4.1.2 Notifications

Receivers transmit a notification when they have decoded all of the packets in the transmission queue. Whenever the sender receives a notification, it resets the corresponding receiver's queue size to $Q(r) = 0$. The receivers' notifications allow the sender to manage the transmission queue size, since the queue will consist of the most recent $\max(Q(1), \dots, Q(R))$ packets.

One advantage of the queue threshold scheme is that it requires relatively little feedback to determine how many of the packets in the transmission queue are still undelivered. Under RLNC, a receiver can generally decode new packets only when the number of innovative transmissions received equals the number of packets in the transmission queue. When this is the case, all packets in the transmission queue are delivered. Therefore it is only necessary for receivers to send a notification to the sender every time they deliver all the packets in the transmission queue.

One disadvantage of this transmission scheme is that, if the receivers deliver packets frequently, it will greatly increase the volume of feedback. However, we reason that, when packets are delivered frequently, only a small number of packets are delivered for each notification. Therefore, these more frequent transmissions have a relatively small impact on the accuracy of $Q(r)$ and the resulting add decisions. Therefore, it is possible to further reduce the amount of feedback by only allowing a receiver to send a notification when the following conditions hold:

- The receiver did not experience an erasure (so that it accurately knows the size of the transmission queue),
- The receiver has decoded all packets in the transmission queue,
- At least $T_{\min}$ time slots have passed since the receiver's last transmitted notification.

Note, however, that reducing the amount of feedback in this fashion also has an impact on the throughput delay performance, since the receivers' queue sizes cannot be determined as accurately. We will observe the effects of reduced feedback later, in Section 4.4.4.

4.4.2 Reduced feedback schemes

We now introduce the *reduced feedback schemes* from the literature, which we will use to compare the performance of our queue threshold scheme. Reduced feedback schemes send notifications at selected time slots, and are to some degree able to cope with the possibility of delayed and lossy feedback. For fair comparison we use

RLNC for the coding scheme, since the feedback requirements of deterministic coding schemes are generally too stringent. The RLNC scheme is paired with three rate control schemes: the block-based and sliding window schemes from the literature, as well as the queue threshold scheme we derived in Section 4.4.1. A summary of these schemes is given below. More detail on these components can be found in Chapter 2.5.

4.4.2.1 RLNC scheme

Under the random linear network coding (RLNC) scheme, the sender transmits a random combination of all packets currently stored in the transmission queue. Coefficients for each packet are selected randomly from $\mathbb{F}_M$, where M is assumed to be sufficiently large to guarantee innovative transmissions will be chosen with high probability.

To obtain more numerically accurate results in a shorter amount of time, we speed up our simulations by making the following assumption.

A receiver decodes its packets iff the number of packets in the transmission queue matches the number of innovative transmissions it has received.

This assumption is accurate if a sufficiently large field size is used [34]. With this being the case, a receiver's ability to decode depends purely on its Markov state. Under RLNC, a receiver can only decode packets when it reaches the zero state, i.e. its Markov state is 0. When this is the case, the receiver delivers all packets in the transmission queue.

4.4.2.2 Block-based scheme

Under the block-based rate control scheme, packets are divided into blocks of size B for transmission. Receivers transmit a notification every time they have decoded all packets in the current block. While receivers are still receiving the current block, $a(t) = 0$ so that the sender does not add new packets to the transmission queue. Once a notification has been received from every receiver, the sender begins transmission of a new block, and sets $a(t) = B$.

4.4.2.3 Sliding window scheme

The sliding rate control scheme only allows the coding of packets within a fixed window of size W , which moves at a predetermined rate V . At each time slot that the window advances, a new packet is added to the transmission queue. If the window size exceeds W , the oldest packet is dropped. Therefore the transmission queue consists of a window of the last W packets.

To ensure that every packet can eventually be decoded, the rate control scheme also ensures that every packet $\mathbf{p}_n$ exiting the window has been *seen*: that is, the receiver knows at least one combination of the form $\mathbf{p}_n + \sum_{i=n+1}^{\infty} \alpha_i \mathbf{p}_i$. Whenever a packet at the end of the window is still unseen by one of the receivers, it transmits a notification to the sender. Whenever the sender receives a notification, it will pause the sliding window, so that no new packets are added. This ensures that every packet $\mathbf{p}_n$ must be seen before the sender continues to advance the window. It is known from [74] that seeing a packet is a sufficient condition for decoding it at a later point in time, hence seen packets can be dropped from the transmission queue.

4.4.3 Throughput delay performance

We first examine the throughput delay performance of all three schemes, under perfect feedback conditions. Fig. 4.5 compares the throughput delay tradeoff under the block coding, sliding window and queue threshold schemes. We discuss these results under the setting $T_{\min} = 1$, where full perfect feedback is available to the sender.

4.4.3.1 Block coding scheme

For small values of B , the receivers experiences low delivery delays, but also low throughputs, since the sender cannot begin transmission of a new block until all receivers have received the current block. Higher values of B result in higher throughputs, but also higher delivery delays, since receivers must wait until they receive the whole block before it can be decoded. Because of these issues, block coding has the worst throughput delay performance out of the three transmission schemes.

4.4.3.2 Sliding window scheme

From our simulations, the window size W does not appear to have a significant effect on the throughput delay performance. Some degradation in the throughput delay performance occurs at low values of W , so we have set $W = 10$ in our simulations. We find that the throughput is simply given by V , the window rate. Higher window rates result in higher delivery delays, since statistically, receivers will reach the zero state less frequently. This is analogous to the Markov state analysis we performed in Chapter 3, for the Bernoulli rate control scheme. The sliding window scheme performs significantly better than the block threshold scheme, as a greater flexibility in the decoding time is permitted by the sliding window. This allows receivers to decode in their own time without affecting the throughput of the other receivers.

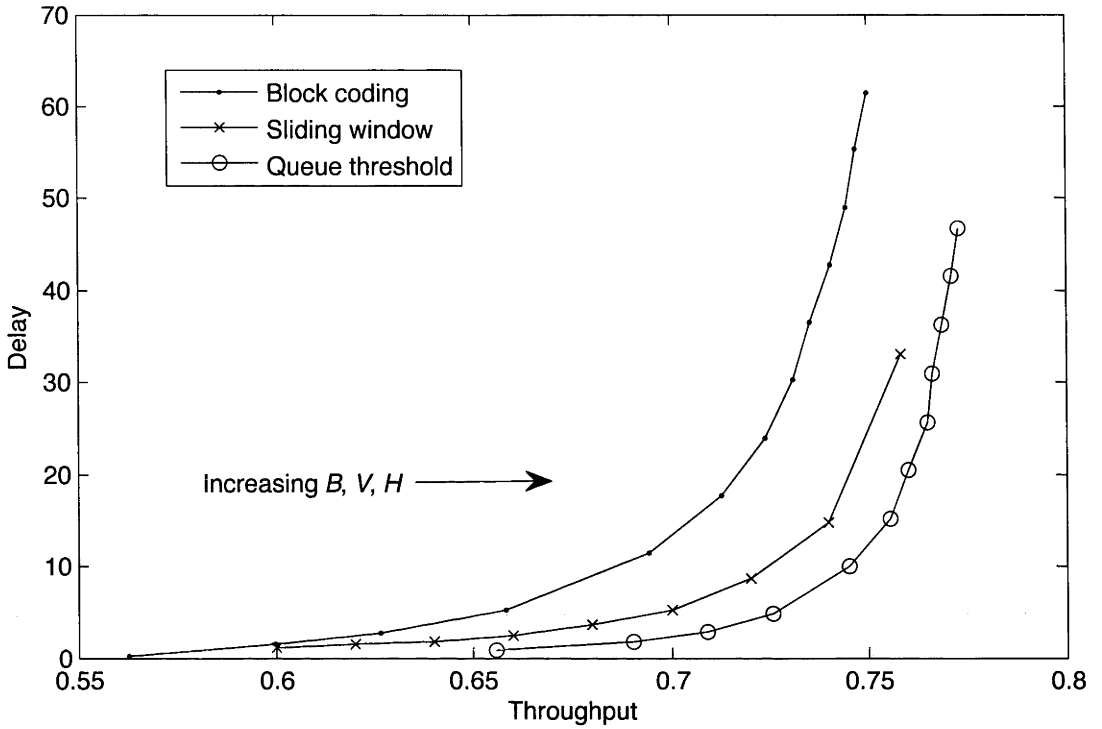


Figure 4.5: Throughput delay tradeoff for the block coding, sliding window and queue threshold schemes. Feedback is assumed to be nonlossy and without delay. For all rate control schemes, $R = 4$, $\mu = 0.8$. Block coding: $B = 1, \dots, 50$. Sliding window: $W = 10$, $V = 0.6, \dots, 0.76$. Queue threshold: $T_{\min} = 1$, $H = 1, \dots, 50$.

4.4.3.3 Queue threshold scheme

At low values of H , the receivers experience both low delays and low throughputs, since only a small number of packets will be added by the sender, before it will wait for a receiver to decode. At higher values of H , both the throughput and the delay increase. The queue threshold scheme further improves upon the sliding window scheme because it ensures that receivers will not accumulate too many undecoded packets. By upper bounding the number of packets in the transmission queue using the threshold value H , we also bound the maximum amount of delay that can be accumulated in the transmission queue at each time slot.

4.4.4 Notification frequency

We now study the notification frequency required by each of the rate control schemes.

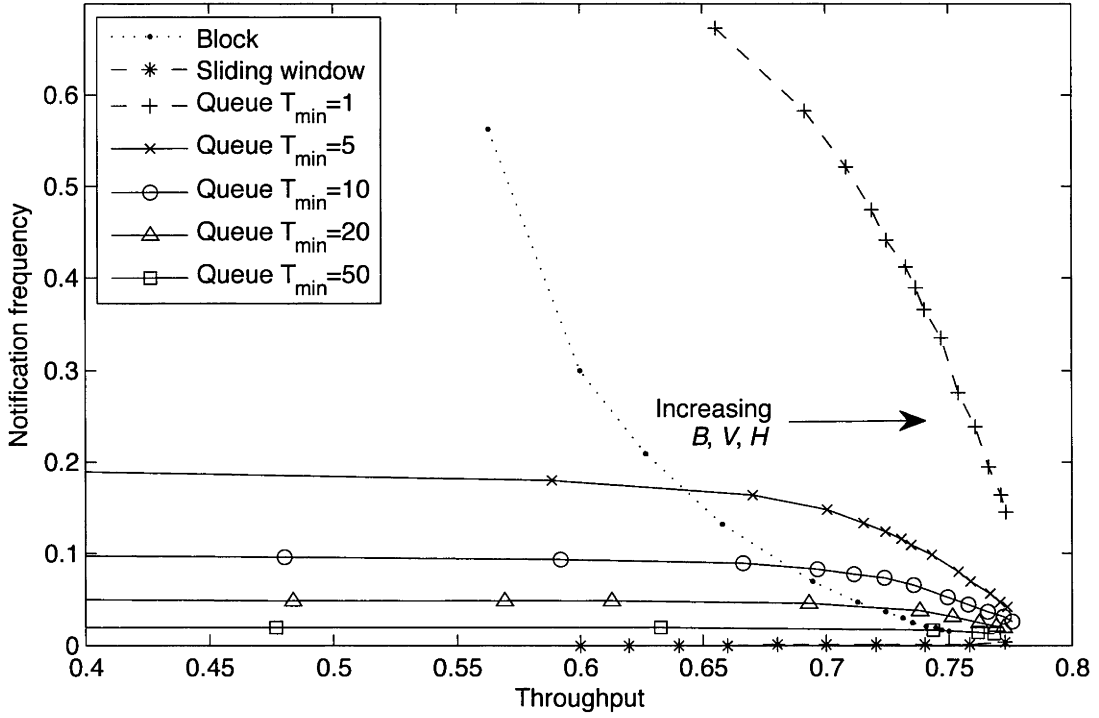


Figure 4.6: The notification frequency of the block coding, sliding window and queue threshold schemes. For all rate control schemes, $R = 4$, $\mu = 0.8$. Block coding: $B = 1, \dots, 50$. Sliding window: $W = 10$, $V = 0.6, \dots, 0.78$. Queue threshold: $H = 1, \dots, 50$; values of $T_{\min}$ are given in the figure.

4.4.4.1 Block coding

Fig. 4.6 shows the notification frequency of the block coding scheme. Since only one notification is required per block, as the block size increases (corresponding to higher throughputs and delays), the notification frequency decreases.

4.4.4.2 Sliding window

Under the sliding window rate control scheme, the window size W may be varied to reduce the notification frequency. If a sufficiently large window size W is used, almost no feedback is sent by the receivers [80]. For example, in Fig. 4.6, at window size of $W = 10$, barely any notifications are required from the receivers. However, the downside of this rate control scheme is that the sender has no way of knowing if the channel rate μ changes. If the channel rate falls to a value $\mu < V$, then the throughput reduces to μ , and the notification frequency will become high. At least one receiver will maintain a Markov state W , which will have an extremely bad effect on the decoding delay.

4.4.4.3 Queue threshold

We now study the effect of reducing the feedback on the throughput delay tradeoff achievable under the queue threshold scheme.

Fig. 4.6 illustrates the notification frequency as a function of throughput, for different values of $T_{\min}$. Under full feedback with $T_{\min} = 1$, the notification frequency is relatively high, although it drops somewhat as the throughput approaches the channel rate μ . We observe that much lower notification frequencies can be achieved by increasing $T_{\min}$.

Although increasing $T_{\min}$ certainly reduces the notification frequency, we must also ensure this does not greatly penalise the throughput delay performance. Figure 4.7 illustrates the throughput delay performance for different values of $T_{\min}$. It is interesting to observe that there is little performance degradation between $T_{\min} = 1$ and $T_{\min} = 10$. However, starting from around $T_{\min} = 20$, we do begin to observe a noticeable impact on the throughput delay performance. However, at high throughputs, even setting $T_{\min} = 50$ does not significantly affect the throughput delay performance. This is true since at high throughputs, the receivers are already expected to decode very infrequently. Therefore, limiting the feedback frequency has little effect on the receivers' throughput delay performance. This tells us that we can significantly reduce the amount of feedback without greatly affecting the throughput delay performance – particularly at high throughputs. For the example given in Fig. 4.7, at low throughputs, setting $T_{\min} = 10$ is sufficient to improve upon the notification frequency of the block coding scheme. At high throughputs, larger values of $T_{\min}$ can be used to maintain a competitive notification frequency.

4.4.5 Upper bounding the delivery delay

Under the queue threshold scheme, we can upper bound the delivery delay that will be experienced for a particular value of H . Let λ be the average throughput of the system. Since H limits the total number of undelivered packets that are permitted in the transmission queue at any given time, it also limits the total amount of delay that can be incurred per time slot. Since this delay is spread over the average of λ packets which enter the queue at each time slot, the delivery delay d can be upper bounded by

$$d < \frac{H}{\lambda}. \quad (4.22)$$

This upper bound is compared with the throughput-delay performance obtained by simulation in Fig. 4.8 for $T_{\min} = 1$, under different channel rates μ . For a fixed value of H , as the throughput increases, the delivery delay decreases, in agreement

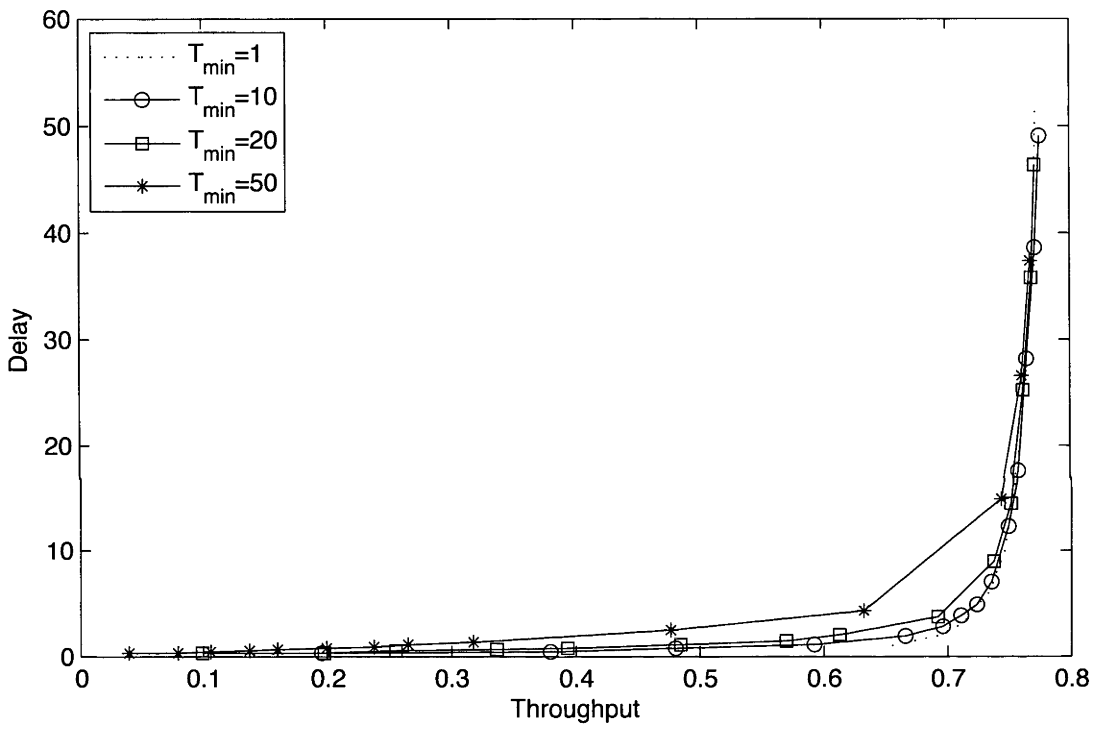


Figure 4.7: Throughput delay tradeoff for the queue threshold scheme, for the values of $T_{\min}$ listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, $H = 1, \dots, 50$.

with (4.22). We observe that the upper bound obtained in (4.22) is reasonably close to the performance observed in our simulations.

The upper bound on the delivery delay obtained in (4.22) still applies for larger values of $T_{\min}$, as reducing the amount of feedback causes the sender to slightly overestimate the queue size. For example, Fig. 4.9 illustrates the delay performance for $T_{\min} = 20$. We can observe that, while the throughput delay estimates still match reasonably well, the upper bound is not as tight as for $T_{\min} = 1$. We should also note that, when $T_{\min}$ is large, only small throughputs are achievable for small values of H . This is because, if the sender has a very small threshold value H , then typically it can only add H packets during the $T_{\min}$ time slots between feedback from the receivers.

4.4.5.1 Stable delivery delays

We now show how it is possible to maintain a stable average delivery delay, even under fluctuating channel conditions. Changes in the channel rate μ will cause the throughput λ of the system to vary over time. We can negate the effects this would have on (4.22) by introducing a feedback-based variable λ_{est} , the estimated throughput of the system.⁶ If λ_{est} is a reasonably accurate estimate of the throughput of the system, we can upper bound the average delivery delay by the value $D_{\max}$. This is achieved by setting the threshold value H to

$$H = \lambda_{\text{est}} D_{\max}. \quad (4.23)$$

4.4.6 Lossy and delayed feedback

We now examine the performance of the queue threshold scheme under lossy or delayed feedback. Again it is expected that the queue threshold scheme should still perform reasonably well, since feedback degradations will only affect the accuracy of the queue size estimates made by the rate control scheme.

4.4.6.1 Feedback delay

We firstly consider the possibility that notifications are received by the sender after a fixed delay. Say that there is a D time slot delay between the transmission of a notification and its reception, where the value of D is known by the sender. Then whenever a notification is received, the sender updates the corresponding receiver's

⁶We only assume here that λ_{est} is a reasonably accurate predictor of future delivery behaviour. How λ_{est} is determined in practice would depend on the properties of the channel.

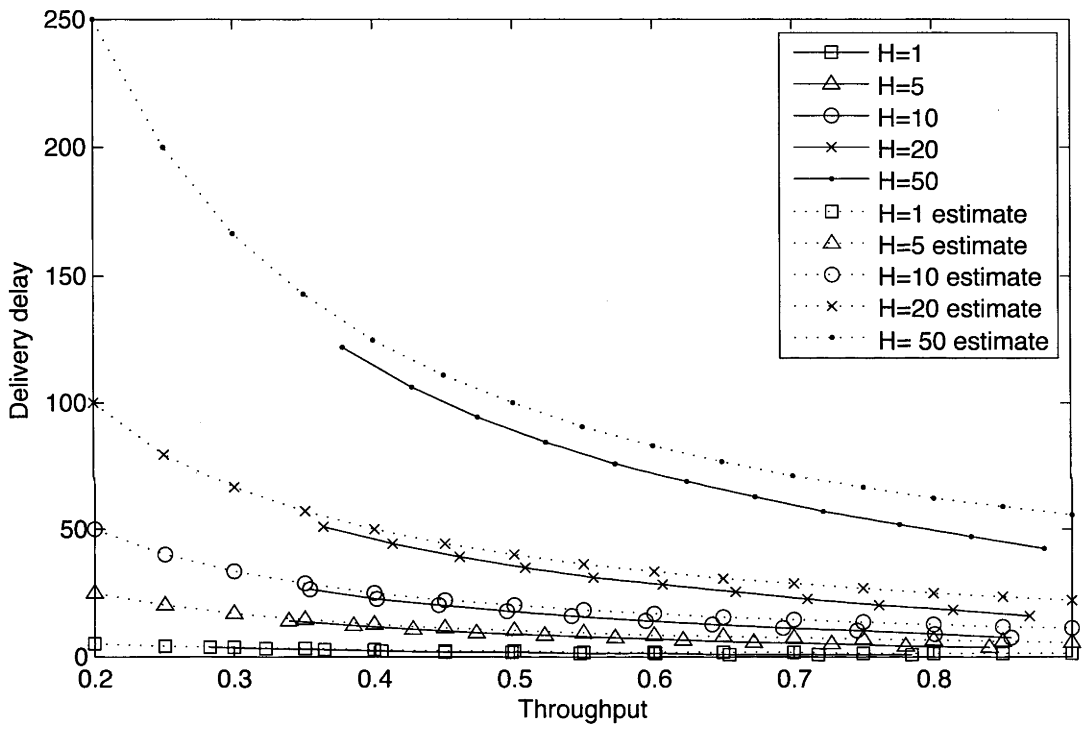


Figure 4.8: Throughput delay tradeoff for different values of H compared with the upper bound of (4.22), for channel rates $\mu = 0.4, \dots, 0.9$. $R = 4$ receivers, $T_{\min} = 1$.

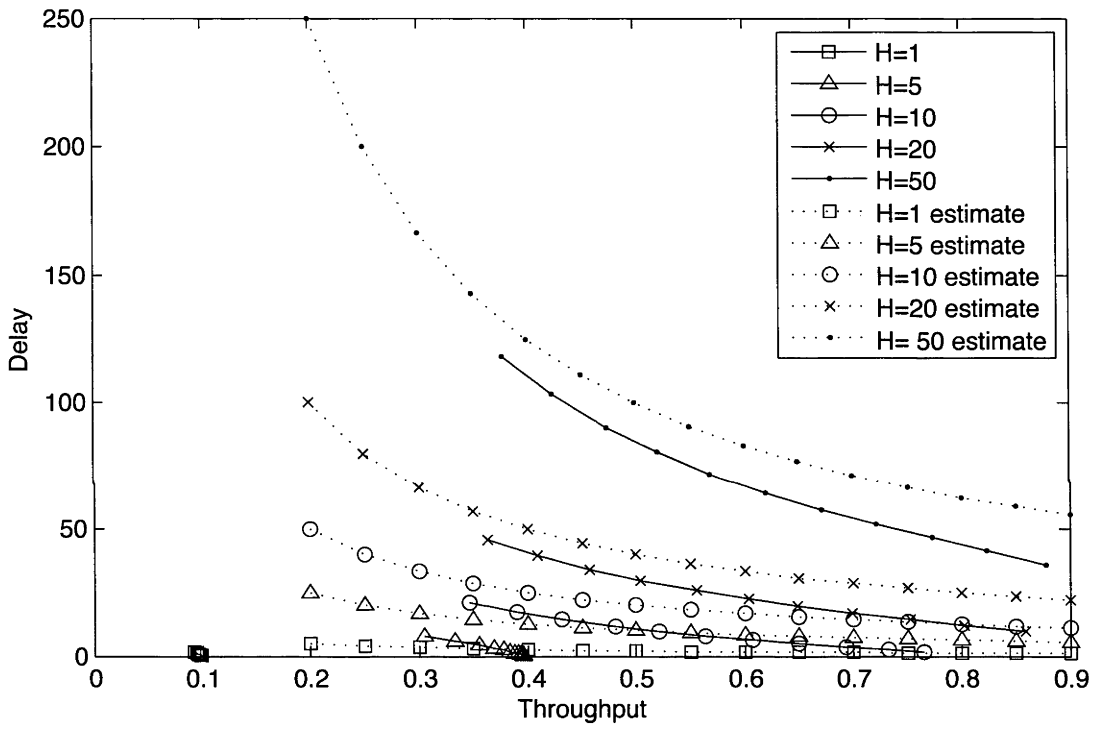


Figure 4.9: Throughput delay tradeoff for different values of H compared with the upper bound of (4.22), for channel rates $\mu = 0.4, \dots, 0.9$. $R = 4$ receivers, $T_{\min} = 20$.

