The Impact of Coding Depth on Sliding Window RLNC Protocols

A Thesis

submitted to the designated
by the Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Foteini Karetsi

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION
IN ADVANCED COMPUTER SYSTEMS

University of Ioannina
School of Engineering
Ioannina 2022

Examining Committee:

- **Evangelos Papapetrou**, Assist. Professor, Department of Computer Science and Engineering, University of Ioannina (Advisor)
- **Christos Liaskos**, Assist. Professor, Department of Computer Science and Engineering, University of Ioannina
- **Lysimachos-Pavlos Kondi**, Professor, Department of Computer Science and Engineering, University of Ioannina

Acknowledgements

First of all, I would like to thank my supervisor, Assistant Professor Mr. Evangelos Papapetrou of University of Ioannina, for his undivided support and guidance over the last few years. I am beyond grateful for sparking my interest in research and believing in my capabilities when I mostly doubt myself. Our endless discussions always encourage me to step over boundaries and make me evolve as a scientist and researcher.

I would also like to thank my family for standing by me all these years, both mentally and financially. Their willingness to facilitate my daily routine in order to pursue my ambitions is an undeniable proof of selfless love that I will always look up to.

Last but not least, I could not be more indebted to Alexandros, without whose boundless patience and affection nothing would have happened. Thank you for being the light of my life and making me smile every day even during the darkest of times.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Glossary

URLLC	Ultra-Reliable Low-Latency Communication
RLNC	Random Linear Network Coding
ARQ	Automatic Repeat reQuest
FEC	Forward Error Correction
SW	Sliding Window
CW	Maximum Coding Window Size
W	Maximum Sliding Window Size
d	Coding Depth
R	Code Rate
$\mathbb{S}_w$	RapidARQ's Sliding Window
$\mathbb{C}_w$	RapidARQ's Coding Window
k	The number of source packets transmitted between two successive coded packets

Abstract

Foteini Karetsi, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2022.
The Impact of Coding Depth on Sliding Window RLNC Protocols.
Advisor: Evangelos Papapetrou, Assistant Professor.

5G and beyond networks are envisioned to provide services for a plethora of heterogeneous applications which