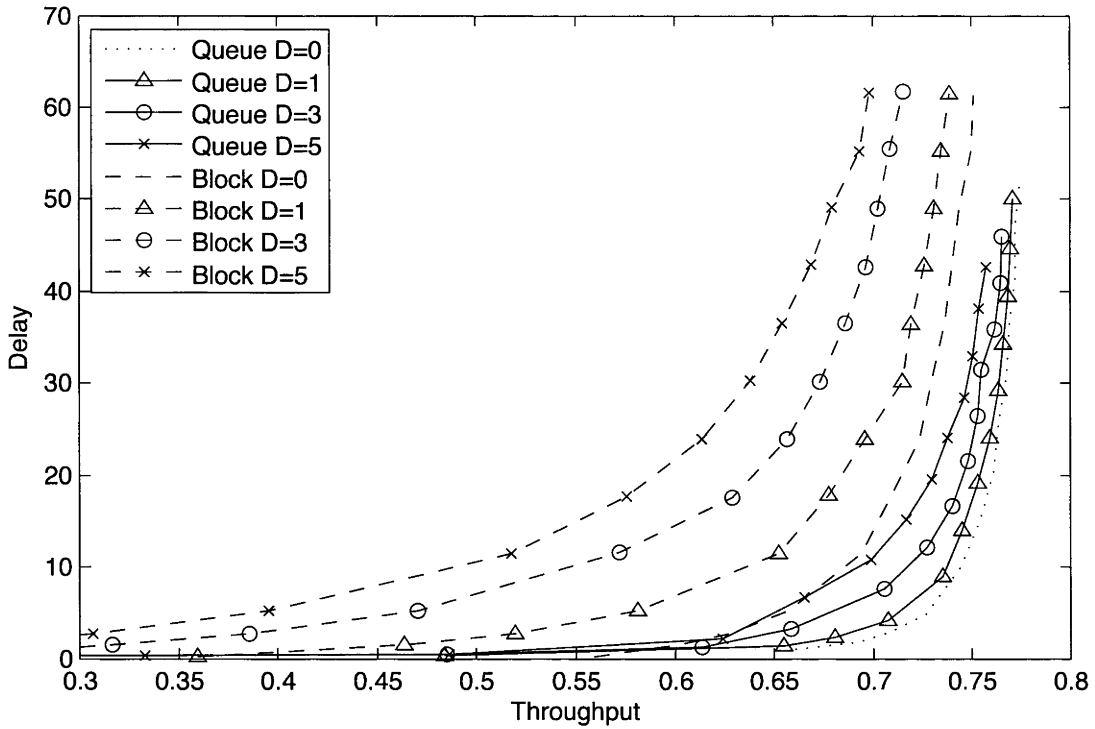


Figure 4.10: Throughput delay tradeoff under the queue threshold and block-based coding schemes, for the notification delays listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, the minimum time between notifications $T_{\min} = 1$, threshold value $H = 1, \dots, 50$.

queue size to be

$$Q(r) = \sum_{i=t-D}^t a(i), \quad (4.24)$$

the number of packets added to the transmission queue in the time since the receiver transmitted the notification.

Fig. 4.10 compares the throughput delay performance under the block coding and queue threshold schemes. Under both rate control schemes, performance degradations can be observed with increasing values of D . The throughput delay performance of the block coding scheme worsens dramatically as the value of D increases. This can be attributed to the D time slots of non-innovative transmissions that must occur, while waiting for the last receiver's notification to be received at the sender. In contrast, the queue threshold scheme experiences much smaller performance degradations in response to delayed feedback, since delays in feedback will simply cause a temporary overestimation of the number of undelivered packets stored at the receivers.

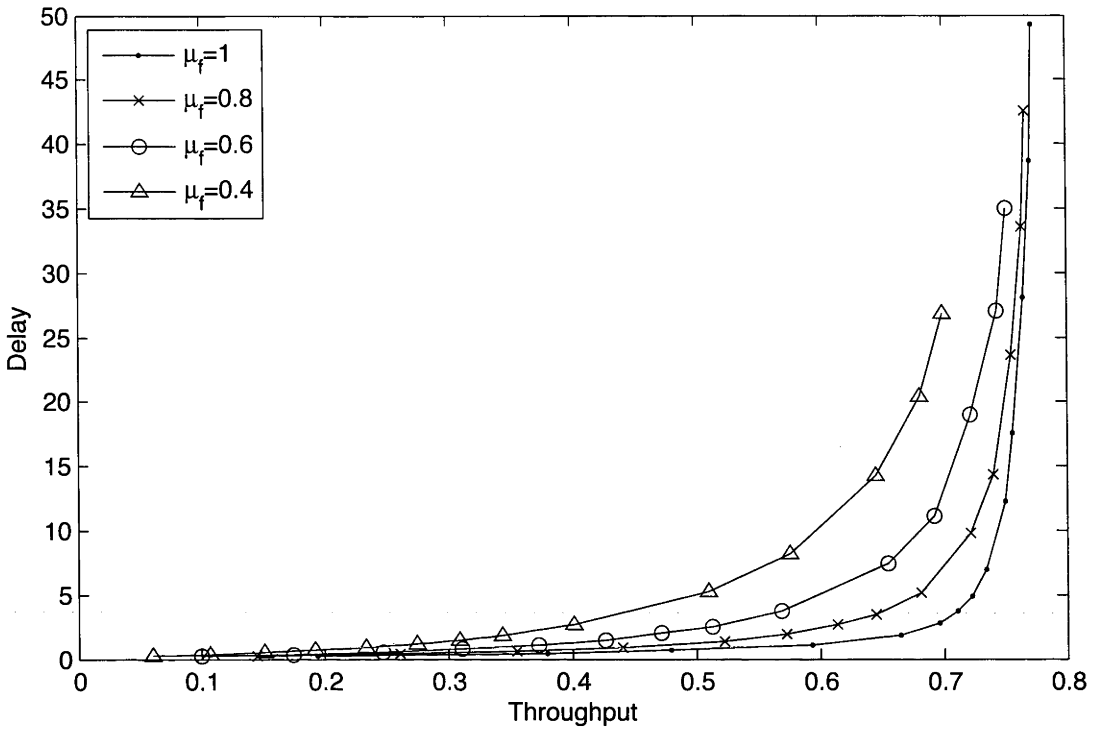


Figure 4.11: Throughput delay tradeoff for the values of μ_f listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, the minimum time between notifications $T_{\min} = 10$, $H = 1, \dots, 50$.

4.4.6.2 Feedback erasures

We now consider the possibility that some of the transmitted notifications will not be received by the sender. Let us introduce another parameter μ_f , which gives the probability that a notification from a receiver will be successfully detected by the sender. In Fig. 4.11 we examine the throughput delay performance for $T_{\min} = 10$, under a range of feedback channel conditions. Only a small difference in the delay throughput performance occurs between $\mu_f = 1$ and $\mu_f = 0.8$, indicating that small proportion of notification erasures will not affect the throughput delay performance greatly. It is useful to observe that the throughput delay performance degrades gracefully, even when μ_f drops to less than half.

Under a similar simulation for $T_{\min} = 1$ in Fig. 4.12, we did not observe any significant degradation in throughput delay performance, since the notification frequency of this scheme was high enough to tolerate erasures well. This indicates that it is possible to compensate for performance losses caused by notification erasures by adjusting $T_{\min}$ to increase the notification frequency.

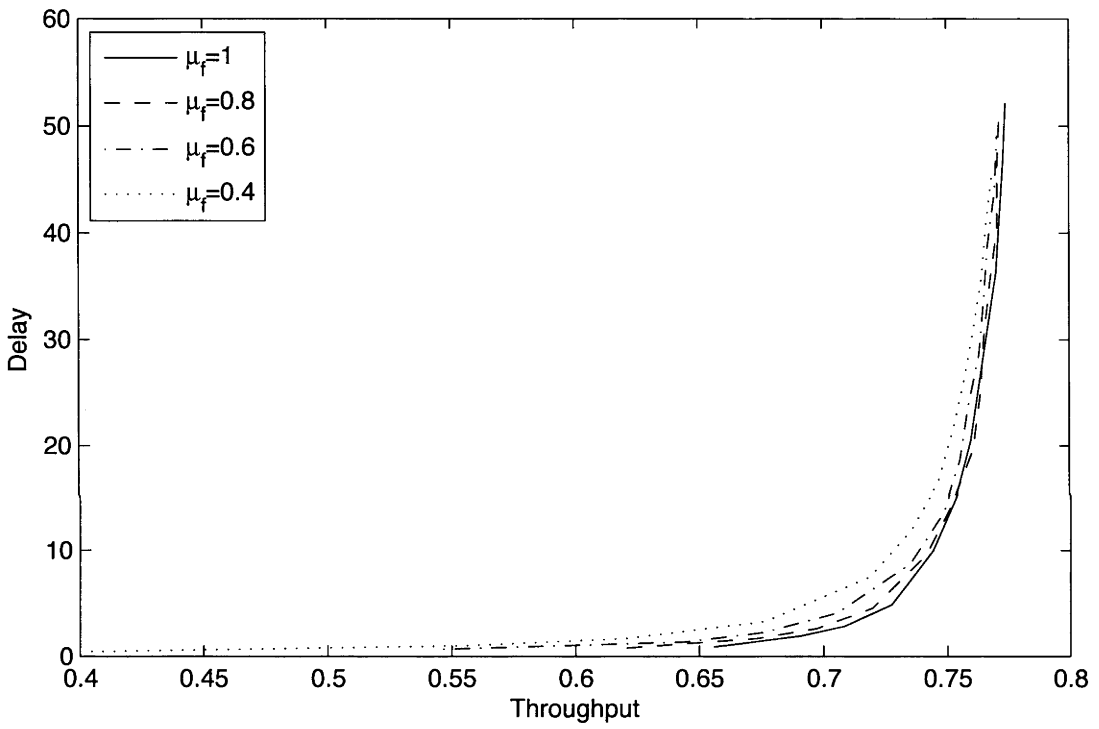


Figure 4.12: Throughput delay tradeoff for the values of μ_f listed in the figure. $R = 4$ receivers, channel rate $\mu = 0.8$, the minimum time between notifications $T_{\min} = 1$, $H = 1, \dots, 50$.

4.5 Conclusion

In this chapter, we formulated a dynamic rate control model that allowed us to estimate the expected throughput delay performance of the receivers. Based on this model, we derived a decision metric for a dynamic rate control scheme, that allowed the sender to use feedback information to adjust its transmission rate, balancing the competing requirements of good throughput and delay performance. Through analysis, this decision-making process was found to be equivalent to regulating the total number of undelivered packets stored among the receivers. The dynamic rate control scheme we proposed was demonstrated to have improved throughput delay performance compared with existing schemes in the literature.

The dynamic rate control scheme was then adapted to perform well with a reduced volume of feedback, and under imperfect feedback conditions. The resulting queue threshold scheme was shown to have noticeably improved throughput delay performance compared with other RLNC schemes in the literature. By modifying the queue threshold scheme to reduce the frequency with which notifications are transmitted, we showed that it is possible to reduce the volume of feedback required to a fraction of the previous amount required, with only a minimal impact on the throughput delay performance. An upper bound was provided for the average delay of the queue threshold scheme, and it was shown that for fixed parameter settings, the delay performance did not vary greatly as a function of the throughput. It was also demonstrated that the throughput delay performance of the queue threshold scheme is fairly robust to small degradations in the feedback, but will degrade gracefully as the quality of feedback is further reduced.

Although we can provide some upper bounds for the delay of the queue threshold scheme, we did not quantify the achievable throughput delay performance of either of the two transmission schemes we have proposed. More work is required to be able to accurately determine the achievable throughput delay performance under both the dynamic and queue threshold rate control schemes. So far, we have determined how rate control can be used to improve the throughput delay performance of the network in a single-hop broadcast network. An interesting question for future work is whether similar rate control strategies can be used to improve the throughput delay performance of multi-hop networks.

Intermediate packet delivery

In a single hop broadcast packet erasure network, we demonstrate that it is possible to use *intermediate packet delivery* to provide multi-rate packet delivery outside of the rate limitations given by the network min-cut. This is achieved by using a deterministic non-block-based network coding scheme to sidestep the limitations of the block coding model used to derive the min-cut result.

Under the network coding scheme we outline, the sender is able to transmit network coded packets above the channel rate of some receivers, while ensuring that they still experience nonzero delivery rates. Interestingly, in this generalised form of asynchronous network coded broadcast, receivers are not required to obtain knowledge of all packets transmitted so far. Instead, causal feedback from the receivers about packet erasures is used by the sender to determine a network coded transmission that will allow at least one, but often multiple receivers, to deliver their next needed packet.

Although the analysis of deterministic coding schemes is generally a difficult problem, by making some approximations we are able to obtain tractable estimates of the receivers' delivery rates, which are shown to match reasonably well with simulation. Using these estimates, we design a fairness algorithm that allocates the sender's resources so all receivers experience fair delivery rate performance relative to their channel rates.

5.1 Introduction

The information capacity of a network is a well studied problem. It has been established [2] that the capacity of any multicast network is given by its *min-cut*, namely the lowest channel rate among receivers. These results have been extended in [14, 53] to the broadcast packet erasure channels studied in this thesis. The min-cut bound is a definitive result that cannot be violated under the system model considered.

Although network capacity is a fundamental property of the network itself, ca-

capacity results in the literature rely on the assumption of *block coding*. Block coding, including rateless codes such as [64], requires that all transmissions are generated from a fixed block of information, which must be completely received by all receivers before transmission of a new block can begin. Two assumptions are generally made regarding block codes:

- Receivers cannot decode the block until it has been completely received.¹
- The same block of information must be concurrently transmitted to all receivers at the same time.

While block-based decoding is predominantly assumed or studied in the literature, the limitations of block coding do not necessarily hold in general. For example, there have been some results on the *intermediate decoding* of packets [5, 54, 61], where packets are decoded *prior* to the completion of the block - albeit without considering the logical packet ordering for delivery to the application.

Network coding, shown in [45] to be capable of achieving network capacity, also provides some interesting counterexamples to the necessity of block coding restrictions. Although a large number of *block-based* network coding schemes exist [11, 19, 30, 34, 47, 51, 57, 59, 60, 65, 66, 75, 84] which enforce the complete reception of each block, there are also some non-block-based *deterministic* network coding schemes [4, 18, 70, 71, 74, 80]. By relying on feedback from the receivers to determine the coded transmissions, and similar to [5, 54, 61], these schemes can in some cases [47, 59, 60, 65, 66, 71, 74] allow receivers to decode packets of information *prior* to receiving all information transmitted by the sender. In Chapter 3, for example, we observed that coefficient-based delivery could also make a significant contribution to the delivery performance of the network.

In this chapter, we consider the potential implications of using deterministic network coding in the case of a broadcast packet erasure network, comprising of a single sender connected to multiple receivers via a single-hop broadcast packet erasure channel, with distinct channel rates. From [14, 52], the capacity of this network under the block coding requirement is given by the channel rate of the most challenged receiver with the highest packet erasure probability, which is the min-cut of the network. As a consequence, even a single poorly-performing receiver can severely degrade the rate at which packets can be delivered in order to the application across the entire network. So a fundamental question that has not been considered before is:

What will happen to the delivery rates and simultaneity of decoding if the sender's transmission rate exceeds the channel rate of one or more receivers?

¹This is equivalent to the case studied in Chapter 3, where only *zero state delivery* is permitted.

Under the traditional block coding assumptions, the delivery rate of these receivers would be zero. Using deterministic network coding, however, it may be possible for the receivers to continue decoding new information, despite never being able to fully receive all information that has so far been transmitted by the sender.

As we will show in this chapter, it is in fact possible to achieve *intermediate packet delivery*, allowing nonzero in-order packet delivery rates to all receivers – even if the sender’s transmission rate and the packet delivery rates of some receivers in the network exceed the channel capacity. We demonstrate the principles behind the somewhat counterintuitive notion of intermediate packet delivery using the following example, which is based on the two-receiver transmission scheme first presented in [18].

5.1.1 Deterministic coding example

Imagine a simple broadcast network consisting of a single sender aiming to deliver a set of packets $\mathbf{p}_1, \mathbf{p}_2, \dots$ in the correct order to two receivers: a primary receiver Rx_1 with channel rate $\mu_1 < 1$ and a secondary receiver Rx_2 with channel rate $0 < \mu_2 < \mu_1$. At each time slot t the sender transmits a single packet, to which the receivers respond with a notification indicating whether or not they were able to receive it. The sender uses this feedback to determine which coded combination should be transmitted next, as outlined in Algorithm 6. A simple example of this algorithm in progress is given in Table 5.1. The sender’s priority is to transmit packets required by Rx_1 , who has delivered the most packets so far, but it will also aid Rx_2 wherever possible.

Packet transmissions intended for Rx_1 are not immediately useful to Rx_2 . However, Rx_2 has the ability to store and make use of any of these packets it overhears. For example, in Table 5.1 at $t = 1$, Rx_2 stores $\mathbf{p}_9$, which was intended for Rx_1 . If, owing to an erasure at Rx_1 , the sender retransmits $\mathbf{p}_9$, it can take the opportunity to code a second packet, $\mathbf{p}_4$, intended for Rx_2 , into $s(t)$. This results in the time $t = 2$ transmission of $s(t) = \mathbf{p}_9 + \mathbf{p}_4$.

The inclusion of $\mathbf{p}_4$ into $s(t)$ does not interfere with Rx_1 ’s ability to receive $\mathbf{p}_9$, since Rx_1 has already decoded all packets that could potentially be required by Rx_2 . Conversely, Rx_2 will be able to decode $\mathbf{p}_4$ from the transmission, since it already received $\mathbf{p}_9$ in the previous time slot. When receivers’ contributions to the transmission $s(t)$ do not impact each others’ ability to decode, what we call a *knowledge differential transmission* occurs. These opportunities will be discussed in a more general setting later on. The end result in this example is that the sender will, as a first priority, deliver packets to Rx_1 , resulting in a delivery rate of μ_1 to that receiver. However,

t	$s(t)$	Rx ₁	Rx ₂
0		$p_1 \cdots p_8$	$p_1 \cdots p_3, p_5$
1	p_9	✗ $p_1 \cdots p_8$	✓ $p_1 \cdots p_3, p_5, p_9$
2	$p_9 + p_4$	✓ $p_1 \cdots p_9$	✓ $p_1 \cdots p_5, p_9$

Table 5.1: An example of the transmission scheme for two receivers, showing which packets have been decoded by each receiver. A ✓ means the transmitted packet $s(t)$ was received; a ✗ indicates a packet erasure.

there are also occasional opportunities to deliver packets to Rx₂, resulting in a small nonzero delivery rate at Rx₂. We shall show that this idea can be generalised to R receivers, using a deterministic coding scheme.

In summary, the key idea in this chapter is to break away from the block-based and rate limited schemes that are prevalent in the literature, and apply a *feedback-based, deterministic* network coding scheme as shown above. In doing so, we can achieve continued intermediate packet delivery at rates that exceed the limitations on capacity determined under the complete information acquisition requirement [14]. In a sense, this encoding scheme relaxes the notion of broadcasting common information. Although it still ensures that all receivers can decode all packets, receivers are permitted to do so at greatly differing times. By taking advantage of the leading receiver's progress and the stored packets in the buffers of lagging receivers, we can establish a more general form of broadcast, which we can call *asynchronous broadcast*, which allows receivers to deliver the same information at different rates.

Algorithm 6 Two-receiver transmission algorithm of [18].

- 1: Let $N_i(t)$ be the first undecoded packet of Rx _{i} .
 - 2: **if** Rx₂ has already received $N_1(t)$ **then**
 - 3: transmit the XOR combination $N_1(t) \oplus N_2(t)$.
 - 4: **else**
 - 5: transmit $N_1(t)$.
 - 6: **end if**
-

5.1.2 Contributions and related work

In this chapter we demonstrate that it is possible to achieve intermediate packet delivery for wireless broadcast transmission, allowing multi-rate packet delivery of ordered information to receivers with nonuniform channel rates. Using a deterministic, non-block based transmission scheme, we are able to improve upon the capacity region established in [14], by maintaining nonzero packet delivery rates to all receivers while also allowing some receivers to deliver packets at rates above the min-cut.

Multi-rate broadcast to receivers with non-uniform channel rates has been previously considered in the context of applications such as video broadcast and layered streaming [13, 69], where different subsets of the information are made available to different receivers. To our knowledge, the transmission scheme we will present is the first to enable the multi-rate delivery of the same set of ordered information to all receivers.

Our transmission scheme is based on a network coding scheme first proposed in [71], which aimed to minimise the in-order packet delivery delay of a network with uniform channel rates. By making some minor modifications to this scheme, we show that it can be adapted for transmission in a broadcast network with non-uniform channel rates, allowing multiple delivery rates to be provided to different receivers. Using an analytical model to approximate the system behaviour, we demonstrate that the delivery rates achieved under this transmission scheme can be accurately estimated. The analysis of feedback-based deterministic coding schemes is a challenging problem, owing to the number of cross-receiver dependencies that must be considered. We believe that our analysis is the first attempt at this level of detail.

With the newfound ability to implement multi-rate broadcast, the provision of fair delivery rates to the receivers becomes a possibility. In this chapter, we consider a transmission scheme to be *fair* if the receivers' in-order packet delivery rates are proportional to their channel rate. Note that there are, however, many other definitions of fairness in the literature, for example [38]. To enable the sender to provide fair multi-rate delivery to the receivers, we introduce the concept of time sharing among different *modes*. Using the results of our delivery rate analysis, a fairness algorithm is designed to adjust the amount of time spent in each mode, allocating the sender's resources to allow good delivery rates to be achieved by all receivers under the fairness constraint.

5.2 System Model

The system model we will study in this chapter is a reduced version of that described in Section 2.3. A single sender aims to transmit a set of backlogged packets $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \dots$ in the correct order to a set of R receivers, $\text{Rx}_1, \dots, \text{Rx}_R$. Time is slotted, and the sender transmits at a rate of one packet per time slot. The channel between the sender and receivers is a wireless broadcast packet erasure channel, where in each time slot, each receiver Rx_i experiences packet erasures independently with a probability μ_i , as illustrated in Fig. 5.1, so that on average μ_i of the broadcasted packets are successfully received. We assume that the receivers' channel rates are

unique, and order the receivers by their channel rates, so that $\mu_1 > \mu_2 > \dots > \mu_R$, and Rx_1 is the *fastest* receiver with the highest channel rate, followed by Rx_2 , Rx_3 and so on. For simplicity of notation, the set of channel rates is represented by the vector $\mu = [\mu_1, \mu_2, \dots, \mu_R]$. The *receiver index* i of a receiver Rx_i is therefore a numerical ranking of its channel rate, compared to all other receivers in the system.

At each time slot t the sender broadcasts a single network coded packet $s(t)$, along with the vector of coding coefficients used. The network coding coefficients are chosen from the Galois field $\mathbb{F}_M$, where the field size $M \geq R$, which in this chapter we will take to be the smallest power of 2 satisfying this requirement. Recall from Section 2.1 that, for simplicity, we treat each packet $\mathbf{p}_n$ as a single symbol from $\mathbb{F}_M$.

Each receiver stores the packets it receives, along with the corresponding coding coefficients, in its own buffer. We assume that all receivers to have enough space in their buffers to store all of the packets transmitted by the sender. To decode the original packets, receivers use the information from the coding coefficients to apply Gaussian elimination [3, 42] operations, which, once enough information has been received, allow them to recover the original uncoded packets. After each packet transmission, the receiver sends a notification to indicate whether or not the packet was successfully received. For the purposes of this work we assume the sender detects these notifications without delay or error. The sender uses the information from the notifications to record which packets each receiver has stored in its buffer. Based on this information, a *transmission scheme* is used to determine the packet combinations the sender will transmit. The components of the transmission scheme we will be outlined shortly.

5.2.1 Definitions

For quick reference, we list the definitions of some common terms that will be used throughout this chapter.

DEFINITION 5.1: A *packet*, $\mathbf{p}_n$, corresponds to the n^{th} packet in the sender's backlog, and is represented by a symbol in $\mathbb{F}_M$.

DEFINITION 5.2: A packet $\mathbf{p}_n$ is *older* than another packet $\mathbf{p}_m$ if its packet index $n < m$, and *newer* if its packet index n is higher.

DEFINITION 5.3: A *transmission* $s(t)$ is a symbol in $\mathbb{F}_M$ that represents the network coded packet transmitted by the sender at time t ,

$$s(t) = \sum_i \alpha_i(t) \mathbf{p}_i, \quad (5.1)$$

where the coding coefficients $\alpha_1(t), \alpha_2, \dots$ are symbols chosen from the field $\mathbb{F}_M$.

DEFINITION 5.4: A packet $\mathbf{p}_n$ is *decoded* by a receiver Rx_i if Gaussian elimination on all transmissions $s(t)$ it has received so far reveals the value of $\mathbf{p}_n$.

DEFINITION 5.5: A packet $\mathbf{p}_n$ is *delivered* to a receiver Rx_i if all older packets $\mathbf{p}_1, \dots, \mathbf{p}_n$ have been decoded by that receiver. $d_i(t)$ is the number of packets delivered to Rx_i by time t , so the most recently delivered packet is $\mathbf{p}_{d_i(t)}$.

DEFINITION 5.6: The *delivery rate* for Rx_i is given by $R_i = d_i(t)/t$, the average rate at which packets are delivered to the receiver, as $t \rightarrow \infty$.

Example: In Table 5.1 at $t = 0$, packets $\mathbf{p}_1, \dots, \mathbf{p}_8$ have been delivered to Rx_1 , while $\mathbf{p}_1, \dots, \mathbf{p}_3$ have been delivered to Rx_2 .

DEFINITION 5.7: At any time t , the *next needed* packet $N_i(t)$ of a receiver Rx_i is the oldest packet which has not been delivered in that receiver's buffer. From Definition 5.5, $N_i(t) = \mathbf{p}_{d_i(t)+1}$.

Example: If at time t , a receiver Rx_i has the packets $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_5 + \mathbf{p}_3, \mathbf{p}_7$ stored in its buffer, its next needed packet is $N_i(t) = \mathbf{p}_3$.

5.2.2 Transmission scheme

Here we give an overview of the transmission scheme we will be studying in this chapter. The transmission scheme model we will use, shown in Fig. 5.1, is adapted from the model discussed in Section 2.4. Because we will study intermediate packet delivery in this chapter, our model differs from Fig. 2.3 in that we have removed the rate control block and transmission queue.² A mode selector component has also been added, which will be used later on to implement fair delivery rates. Our transmission scheme therefore consists of two components:

- A *mode selector* which determines the *mode* m (to be defined later) in which the coding scheme transmits,
- A *coding scheme*, which in this chapter is moded scheme B, that determines which network coded combination of the packets from the backlog will be selected for transmission.

The purpose of the transmission scheme is to deliver the packets in order, though not necessarily at the same rate, to all of the receivers.

²Our modified transmission scheme could, however, be equivalently modeled with a rate control scheme that immediately passes all packets from the backlog into the transmission queue.

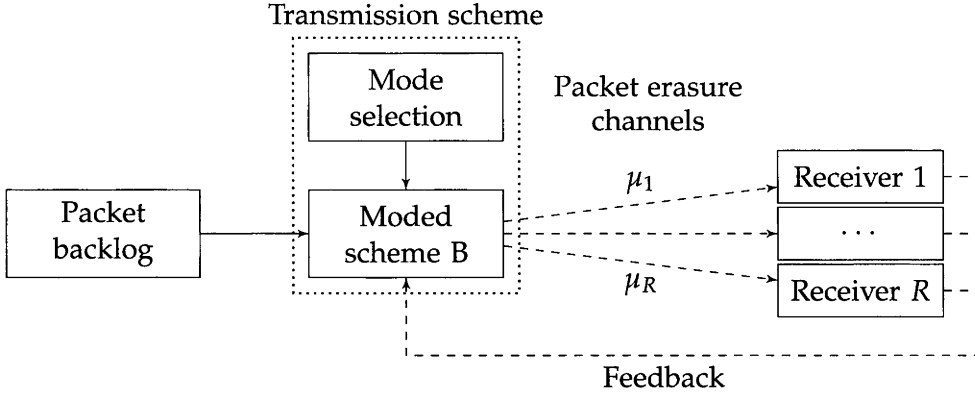


Figure 5.1: A block diagram of the components of the transmission scheme.

5.2.3 Mode selector

The mode selector component determines which subset of receivers the sender should code its transmission for.

DEFINITION 5.8: In *mode m*, a transmission is restricted to coding the next needed packets of receivers $Rx_m, \dots, Rx_R$. These receivers are said to be the *mode m receivers*. A transmission $s(t)$ made while the sender is in mode m is called a *mode m transmission*.

DEFINITION 5.9: The *mode vector* $\beta = [\beta_1, \dots, \beta_R]$ determines the proportion of time spent in each mode. β_m is the proportion of time spent in mode m , and $\sum_{m=1}^R \beta_m = 1$.

At each time slot, the mode selector randomly chooses a mode for the sender to transmit in, selecting transmission mode m with probability β_m independently of previous transmission mode choices.

5.2.4 Moded scheme B

In this chapter, we will make use of moded scheme B to determine each transmission $s(t)$. The method for determining each transmission $s(t)$ under moded scheme B is given in Algorithm 7.³ Under moded scheme B, the sender is restricted to coding from the next needed packets of the mode m receivers, so that

$$s(t) = \sum_{i=m}^R \alpha_i(t) N_i(t). \quad (5.2)$$

Under Algorithm 7, the sender makes a list containing each oldest undelivered packet p_j and the corresponding set of receivers G_j . Then, beginning with the newest packet

³More discussion of moded scheme B can be found in Section 2.5.2.

Algorithm 7 Moded scheme B.

-
- 1: Organise receivers $Rx_m, \dots, Rx_R$ into groups G_j , so that G_j contains all receivers Rx_i whose next needed packet $N_i(t) = p_j$. Let G be the set of all groups G_j which contain at least one receiver.
 - 2: Initialise $s(t) = 0$.
 - 3: **for** each group $G_j \in G$, from high to low j , **do**
 - 4: Initialise the empty *veto list* $v_j = \{\}$.
 - 5: **for** each receiver $Rx_i \in G_j$ **do**
 - 6: Calculate r_i , the result of performing Gaussian elimination on $s(t)$ with the transmissions stored in Rx_i 's buffer.
 - 7: **if** $r_i = 0$ **then**
 - 8: $v_j \leftarrow v_j \cup \{0\}$.
 - 9: **else if** $r_i = \alpha p_j$ for some field element α **then**
 - 10: $v_j \leftarrow v_j \cup \{\alpha\}$.
 - 11: **end if**
 - 12: **end for**
 - 13: **if** $0 \in v_j$ **then**
 - 14: $a_j \triangleq \min(\mathbb{F}_M \setminus v_j)$.
 - 15: Set $s(t) = s(t) + a_j p_j$.
 - 16: **end if**
 - 17: **end for**
-

in the list, it adds each packet p_j into $s(t)$ only if the receiver(s) in G_j would not otherwise receive an innovative packet.

In Algorithm 7, lines 13-15 ensure that a packet p_j is only coded into $s(t)$ if $0 \in v_j$. From lines 7-8, this is contingent on at least one of the receivers in G_j being able to cancel out the existing packets in $s(t)$ using Gaussian elimination. Therefore Algorithm 7 has the following property.

Property 5.1. *In mode m , only the next needed packets $N_m(t), \dots, N_R(t)$ of receivers $Rx_m, \dots, Rx_R$ can be coded into the transmission $s(t)$. Of these, a packet $N_i(t)$ will only be coded into $s(t)$ if it allows Rx_i to deliver its next needed packet.*

Example: An example of Algorithm 7 is given in Table 5.2 for the case where each group G_j consists of a single receiver. Consider $t = 2$. The coding scheme starts by setting $s(2) = p_{11}$. However, Rx_2 can already cancel out this transmission using p_{11} in its buffer ($0 \in v_2$ in line 13 of Algorithm 7). Hence, $s(2) = p_{11} + p_5$, because $N_2(2) = p_5$. This is already innovative for Rx_3 . Therefore, p_3 need not be added.

In [71] it is shown that, using this coding scheme, we can always find a transmission that is innovative for all relevant receivers, if the field size $M \geq R$. It should be noted, however, that under our transmission scheme, although every packet is innovative, receivers will still experience different delivery rates, since the packets they

receive are not always contiguous, and, from Definition 5.5, delivery can only take place if all older packets have been decoded at that receiver.

t	$s(t)$	Rx ₁	Rx ₂	Rx ₃
0		$p_1 \cdots p_{10}$	$p_1 \cdots p_4, p_6$	p_1, p_2, p_7
1	p_{11}	✗ $p_1 \cdots p_{10}$	✓ $p_1 \cdots p_4, p_6, p_{11}$	✗ p_1, p_2, p_7
2	$p_{11} + p_5$	✗ $p_1 \cdots p_{10}$	✓ $p_1 \cdots p_6, p_{11}$	✓ $p_1, p_2, p_7, p_{11} + p_5$
3	$p_{11} + p_7$	✗ $p_1 \cdots p_{10}$	✗ $p_1 \cdots p_6, p_{11}$	✓ $p_1, p_2, p_5, p_7, p_{11}$
4	$p_{11} + p_7 + p_3$	✓ $p_1 \cdots p_{11}$	✓ $p_1 \cdots p_7, p_{11}$	✓ $p_1 \cdots p_3, p_7, p_{11}$

Table 5.2: An example of mode 1 transmission, and the effect on receivers' buffers. Ticks and crosses represent receivers' channel states (successful and unsuccessful reception, respectively) at each transmission.

5.3 Analytical model

It is difficult to model the transmission scheme presented in Section 5.2 precisely. This is due to the complicated interdependencies caused by the sender's transmissions being determined by the contents of the receivers' buffers, which are in turn determined by a combination of past transmissions, the receivers' erasure patterns and the effects of Gaussian elimination. Therefore, in order to obtain a tractable model, we must make some simplifying approximations in our transmission scheme analysis. Here we list the approximations we will use in this chapter to obtain a tractable model for estimation the receivers' delivery rates under our transmission scheme. In Sections 5.5 to 5.7 we will demonstrate that the numerical results obtained under this analytical model match well with simulations.

5.3.1 Simplified encoding scheme

We begin by showing that, in most cases, Algorithm 7 naturally degrades to the simpler algorithm given in Algorithm 8. Where the conditions required to use this simplified coding scheme are satisfied, we can prove that a field size $M = 2$ is sufficient to guarantee that a sender transmission $s(t)$ which is innovative for all mode m receivers can always be found. We begin by introducing two approximations.

Approximation 5.1. *If the receivers' channel rates are distinct and ordered so that $\mu_1 > \mu_2 > \cdots > \mu_R$, then their delivery rates will also obey the ordering $R_1 > R_2 > \cdots > R_R$.*

Intuitively it makes sense that higher channel rates will result in higher packet delivery rates for the corresponding receivers. Where Approximation 5.1 holds, the

total number of packets delivered by each of the receivers will diverge over time. Therefore, after a sufficient amount of time has passed the next needed packets $N_1(t), \dots, N_R(t)$ of each receiver will also be distinct with high probability.⁴ Therefore we state the next approximation that we will use for this chapter,

Approximation 5.2. *The next needed packet of each receiver is distinct and ordered by receiver index, so that $N_1(t) > N_2(t) > \dots > N_R(t)$.*

Under Approximation 5.2, each receiver's next needed packet is distinct. When this is the case, there is only a single receiver in group G_j corresponding to each next needed packet p_j and Algorithm 7 reduces to the simplified encoding scheme presented in Algorithm 8. This simplified encoding scheme, which is based on the much smaller field size $\mathbb{F}_2$, will form the basis of our analysis.

Algorithm 8 Simplified encoding scheme.

- 1: Initialise $s(t) = 0$.
 - 2: **for** $i = m, \dots, R$ **do**
 - 3: Calculate r_i , the result of performing Gaussian elimination on $s(t)$ using the packets in receiver Rx_i 's buffer.
 - 4: **if** $r_i = 0$ **then**
 - 5: $s(t) = s(t) \oplus N_i(t)$.
 - 6: **end if**
 - 7: **end for**
-

Property 5.2. *Under Algorithm 8, a mode m transmission is innovative to every mode m receiver, and non-innovative to receivers $Rx_1, \dots, Rx_{m-1}$.*

Proof: Under Approximation 5.2, Rx_1 will have the newest next needed packet, followed by Rx_2, Rx_3 , and so on. At each iteration i of the **for** loop, it is ensured that $s(t)$ is innovative to Rx_i . By Definition 5.7 and Approximation 5.2, $N_{i+1}(t), \dots, N_R(t)$ have already been decoded by Rx_i , so the addition of these packets to $s(t)$ in later iterations of the **for** loop does not affect the innovativeness of $s(t)$ to Rx_i . Algorithm 8 guarantees a transmission $s(t)$ will be found which is innovative to all receivers $Rx_m, \dots, Rx_R$.

If the ordering of next needed packets is maintained, packets coded for $Rx_m, \dots, Rx_R$ have already been decoded by $Rx_1, \dots, Rx_{m-1}$. Therefore the m^{th} mode will not contain innovative information for receivers Rx_1 to Rx_{m-1} .

The simplified transmission scheme of Algorithm 8 also has the following property:

Property 5.3. *Implementing Algorithm 8 with the binary field $\mathbb{F}_2$ is sufficient to ensure that Property 5.2 holds.*

⁴Note, however, that this is not always the case: for example at $t = 0$, $N_1(t) = \dots = N_R(t) = p_1$ since none of the receivers has received any transmissions yet.

Proof: Under Approximation 5.2, the next needed packets of each receiver are distinct. Therefore each group $G_j \in G$ contains exactly one receiver. It follows that in each iteration of the **for** loop corresponding to a group $G_j \in G$, only one element can be added to the corresponding veto list v_j . Since the binary field $\mathbb{F}_2$ contains two elements, this field is sufficient to ensure an appropriate coefficient can always be found.

An example of the simplified transmission scheme for three receivers in mode 1 is given in Table 5.2. Note that each packet coded into $s(t)$ corresponds to a receiver being able to decode its next needed packet. Two important features of this coding scheme can be observed:

1. It is possible for packets to be delivered to any of the receivers. For example, at time $t = 2$, Rx₂ delivers its next needed packet $\mathbf{p}_5$, and at $t = 4$ receivers Rx₁, Rx₂, Rx₃ all deliver their next needed packets $\mathbf{p}_{11}$, $\mathbf{p}_7$ and $\mathbf{p}_3$, respectively.
2. Every next needed packet coded into a transmission $s(t)$ corresponds to a receiver who can decode and hence deliver its next needed packet, if it receives the transmission. For example, at time $t = 4$, all three receivers' next needed packets are encoded. Therefore, receiving the transmission allows all three receivers to deliver in the same time slot.

5.3.2 Buffer model

Now that the assumptions for the analytical model have been detailed, we outline the model we will use to analyse the contents of the receivers' buffers. The packets stored in the buffer serve a dual purpose. Firstly, they provide information that will facilitate the decoding of the received packets. Secondly, they store information about packets that will need to be delivered at some point in the future.

For the purpose of analysis, we can imagine that every receiver's buffer has space allocated for all of the packets in the sender's backlog, as illustrated in Fig. 5.2. This can be imagined as a large vector, with each element corresponding to a packet, ordered left to right from oldest to newest. The associated coding coefficients are also stored along with each element of the vector, so that the receiver has the information required to perform Gaussian elimination. As shown in Fig. 5.2, each position n in the buffer corresponds to the packet $\mathbf{p}_n$. The following two definitions will be used to describe the contents of a receiver's buffer.

DEFINITION 5.10: For a receiver Rx _{i} , The *buffer coefficient matrix* $\mathbf{B}_i$ stores the coding coefficients associated with each buffer position. Each packet $\mathbf{p}_n$ corresponds to the n^{th} column of the buffer coefficient matrix, which we denote by $\mathbf{B}_i(n)$. If packet $\mathbf{p}_i$ is coded into the combination stored at $\mathbf{p}_n$, then the i^{th} element of that column vector

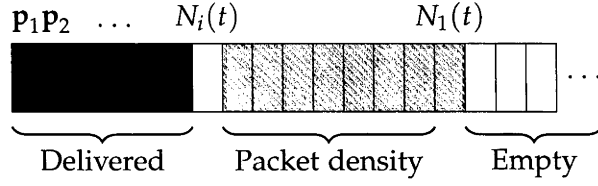


Figure 5.2: Our model of receiver Rx_i 's buffer. Dark coloured packets have been all received and delivered; some fraction of the shaded packets have been received; white packets have not been received. Packets are ordered left to right by ascending packet index.

will correspond to its coding coefficient. Otherwise, this element will be 0. If buffer position n is empty, then $B_i(n) = 0$.

To decode the packets stored in the buffer, identical Gaussian elimination operations are applied to the columns of the buffer coefficient matrix, and the packets in the buffer. Once a column of the buffer coefficient matrix $B_i(n)$ has been reduced to a standard basis vector e_n , then the corresponding packet has been decoded.

DEFINITION 5.11: For a receiver Rx_i , the *buffer fill vector* $\mathbf{b}_i$ indicates whether any information has been stored at each position n of the buffer. We say that $\mathbf{b}_i(n) = 0$ if $B_i(n)$ is the zero vector, otherwise $\mathbf{b}_i(n) = 1$.

Generally, the receiver will be required to store both coded and uncoded packets. Algorithm 9 outlines procedure for storing a received transmission $s(t)$ in the buffer. This procedure can be divided into three stages: using Gaussian elimination to reduce $s(t)$, storing the result in the correct buffer position, and using this new packet to further reduce other packets in the buffer. We now give an example of Algorithm 9 at work.

Example: In Table 5.2, at time $t = 2$, Rx_3 cannot further reduce $s(2) = \mathbf{p}_{11} + \mathbf{p}_5$, nor can $s(2)$ further reduce any of the packets stored by Rx_3 . Therefore the received packet $\mathbf{p}_{11} + \mathbf{p}_5$ will be stored at buffer position 11. By contrast, at time $t = 3$ the presence of packets $\mathbf{p}_7$ and $\mathbf{p}_{11} + \mathbf{p}_5$ allows the received packet $s(3) = \mathbf{p}_{11} + \mathbf{p}_7$ to be reduced down to $s'(3) = \mathbf{p}_5$. After storing $s'(3) = \mathbf{p}_5$ at buffer position 5, we use it to reduce $\mathbf{p}_{11} + \mathbf{p}_5$ down to $\mathbf{p}_{11}$ which will replace the content of buffer position 11.

This method of representing the buffer has the following useful properties:

Property 5.4. For every receiver Rx_i , Algorithm 9 ensures that for every filled element n in the buffer, $\mathbf{p}_n$ is the packet with the highest index coded.

Proof: Algorithm 9 only modifies the contents of the buffer at lines 3 and 4. At line 3, $s'(t)$ is stored at the correct position. Since Gaussian elimination has already been applied to $s'(t)$, applying line 4 will not change the highest packet index of any of the stored combinations.

Algorithm 9 Adding a received packet to the buffer of Rx_i .

-
- 1: Find $s'(t)$, the remainder when Gaussian elimination is applied to $s(t)$ using the columns of B_i .
 - 2: **if** $s'(t) \neq 0$ **then**
 - 3: Add $s'(t)$ to buffer position n , where n is the highest packet index in $s'(t)$.
 - 4: Use $s'(t)$ to apply Gaussian elimination to the non-empty buffer positions, to obtain column echelon form for B_i .
 - 5: **end if**
-

We now introduce some terminology that will be useful later.

DEFINITION 5.12: A packet p_n is *seen* by receiver Rx_i if $b_i(n) = 1$, i.e. there is a packet stored at buffer position n .

Example: In Table 5.2, at time $t = 2$, receiver Rx_3 has packets $p_1, p_2, p_7, p_{11} + p_5$ in its buffer. In our model of the buffer, positions 1, 2, 7 and 11 are filled, therefore $b_3(1) = b_3(2) = b_3(7) = b_3(11) = 1$ and packets p_1, p_2, p_7 and p_{11} have been seen. The nonzero columns of the buffer coefficient matrix are given by $B_3(1) = e_1$, $B_3(2) = e_2$, $B_3(7) = e_7$, and $B_3(11) = e_{11} + e_5$.

DEFINITION 5.13: A transmission $s(t)$ is *innovative* to a receiver Rx_i if $s(t)$ cannot be computed from packets in Rx_i 's buffer.

Property 5.5. A receiver Rx_i 's next needed packet is always the first unseen packet in its buffer. In other words, its next needed packet is given by $N_i(t) = p_n$, where n is the first buffer position for which $b_i(n) = 0$.

Proof: If $b_i(1) = \dots = b_i(n-1) = 1$, then packets $p_1, \dots, p_{n-1}$ must have been decoded and delivered to the receiver (since we have received $n-1$ innovative combinations of $n-1$ packets). By Property 5.4, if p_n has been delivered by Rx_i , then we must have $B_i(n) = 1$. Since $B_i(n) = 0$, it follows that the first unseen packet p_n is the next needed packet of Rx_i .

5.3.2.1 Regions of the buffer

As illustrated in Fig. 5.2, our model of the buffer of a receiver Rx_i can be broken down into several distinct regions. The *delivered region* consists of all packets $p_1, \dots, p_{d_i(t)}$ which have been delivered to the receiver. By definition this is immediately followed by $N_i(t) = p_{d_i(t)+1}$, the oldest undelivered packet in the buffer. Following this is the *packet density region*, consisting of combinations of transmitted but undelivered packets $p_{d_i(t)+2}, \dots, N_1(t)$. In this region it is possible for Rx_i to overhear both coded and uncoded transmissions intended for faster receivers. All packets lying beyond $N_1(t)$ form the *empty region* where no packets are yet stored, since no information about them has been broadcasted yet.

For a receiver Rx_i , there are three possible outcomes regarding the storage of every received packet in the buffer.

- If $s(t)$ is not innovative to Rx_i , then the packet will be discarded.
- If $s(t)$ is innovative and allows $N_i(t)$ to be decoded, then $s'(t)$ will be stored at position $d_i(t) + 1$. This results in the delivery of one or more new packets, so that $d_i(t+1) > d_i(t)$.
- If $s(t)$ is innovative but it does not help decode $N_i(t)$, then the transmission is stored in the packet density region of the buffer according to Algorithm 9.

5.3.2.2 Subregions

Taking a closer look at the packet density region of Fig. 5.2, this region can be further divided into subregions $1, 2, \dots, R$, each with its own distinct packet density. This is illustrated in Fig. 5.3. The reason each of these subregions exists, is that the transmissions from each mode m contribute to the packet density of each subregion. However, it is not a uniform contribution. Mode 1 transmissions evenly cover packets in subregions $1, 2, \dots, R$, while mode 2 transmissions cover subregions $2, \dots, R$, and so on. The impact of these transmissions depends on how frequently they occur, which is in part determined by the mode vector β . For a given receiver Rx_r , the primary subregion of interest is subregion r , which gives us the packet density in the subregion where Rx_r is delivering its next needed packets. The packet density of subregion r is called the *buffer density* B_r , which will be useful in determining the delivery rate later on.

DEFINITION 5.14: For a receiver Rx_r , the *buffer density* B_r is defined as the average proportion of buffer positions filled inside subregion r ,

$$B_r = \begin{cases} 0, & r = 1 \\ \frac{\sum_{j=d_r(t)+2}^{d_{r-1}(t)} \mathbf{b}_r(j)}{(d_{r-1}(t) - d_r(t) - 1)}, & 1 < r \leq R. \end{cases} \quad (5.3)$$

This is illustrated in Fig. 5.3.

5.3.2.3 Buffer model approximations

In our delivery rate analysis of the transmission scheme, we model our system for the simple coding scheme presented in Section 5.3.1. However, even to this coding scheme we need to apply some further simplification for tractability. Therefore

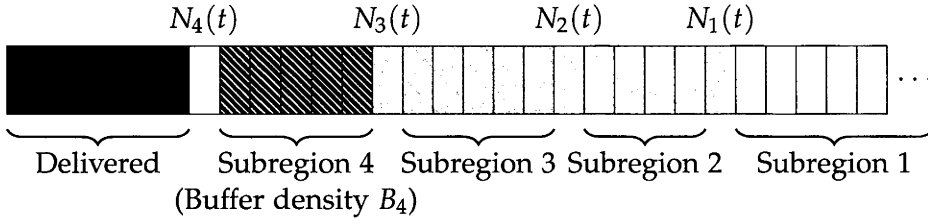


Figure 5.3: Subregions within the packet density region of receiver Rx_4 's buffer. Darker colouring corresponds to a higher packet density; white indicates a packet has not been received. Packets are ordered left to right by ascending index.

we now present a list of approximations that will be used to model the receivers' buffers.⁵

Approximation 5.3. All packets stored in a receiver Rx_i 's buffer are uncoded, so if $\mathbf{b}_i(n) = 1$, then $\mathbf{B}_i(n) = \mathbf{e}_n$. This means that once a receiver Rx_i has seen a packet $\mathbf{p}_n$, we assume decoding of it as well.

This approximation results a somewhat optimistic estimate of the probability of decoding. However, this is not an unreasonable approximation as our coding scheme is *near-systematic*, meaning that whenever Property 5.2 permits, the sender transmits an uncoded packet. And of the coded transmissions received, many will be reduced to uncoded packets when they are stored in the buffer, through Gaussian elimination. From our simulations we observe that, after Gaussian elimination has been applied, the vast majority of packets stored in the buffer are uncoded. Some examples are given in Table 5.3. In general, we observe that a few factors influence the accuracy of Approximation 5.3. Higher channel rates and a lower receiver index i will reduce the fraction of undecoded packets a receiver Rx_i has stored in its buffer. On the other hand, more time spent in modes $2, \dots, R$ will reduce the proportion of undecoded packets stored in the receivers' buffers.

Example: In Table 5.2 at time $t = 1$, we see a typical situation where the sender sends an uncoded packet, in this case $s(t) = \mathbf{p}_{11}$. Although occasionally a coded packet may be stored in the receiver's buffer, for example the buffer of Rx_3 at $t = 2$, often Gaussian elimination will help decode the packet, illustrated for Rx_3 at $t = 3$.

Approximation 5.3 leads to the following property, which simplifies our model of the receivers' packet delivery mechanisms. This property will be useful for the knowledge differential calculations of Section 5.6.

⁵It can be observed that all of the approximations in this section are used to remove the more unpredictable elements of cross-receiver interactions. For Rx_1 and Rx_2 , the effective lack of interaction with other receivers means that when analysing these two receivers, our approximations will be exact. As the receiver index increases however, the accuracy of these approximations will slowly decrease.

μ	β	Percentage of undecoded packets
[0.8, 0.6, 0.4, 0.2]	[0.85, 0.05, 0.05, 0.05]	0.0, 0.0, 2.66, 10.26
[0.9, 0.8, $\dots$, 0.2]	[0.65, 0.05, $\dots$, 0.05]	0.0, 0.01, 0.63, 1.06, 2.88, 5.23, 11.5, 16.22
[0.8, 0.75, $\dots$, 0.45]	[1, 0, $\dots$, 0]	0.0, 0.03, 0.31, 0.60, 3.20, 6.10, 11.93, 12.07

Table 5.3: Percentage of undecoded packets in each receiver's buffer for $R = 4, 8$, and 8 , under the indicated values of μ and β .

Property 5.6. For a receiver Rx_r to decode its next needed packet $N_r(t)$, every packet coded into $s(t)$ apart from $N_r(t)$ must have been seen by Rx_r .

DEFINITION 5.15: The *leader* in a mode m transmission is the receiver Rx_m .

DEFINITION 5.16: A *leader transmission* of a packet p_n is a transmission $s(t)$ where p_n is the packet with the highest index coded into the transmission, i.e. the next needed packet of the leader. If the sender is in mode m when this transmission occurs, we can more precisely describe it as a *mode m leader transmission*.

Example: If the transmission $s(t) = p_3 + p_5 + p_{10}$, then it is a leader transmission of packet p_{10} , the newest packet in the transmission. If the sender was in mode 2 at time t , then $s(t)$ is a mode 2 leader transmission of p_{10} .

We can further simplify our model of the receivers' packet delivery behaviour by limiting the ways in which packets can be seen.

Approximation 5.4. A receiver Rx_i can only have seen an undelivered packet p_n through the reception of a leader transmission of p_n .

In practice, due to the effects of Gaussian elimination, it is possible for a receiver Rx_i to see an undelivered packet in the packet density region at any time slot. But for the purpose of analysis, we only allow the receiver to see a packet by the most likely method, namely receiving a leader transmission. As a result of this approximation, we obtain the following property, which will also be useful in Section 5.6.

Property 5.7. For a packet p_n in a receiver's packet density region to be seen, Rx_r must have received a leader transmission of p_n .

One issue this property raises is that, since there is a common sender, the same number of leader transmissions of each packet will be sent to the receivers. As a result, we can expect some correlation between which packets each receiver has seen. These cross receiver dependencies would significantly increase the complexity of our analysis, so for the sake of simplicity we make the following approximation.

Approximation 5.5. The number of mode m leader transmissions of a packet p_n is independent of the number of mode b leader transmissions of p_n , for all $m \neq b$.

It can be observed that most of the approximations made in this section are designed to minimise the number of cross-receiver interactions we will need to take into account in our calculations. With the exception of Approximation 5.2, our analytical model is exact for receivers Rx_1 and Rx_2 . But, as the receiver index increases, so does the impact of cross receiver interactions on the receiver's performance. As a result, it is expected that our analytical model will lose some accuracy as the number of receivers increases. Our simulation results indicate that the delivery rates predicted by the analytical model are accurate for $R = 4$ and 8, however, the computational complexity of the simulations prevent us from investigating higher values of R with accuracy.

5.4 Outline of delivery rate analysis

In this section we outline the steps that will be taken in Sections 5.5 to 5.7 to determine the delivery rate R_r for each receiver Rx_r under the transmission scheme outlined in Section 5.2.4. The estimation of the delivery rate is a challenging task, given the somewhat complicated dependency between sender transmissions and the contents of each of the receivers' buffers. The approximations made in Section 5.3 go some way toward simplifying these relationships, but the process of calculating the receivers' delivery rates still requires some detail. A map of the quantities we will calculate on our journey towards finding the delivery rate is illustrated in Fig. 5.4, with definitions given in Table 5.4 for quick reference. The values in this figure will be discussed later, but for now we simply observe that a number of dependencies exist between different components in the calculation.

To divide the problem into manageable pieces, we make the observation that there are two ways in which a packet may be delivered to a receiver. Either:

- The packet was previously seen from leader transmissions of the packet in other modes,
- The packet is delivered through a *knowledge differential*, while it is the receiver's next needed packet.

The rate analysis is therefore divided into three parts. Analysis of leader transmissions takes place in Section 5.5, while knowledge differentials are studied in Section 5.6. The results from these two sections are combined to determine the delivery rate in Section 5.7. Each of these sections, corresponding to the dotted rectangles in Fig. 5.4, will now be briefly discussed.

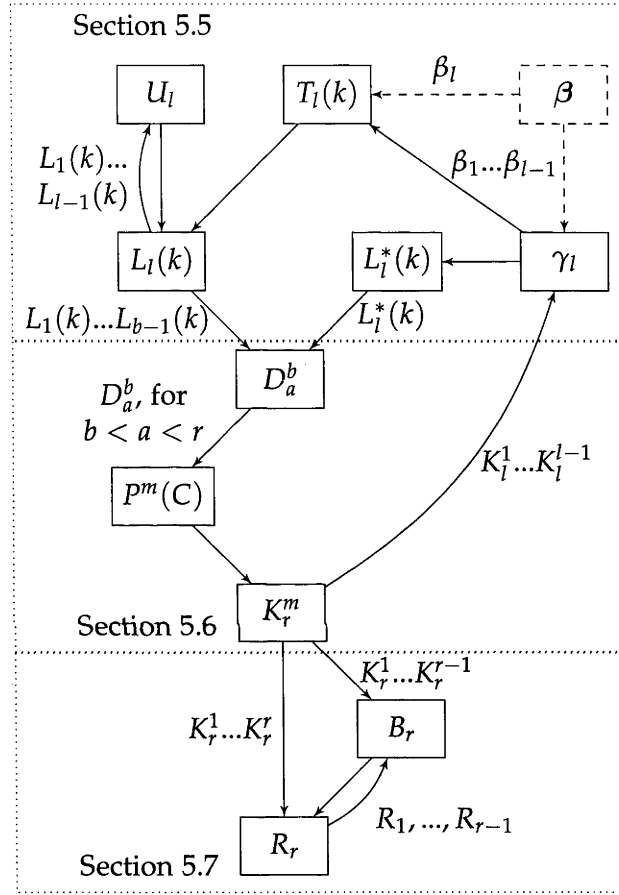


Figure 5.4: Dependency graph for quantities calculated in Sections 5.5 to 5.7. See Table 5.4 for a list of definitions. Labeled arrows indicate which values are required to calculate the next block, while unlabeled arrows indicate the following block depends directly on the value from the previous block. Curved arrows indicate dependencies on previously calculated values.

U_l	Probability that a packet will not be seen from Rx_l from mode $1, \dots, l-1$ transmissions.
$T_l(k)$	Probability that there will be k total mode l leader transmissions of a packet, given that it was not seen from mode $1, \dots, l-1$ leader transmissions.
β	The mode vector set by the sender (see Definition 5.9).
$L_l(k)$	Probability that a total of k mode l leader transmissions were made of a packet that has been delivered by Rx_l .
$L_l^*(k)$	Probability that so far, there have been exactly k mode l leader transmissions of $N_l(t)$, Rx_l 's current next needed packet.
γ_l	Probability of a knowledge differential transmission for Rx_l .
D_a^b	Probability that the next needed packet $N_b(t)$ of a receiver Rx_b is seen by another receiver Rx_a , where $a > b$.
$P^m(C)$	Probability of a coding C being transmitted in mode m .
K_r^m	Probability that Rx_r has a knowledge differential while the sender is in mode m .
B_r	Expected buffer density.
R_r	Estimated delivery rate for receiver Rx_r .

Table 5.4: Definition of the values used in Fig. 5.4.

5.4.1 Leader transmissions

In Section 5.5, we characterise the number of mode l leader transmissions of a packet $\mathbf{p}_n$ in the packet backlog. The process for calculating this probability is illustrated by the top box of Fig. 5.4. By Approximation 5.4, this characterisation is important because it can be used to determine the likelihood that a packet $\mathbf{p}_n$ has been seen by a receiver.

There are three possible cases to consider when finding the likelihood that a packet $\mathbf{p}_n$ has been transmitted k times in mode l . These depend on Rx_l 's progress towards delivering $\mathbf{p}_n$.

- If $n \leq d_l(t)$ then $\mathbf{p}_n$ has already been delivered by Rx_l , and $L_l(k)$, calculated in Section 5.5.4, gives the probability there were k mode l leader transmissions of $\mathbf{p}_n$.
- If $n = d_l(t) + 1$, then $\mathbf{p}_n = N_l(t)$, and k takes the probability distribution $L_l^*(k)$, calculated in Section 5.5.3.
- If $n > d_l(t) + 1$, then there have been no mode l leader transmissions of $\mathbf{p}_n$ so far, i.e. $k = 0$, so we do not need to consider the possibility of any mode l leader transmissions of this packet.

The probabilities $L_l(k)$ and $L_l^*(k)$ are calculated in Section 5.5.

5.4.2 Knowledge differentials

In Section 5.6 we find the probability of a knowledge differential. This process is illustrated by the middle box in Fig. 5.4. Knowledge differentials are a key concept in this chapter, as they are necessary to allow receivers to deliver the remaining packets that were not seen from leader transmissions.

DEFINITION 5.17: At time t , a receiver Rx_r has a *knowledge differential* if the portion of $s(t)$ coded for faster receivers $\text{Rx}_1, \dots, \text{Rx}_{r-1}$ can be decoded by Rx_r using the information stored in its buffer. If the sender is in mode m , then the probability of a knowledge differential is given by K_r^m .

If $s(t)$ meets this criterion, it is a *knowledge differential transmission*. Otherwise, it is a *non-knowledge differential transmission*.

Knowledge differentials occur when portions of the transmission $s(t)$ intended for faster receivers do not affect Rx_r 's ability to decode the transmission. Knowledge differentials make it possible to provide different delivery rates to each of the receivers, since some packets will be received effectively without interference from the

faster receivers. Whenever a knowledge differential transmission is received, Rx_r is able to decode and hence deliver its next needed packet $N_r(t)$.

Property 5.8. *A non-leading receiver Rx_i 's next needed packet $N_i(t)$ can only be seen through the reception of a knowledge differential transmission.*

Combining Property 5.1 and Approximation 5.2, a new packet can be delivered to Rx_r iff its next needed packet $N_r(t)$ is encoded into the transmission $s(t)$. In these time slots, the sender can take advantage of the knowledge differential to effectively send Rx_r an uncoded packet.

There are two components required to calculate K_r^m : D_a^b , the probability that a receiver Rx_a has a Rx_b knowledge difference, and $P^m(C)$, the probability of a particular coding C . These are now discussed.

5.4.2.1 Knowledge differences

From Property 5.6, each of the packets coded into $s(t)$ for other receivers must be seen by a receiver Rx_a in order to obtain a knowledge differential. The probability of each of these packets being seen is given by the probability of a Rx_b knowledge difference, defined as follows.

DEFINITION 5.18: At time t , a receiver Rx_a obtains a Rx_b knowledge difference if $\mathbf{b}_a(d_b(t) + 1) = 1$, i.e. Rx_a has seen $N_b(t)$. The probability of Rx_a having a Rx_b knowledge difference is D_a^b . If $a \leq b$, then we set $D_a^b = 1$.⁶

This means that when Rx_a has a Rx_b knowledge difference, even if $N_b(t)$ is coded in the sender transmission $s(t)$, it will not affect Rx_a 's ability to decode $s(t)$. Examples of this can be seen in Table 5.2. At times $t = 2, 3, 4$, Rx_2 has seen Rx_1 's next needed packet, $\mathbf{p}_{11}$. At time $t = 4$, Rx_3 has seen both Rx_1 and Rx_2 's next needed packets, $\mathbf{p}_{11}$ and $\mathbf{p}_7$, respectively.

It is important to note that knowledge differences are a common occurrence, since the broadcast nature of the system means that all transmissions may be overheard by any of the receivers. A packet seen by one receiver, but not another, causes a knowledge difference to occur in the future. Generally the higher a receiver's channel rate, the better its chances of obtaining a knowledge difference, since it is more likely to overhear packets that are missed by other receivers.

⁶We set $D_a^b = 1$ when $a \leq b$, since if $a < b$ then Rx_b has already decoded $N_a(t)$, and when $a = b$, a receiver cannot affect its own probability of decoding.

5.4.2.2 Codings

The probability of a knowledge differential K_r^m depends on which of the receivers' next needed packets are coded into $s(t)$. Therefore in order to determine the probability of a knowledge differential, it is first necessary to find the probability of every possible coding, C .

DEFINITION 5.19: The *transmission coding* C is an R -length binary vector representing which of the receivers' next needed packets have been coded into $s(t)$. If the r^{th} element of C , $C(r) = 1$, then $N_r(t)$ is one of the packets coded into $s(t)$. Otherwise, $C(r) = 0$ and $s(t)$ does not code $N_r(t)$.

Example: If $R = 4$ and $s(t) = N_1(t) + N_3(t) + N_4(t)$, then $C = [1, 0, 1, 1]$. If $R = 5$ and $s(t) = N_2(t) + N_4(t)$, then $C = [0, 1, 0, 1, 0]$.

5.4.3 Delivery rate

In Section 5.7 we will find the delivery rate R_r . This process is illustrated in the bottom box in Fig. 5.4. The knowledge differential probabilities established in Section 5.6 are used to determine the delivery rate of all of the receivers as follows.

By Property 5.2, every mode l transmission received by a receiver Rx_r is innovative for $r \geq l$. Therefore, it either allows the delivery of $N_r(t)$, or is stored in the packet density region. Knowledge differential transmissions allow delivery of next needed packets, and non-knowledge differential transmissions contribute seen packets to the buffer. The buffer density B_r (recall Definition 5.14) can be determined from the probability of non-knowledge differential transmissions and the delivery rates of receivers $Rx_1, \dots, Rx_{r-1}$.

For a receiver Rx_r , the buffer density B_r gives us the fraction of packets which are already seen by Rx_r , prior to the contribution of knowledge differential transmissions. The delivery rate can be found by determining how quickly knowledge differential transmissions can fill the remaining fraction of unseen packets.

5.5 Leader transmissions

In this section, we find the number of mode l leader transmissions of any packet $\mathbf{p}_n$ we can expect, at any time t . All calculations in this section are implicitly dependent on the mode vector β , however, for readability we omit direct mention of β in our notation.

5.5.1 Mode 1 and 2 leader transmissions

We begin with a walkthrough of the mode 1 and 2 leader transmissions of an arbitrary packet $\mathbf{p}_n$. Recall the definitions of a seen packet (Definition 5.12) and a leader transmission (Definition 5.16).

The first time a packet $\mathbf{p}_n$ will be transmitted by the sender is as a mode 1 leader transmission. This will only happen once $\mathbf{p}_n = N_1(t)$, i.e. it is the next needed packet of Rx_1 . The total number of times k that $\mathbf{p}_n$ is transmitted in mode 1 is simply the number of mode 1 transmissions it takes until it is seen by Rx_1 .

Example: Say that at time $t = 10$, the receiver Rx_1 has delivered $\mathbf{p}_1, \dots, \mathbf{p}_7$, so its next needed packet is $N_1(10) = \mathbf{p}_8$. There are mode 1 transmissions at $t = 10, 12, 13$ and the corresponding Rx_1 channel state at these time slots is $\mathbf{x}, \mathbf{x}, \checkmark$. In this case, at times $t = 10, 11, 12$, there will have been $k = 1, 2, 3$ mode 1 leader transmissions of $\mathbf{p}_8$ respectively, after which it is seen by Rx_1 . There are no more mode 1 leader transmissions of $\mathbf{p}_8$ once it is seen. After this time $\mathbf{p}_9$ will become Rx_1 's next needed packet, and there will have been a total of $k = 3$ mode 1 leader transmissions of $\mathbf{p}_8$.

Similarly for mode 2, once $N_2(t) = \mathbf{p}_n$, all mode 2 transmissions will be leader transmissions of $\mathbf{p}_n$ until it is seen by Rx_2 . The difference from the mode 1 case is that some of the earlier mode 1 leader transmissions of a packet $\mathbf{p}_n$ may have been overheard by Rx_2 , therefore there is a probability that $\mathbf{p}_n$ is seen *prior* to any mode 2 transmissions. If this is the case, then $k = 0$ since no mode 2 leader transmissions of $\mathbf{p}_n$ are required.

Example: At time $t = 11$, the receiver Rx_2 has delivered $\mathbf{p}_1, \dots, \mathbf{p}_4$ and also has seen $\mathbf{p}_6$ and $\mathbf{p}_8$. Mode 2 transmissions occur at $t = 11, 14, 17$, and the corresponding Rx_2 channel state at these time slots is $\checkmark, \mathbf{x}, \checkmark$. At $t = 11$, $N_2(11) = \mathbf{p}_5$, and Rx_2 receives the first mode 2 leader transmission of $\mathbf{p}_5$. As a result, there is exactly one mode 2 leader transmission of $\mathbf{p}_5$. Since $\mathbf{p}_6$ is already seen by Rx_2 , it is immediately delivered and there are no mode 2 leader transmissions of that packet. At time $t = 14$, the sender skips straight to $N_2(14) = \mathbf{p}_7$. Rx_2 experiences an erasure at this time slot, so $\mathbf{p}_7$ is transmitted again at $t = 17$, giving $k = 2$ total mode 2 leader transmissions of $\mathbf{p}_7$. Since $\mathbf{p}_8$ is already seen, the sender will skip to $\mathbf{p}_9$ as Rx_2 's next needed packet the next time a mode 2 leader transmission occurs.

From these examples we can make the following observation about mode l leader transmissions.

Property 5.9. *There can only be mode l leader transmissions of a packet $\mathbf{p}_n$ while it is unseen by Rx_l .*

Proof: By Property 5.1, only the receivers' next needed packets can be coded into a transmission. From Property 5.5, the next needed packet of a receiver must always be unseen by that receiver. It follows that there will be no mode l leader transmissions of a packet $\mathbf{p}_n$ that is seen by Rx_l .

$\mathbf{p}_n =$	$\mathbf{p}_1$	$\cdots$	$N_3(t)$	$\cdots$	$N_2(t)$	$\cdots$	$N_1(t)$
$m = 1$	$L_1(k)$						$L_1^*(k)$
$m = 2$	$L_2(k)$				$L_2^*(k)$		
$m = 3$	$L_3(k)$	$L_3^*(k)$					

Table 5.5: A schematic of the three-receiver probability that the sender will transmit k mode l leader transmissions of each packet $\mathbf{p}_n$.

5.5.2 Buffer position and seen packets

We now examine the distribution of k , the number of mode l leader transmissions, in terms of buffer position. At a given time t , a receiver Rx_l has delivered packets $\mathbf{p}_1, \dots, \mathbf{p}_{d_l(t)}$. This means that there have also been mode l leader transmissions of some or all of the packets in the corresponding region of the buffer, as illustrated in Table 5.5.

We define the probability $L_l(k)$ of there being k mode l leader transmissions of a given packet $\mathbf{p}_n$ for $n \leq d_l(t)$. By Approximation 5.5 this probability distribution is independent for each value of l , so the number of leader transmissions k of a packet $\mathbf{p}_n$ in mode l is not correlated to the number of leader transmissions made in other modes.

In contrast, the next needed packet $N_l(t) = \mathbf{p}_{d_l(t)+1}$ has a slightly different probability distribution. Since it is in the process of being delivered, there may or may not have been previous mode l leader transmissions of this packet. Therefore, we allow for this packet to have a potentially different probability of k mode l leader transmissions and denote it by $L_l^*(k)$, the probability that there have been exactly k mode l leader transmissions of the packet so far.

5.5.2.1 Seen packets

From this information, we can make some general comments about the methods by which a next needed packet $N_b(t)$ of a receiver Rx_b can be seen by another receiver Rx_a , based on its position n in the receiver's buffer. We are primarily interested in studying the next needed packets, since only these packets will be encoded by the sender.

If $b > a$ then Rx_a has already seen $N_b(t)$, since by Approximation 5.2 it has already been delivered to Rx_a . If $a > b$, then by Approximation 5.4, the probability of a packet $N_b(t)$ being seen by another receiver Rx_a depends on the number of

leader transmissions of the packet in modes $1, \dots, b$.⁷ Generally, the more leader transmissions of $N_b(t)$ there have been, the more likely it is that Rx_a has already seen this packet.

5.5.3 Calculating $L_l^*(k)$

We begin with the simpler case: finding the probability $L_l^*(k)$ that $N_l(t)$, the next needed packet of Rx_l , has been transmitted k times so far in mode l .

The sender only transmits mode l leader transmissions of a new packet once the previous next needed packet has been seen by Rx_l . Recall from Approximation 5.4 that a next needed packet can be received by two methods:

- Mode l leader transmissions⁸
- Mode $1, \dots, l-1$ knowledge differential transmissions.

The probability of a mode l transmission is simply β_l , while the probability of a mode $i = 1, \dots, l-1$ knowledge differential transmission for Rx_l is given by

$$\gamma_l = \sum_{m=1}^{l-1} \beta_m K_l^m, \quad (5.4)$$

where K_l^i , the probability that Rx_l has a knowledge differential from a mode i transmission, will be calculated in Section 5.6.

So from Rx_l 's perspective, there are four possible actions from the sender at each time slot, which we now list. The sender can transmit:

1. A mode l transmission, with probability β_l
2. A mode $1, \dots, l-1$ knowledge differential transmission, with probability γ_l
3. A mode $1, \dots, l-1$ non-knowledge differential transmission
4. A mode $l+1, \dots, R$ transmission

The probability of each outcome can be multiplied by μ_l or $\bar{\mu}_l$ to give the probability that it will or will not be received by Rx_l , respectively.

Note that once the sender begins mode l leader transmissions of a new packet, it will continue to do so until it is seen, i.e. Rx_l receives either a mode l or mode $1, \dots, l-1$ knowledge differential transmission of that packet. Therefore, $L_l^*(k)$ can

⁷Modes $b+1, \dots, R$ do not contribute leader transmissions because $N_b(t)$ has not yet been transmitted in those modes.

⁸Mode l transmissions are also knowledge differential transmissions, but because we are counting the number of mode l leader transmissions, we treat them separately.

be considered as the probability there have been k unreceived mode l transmissions since the most recently received mode l or knowledge differential transmission.

In this context, mode $1, \dots, l-1$ non-knowledge differential transmissions as well as mode $l+1, \dots, R$ transmissions are not of interest, since they have no impact on the number of mode l leader transmissions. These correspond to items 3 and 4 in our list of sender actions. We can ignore these transmissions by normalising the probability of each event of interest by $\beta_l + \mu_l \gamma_l$, the likelihood that item 1 or 2 from our list will occur.

Then the probability that an unreceived mode l transmission occurs before either a knowledge differential transmission or a received mode l transmission is given by

$$\frac{\bar{\mu}_l \beta_l}{\beta_l + \mu_l \gamma_l} \quad (5.5)$$

and the probability of a received mode l or knowledge differential transmission occurring instead of an unreceived mode l transmission is given by

$$\frac{\mu_l \beta_l + \mu_l \gamma_l}{\beta_l + \mu_l \gamma_l}. \quad (5.6)$$

We are now ready to determine the value of $L_l^*(k)$. For there to have been exactly k prior mode l leader transmissions of Rx_l 's current next packet $N_l(t)$, two requirements must be satisfied. Firstly, for there to have been at least k transmissions of $N_l(t)$, the previous k transmissions of interest must have been unreceived mode l transmissions. Secondly, to ensure that there have been no more than k transmissions of $N_l(t)$, these unreceived mode l transmissions must have been preceded by either a received mode l or knowledge differential transmission. Combining (5.5) and (5.6), $L_l^*(k)$ is then given by

$$L_l^*(k) = \left(\frac{\bar{\mu}_l \beta_l}{\beta_l + \mu_l \gamma_l} \right)^k \frac{\mu_l \beta_l + \mu_l \gamma_l}{\beta_l + \mu_l \gamma_l}. \quad (5.7)$$

5.5.4 Calculating $L_l(k)$

In this part, we calculate the probability $L_l(k)$ that a packet $\mathbf{p}_n$ that has been delivered by Rx_l was transmitted k times in total in mode l . The calculation of $L_l(k)$ depends on two separate probabilities: U_l , the probability that $\mathbf{p}_n$ was unseen by Rx_l from mode $1, \dots, l-1$ leader transmissions of $\mathbf{p}_n$, and $T_l(k)$, the probability that, given $\mathbf{p}_n$ was unseen Rx_l from mode $1, \dots, l-1$ transmissions, there were exactly k mode l leader transmissions of $\mathbf{p}_n$. We now calculate these values.

5.5.4.1 Calculating U_l

We begin by calculating U_l , the probability that a packet $\mathbf{p}_n$ will remain unseen to receiver Rx_l from mode $1, \dots, l-1$ leader transmissions of the packet. The probability that Rx_l will not see $\mathbf{p}_n$ from mode i transmissions, where $i < l$, is given by

$$\sum_{k=0}^{\infty} L_i(k) \bar{\mu}_l^k. \quad (5.8)$$

Under Approximation 5.5, we can assume that seen packets are distributed uniformly across the buffer. Therefore the probability that a packet $\mathbf{p}_n$ will still be unseen by Rx_l from mode $1, \dots, l-1$ transmissions is given by

$$U_l = \prod_{i=1}^{l-1} \sum_{k=0}^{\infty} L_i(k) \bar{\mu}_l^k. \quad (5.9)$$

5.5.4.2 Calculating $T_l(k)$

Next we find $T_l(k)$, the probability that there will be exactly k mode l leader transmissions of a packet $\mathbf{p}_n$, given it was unseen from mode $1, \dots, l-1$ leader transmissions. As in Section 5.5.3, only mode l transmissions and received knowledge differential transmissions impact the value of k . So again, we ignore all other transmissions by normalising the probability of each relevant transmission by $\beta_l + \gamma_l \mu_l$.

If $k = 0$, then a knowledge differential transmission must have been received before any mode l leader transmissions of $\mathbf{p}_n$ could occur. This occurs with probability

$$T_l(0) = \frac{\mu_l \gamma_l}{\beta_l + \mu_l \gamma_l}. \quad (5.10)$$

For $k > 0$, using (5.5) we find that the probability of $k-1$ unreceived mode l transmissions occurring prior to any receptions of knowledge differential or mode l transmissions is given by

$$\left(\frac{\bar{\mu}_l \beta_l}{\beta_l + \mu_l \gamma_l} \right)^{k-1}. \quad (5.11)$$

Following these $k-1$ unreceived mode l transmissions, one of two things can happen to limit the sender to exactly k mode l leader transmissions of $\mathbf{p}_n$.

- A received mode l transmission occurs,
- An unreceived mode l transmission occurs, followed by a received knowledge differential transmission.

The probability that either of two events will occur is given by

$$\frac{\mu_l \beta_l}{\beta_l + \mu_l \gamma_l} + \frac{\bar{\mu}_l \beta_l \mu_l \gamma_l}{(\beta_l + \mu_l \gamma_l)^2}. \quad (5.12)$$

Combining (5.10), (5.11) and (5.12), the probability that exactly k mode l transmissions are required to obtain a previously unseen packet $N_l(t)$ is given by

$$T_l(k) = \begin{cases} \frac{\mu_l \gamma_l}{\beta_l + \mu_l \gamma_l}, & k = 0 \\ \frac{\mu_l \beta_l (\beta_l + \gamma_l) (\bar{\mu}_l \beta_l)^{k-1}}{(\beta_l + \mu_l \gamma_l)^{k+1}}, & k > 0. \end{cases} \quad (5.13)$$

We now scale this by $U_l(k)$, which from (5.9) is the probability that $\mathbf{p}_n$ is unseen after mode $1, \dots, l-1$ transmissions, to obtain the probability that the sender will transmit a packet k times,

$$L_l(k) = \begin{cases} 1 - U_l + U_l T_l(0), & k = 0 \\ T_l(k) U_l, & k > 0. \end{cases} \quad (5.14)$$

5.5.5 Comparison with simulation

We examine the accuracy of our $L_l(k)$ calculations by comparing the calculated and simulation-based probabilities in Fig. 5.5. We observe that calculated and simulation values match extremely well for receivers $l = 1, 2, 3$, but lose some accuracy for Rx₄, although it does follow the same general trend. This can be attributed to the fact that the approximations introduced in Section 5.3 lead to some loss of accuracy, particularly for higher receiver indices.

We now examine the shape of $L_l(k)$, for different values of l . For $l = 1$, every packet must be transmitted at least once for Rx₁ to receive it. Therefore there is no possibility of $k = 0$ for this receiver. The most likely case is $k = 1$, since Rx₁'s high channel rate means the majority of the time, a packet will be received after one transmission. For the same reason, $L_1(k)$ decays relatively quickly as k increases. At higher values of l , however, it is reasonably likely that $k = 0$ because the receiver can overhear the packet from faster receivers' leader transmissions. Although the number of leader transmissions a receiver can overhear increases as the receiver index increases, the receiver's channel rate also decreases. In this simulation, it results in a smaller probability of $k = 0$ as l increases. We also observe that the distribution of $L_l(k)$ decays more slowly for higher values of l . This can also be attributed to the lower channel rates of the corresponding receivers.

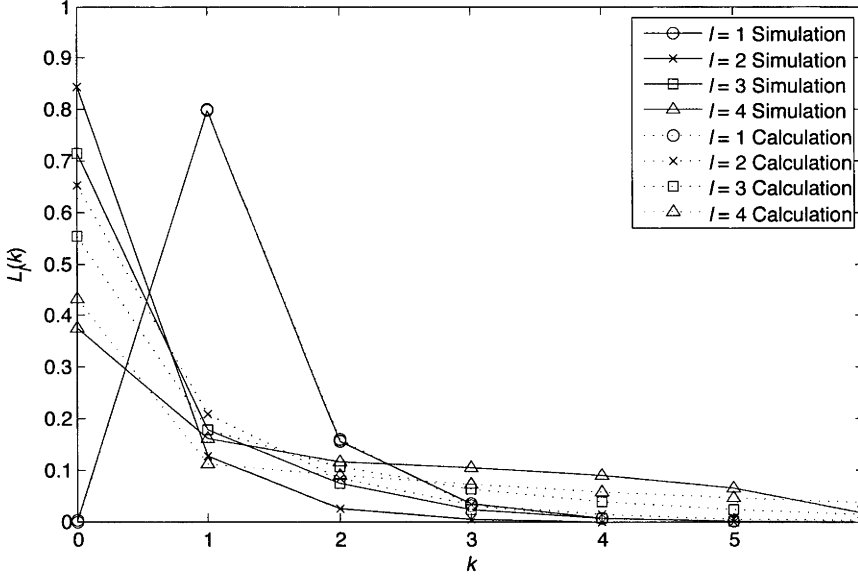


Figure 5.5: Probability of k leader transmissions of a packet, for $R = 4$ receivers, channel rates $\mu = [0.8, 0.6, 0.4, 0.2]$ and mode vector $\beta = [0.85, 0.05, 0.05, 0.05]$.

5.6 Knowledge differentials

In this section, we will determine K_r^m , the probability that a receiver Rx_r obtains a knowledge differential from a mode m transmission, given the mode vector β . Through the dependencies on $L_l^*(k)$ and $L_l(k)$, all calculations in this section are also implicitly dependent on β , but for readability we again omit direct mention of β in our notation.

5.6.1 Knowledge differences

We calculate D_a^b , the probability of a receiver Rx_a having a Rx_b knowledge difference. Recall from Definition 5.18 that if $a \leq b$ then $D_a^b = 1$.

If $b < a$, then the probability D_a^b that a receiver Rx_a has a Rx_b knowledge difference is given by the probability that Rx_a has seen $N_b(t)$ from at least one of the mode $1, \dots, b$ transmissions. This can be determined from the complementary probability of the packet *not* being seen from mode $1, \dots, b$ transmissions, which can be calculated using (5.8) and (5.9).

$$D_a^b = 1 - \left(\sum_{k=0}^{\infty} L_b^*(k) \bar{\mu}_a^k \right) \prod_{l=1}^{b-1} \sum_{k=0}^{\infty} L_l(k) \bar{\mu}_a^k. \quad (5.15)$$

Recall that in mode b transmissions, $N_b(t)$ is the next needed packet of Rx_b and hence

$L_b^*(k)$ is used instead of $L_l(k)$.

5.6.2 Probability of a coding

We next establish the probability $P^m(C)$ that in mode m , a particular coding C will occur. The notation $P^m(\cdot)$ represents the mode m probability of the event inside the brackets.

DEFINITION 5.20: The *subcoding* C_r takes the first r elements of the coding C . C_r is the r -length binary vector representing which of receivers Rx_1 to Rx_r 's next needed packets have been coded into $s(t)$.

Example: For the coding $C = [1, 0, 1, 1]$, the corresponding subcodings are $C_1 = [1]$, $C_2 = [1, 0]$, $C_3 = [1, 0, 1]$ and $C_4 = [1, 0, 1, 1]$.

Combining Property 5.1, Property 5.6 and Approximation 5.3, a packet $N_r(t)$ will only be coded into $s(t)$ if Rx_r has seen all of the other packets coded into $s(t)$. But by Approximation 5.2, receiver Rx_r has already decoded the next needed packets of receivers $Rx_{r+1}, \dots, Rx_R$. Therefore the subcoding C_r corresponds to the portion of the transmission coding C that will affect whether or not $N_r(t)$ is coded into $s(t)$, and therefore the value of $C(r)$. From this information, we can break down finding the probability $P^m(C)$ into the problem of finding the probability, $P^m(C_r)$ for $r = 1, \dots, R$.

In practice, it is usually not necessary to perform calculations starting from $P^m(C_1)$. A mode m transmission will always code the next needed packet of the leader, so

$$P^m(C(m) = 1) = 1. \quad (5.16)$$

Furthermore, since $N_1(t), \dots, N_{m-1}(t)$ will not be coded into a mode m transmission,

$$C(1) = \dots = C(m-1) = 0. \quad (5.17)$$

Combining these two pieces of information, we obtain the result,

$$P^m(C_m) = \begin{cases} 1, & \text{for } C_m = [0, \dots, 0, 1] \\ 0, & \text{for all other } C_m. \end{cases} \quad (5.18)$$

This means that we can start our calculations from C_m , since all possible mode m subcodings begin with $[0, \dots, 0, 1]$. Equation (5.18) can be used as the base case to determine the probability of the other subcodings C_r , for $r = m+1, \dots, R$.

Under Property 5.6 the only way for a packet $N_r(t)$ to be coded is if the corresponding receiver Rx_r has seen all packets coded into the transmission $s(t)$. Then

the probability of $N_r(t)$ being added to $s(t)$ is the probability that Rx_r has a Rx_b knowledge difference for all receivers Rx_b corresponding to packets coded into $s(t)$,

$$\begin{aligned} P^m(C(r) = 1|C_{r-1}) &= \prod_{b \in f_1^m(C_{r-1})} D_r^b \\ &= \prod_{b \in f_1^m(C)} D_r^b, \end{aligned} \quad (5.19)$$

where for a vector v , we define $f_1^m(v)$ as the list of all indices $r \geq m$ where $v(r) = 1$, and $f_0^m(v)$ as the list of all indices $r > m$ where $v(r) = 0$. Note that the second part of this equation is true because $D_r^b = 1$ for all $r \leq b$.

The probability $P^m(C_r)$ of each subcoding C_r can then be defined in terms of the previously calculated probability $P^m(C_{r-1})$,

$$P^m(C(r) = 1) = P^m(C_{r-1})P^m(C(r) = 1|C_{r-1}), \quad (5.20)$$

$$P^m(C(r) = 0) = P^m(C_{r-1})P^m(C(r) = 0|C_{r-1}). \quad (5.21)$$

Combining this pair of equations with (5.19), we can derive the probability of a particular subcoding C_r :

$$P^m(C_r) = \left(\prod_{a \in f_1^m(C_r)} \prod_{b \in f_1^m(C_r)} D_a^b \right) \left(\prod_{a \in f_0^m(C_r)} 1 - \prod_{b \in f_1^m(C_r)} D_a^b \right), \quad (5.22)$$

if the first m elements of C_r are $C_m = [0, \dots, 0, 1]$. Otherwise, $P^m(C_r) = 0$.

5.6.3 Knowledge differentials

We now find the probability K_r^m of a knowledge differential occurring for a receiver Rx_r under mode m transmission.

By Algorithm 8 receiver Rx_r 's next needed packet is coded iff a knowledge differential occurs. This means that we can determine the probability of a knowledge differential by finding the probability that the next needed packet of Rx_r is coded into a transmission,

$$\begin{aligned} K_r^m &= P^m(C(r) = 1) \\ &= \sum_C C(r) P^m(C) \\ &= \sum_{C_r} C_r(r) P^m(C_r). \end{aligned} \quad (5.23)$$

It can be noted that, in the special case where $r = m$, we can substitute (5.18) to

obtain

$$K_m^m = 1. \quad (5.24)$$

Property 5.10. *The probability K_r^m of a mode m knowledge differential for a receiver Rx_r , depends only on the ratio of $\beta_1 : \beta_2 : \dots : \beta_r$.*

Proof: We use induction, starting with the base case of $r = 1$. From (5.24) and Property 5.2, $K_1^1 = 1$, and $K_1^m = 0$ for $m > 1$. This is true regardless of what is happening to the other receivers, so for the fastest receiver corresponding to $r = 1$, Property 5.10 is satisfied.

For each subsequent value of r , we examine the dependency of K_r^m on β using Fig. 5.4. We observe that dependencies on the mode vector β are fed in through two variables: $T_l(k)$ and $L_l^*(k)$, via γ_l . In particular, K_r^m is calculated from $T_1(k), \dots, T_{r-1}(k)$ and $L_r^*(k)$. Therefore we need only prove that $T_l(k)$ and $L_l^*(k)$ depend on the ratio $\beta_1 : \beta_2 : \dots : \beta_l$.

From (5.4), if K_l^i satisfies Property 5.10 for $i < l$, then scaling $\beta_1, \dots, \beta_l$ by some factor $\omega > 0$ will also scale γ_l by a factor of ω . We then make the key observation that, in each case of (5.13), *every term in the numerator and denominator has the same multiplicity of either β_l or γ_l* . This means that multiplying $\beta_1, \dots, \beta_l$ and γ_l by a factor of ω in (5.13) will not change the value of $T_l(k)$. Therefore, scaling $\beta_1, \dots, \beta_l$ by some value ω will not affect the value of $L_l(k)$.

In (5.7), we similarly observe that the same multiplicity of β_l and γ_l in the numerator and denominator of $L_l^*(k)$. Therefore both $T_l(k)$ and $L_l^*(k)$ depend only on the ratio $\beta_1 : \beta_2 : \dots : \beta_l$.

5.6.4 Simulation results

We examine the accuracy of our knowledge differential calculations in Fig. 5.6. We can observe here that, for $r = 1, 2, 3$ the calculated values of K_r^m match the simulated values extremely well, while at $r = 4$ some inaccuracy is observed at $m = 1, 2$. This can be attributed to the cross-receiver effects from Rx_2 and Rx_3 in modes $m = 1, 2$. For $m = 3, 4$ only Rx_3 and Rx_4 's packets are coded. Since only Rx_3 can affect Rx_4 's knowledge differential probability, there are few cross-receiver interactions to take into account when determining K_r^m , so the calculations remain accurate.

The shape of the K_r^m can be explained as follows. For a given receiver Rx_r , K_r^m is a relatively small probability at $m = 1$, that increases as m increases. This increasing probability can be attributed to the lesser number of coded receivers, which makes it more likely that a knowledge differential will occur. At mode $m = r$, K_r^m reaches its maximum value, with $K_r^r = 1$. This matches with (5.24). In modes $m > r$, transmissions are no longer innovative, in line with Property 5.2. As a result, the probability of a knowledge difference is $K_r^m = 0$.

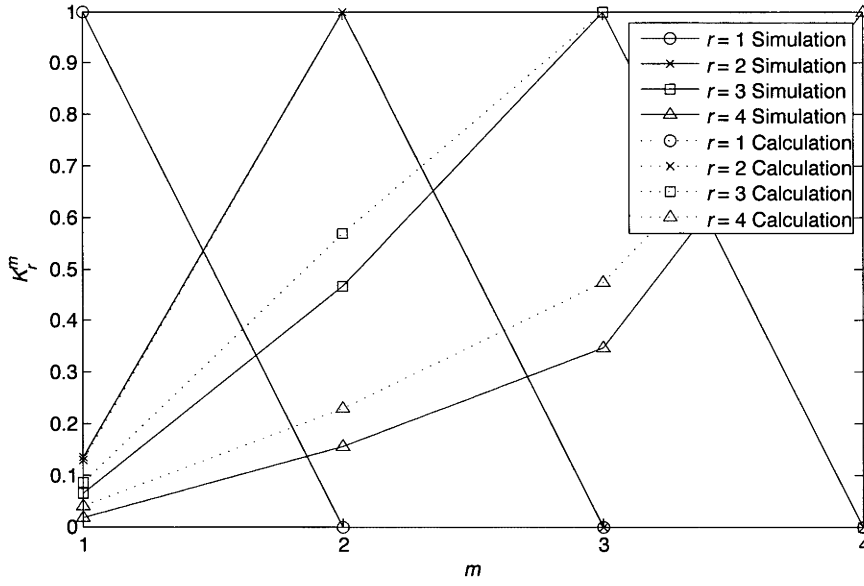


Figure 5.6: Probability K_r^m of a knowledge differential for a receiver Rx_r in mode m , for $R = 4$ receivers, channel rates $\mu = [0.8, 0.6, 0.4, 0.2]$ and mode vector $\beta = [0.85, 0.05, 0.05, 0.05]$.

5.7 Delivery Rates

Now that the probability of K_r^m of a mode m knowledge differential is known, we are ready to estimate R_r , the delivery rate of each receiver Rx_r . It can be observed in Fig. 5.4 that B_r and R_r are recursively dependent on one another.

As outlined in Section 5.3.2, we can divide all transmissions received by Rx_r into three categories. Those that are non-innovative, those that allow the delivery of $N_r(t)$, and those that contribute to the buffer density. We use this information to estimate the delivery rate for the receivers.

5.7.1 Buffer density

We firstly determine B_r , receiver Rx_r 's buffer density. The first step is to determine which of the sender's transmissions can contribute to the buffer density. From Section 5.3.2.1, a transmission must be both innovative and a non-knowledge differential transmission in order to be stored in the packet density region of the buffer. By Property 5.2, transmissions from modes $m = r + 1, \dots, R$ will be noninnovative to Rx_r . From Section 5.6, K_r^m is the fraction of the sender's mode m transmissions that will result in knowledge differentials for Rx_r (for $m \leq r$), and from (5.24) $K_r^r = 1$. Therefore only the $\overline{K_r^m}$ fraction of mode $m = 1, \dots, r - 1$ non-knowledge differential transmissions can contribute to Rx_r 's buffer density.

Since the distribution of leader transmissions studied in Section 5.5 is uniform, it is reasonable to say that within a subregion, the buffer density will also be uniform. Therefore, the buffer density B_r is the average number of non-knowledge differential transmissions received from modes $m = 1, \dots, r-1$ at each packet index in subregion r .

We consider the contribution from each mode $m < r$. For a given mode m , we assume all received mode m non-knowledge differential packets are distributed uniformly between $\mathbf{p}_1, \dots, N_m(t)$. Then the average number of sender transmissions corresponding to each packet index in subregions $m, \dots, R$ is the inverse of Rx_m 's delivery rate, $1/R_m$. Of the β_m fraction of time spent in mode m , $\overline{K_r^m}$ of the packets transmitted will be non-knowledge differential transmissions for Rx_r . These are distributed uniformly across $\mathbf{p}_1, \dots, N_m(t)$ so there is on average β_m/R_m mode m transmissions per packet. Of these, μ_r will be received by Rx_r . So summing over all modes $m = 1, \dots, r-1$,

$$B_r = \mu_r \sum_{m=1}^{r-1} \frac{\beta_m \overline{K_r^m}}{R_m}. \quad (5.25)$$

For $r = 1$ there would be no summation, so we use this as the base case for our recursive calculation,

$$B_1 = 0. \quad (5.26)$$

5.7.2 Delivery rate

The task of delivering new packets to a receiver Rx_r is equivalently the process of making the unseen packets in the buffer seen to Rx_r , using knowledge differential transmissions. On average, B_r of the undelivered packets in the buffer have already been seen by the receiver. Therefore it is necessary to fill in the remaining $(1 - B_r)$ fraction of packets using knowledge differential transmissions. On average, μ_r of the knowledge differential transmissions will be received. Therefore, the delivery rate is given by

$$R_r = \frac{\mu_r \sum_{m=1}^r \beta_m K_r^m}{1 - B_r}. \quad (5.27)$$

Here we must sum the knowledge differentials contributed by each mode. Note that the summation only goes from 1 to r , since from Property 5.2, a mode m transmission can only be innovative to receivers $Rx_m, \dots, Rx_R$.

We can observe from (5.25) and (5.27) that there is a recursive relationship between the delivery rates of the receivers. Note that B_r only depends on $R_1, \dots, R_{r-1}$ and K_r^m , calculated in Section 5.6, while R_r only depends on B_r and K_r^m . Therefore starting from the Rx_1 base case, where the delivery rate is $R_1^m = \beta_1 \mu_1$ and the buffer

density is $B_1^m = 0$, we can sequentially calculate the delivery rates of all receivers.

It is also important to observe the dependency of a receiver Rx_r 's delivery rate R_r on the parameters β_i and μ_i , for $1 \leq i < r$. As an example, in the extreme case, if receiver Rx_1 has channel rate $\mu_1 = 1$ and mode vector is $\beta = [1, 0, \dots, 0]$, then we find that the delivery rate of the other receivers is $R_2 = \dots = R_R = 0$. This is the case because if Rx_1 never fails to receive a packet, there are no opportunities for knowledge differences, and therefore knowledge differentials, to occur. Without knowledge differentials, intermediate packet delivery is not possible. However we can compensate for these effects using the mode vector β . By reducing the time spent in mode 1 and redistributing it to the other modes, we reduce the delivery rate of Rx_1 but increase the delivery rate of the slower receivers.

5.7.3 Rate calculation algorithm

The process for calculating the delivery rate has many recursive dependencies that make it difficult to determine how to implement. Therefore, we provide Algorithm 10 to outline the order in which variables shown in Fig. 5.4 can be calculated.

Making use of the fact that a receiver's delivery rate is only influenced by faster receivers, we sequentially calculate the delivery rate R_r for each receiver Rx_r , starting from $r = 1, \dots, R$. Following the same structure as Sections 5.5 to 5.7.2, we perform the leader transmission calculations, followed by knowledge differential calculations and finally the delivery rate calculations.

It is interesting to note that, in the special case of $r = 1$, the complexity of the calculations reduces significantly.⁹ Because there are no earlier receivers for Rx_1 's behaviour to be influenced by, the calculation dependencies illustrated by curved arrows in Fig. 5.4 can be removed.

5.7.4 Simulation results

We now compare the simulated and calculated delivery rates for the receivers. Fig. 5.7 illustrates the simulated and calculated delivery rates of a set of receivers under the settings listed in Table 5.6. Observe that, under these settings, the delivery rates of the faster receivers greatly exceed the min-cut of the network, given by the slowest receiver's channel rate. For example, under Setting A, the delivery rates of Rx_1 and Rx_2 are 0.68 and 0.28 respectively, which are both much larger than the network min-cut of 0.2. Note, however, that this is achieved at the expense of the slower receivers, who experience delivery rates below the network min-cut. Under Setting

⁹In our calculations, empty summations are equal to zero, and empty products are equal to one.

Algorithm 10 Delivery rate calculation.

- 1: Set $D_a^b = 1$ for all $a \geq b$.
 - 2: Set $K_r^m = 1$ for all $r = m$, and $K_r^m = 0$ for $r < m$.
 - 3: **for** $m = 1, \dots, R$ **do**
 - 4: Calculate γ_m , using $K_m^1, \dots, K_m^{m-1}$, by (5.4).
 - 5: Calculate $L_m^*(k)$ using γ_m , by (5.7).
 - 6: Calculate U_m using $L_1, \dots, L_{m-1}(k)$, by (5.9).
 - 7: Calculate $T_m(k)$ using γ_m , by (5.13).
 - 8: Calculate $L_m(k)$ using U_m and $T_m(k)$, by (5.14).
 - 9: **for** $a = m + 1, \dots, R$ **do**
 - 10: Calculate D_a^m using $L_1(k), \dots, L_{m-1}$ and $L_m^*(k)$, by (5.15).
 - 11: **end for**
 - 12: **for** $n = 1, \dots, m$ **do**
 - 13: **for all** C_m **do**
 - 14: Calculate $P^n(C_m)$ using D_a^b for $b < a < m$, by (5.22).
 - 15: **end for**
 - 16: **for** $r = n, \dots, m$ **do**
 - 17: Calculate K_r^n using all $P^n(C_r)$, by (5.23).
 - 18: **end for**
 - 19: **end for**
 - 20: Calculate B_m using $K_m^1, \dots, K_m^{m-1}$ and $R_1, \dots, R_{m-1}$, by (5.25).
 - 21: Calculate R_m using $K_m^1, \dots, K_m^m$ and B_m , by (5.27).
 - 22: **end for**
-

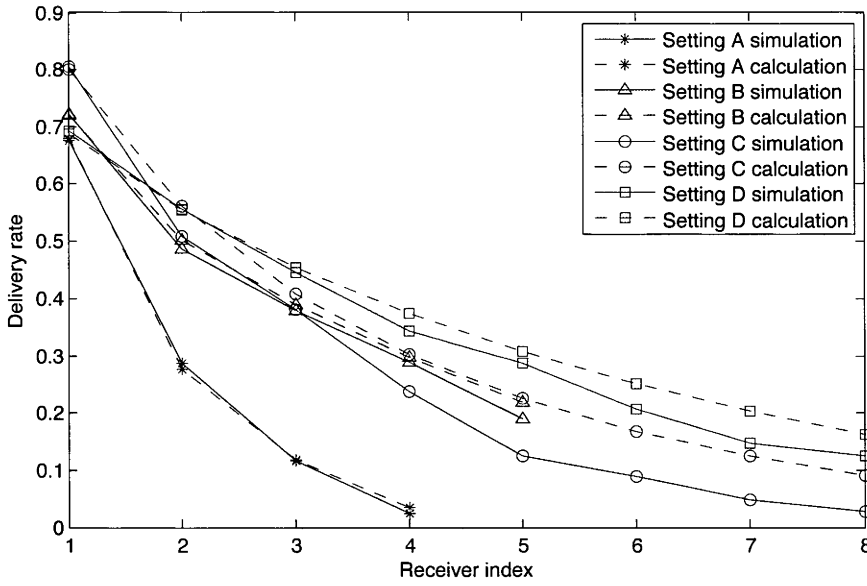


Figure 5.7: Probability of a knowledge differential for a receiver Rx_r in mode m , for the simulation settings given in Table 5.6.

A, for example, Rx_3 and Rx_4 experience delivery rates of 0.12 and 0.035 respectively, which are well below the network min-cut.

Under Settings A and B, the calculated delivery rates match well with the values observed in simulation. In settings C and D on the other hand, the approximate calculations are fairly accurate for the first few receivers, but gradually lose accuracy for higher receiver indices. The calculated values tend to be a little optimistic as the receiver index grows, since at higher receiver indices, Approximation 5.3 loses its accuracy. Setting C represents the extreme case, where all sender transmissions are allocated to mode 1. For a receiver Rx_r , the closer the mode m is to r , the more accurate the knowledge differential estimate will be. Therefore the greater the amount of time allocated to higher modes, the more accurate the calculated rates will be.¹⁰

It is interesting to note that, for Setting A, despite some slight inaccuracies in the knowledge differential estimates of Fig. 5.6, the delivery rates under both simulation and calculation match quite well. In part, this can be attributed to the inverse relationship between the buffer density and the probability of knowledge differential that can be observed in (5.25). As a result of this relationship, inaccuracies in the calculated probability of a knowledge differential are partially mitigated by their inverse effect on the buffer density.

¹⁰Note, however, that β must be restricted to values that maintain the ordering of the receivers under Approximation 5.2.

Setting	Receivers R	Channel rate μ	Mode vector β	Min-cut
A	4	$[0.8, 0.6, 0.4, 0.2]$	$[0.85, 0.05, 0.05, 0.05]$	0.2
B	5	$[0.9, 0.8, 0.7, 0.6, 0.5]$	$[0.8, 0.05, \dots, 0.05]$	0.5
C	8	$[0.8, 0.75, 0.7, \dots, 0.45]$	$[1, 0, \dots, 0]$	0.45
D	8	$[0.8, 0.75, \dots, 0.45]$	$[0.86, 0.02, \dots, 0.02]$	0.45

Table 5.6: Simulation conditions used in Fig. 5.7, and the corresponding min-cut values.

5.8 Fairness

We now use the delivery rate analysis of Sections 5.6 and 5.7 to determine the value of the mode vector β that will result in fair delivery rates for all receivers. We define fairness as follows:

DEFINITION 5.21: A transmission scheme is *fair* for all receivers if their delivery ratios are equal, so that

$$\frac{R_1}{\mu_1} = \frac{R_2}{\mu_2} = \dots = \frac{R_R}{\mu_R}. \quad (5.28)$$

Our aim then is to achieve equal *delivery ratios* for all receivers, where the delivery ratio Q_r of a receiver Rx_r is defined as

$$Q_r = \frac{R_r}{\mu_r} \quad (5.29)$$

In practice, the method we will outline can be modified to equalise any set of delivery ratios, so long as doing so will maintain the condition

$$R_1 \geq R_2 \geq \dots \geq R_R. \quad (5.30)$$

5.8.1 The rate scaling property

Before we can begin, it is necessary to highlight the key property of our transmission scheme that will help us build our fairness algorithm.

Property 5.11. *Scaling $\beta_1, \dots, \beta_r$ by some amount $\omega > 0$ does not affect the buffer density of receivers $Rx_1, \dots, Rx_r$, and scales their delivery rates by ω .*

Proof: We will prove this property using induction. Let us substitute $\beta'_m = \omega\beta_m$ for all $m = 1, \dots, r$ into (5.27) and (5.25). Using the notation $B_r(\beta)$ and $R_r(\beta)$ to show the explicit dependency on the mode vector β , we obtain

$$B_r(\beta') = \mu_r \sum_{m=1}^{r-1} \frac{\omega\beta_m \overline{K}_r^m}{R_m(\beta')} \quad (5.31)$$

$$R_r(\beta') = \frac{\sum_{m=1}^r \omega \beta_m K_r^m}{1 - B_r(\beta')} \quad (5.32)$$

As the base case, $R_1(\beta') = \omega \beta_1 \mu_1$ and $B_1(\beta') = 0$. $R_1(\beta')$ is linearly dependent on ω , and $B_1(\beta')$ is a constant. Therefore Property 5.11 holds for the base case.

We now show that, if Property 5.11 is true for $R_{x_1}, \dots, R_{x_{r-1}}$, then it also holds for R_{x_r} . From Property 5.10, we know that scaling $\beta_1, \dots, \beta_r$ by ω will not affect the probability K_r^m of a knowledge differential. Now, assume that $R_m(\beta') = \omega R_m(\beta)$ and $B_m(\beta') = B_m(\beta)$ for $m < r$. Then by substituting these values into (5.32) and (5.31), we obtain

$$\begin{aligned} B_r(\beta') &= \mu_r \sum_{m=1}^{r-1} \frac{\omega \beta_m \overline{K_r^m}}{\omega R_m(\beta)} \\ &= B_r(\beta) \end{aligned} \quad (5.33)$$

and

$$\begin{aligned} R_r(\beta') &= \frac{\sum_{m=1}^r \omega \beta_m K_r^m}{1 - B_r(\beta)} \\ &= \omega R_r(\beta). \end{aligned} \quad (5.34)$$

Therefore, Property 5.11 holds for all receivers.

This is a key result since it tells us that, for a receiver R_{x_r} , scaling the time spent in modes $1, \dots, r$ by a factor of ω will scale the delivery rates $R_1, \dots, R_r$ by a factor of ω as well. This will give us a predictable way to determine the receivers' delivery rates as we adjust the value of the mode vector β .

5.8.2 Fairness algorithm

We now outline the fairness algorithm used to determine the value of β required to achieve delivery rate fairness. Recall that, in Section 5.7, we calculated both the expected buffer density and delivery rates, given β . The aim now is to determine the value of β that will result in fair delivery rates for all receivers.

The receivers' delivery ratios can be matched using the procedure outlined in Algorithm 11. This fairness algorithm performs two functions as it iterates through each value of r . Firstly, it scales down the amount of time allocated to modes $1, \dots, r-1$ by the calculated quantity $1 - \omega[r]$. Secondly, it allocates the remaining $\omega[r]$ fraction of time to mode r . By Properties 5.2 and 5.11, neither of these actions affects the delivery ratio of receivers $1, \dots, r-1$. Therefore, Algorithm 11 ensures that equalised delivery ratios will remain matched through subsequent iterations of the algorithm. After the r^{th} iteration is complete, $\beta[r]$ will allow the sender to achieve $Q_1[r] = \dots = Q_r[r]$.

5.8.2.1 Notation

To reflect their changing values at each iteration, we append all variables with an index $[r]$ to represent the updated values as of the r^{th} iteration. $\beta_m[r]$ represents the time allocated to mode m after iteration r , and $\omega[r]$ is an equalisation factor which determines $\beta[r]$ based on $\beta[r-1]$, the value at the previous iteration. The delivery ratio $Q_m[r]$ of each receiver Rx_r is also determined from $\beta[r]$. At each iteration r , the value $\beta[r]$ is updated to match the delivery ratio of Rx_r to that of receivers $Rx_1, \dots, Rx_{r-1}$. In the remainder of this section, we will explicitly show any dependencies on $\beta[r]$, for example $K_r^m(\beta[r])$ represents receiver Rx_r 's mode m knowledge differential probability given that $\beta[r]$ is used as the mode vector.

Algorithm 11 Fairness algorithm.

- 1: Set $\beta_1[1] = 1$, and $\beta_2[1], \dots, \beta_R[1] = 0$.
 - 2: **for** $r = 2$ to R **do**
 - 3: Calculate $Q_{r-1}[r-1]$ and $Q_r[r-1]$ using (5.29), substituting $\beta = \beta[r-1]$.
 - 4: Calculate $\omega[r]$ from (5.37).
 - 5: Set $\beta_r[r] = \omega[r]$
 - 6: **for** $m = 1$ to $r-1$ **do**
 - 7: Set $\beta_m[r] = (1 - \omega[r])\beta_m[r-1]$
 - 8: **end for**
 - 9: **end for**
 - 10: Set $\beta = \beta[r]$.
-

5.8.2.2 Calculating the mode vector β

We begin by allocating all transmissions to mode 1, so that $\beta_1[1] = 1$, and $\beta_m[1] = 0$, for $m = 2, \dots, R$. By (5.24), all mode 1 transmissions are knowledge differential transmissions for Rx_1 . Therefore the delivery ratio for Rx_1 after iteration 1 is $Q_1[1] = 1$.

In each iteration $2 \leq r \leq R$ of the for loop, we find a value of $\beta[r]$ which will achieve $Q_1[r] = \dots = Q_r[r]$, equalising the delivery ratio of Rx_r with that of receivers Rx_1 to Rx_{r-1} . This is determined by reducing the time spent in modes $1, \dots, r-1$ by a factor of $(1 - \omega[r])$ to allow a fraction of time $\omega[r]$ to be spent in mode r .

By Property 5.11, this has the effect of scaling $Q_1[r], \dots, Q_{r-1}[r]$ by a factor $1 - \omega[r]$, so that

$$Q_m[r] = (1 - \omega[r])Q_m[r-1] \quad (5.35)$$

for all $m < r$.

Combining (5.27), (5.29), Property 5.11 and (5.24), the delivery ratio of receiver

Rx_r in iteration r is given by

$$\begin{aligned} Q_r[r] &= \frac{\sum_{m=1}^r \beta_m[r] K_r^m(\beta[r])}{1 - B_r[r]} \\ &= (1 - \omega[r]) Q_r[r-1] + \frac{\omega[r]}{1 - B_r(\beta[r-1])}. \end{aligned} \quad (5.36)$$

Therefore to equalise the delivery ratios of Rx_r and Rx_{r-1} the task is to find the value of $\omega[r]$ for which $Q_r[r] = Q_{r-1}[r]$. This can be solved using (5.35) and (5.36) to give

$$\omega[r] = \frac{Q_{r-1}[r-1] - Q_r[r-1]}{Q_r[r-1] - Q_{r-1}[r-1] + \frac{1}{1 - B_r(\beta[r-1])}}. \quad (5.37)$$

For $m = 1, \dots, r-1$ the new values of $\beta_m[r]$ are now given by

$$\beta_m[r] = \begin{cases} (1 - \omega[r]) \beta_m[r-1], & m < r \\ \omega[r], & m = r \\ 0, & m > r \end{cases} \quad (5.38)$$

This process is repeated for each receiver to equalise the delivery ratio of each subsequent receiver, until all receivers are equal. After this process is complete, the amount of time that should be spent in each mode m to achieve fairness is given by

$$\beta = \beta[R]. \quad (5.39)$$

The delivery ratio of every receiver Rx_r using the mode vector β will be equal. Substituting (5.24) and (5.26) into (5.27), the delivery rate of Rx_1 is found to be $\mu_1 \beta_1$, so by (5.29), the delivery ratio of Rx_1 is $Q_1 = \beta_1[R]$. Therefore, using the mode vector β will give the delivery ratios

$$Q_1 = Q_2 = \dots = Q_R = \beta_1[R]. \quad (5.40)$$

5.8.3 Simulation results

We can now compare the delivery rates achieved both in theory and in practice. The channel rates for each setting we will investigate are given in Table 5.7. Observe from Fig. 5.9 that, similar to the results of Fig. 5.7, a number of receivers are able to experience channel rates above the min-cut given in Table 5.7.

In Fig. 5.8 we can observe the delivery rate estimates after each iteration r in Algorithm 11. At each iteration r , the theoretical delivery rates are those that the sender would enforce if it only took into consideration the delivery ratios of the

Table 5.7: The settings used in fairness algorithm simulations, and their corresponding min-cuts.

Setting	Receivers R	Channel rate μ	Min-cut
A	4	$[0.8, 0.7, 0.6, 0.5]$	0.5
B	5	$[0.9, 0.8, \dots, 0.5]$	0.5
C	8	$[0.9, 0.85, \dots, 0.55]$	0.55

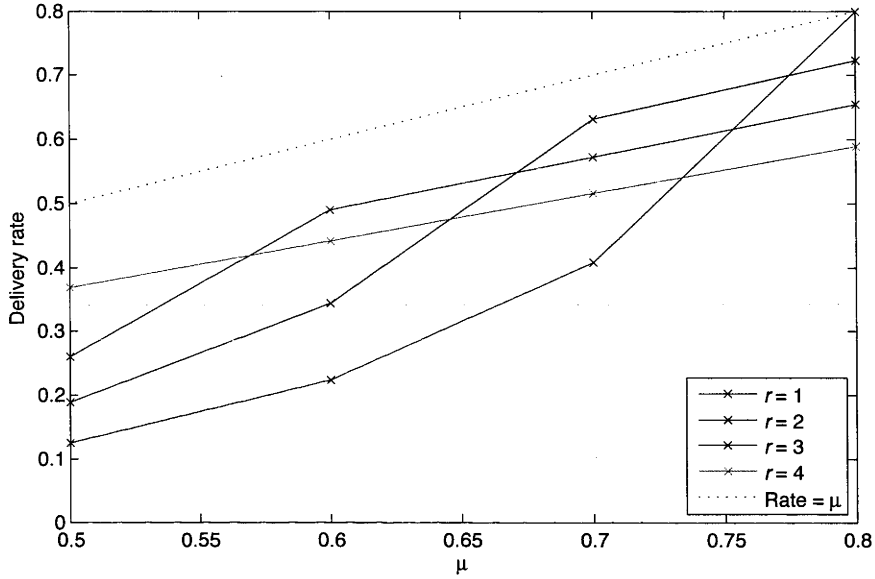


Figure 5.8: Calculated delivery rates achieved after each iteration r of Algorithm 11, under Setting A from Table 5.7. These are compared against μ , the channel rate for each receiver.

first r receivers. At iteration $r = 1$, Rx_1 achieves the best possible performance with delivery rate $R_1 = \mu_1$, however, the other receivers' delivery delays are significantly worse. At each subsequent value of r , the delivery rates of receivers $Rx_1, \dots, Rx_{r-1}$ are scaled down to allow Rx_r to match the delivery ratios of receivers $Rx_1, \dots, Rx_{r-1}$.

It is interesting to note that, after iteration r , there is improvement to the delivery rates of *all* mode r receivers (that is receivers Rx_r to Rx_R), not simply Rx_r . This occurs because in higher modes, the transmission $s(t)$ will be coded from a smaller set of receivers, whose next needed packets are more likely to have been seen. This improves the likelihood that a receiver will obtain a knowledge differential.

In Fig. 5.9 we compare the calculated and simulated delivery rates under the values of β determined from Algorithm 11. Similar to Fig. 5.7, we observe that for low receiver indices, estimates match well with calculated values, while some loss of accuracy is observed for higher receiver indices.

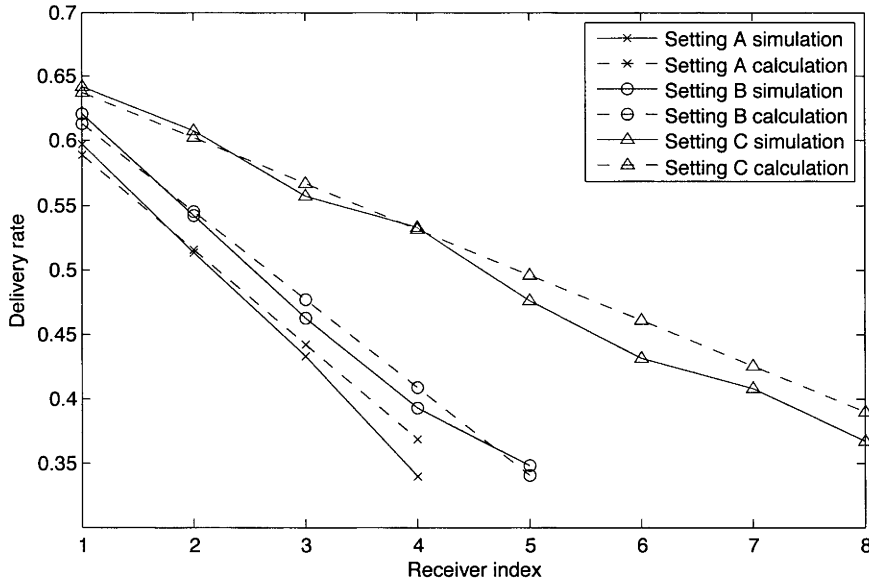


Figure 5.9: Calculated and simulated delivery rates under the settings of Table 5.7, using the values of mode vector β derived in Algorithm 11.

5.9 Conclusion

In this chapter we have demonstrated that intermediate packet delivery can be utilised to exceed the min-cut capacity of a broadcast packet erasure network. In particular, we showed that nonzero packet delivery rates could be maintained for all receivers, while the sender's transmission rate, as well as the packet delivery rate of some receivers, was able to exceed the channel rate of the slowest receiver. By implementing a deterministic coding scheme, we were able to achieve multi-rate delivery of the same set of ordered information to receivers with nonuniform channel rates.

Despite the complexity associated with the analysis of deterministic coding schemes, through the use of some approximations we were able to obtain tractable estimates of the receivers' delivery rates. These estimates were shown to be reasonably accurate, although some loss of accuracy was observed with a larger number of receivers. Based on our approximate analysis, we introduced a fairness algorithm that determined how the sender should allocate its resources to achieve fair delivery rate performance for the receivers.

While the rate analysis we have presented in this chapter lends insight into the process of multi-rate delivery, it is specific to the network coding scheme we have studied. It remains unclear whether higher delivery rates could in fact be achieved using other deterministic transmission schemes. Future work might include the more

general characterisation of the capacity of a broadcast network, given the relaxation of the complete information acquisition requirement. In its current form, the transmission scheme we have outlined is not suitable for practical applications, due to its large memory and computational requirements. In the following chapter, we will outline a more practical transmission scheme, which utilises intermediate packet delivery in a network with uniform channel rates.

Video on demand using intermediate packet delivery

In this chapter, we build on the intermediate packet delivery results of Chapter 5, studying its effects on the delivery rates and delay experienced by a network with uniform channel rates. We show through analysis that the use of intermediate packet delivery allows transmission optimality to be achieved, where every transmission is guaranteed to provide new information to all receivers. Receivers are able to maintain in-order packet delivery at the channel rate.

We illustrate how intermediate packet delivery can be used to provide multicast video on demand using a single server channel. Receivers are permitted to join the broadcast at any time, and after an initial startup delay, are able to deliver packets at their channel rate. We analyse the receivers' performance, providing a lower bound for the average startup delay. We anticipate that this transmission scheme can be combined with existing multicast video on demand algorithms to reduce both the latency and number of server channels required for implementing video on demand services, and increase the capacity of the system.

6.1 Introduction

Much research has been devoted to the implementation of *video on demand* [36]. This application requires that the packets of a video be made available from start to finish to a set of receivers, who are permitted to join the broadcast at any time. Solutions must be bandwidth efficient, allow the video to be viewable in a timely manner, and ensure that the packets of the video are delivered in the correct playback order.

Numerous methods for implementing multicast video on demand have been proposed in the literature [36]. These systems generally make use of a number of *server channels* to distribute each video, each of which is responsible for broadcasting a

different component of the video at any given time.

In *server initiated* algorithms, for example [39, 76], the video is split up across different server channels. Each server channel sequentially transmits the packets in its designated portion of the video, looping back to the start once it reaches the end of its portion. Receivers are able to view the video by subscribing to the appropriate channels. Any number of receivers can subscribe to a given video, however, upon joining, receivers often experience a *startup delay*, as they must wait until the relevant server channel loops back to the start of the stream before they can begin watching the video.

In contrast, *client initiated* algorithms such as [1, 28] assign server channels dynamically, according to users' requests for videos. Making use of batching techniques, these client initiated algorithms can provide instantaneous playback, as seen in [28], however, the number of channels required to support this service increases with the number of users.

The majority of the proposed algorithms are uncoded, although network coding has been considered in works such as [9, 48] in the context of peer-to-peer video on demand. These multicast video on demand solutions rely on at least two server channels, but often more, for implementation. The use of intermediate packet delivery in network coding, which we introduced in Chapter 5, has not yet been explored for such applications. We will demonstrate in this chapter that, if receivers are willing to tolerate some delay, multicast video on demand can be implemented using only a single server channel. By making use of intermediate packet delivery, we will show that it is possible to simultaneously deliver the same video asynchronously to multiple receivers.

6.1.1 Contributions and related work

In Chapter 5, we showed for the first time that intermediate packet delivery is consistently achievable under network coded transmission. We demonstrated that under a lossy single hop network, the same set of packets could be simultaneously delivered at multiple rates to a set of receivers with nonuniform channel rates. The assumptions that were required in Chapter 5 meant that results were limited to the case where receivers had strictly different channel rates and joined the broadcast at the same time. In this chapter, we extend this work to cover the case where receivers have a uniform channel rate, and receivers are permitted to join the broadcast at different times.

In this chapter we will demonstrate that it is possible to use network coding to provide video on demand services using a single multicast server channel. We

demonstrate how, despite joining at different times, all receivers are able to deliver packets at full rate. The transmission scheme we outline is fully *transmission optimal*, meaning that every received transmission provides innovative information to every receiver. This has been achieved in [18] for the two receiver case, but does not generalise to larger numbers of receivers. The transmission scheme we discuss in this chapter is the first to permit any number of receivers to continuously deliver packets at their channel rate.

6.2 System Model

The model we study in this chapter is similar to that of Section 5.2, with the key difference that, in this chapter, we model a system where receivers experience uniform channel rates. For the sake of completeness, we describe the system model in full. A single sender aims to transmit an infinite backlog of data packets $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \dots$ in the correct order to a set of R receivers, $Rx_1, Rx_2, \dots, Rx_R$. Time is slotted, denoted by $t = 1, 2, \dots$ and the sender broadcasts packets at the rate of one network coded packet per time slot. The receivers are connected to the sender via independent erasure channels with *channel rate* μ , so that they successfully receive transmissions independently with probability μ at each time slot. Each receiver Rx_r joins the broadcast at time t_r . Without loss of generality we set $t_1 = 0$, and order the receivers so that $t_1 \leq t_2 \leq \dots \leq t_R$.

Each receiver stores the packets it receives, along with the corresponding coding coefficients, in its own buffer. To decode the original packets, receivers apply Gaussian elimination [3, 42], using information from the coding coefficients to recover the original uncoded packets. A more detailed summary of the procedure can be found in Section 2.1. After each packet transmission, the receiver sends a notification to indicate whether or not the packet was successfully received. For simplicity of analysis we assume the sender detects these notifications without error or delay. The sender uses this information to record which packets each receiver has stored in its buffer. Based on this information, a transmission scheme is used to determine the packet combinations the sender will transmit.

6.2.1 Definitions

For quick reference, we first provide a list of some of the basic terminology that will be used throughout this chapter.

DEFINITION 6.1: The packet $\mathbf{p}_n$ has a *packet index* n .

DEFINITION 6.2: A packet $\mathbf{p}_n$ is *newer* than a packet $\mathbf{p}_m$ if its packet index is higher, i.e. $n > m$. The packet $\mathbf{p}_n$ is *older* if its packet index is lower.

DEFINITION 6.3: A packet is *delivered* to a receiver Rx_r when it and all older packets have been decoded.

DEFINITION 6.4: A packet is *innovative* to a receiver Rx_r if it provides new information about an undecoded packet.

DEFINITION 6.5: At any time t , the *next needed* packet $N_r(t)$ of a receiver Rx_r is the oldest packet $\mathbf{p}_n$ which has not been delivered to that receiver.

DEFINITION 6.6: A receiver Rx_r is *faster* than another receiver Rx_i if its next needed packet is newer. Rx_i is then said to be *slower* than Rx_r .¹

DEFINITION 6.7: At a given time t , the fastest receiver is called the *leader*.

6.2.2 Transmission scheme

As shown in Fig. 6.1, a single block, coding scheme B, is used to determine each transmission $s(t)$. For reference, the decision process used for coding scheme B is given in Algorithm 12. All packets from the packet backlog are available for transmission as needed. This model is almost identical to the transmission scheme used in Chapter 5, except that the mode selection block has been removed, coding scheme B is used, and the channel rates are uniform. At each time slot, the sender orders the receivers from fastest to slowest. Starting with $s(t) = 0$, the sender transmission is sequentially updated to ensure that $s(t)$ is innovative to each receiver. If the transmission so far is non-innovative to a particular receiver Rx_r , $s(t)$ is updated to include that receiver's next needed packet. Because packets are only added if they are necessary to ensure $s(t)$ is innovative, Algorithm 12 ensures that the sender transmits parsimonious combinations of the next needed packets of the receivers.

Algorithm 12 has a few simple properties which will be useful throughout this chapter.

Property 6.1. *Each transmission $s(t)$ is coded from the next needed packets of the receivers.*

Property 6.2. *The next needed packet of a receiver Rx_r is only coded into $s(t)$ if it allows Rx_r to decode that packet from $s(t)$ upon successful reception.*

Property 6.3. *The next needed packet of the leader will always be coded into each transmission.*

¹Note that due to fluctuations in the receivers' channel conditions, their relative progress may change over the course of the broadcast. This differs from Chapter 5, where it was assumed that the receivers maintain a fixed ordering from fastest to slowest.

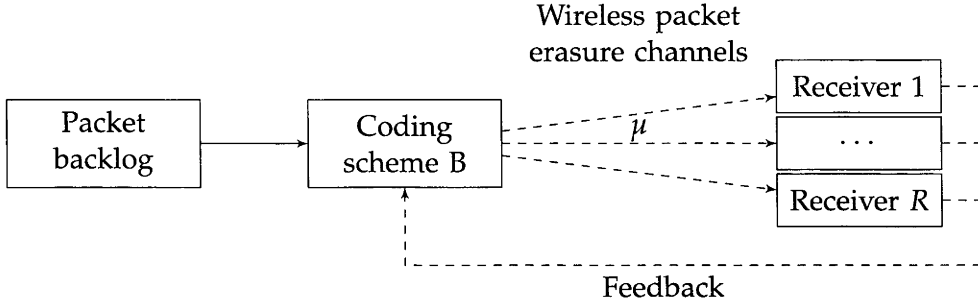


Figure 6.1: A block diagram of the components of the transmission scheme.

Algorithm 12 Coding algorithm from [71].

- 1: Organise receivers $Rx_1, \dots, Rx_R$ into groups G_j , so that G_j contains all receivers Rx_i whose next needed packet $N_i(t) = \mathbf{p}_j$. Let G be the set of all groups G_j which contain at least one receiver.
 - 2: Initialise $s(t) = 0$.
 - 3: **for** each group $G_j \in G$, from high to low j , **do**
 - 4: Initialise the empty *veto list* $v_j = \{\}$.
 - 5: **for** each receiver $Rx_i \in G_j$ **do**
 - 6: Calculate r_i , the result of performing Gaussian elimination on $s(t)$ with the packet in Rx_i 's buffer.
 - 7: **if** $r_i = 0$ **then**
 - 8: Add 0 to v_j .
 - 9: **else if** $r_i = \alpha \mathbf{p}_j$ for some field element α **then**
 - 10: Add α to v_j .
 - 11: **end if**
 - 12: **end for**
 - 13: **if** $0 \in v_j$ **then**
 - 14: Set a_j to the smallest value not listed in v_j .
 - 15: Set $s(t) = s(t) + a_j \mathbf{p}_j$.
 - 16: **end if**
 - 17: **end for**
-

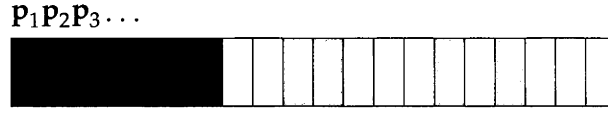


Figure 6.2: Example illustration of the contents of a receiver's buffer. Each packet $p_1, p_2, \dots$ is allocated a position in the buffer, from left to right. Dark coloured packets have been delivered, and grey packets have been seen but not delivered. White packets have not yet been seen by the receiver.³

6.2.3 Buffer model

To help the reader understand the packet delivery process, we use a buffer model, similar to the model outlined in Section 5.3.2, to describe which packet combinations are known by each of the receivers. Imagine that each receiver has an infinite buffer that has one space allocated for each packet in the sender's backlog. The n^{th} position of this buffer corresponds to p_n in the sender's backlog, as illustrated in Fig. 6.2.

DEFINITION 6.8: A packet p_n has been *seen* if a network coded packet of the form $p_n + c$ is known, where c is some network coded combination of older packets $p_1, \dots, p_{n-1}$.

If packet p_n has been seen by a receiver, then the corresponding packet combination is stored at position n in the receiver's buffer.

Gaussian elimination is performed on the received packets as they are stored in the receivers' buffers. Each packet is stored in a receiver's buffer according to the newest packet coded into the transmission, after Gaussian elimination has been applied. The buffer model has the following property.

Property 6.4. *Once packets $p_1, \dots, p_n$ have been seen by a receiver, they have also been delivered.*

Proof. Once a receiver has obtained n coded combinations of n unknown packets, it has a sufficient amount of information to decode all packets. $\square$

6.2.4 Organisation of the chapter

In the remainder of this chapter, we will study the delivery behaviour of the receivers under the transmission scheme we have outlined. To achieve this, we divide the problem into two parts. In Section 6.3, we estimate the probability of *knowledge*

³This is similar to the buffer model introduced in Chapter 5, but because only mode 1 transmissions are used, only one buffer region exists.

differential transmissions, the key feature of our transmission scheme which allows a single channel multicast implementation. Based on these results, in Section 6.4 we find both the delivery rate and the expected number of undelivered packets at each receiver.

6.3 Knowledge differential transmissions

In this section we will estimate the probability of *knowledge differential transmissions* for any given receiver.

DEFINITION 6.9: For a receiver Rx_r , a transmission $s(t)$ is classified as a *knowledge differential transmission* if the portion of $s(t)$ coded for faster receivers $Rx_1, \dots, Rx_{r-1}$ can be decoded by Rx_r using the information stored in its buffer.

Successfully receiving a knowledge differential transmission allows a receiver to decode its next needed packet.

6.3.1 Approximations

The analysis of this transmission scheme, while simpler than that of Chapter 5, is still complicated by the feedback-related dependencies between the sender and receivers. Therefore, many (but not all) of the approximations from Chapter 5 are carried over into our analysis here. We provide an overview of the approximations that are used in this chapter to estimate the probabilities of knowledge differential transmissions.

The reader may recall from Chapter 5 that it is difficult to precisely characterise the contents of the receivers' buffers, as any number of network coded combinations could potentially be stored. We simplify our representation of the receivers' buffers by making the following approximation.

Approximation 6.1. *All packets stored in a receiver's buffer are uncoded.*

This statement is true for the vast majority of packets stored in the receiver's buffer. Algorithm 12 ensures that the first transmission of a new packet will be uncoded, therefore, at least μ of the sender's transmissions must be uncoded. Furthermore, Gaussian elimination applied inside the receivers' buffers allows the decoding of some of the coded transmissions received. As a consequence of Approximation 6.1, we obtain the following property.

Property 6.5. *A receiver Rx_r can decode its next needed packet $N_r(t)$ from $s(t)$ if it has seen all other packets coded into $s(t)$.*

It is not immediately clear how likely it is for a packet to have been seen, since sender transmissions can be stored nearly anywhere in a receiver's buffer, depending on the effects of Gaussian elimination. Therefore, we introduce Approximation 6.2.

DEFINITION 6.10: Let $\mathbf{p}_n$ be the newest packet coded into a transmission $s(t)$. Then $s(t)$ is said to be a *leader transmission* of the packet $\mathbf{p}_n$.

Approximation 6.2. A receiver Rx_r has seen a packet $\mathbf{p}_n$ iff Rx_r has received a leader transmission of the packet $\mathbf{p}_n$.

If $\mathbf{p}_n$, the newest packet coded into $s(t)$, has not yet been seen, then it will be stored at position n . This is the most typical scenario, accounting for more than μ of the received transmissions. Using Approximation 6.2, the likelihood of a packet being seen then depends on the number of leader transmissions of the packet.

The two approximations we have introduced allow us to determine a receiver's probability of a knowledge differential transmission by answering three questions. Firstly, what is the probability that there have been k leader transmissions of a packet? Secondly, given this information, what is the probability that a packet has been seen? And finally, what is the likelihood that a particular set of packets is coded into $s(t)$? We will answer each of these questions in Sections 6.3.2 to 6.3.4, and use the results to find the probability of a knowledge differential transmission in Section 6.3.5. Note that, because the receiver's channel rates are uniform, unlike Chapter 5 we do not assume any particular ordering of the receivers' next needed packets.

6.3.2 Leader transmissions

Our first task is to determine how many leader transmissions there have been of any given packet in the buffer. By Property 6.3, $s(t)$ is a leader transmission of the current leader Rx_l 's next needed packet, $N_l(t)$. It follows that leader transmissions of a packet $\mathbf{p}_n$ only occur while $N_l(t) = \mathbf{p}_n$. The following property explains how $N_l(t)$ evolves over time.⁴

Property 6.6. The leader Rx_l 's next needed packet $N_l(t)$ remains the same while Rx_l is in erasure, but will progress to the next packet every time the leader receives a packet.

We can prove Property 6.6 as follows. The leader can always decode its next needed packet from $s(t)$, since it has already delivered all older packets which might

⁴We assume here the leader is a fixed receiver which does not change. In reality it is possible for other receivers to take the place of the leader and affect the number of leader transmissions of a packet. However from simulations, we observe that the leader changes very infrequently, and therefore have a negligible effect on the observed performance.

be coded into $s(t)$. Therefore the leader's next needed packet remains the same while it is in erasure (since it receives no new information), but will progress with every successful reception. The leader's next needed packet is always the newest packet to have been transmitted by the sender so far, so it is not possible for the leader to 'skip' any next needed packets.

We can use Property 6.6 to determine the probability that there have been k leader transmissions of a given packet $\mathbf{p}_n$.

If $\mathbf{p}_n$ is older than $N_l(t)$, this is the probability it took exactly k leader transmissions of $\mathbf{p}_n$ for it to be received,

$$L(k) = \begin{cases} 0, & k = 0, \\ \bar{\mu}^{k-1}\mu, & k > 0. \end{cases} \quad (6.1)$$

This is the probability that, when leader transmissions of $\mathbf{p}_n$ began, Rx_l experienced $k - 1$ erasures, followed by a successful reception. Note that k cannot be zero, since a packet must be transmitted at least once for Rx_l to receive it.

On the other hand, if the packet $\mathbf{p}_n = N_l(t)$ is still being transmitted to Rx_l , then the likelihood $L^*(k)$ of there having been k transmissions of $\mathbf{p}_n$ so far is given by

$$L^*(k) = \mu\bar{\mu}^k. \quad (6.2)$$

This is the likelihood that the last change of next needed packet occurred exactly k time slots ago.

Packets $\mathbf{p}_n$ that are newer than $N_l(t)$ have not yet been transmitted by the sender, so in this case $k = 0$.

6.3.3 Probability that a packet is seen

We now find the probability that a packet $\mathbf{p}_n$, corresponding to the next needed packet of another receiver, has been seen by a receiver Rx_r who is not the leader. We let D_s , D_f and D_l denote the probability of having seen the next needed packet of a slower, faster and leading receiver, respectively.⁵

Our simulations show that it is uncommon for two receivers to have exactly the same next needed packet.⁶ We therefore introduce Approximation 6.3.

Approximation 6.3. *Each receiver Rx_r 's next needed packet $N_r(t)$ is distinct.*

⁵Note that the simpler structure of the transmission scheme studied in this chapter means that, unlike Chapter 5, we do not need to individually calculate the knowledge difference probability for every pair of receivers.

⁶See, for example, Figures 6.4 and 6.5, which will be discussed in Section 6.4.

If $\mathbf{p}_n$ is the next needed packet of a slower receiver, then Rx_r has already delivered, and therefore seen, the packet. So

$$D_s = 1. \quad (6.3)$$

By Approximation 6.2, undelivered packets can only be seen from leader transmissions. Therefore, the probability that a packet $\mathbf{p}_n$ has been seen by Rx_r is given by the probability that it received at least one of the leader transmissions of the packet.

If $\mathbf{p}_n$ is the next needed packet of the leader,

$$\begin{aligned} D_l &= \sum_{k=0}^{\infty} (1 - \bar{\mu}^k) L^*(k) \\ &= \frac{\bar{\mu}}{1 + \bar{\mu}}. \end{aligned} \quad (6.4)$$

If $\mathbf{p}_n$ is the next needed packet of a faster non-leading receiver, then

$$\begin{aligned} D_f &= \sum_{k=0}^{\infty} (1 - \bar{\mu}^k) L(k) \\ &= \frac{1}{1 + \bar{\mu}}. \end{aligned} \quad (6.5)$$

6.3.4 Codings

From Property 6.5, the probability of a knowledge differential transmission depends on which other packets are coded into $s(t)$. This is represented using a coding C , defined as follows.

DEFINITION 6.11: The *coding* C is a binary vector of length R that indicates which of the receivers' next needed packets, *ordered from fastest receiver to slowest*,⁷ have been coded into the transmission $s(t)$. The r^{th} element of C , $C(r) = 1$ if the r^{th} fastest receiver's next needed packet is coded into the transmission $s(t)$; otherwise it is 0.⁸

From Property 6.3 we can establish that

$$\Pr(C(1) = 1) = 1. \quad (6.6)$$

In practice, the probability that a given packet $\mathbf{p}_n$ has been seen is correlated among receivers. For example, $\mathbf{p}_n$ is more likely to have been seen by receivers if

⁷This differs from the equivalent analysis in Chapter 5, in that we do not assume any fixed ordering of the receivers from fastest to slowest.

⁸Note that we still make use of Approximation 6.3, so receivers' next needed packets are assumed to be distinct.

there were four leader transmissions of $\mathbf{p}_n$, compared with only one. However this correlation makes coding probabilities more difficult to calculate. Approximation 6.4 simplifies our calculations, in exchange for a small loss of accuracy.

Approximation 6.4. *The probability of a packet $\mathbf{p}_n$ being seen is independent for each receiver.*

Using Approximation 6.4, the probability that a coded packet $\mathbf{p}_n$ has been seen by a non-leading receiver R_{x_r} depends only on which receiver's next needed packet it is. If it is the next needed packet of the leader, the probability is D_l . If $\mathbf{p}_n$ belongs to a faster receiver who is not the leader, the probability is D_f . If $\mathbf{p}_n$ belongs to a slower receiver, then it has already been seen, since $D_s = 1$.

Say that C_{r-1} , the first $r-1$ elements of C , are already determined. Then let $F_r = \sum_{i=2}^r C(i)$ denote how many of the r fastest receivers' next needed packets are coded into $s(t)$, excluding the leader. Making use of (6.6), the likelihood that the r^{th} fastest receiver is coded into $s(t)$ can be written as

$$\Pr(C(r) = 1 | C_{r-1}) = D_l D_f^{F_{r-1}}. \quad (6.7)$$

Combining (6.6) and (6.7), we can determine the probability of a coding C ,

$$\Pr(C) = \left(\prod_{r \in v_1(C)} D_l D_f^{F_{r-1}} \right) \prod_{r \in v_0(C)} (1 - D_l D_f^{F_{r-1}}), \quad (6.8)$$

where $v_1(C)$ is the set of values r for which $C(r) = 1$, and $v_0(C)$ is the set of values r for which $C(r) = 0$.

6.3.5 Probability of a knowledge differential transmission

We now derive a receiver R_{x_r} 's probability of a knowledge differential transmission K_r .

By Property 6.2, knowledge differential transmissions almost always⁹ result from the receiver's next needed packet being included in $s(t)$. To exclude the small possibility of knowledge differential transmissions occurring by other means, we introduce Approximation 6.5.

Approximation 6.5. *A knowledge differential transmission occurs for a receiver R_{x_r} iff its next needed packet $N_r(t)$ is coded into $s(t)$.*

⁹Theoretically it is also possible for a next needed packet to be decoded via Gaussian elimination, however this probability is extremely small.

Then the probability a transmission $s(t)$ is a knowledge differential transmission for the r^{th} fastest receiver is given by the likelihood that $C(r) = 1$,

$$K_r = \sum_C C(r) \Pr(C). \quad (6.9)$$

Note that this equation differs from a similar result of (5.23) in that we do not assume any fixed ordering of the receivers from fastest to slowest.

6.4 Delivery Rate Analysis

In this section, we find the receivers' expected delivery rates. We can divide packet delivery into two stages: the *startup phase* and *delivery phase*. Fig. 6.3 illustrates a receiver's buffer storage distribution during each of these phases.

Startup phase: The startup phase applies to non-leading receivers. During the startup phase, a receiver Rx_r first joins the broadcast with an empty buffer. However, the sender's transmissions will allow the receiver to see some of the next needed packets of the leader. As a result of receiving these transmissions, at any time $t > t_r$, a fraction of the packets in the range $N_l(t_r), \dots, N_l(t)$ will have been seen by Rx_r .

In addition to this, the receiver will occasionally receive knowledge differential transmissions, which allow the receiver to slowly begin delivering its next needed packets. Higher delivery rates are only possible once the receiver is close to completing the startup phase. Once the receiver has delivered all the packets up to $N_l(t_r)$, the startup phase is complete. Fig. 6.4 illustrates the small, nonzero delivery rate experienced by receivers during the startup phase.

Delivery phase: The delivery phase begins once Rx_r has delivered all packets up to $N_l(t_r)$. The receiver is still able to make use of knowledge differential transmissions, but seen packets already stored in its buffer allow the Rx_r to deliver packets at a higher rate than during the startup phase.

6.4.1 Delivery phase analysis

We now calculate the average delivery rate of the r^{th} fastest receiver during the delivery phase. All sender transmissions can be classified as either:

- Knowledge differential transmissions, which account for K_r of the sender's transmissions. These deliver the receiver's next needed packets.
- Non-knowledge differential transmissions, which account for the remaining $1 - K_r$ fraction of the sender's transmissions. These transmissions are dis-

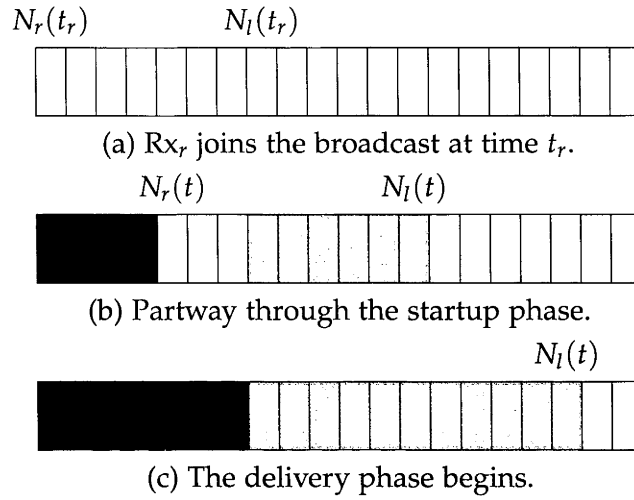


Figure 6.3: The contents of a non-leading receiver's buffer, after it joins the network. Some fraction of the shaded rectangles correspond to packets which have been seen. Initially the buffer is empty, but as the transmission progresses, Rx_r is able to both deliver new packets from the start of the buffer (dark packets), and see some of the leader Rx_l 's next needed packets (shaded packets).

tributed evenly through the shaded portion of the buffer, as illustrated in Fig. 6.3 (b) and (c).

Non-knowledge differential transmissions are stored uniformly across a receiver's buffer, corresponding to the shaded regions in Fig. 6.3. Knowledge differential transmissions deliver the receiver's next needed packets, filling in any missing packets from oldest to newest. Therefore, the delivery rate is given by the average rate at which knowledge differential transmissions can fill in the packets which were missed by non-knowledge differential transmissions.

The *buffer density* is the average fraction of packets in the shaded region that have been seen from non-knowledge differential transmissions. An average of $\mu(1 - K_r)$ non-knowledge differential transmissions are received per time slot. Since the leader progresses at rate μ , an average of $1/\mu$ time slots will be dedicated to each position in the buffer. Therefore, the buffer density is given by

$$B_r = 1 - K_r. \quad (6.10)$$

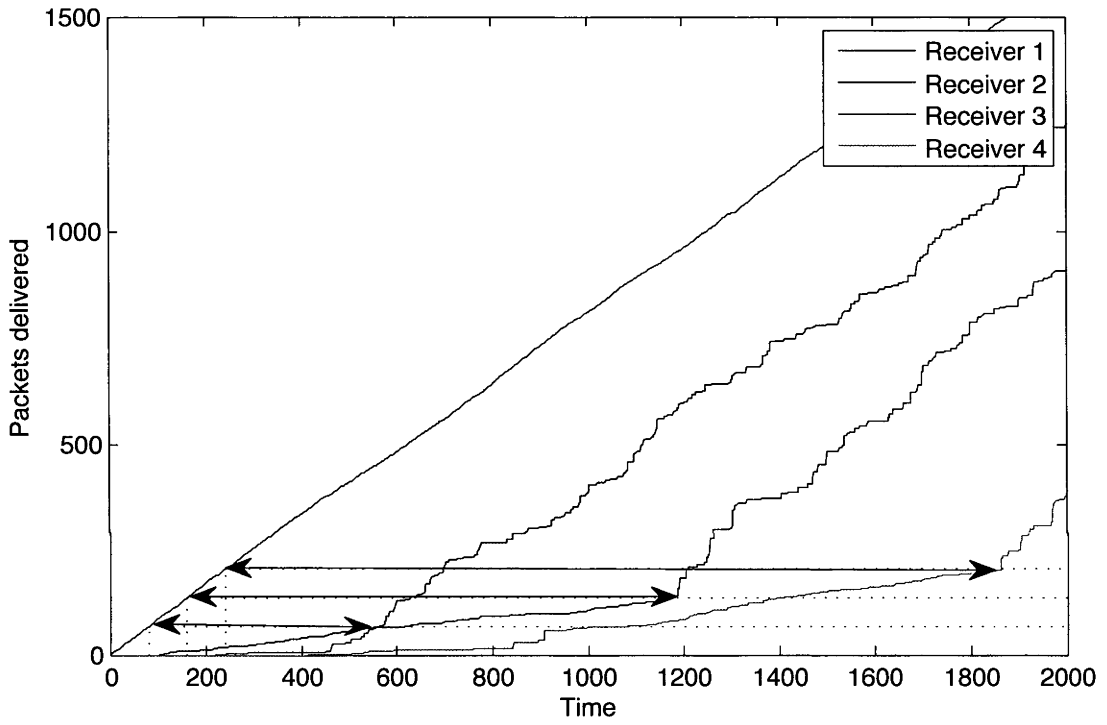


Figure 6.4: Simulated number of packets delivered at each receiver, as a function of time. $R = 4$ receivers, channel rate $\mu = 0.8$, receivers join at times $t_1 = 0$, $t_2 = 80$, $t_3 = 160$, $t_4 = 240$. Vertical dotted lines indicate the starting times of new receivers, and number of packets delivered to the leader Rx_i at these times. Double arrows indicate the duration of the startup phase for receivers.

On average, μK_r knowledge differential transmissions are received per time slot. Therefore, the delivery rate of the r^{th} fastest receiver is given by

$$\begin{aligned} R_r^d &= \frac{\mu K_r}{1 - B_r} \\ &= \mu. \end{aligned} \tag{6.11}$$

We therefore come to the interesting conclusion that, so long as there is a nonzero probability of knowledge differential transmissions, the receiver's average delivery rate during the delivery phase will match that of the leader. In Fig. 6.4, once receivers have delivered up to $N_l(t_r)$, the delivery phase begins. A sudden increase in the delivery rate is observed, and receivers' delivery progress runs roughly parallel to that of the leader. Note, however, that packet delivery is not always smooth, since packet delivery is dependent upon sometimes rare knowledge differential transmissions.

6.5 Undelivered packets

Although the average delivery rate of the receivers is now known, our analysis so far does not give us any indication of how many packets are undelivered at a particular receiver, nor how soon a receiver can begin full rate delivery. As we observe in Fig. 6.5, the number of undelivered packets at a receiver can fluctuate dramatically – in some cases even causing slower receivers to overtake and replace the leader.

6.5.1 Expected number of undelivered packets

We now show that the difference between the number of packets delivered at the leader and any other receiver can be predicted by its *Markov state*.

DEFINITION 6.12: A receiver R_{x_r} 's *Markov state* is the difference between the number of innovative packets stored at the leader and R_{x_r} .

From (6.5) we have an estimate of D_f , the probability that a packet in the buffer density region has been seen. If a receiver's Markov state is k , then there are k unseen packets in the buffer. This corresponds to the $1 - D_f$ unseen packets, which are distributed evenly among the D_f fraction of seen packets. Therefore, if a receiver's Markov state is k , we can expect that the number of undelivered packets U is given by

$$\begin{aligned} U &\approx \frac{1}{1 - D_f} k \\ &= \left(1 + \frac{1}{\bar{\mu}}\right) k. \end{aligned} \tag{6.12}$$

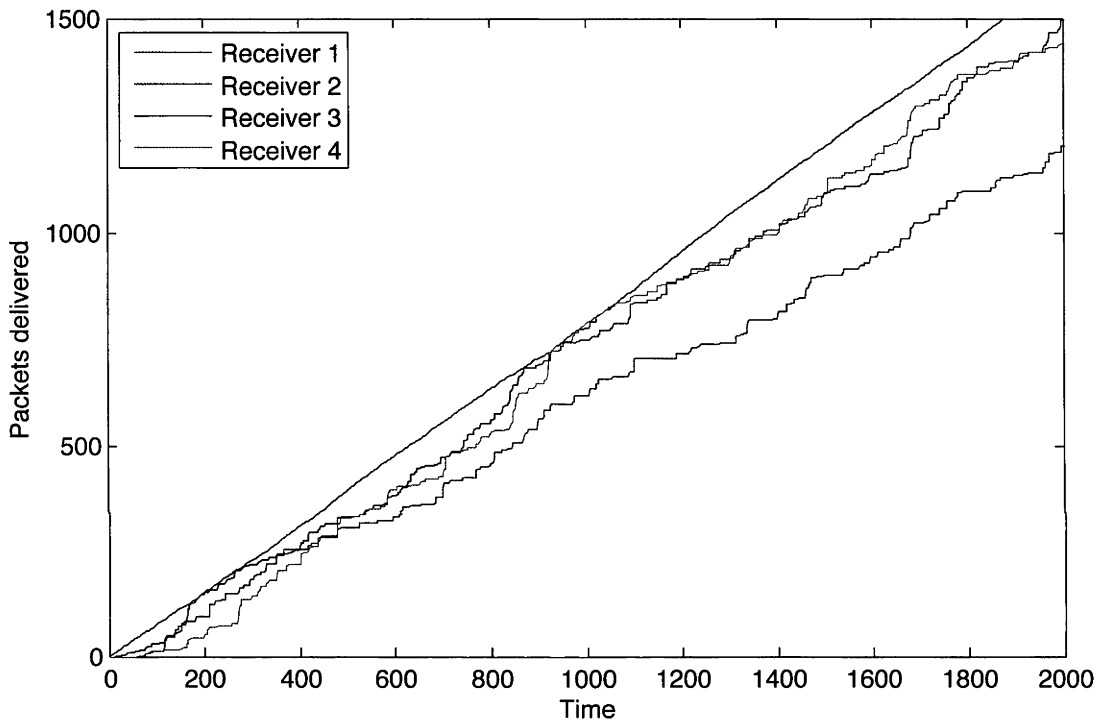


Figure 6.5: Packets delivered at the receivers as a function of time. $R = 4$, $\mu = 0.8$, $t_1 = 0$, $t_2 = 10$, $t_3 = 20$, $t_4 = 30$. We can observe that fluctuations in the receivers' delivery rates can cause changes in the receivers' ordering.

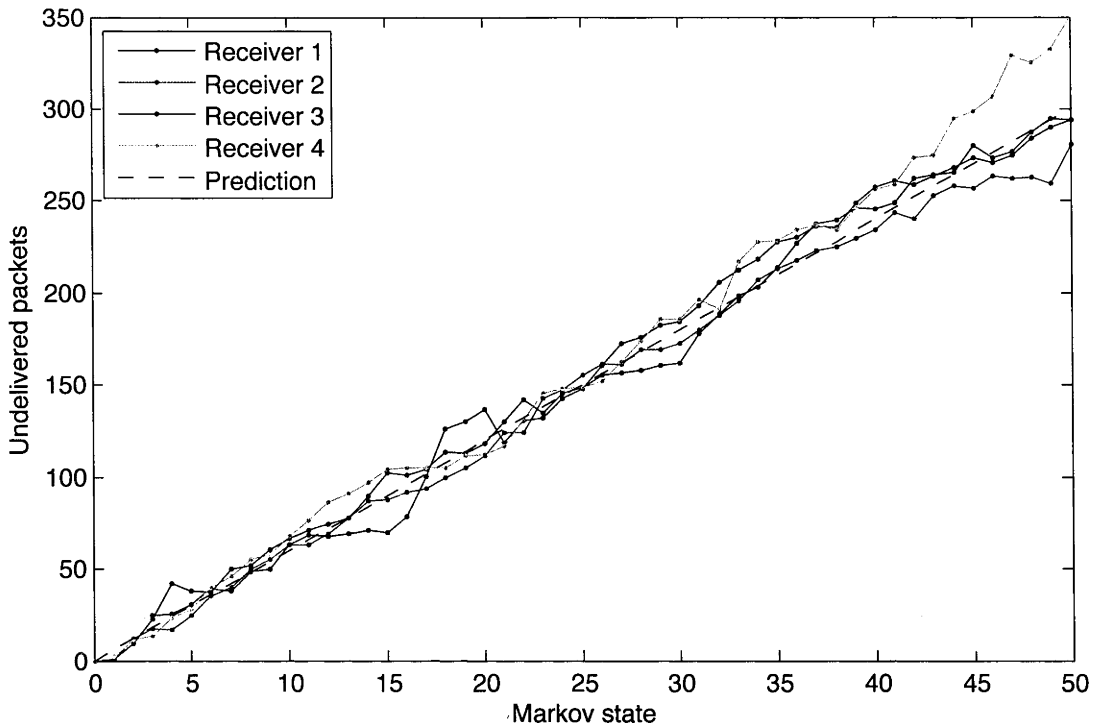


Figure 6.6: Number of undelivered packets as a function of Markov state. $R = 4$, $\mu = 0.8$, $t_1 = 0$, $t_2 = 10$, $t_3 = 20$, $t_4 = 30$. Averaged over 5 simulations of 5000 time slots. This is compared against the prediction made in (6.12).

Fig. 6.6 compares the average number of undelivered packets at each receiver against its Markov state. As we can observe, (6.12) is shown to be a fairly accurate approximation of these values.

Equation (6.12) also indicates that, as the channel rate $\mu \rightarrow 1$, the expected number of undelivered packets approaches infinity. Intuitively this makes sense, since if $\mu = 1$, then there are no opportunities for knowledge differentials to occur. Therefore, this transmission scheme is more effective at lower values of μ .

6.5.2 Startup phase analysis

This model for the number of undelivered packets at a receiver also allows us to obtain a lower bound for the startup delay. Although we cannot make any performance guarantees, it does allow us to upper bound the number of receivers who can be simultaneously supported if only a limited startup delay is permitted. Let us assume that Rx_1 provides a reasonable approximation of the leader. Then, when a receiver Rx_r joins the transmission at time t_r , it also begins with a Markov state of approximately μt_r , the number of packets received by the leader, Rx_1 . Since the Rx_1

and Rx_r both have delivery rates of μ , on average this Markov state is more or less maintained as time passes. Therefore, making use of (6.12), we can expect that on average, the earliest time a receiver can begin full rate transmission will be at time τ , the last time slot at which the predicted number of undelivered packets will be zero. This can be quantified by the relationship,

$$N_1(\tau) = \left(1 + \frac{1}{\mu}\right) \mu t_r. \quad (6.13)$$

On average $N_1(t) = \mu t$, so we obtain

$$\tau = \left(1 + \frac{1}{\mu}\right) t_r. \quad (6.14)$$

Therefore, we can lower bound the average startup delay by

$$\begin{aligned} S_r &> \tau - t_r \\ &= \frac{t_r}{1 - \mu}. \end{aligned} \quad (6.15)$$

A comparison of the receivers' simulated and estimated startup delays is given in Fig. 6.7. The leader is able to deliver at full rate from the first time slot. For subsequent receivers, packets are delivered at slower rates during the startup phase, which generally decreases as more receivers are added. The more faster receivers there are, and the larger the starting time t_r is, the longer the startup phase is for a receiver Rx_r .

While this bound is reasonably close for the faster receivers, as the number of receivers increases, so too does the gap between the predicted and observed startup delay. This can be attributed to the fact that, when there are few receivers, the startup delay is primarily determined by the time taken to obtain the correct ratio of delivered to undelivered packets in the buffer. However, with larger number of receivers the probability of knowledge differential transmissions decreases, making knowledge differential probability the dominant factor in determining the startup delay. It is possible to obtain more accurate estimates using a version of knowledge differential analysis customised for the startup phase, however, this rather complicated procedure lies outside the scope of this thesis.

6.6 Future directions

There is still some way to go before our transmission scheme can be viable for practical applications. While it is able to achieve optimal throughput for all receivers,

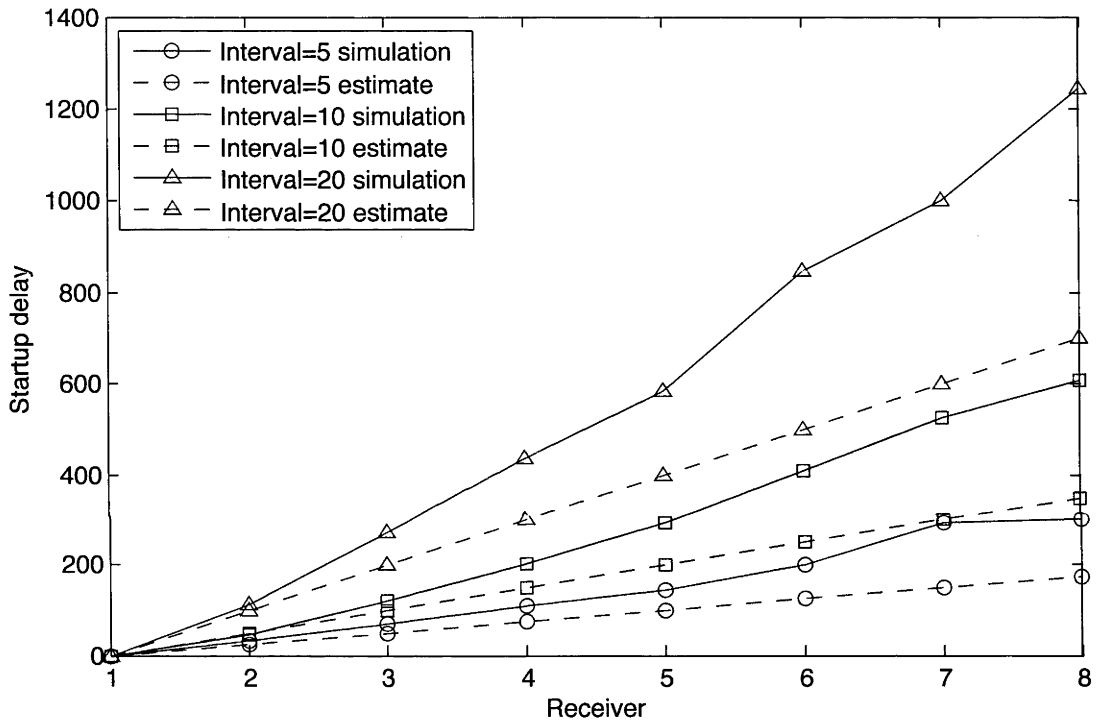


Figure 6.7: Delivery time for receivers, averaged over 10 simulations. $R = 4$, $\mu = 0.8$, $t_1 = 0$, and the difference between receiver starting times $t_{i+1} - t_i$ is given by the intervals listed in figure. This is compared against the lower bound given in (6.15).

to be of practical use, improvements are required in both the startup delay and the number of undelivered packets at the receivers. In video on demand applications, for example, large startup delays would result in unnecessarily long waiting times for viewers. Additionally, higher numbers of undelivered packets would burden the sender and receivers with significantly higher memory and processing requirements.

There are many potential ways to improve the performance of the network coding scheme we have proposed. Although a complete analysis of each is outside the scope of this work, we provide an outline of two promising extensions for future research.

6.6.1 Use of redundant transmissions

Within a single server channel, we may sacrifice a small amount of throughput in order to reduce both the startup delay and number of undelivered packets at the receivers. By allocating selected time slots to different modes, as we did in Chapter 5, we may permit some transmissions that are redundant for faster receivers, but give slower receivers a higher probability of knowledge differential transmissions. Using this technique, we can reduce the number of undelivered packets, and also decrease the startup delay for the slower receivers. The tradeoffs between different receivers' throughputs, startup delays and number of undelivered packets form an interesting topic for future study.

6.6.2 Hybridisation with multiple server channel algorithms

Our proposed network coding algorithm may also be combined with *batch based algorithms* [1, 28] for video on demand applications over multiple server channels. Batch based algorithms group together receivers who have requested the same video within a given time interval, known as the *batching interval*. Once the batching interval has elapsed, the sender begins broadcasting the video stream to all receivers in the group.

One disadvantage of batching is that receivers may potentially be required to wait the entire batching interval before they can begin watching the video. One server is required to transmit to each batch of receivers, so if server resources are limited or the video is particularly long, then the startup delay may be unacceptably high. If, on the other hand, we employ our network coding scheme within each batch, receivers requesting a video slightly *after* a batching interval has elapsed are able to tag along with this broadcast, rather than being forced to wait for the next batching interval. This ensures that lower startup delays can be achieved, or alternatively, means that fewer sender resources are required to serve the video.

One advantage of combining batch based algorithms with our proposed network coding scheme is that the batching interval will limit the network coding induced startup delay. To further reduce the startup delay, a dedicated server channel could also be employed to transmit network coded packets from the start of the video, reducing the Markov state of late joining receivers and allowing them to complete the startup phase more quickly.

6.7 Conclusion

In this chapter, we outlined a network coding scheme that enables multicast video on demand using a single server channel. This network coding scheme is transmission optimal, so the sender's transmissions are guaranteed to simultaneously provide innovative information to all receivers. Through the use of intermediate packet delivery, multiple receivers joining the transmission at different times are able to simultaneously deliver all packets in the correct order. Analysis showed that, after a startup delay has been incurred, receivers can achieve full rate packet delivery, and a lower bound was found for the expected startup delay. We determined that the average number of undelivered packets at any receiver could be effectively predicted by the receiver's Markov state.

Some weaknesses of the transmission scheme in its current form include stringent feedback requirements, increased memory and processing at both the sender and receivers, and the potential for unacceptably large startup delays at the receivers. It is anticipated that some of these weaknesses can be alleviated by hybridising our proposed single server channel transmission scheme with existing algorithms in the literature. The balance between the priorities of transmission optimality, server channel resources and receiver startup delays is an interesting topic for future study.

Conclusion

In this thesis, we analysed and designed a number of deterministic network coding schemes. In particular, we obtained new insights into the characteristics of network coding schemes that allow good throughput and delay performance to be achieved. Making use of the knowledge we gained from our analysis, we were able to design adaptive network coding schemes with greatly improved throughput delay performance compared with existing schemes in the literature.

We demonstrated for the first time that intermediate packet delivery can be achieved using deterministic network coding. Using our performance analysis, we modified an existing transmission scheme to permit fair multi-rate packet delivery to be simultaneously achieved by receivers with multiple channel rates. It was shown that similar methods could be employed for receivers with identical channel rates to achieve asynchronous, transmission optimal packet delivery.

7.1 Summary of results

We now provide a summary of the key results and contributions of this thesis.

7.1.1 Throughput delay analysis

In the first part of this thesis, we gained insight into the throughput delay performance of selected deterministic coding schemes. We used Markov state modelling to distinguish between three forms of packet delivery: namely zero state, leader state and coefficient-based packet delivery. The contributions of each form of packet delivery to the throughput delay performance of the system were characterised. It was discovered that, contrary to what has been speculated in the literature, coefficient-based delivery can have a potentially significant impact on the throughput delay performance of a system.

7.1.2 Dynamic rate control

Using the Markov state model, we formed an understanding of the impact of rate control on the throughput delay performance of a network coding scheme. Using this analytical model, we designed a high throughput, low delay dynamic rate control scheme. By using feedback to adapt the transmission rate to the system performance, this dynamic rate control scheme was able to achieve noticeably improved throughput delay performance compared with existing transmission schemes in the literature. It was found that dynamic rate control algorithms we developed could be reduced down to the simple task of regulating the number of undelivered packets stored across receivers.

To meet the more robust feedback requirements of many practical systems, we designed an adaptation of the dynamic rate control scheme called the queue threshold scheme. Through analysis we demonstrated how the decision parameters could be modified to upper bound the average delay performance, regardless of the channel rate experienced by the receivers. The queue threshold scheme was shown to perform well with reduced volume of feedback, and be robust to delayed and lossy feedback from the receivers.

7.1.3 Intermediate packet delivery

We showed for the first time that intermediate packet delivery can be used to maintain continued in-order packet delivery, without ever making use of the complete information property. Using intermediate packet delivery, we demonstrated that the rate limitations of the min-cut bound could be sidestepped, by allowing a single sender to simultaneously provide multi-rate packet delivery to a number of receivers. We designed a fairness algorithm to allocate the sender's resources so that receivers achieve approximately fair delivery rates, relative to their channel rates. This is the first time that delivery rate fairness has been achieved for a single hop broadcast channel.

We extended the intermediate packet delivery results to show that, if receivers have the same channel rate, video on demand services can be provided using a single server channel. The network coding scheme we outlined is transmission optimal, and after an initial startup delay, all receivers are able to deliver the packets of the video in the correct order at their channel rate.

7.2 Future work

While this thesis has contributed some important insights into the nature of the throughput delay tradeoff in network coding schemes, there remain a number of open questions to be addressed. The throughput delay performance has yet to be characterised for a great number of transmission schemes, including the dynamic rate control schemes we designed. While the performance of these schemes was shown through simulation to be better than existing transmission schemes, one might ask whether even better transmission schemes could be designed. It is not known how much potential for improvement remains, as the fundamental limits on the achievable throughput delay performance have not yet been determined. We have yet to explore how the results we have derived for a single-hop broadcast network can be extended to more general multi-hop networks.

We introduced the concept of consistent, in-order packet delivery for the first time, but have only just begun to explore the range of its potential applications. The independence that intermediate packet delivery permits among the receivers could, for example, be used to maintain good throughput delay performance under fading channels, or achieve fairness under any number of performance measures. Much analytical work remains to be done, as we cannot yet predict the range of simultaneously achievable packet delivery rates, and the delay performance of this transmission scheme is still not well understood. We have not yet ascertained if the transmission scheme we outlined is the only one that will permit intermediate packet delivery, or if other such coding schemes exist. Issues associated with intermediate packet delivery, such as the buffer space required, delay performance and computational requirements, must also be considered in future applications.

Bibliography

1. C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, 1996, pp. 253–258. (cited on pages 138 and 156)
2. R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, 2002. (cited on pages 1, 3, 11, 12, 31, and 91)
3. M. Artin, *Algebra*, 1st ed. Prentice-Hall, 1991. (cited on pages 96 and 139)
4. J. Barros, R. Costa, D. Munaretto, and J. Widmer, "Effective delay control in online network coding," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, April 2009, pp. 208 –216. (cited on pages 3, 4, 5, 9, 16, 21, 32, 34, 44, 60, 61, 72, and 92)
5. A. Beimel, S. Dolev, and N. Singer, "RT oblivious erasure correcting," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1321–1332, 2007. (cited on pages 6 and 92)
6. R. Brualdi and K. Bogart, *Introductory Combinatorics*. Prentice Hall, 1999. (cited on page 42)
7. N. Cai and R. Yeung, "Secure network coding," in *Proc. IEEE International Symposium on Information Theory*, 2002, pp. 323–. (cited on page 1)
8. J. Chakareski and P. Frossard, "Rate-distortion optimized distributed packet scheduling of multiple video streams over shared communication resources," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 207–218, 2006. (cited on page 61)
9. H. Chi, Q. Zhang, J. Jia, and X. Shen, "Efficient search and scheduling in P2P-based media-on-demand streaming service," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 1, pp. 119–130, 2007. (cited on page 138)
10. P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, 2006. (cited on page 61)

11. P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Annual Allerton Conference on Communication, Control and Computing*, Oct. 2003. (cited on page 92)
12. J. Cloud, D. J. Leith, and M. Médard. (2013) Network coded TCP (CTCP) performance over satellite networks. [Online]. Available: <http://arxiv.org/abs/1310.6635> (cited on page 1)
13. T. Cover, "Broadcast channels," *IEEE Trans. Inf. Theory*, vol. 18, no. 1, pp. 2–14, 1972. (cited on page 95)
14. A. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros, "Capacity of wireless erasure networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 789–804, 2006. (cited on pages 6, 91, 92, and 94)
15. A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010. (cited on page 1)
16. A. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proc. IEEE*, vol. 99, no. 3, pp. 476–489, 2011. (cited on page 1)
17. A. Douik, S. Sorour, M.-S. Alouini, and T. Al-Naffouri, "Delay reduction in lossy intermittent feedback for generalized instantly decodable network coding," in *IEEE Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2013, pp. 388–393. (cited on page 61)
18. M. Durvy, C. Fragouli, and P. Thiran, "Towards reliable broadcasting using ACKs," in *IEEE International Symposium on Information Theory (ISIT)*, 2007, pp. 1156–1160. (cited on pages 5, 92, 93, 94, and 139)
19. A. Eryilmaz, A. Ozdaglar, M. Médard, and E. Ahmed, "On the delay and throughput gains of coding in unreliable networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 12, pp. 5511–5524, December 2008. (cited on pages 3, 4, 16, 32, 33, and 92)
20. C. Feng and B. Li, "On large-scale peer-to-peer streaming systems with network coding," in *Proc. ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2008, pp. 269–278. (cited on page 4)
21. C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "A network coding approach to energy efficient broadcasting: From theory to practice," in *Proc. IEEE International*

-
- Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–11. (cited on page 1)
22. A. Fu and P. Sadeghi, “A network coding algorithm for single server channel multicast video on demand,” to appear in *Australian Communication Theory Workshop AusCTW*, 2014. (cited on page 9)
23. —, “Queue-based rate control for low feedback rlnc,” to appear in *IEEE International Conference on Communications (ICC)*, 2014. (cited on page 8)
24. A. Fu, P. Sadeghi, and M. Médard, “Beyond the mincut bound: Deterministic network coding for asynchronous multirate broadcast,” submitted to *IEEE Trans. Inf. Theory*. [Online]. Available: <http://arxiv.org/abs/1401.0543> (cited on page 8)
25. —, “Dynamic rate adaptation for improved throughput and delay in wireless network coded broadcast,” to appear in *IEEE/ACM Trans. Netw.* [Online]. Available: <http://arxiv.org/abs/1208.3806> (cited on pages 7 and 8)
26. —, “Delivery delay analysis of network coded wireless broadcast schemes,” in *Wireless Communications and Networking Conference*, 2012. (cited on page 7)
27. H. Gangammanavar and A. Eryilmaz, “Dynamic coding and rate-control for serving deadline-constrained traffic over fading channels,” in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2010, pp. 1788–1792. (cited on page 3)
28. L. Gao and D. Towsley, “Supplying instantaneous video-on-demand services using controlled multicast,” in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, vol. 2, 1999, pp. 117–121 vol.2. (cited on pages 138 and 156)
29. C. Gkantsidis, J. Miller, and P. Rodriguez, “Comprehensive view of a live network coding P2P system,” in *Proc. ACM SIGCOMM Conference on Internet Measurement*, 2006, p. 188. (cited on page 1)
30. C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution,” in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, vol. 4, 2005. (cited on page 92)
31. A. Gkeliass and K. Leung, “A distributed opportunistic scheduling scheme for wireless network coding,” in *International Symposium on Wireless Communication Systems (ISWCS)*, 2009, pp. 313–317. (cited on page 6)

32. M. Heindlmaier, D. S. Lun, D. Traskov, and M. Médard, "Wireless inter-session network coding-an approach using virtual multicasts," in *IEEE International Conference on Communications (ICC)*, 2011, pp. 1–5. (cited on page 1)
33. T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE International Symposium on Information Theory (ISIT)*. Citeseer, 2003, pp. 442–442. (cited on page 3)
34. T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006. (cited on pages 3, 5, 26, 33, 77, and 92)
35. I. Hou, P. Kumar *et al.*, "Broadcasting delay-constrained traffic over unreliable wireless links with network coding," in *Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2011. (cited on page 61)
36. A. Hu, "Video-on-demand broadcasting protocols: A comprehensive study," in *Proc. Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 1, 2001, pp. 508–517. (cited on page 137)
37. S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2007, pp. 616–624. (cited on page 1)
38. R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *DEC Research Report TR-301*, 1984. (cited on pages 6 and 95)
39. L.-S. Juhn and L.-M. Tseng, "Harmonic broadcasting for video-on-demand service," *IEEE Trans. Broadcast.*, vol. 43, no. 3, pp. 268–271, 1997. (cited on page 138)
40. S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 497–510, 2008. (cited on page 1)
41. M. Kim, J. Cloud, A. ParandehGheibi, L. Urbina, K. Fouli, D. J. Leith, and M. Médard. (2012) Network coded TCP (CTCP). [Online]. Available: <http://arxiv.org/abs/1212.2291> (cited on page 1)
42. R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, 2003. (cited on pages 1, 11, 12, 96, and 139)

-
43. F.-C. Kuo, K. Tan, X. Li, J. Zhang, and X. Fu, "XOR rescue: Exploiting network coding in lossy wireless networks," in *IEEE International Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2009, pp. 1–9. (cited on page 6)
 44. F. Kuo, K. Tan, X. Li, J. Zhang, and X. Fu, "Network coding-aware fair opportunistic scheduling in lossy wireless networks," *Institute of Computer Science, University of Goettingen, Goettingen, Germany, Technical Report IFI-TB-2008-003*, 2008. (cited on page 6)
 45. S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, 2003. (cited on pages 1, 3, 11, and 92)
 46. X. Li, C.-C. Wang, and X. Lin, "Throughput and delay analysis on uncoded and coded wireless broadcast with hard deadline constraints," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, March 2010, pp. 1–5. (cited on pages 1, 3, 4, 16, 32, and 60)
 47. —, "On the capacity of immediately-decodable coding schemes for wireless stored-video broadcast with hard deadline constraints," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 1094–1105, May 2011. (cited on pages 3, 5, 34, 61, and 92)
 48. Z. Liu, C. Wu, B. Li, and S. Zhao, "UUSee: Large-scale operational on-demand streaming with random network coding," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2010, pp. 1–9. (cited on pages 1, 4, and 138)
 49. M. Luby, "LT codes," in *Proc. IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–280. (cited on page 4)
 50. D. E. Lucani, M. Médard, and M. Stojanovic, "Random linear network coding for time-division duplexing: field size considerations," in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 2009, pp. 1–6. (cited on page 3)
 51. D. Lucani, M. Stojanovic, and M. Médard, "Random linear network coding for time division duplexing: When to stop talking and start listening," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, vol. 9, 2009, pp. 1800–1808. (cited on pages 1, 3, 61, and 92)
 52. D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, 2008. (cited on pages 6 and 92)

53. D. Lun, M. Médard, and R. Koetter, "Network coding for efficient wireless unicast," in *Proc. International Zurich Seminar on Communications (IZS)*, 2006, pp. 74–77. (cited on pages 1 and 91)
54. E. Martinian, "Dynamic information and constraints in source and channel coding," Ph.D. dissertation, Massachusetts Institute of Technology, 2004. (cited on pages 6 and 92)
55. D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Trans. Veh. Technol.*, vol. 58, no. 2, pp. 914–925, Feb. 2009. (cited on pages 1, 3, 16, 32, and 34)
56. A. ParandehGheibi, M. Médard, S. Shakkottai, and A. Ozdaglar, "Avoiding interruptions - QoE trade-offs in block-coded streaming media applications," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2010, pp. 1778–1782. (cited on page 33)
57. J. Park, M. Gerla, D. Lun, Y. Yi, and M. Médard, "Codecast: a network-coding-based ad hoc multicast protocol," *IEEE Wireless Commun. Mag.*, vol. 13, no. 5, pp. 76–81, 2006. (cited on page 92)
58. A. Rezaee, F. du Pin Calmon, L. M. Zeger, and M. Médard, "Speeding multicast by acknowledgment reduction technique (SMART) enabling robustness of qoe to the number of users," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 7, pp. 1270–1280, 2012. (cited on pages 33 and 61)
59. P. Sadeghi, R. Shams, and D. Traskov, "An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels," *EURASIP Journal on Wireless Communications and Networking*, pp. 1–14, April 2010. (cited on pages 3, 5, 34, 60, 61, and 92)
60. P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *Workshop on Network Coding, Theory and Applications (NETCOD)*, June 2009, pp. 80–85. (cited on pages 3, 5, 61, and 92)
61. S. Sanghavi, "Intermediate performance of rateless codes," in *IEEE Information Theory Workshop (ITW)*, 2007, pp. 478–482. (cited on pages 6 and 92)
62. H. Seferoglu and A. Markopoulou, "Video-aware opportunistic network coding over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 713–728, 2009. (cited on page 61)

-
63. A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, 2006. (cited on page 4)
 64. N. Shulman, "Communication over an unknown channel via common broadcasting," Ph.D. dissertation, Tel Aviv University, 2003. (cited on page 92)
 65. S. Sorour and S. Valaee, "Minimum broadcast decoding delay for generalized instantly decodable network coding," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, 2010, pp. 1–5. (cited on pages 3, 5, 34, 61, and 92)
 66. —, "On minimizing broadcast completion delay for instantly decodable network coding," in *Proc. IEEE International Conference on Communications (ICC)*, May 2010, pp. 1–5. (cited on pages 3, 5, 34, 60, 61, and 92)
 67. S. Sorour, A. Douik, S. Valaee, T. Y. Al-Naffouri, and M.-S. Alouini. (2013) Partially blind instantly decodable network codes for lossy feedback environment. [Online]. Available: <http://arxiv.org/abs/1307.3791> (cited on page 61)
 68. S. Sorour and S. Valaee, "Completion delay minimization for instantly decodable network coding with limited feedback," in *IEEE International Conference on Communications (ICC)*, 2011. (cited on page 61)
 69. N. Sundaram, P. Ramanathan, and S. Banerjee, "Multirate media streaming using network coding," in *Proc. Annual Allerton Conference on Communication, Control, and Computing*, September 2005. (cited on page 95)
 70. J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: Theory and implementation," *Proc. IEEE*, vol. 99, no. 3, pp. 490–512, 2011. (cited on pages 1, 4, 28, 61, and 92)
 71. J. Sundararajan, P. Sadeghi, and M. Médard, "A feedback-based adaptive broadcast coding scheme for reducing in-order delivery delay," in *Proc. Workshop on Network Coding, Theory, and Applications (NETCOD)*, 2009, pp. 1–6. (cited on pages 4, 5, 6, 7, 9, 18, 19, 20, 23, 28, 34, 36, 60, 61, 92, 95, 99, and 141)
 72. J. Sundararajan, D. Shah, and M. Médard, "Online network coding for optimal throughput and delay - the three-receiver case," in *International Symposium on Information Theory and Its Applications (ISIT)*, 2008, pp. 1–6. (cited on pages 5, 31, and 34)
 73. —. (2009) Feedback-based online network coding. [Online]. Available: <http://arxiv.org/abs/0904.1730> (cited on pages 5, 31, 34, 36, 38, 41, and 44)

74. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2008, pp. 1651–1655. (cited on pages 4, 5, 6, 7, 9, 18, 19, 20, 22, 23, 28, 34, 36, 52, 60, 61, 78, and 92)
75. B. Swapna, A. Eryilmaz, and N. Shroff, "Throughput-delay analysis of random linear network coding for wireless broadcasting," in *Proc. International Symposium on Network Coding (NETCOD)*, June 2010, pp. 1–6. (cited on pages 3, 4, 33, and 92)
76. S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems*, vol. 4, no. 4, pp. 197–208, 1996. (cited on page 138)
77. G. Wang, X. Zhao, and X. Dai, "On efficient network coding scheme with lossy and delayed feedback," in *International Symposium on Network Coding*, July 2011, pp. 1–6. (cited on pages 4, 5, and 61)
78. M. Wang and B. Li, " R^2 : Random push with random network coding in live peer-to-peer streaming," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, p. 1655, 2007. (cited on pages 1 and 4)
79. J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," in *Proc. ACM SIGCOMM Workshop on Delay-tolerant Networking*, 2005, pp. 284–291. (cited on page 1)
80. F. Wu, C. Hua, H. Shan, and A. Huang, "Reliable network coding for minimizing decoding delay and feedback overhead in wireless broadcasting," in *IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, 2012, pp. 796–801. (cited on pages 4, 16, 21, 22, 28, 33, 34, 61, 80, and 92)
81. Y. Wu, P. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1906–1918, 2005. (cited on page 1)
82. L. Yang, Y. Sagduyu, J. Li, and J. Zhang. (2012) Adaptive network coding for scheduling real-time traffic with hard deadlines. [Online]. Available: <http://arxiv.org/abs/1203.4008> (cited on pages 1, 4, 21, 33, 60, and 61)
83. M. Yang and Y. Yang, "Adaptive Network Coding for Heterogeneous Peer-to-Peer Streaming Systems," in *Proc. IEEE International Symposium on Network Computing and Applications (NCA)*, 2009, pp. 274–277. (cited on page 4)
84. W. Zeng, C. Ng, and M. Médard, "Joint coding and scheduling optimization in wireless systems with varying delay sensitivities," in *IEEE International Conference*

on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012, pp. 416–424. (cited on pages 1, 4, 32, 60, 68, and 92)

85. X. Zhang, J. Liu, B. Li, and T. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, vol. 3, 2005, pp. 13–17. (cited on page 1)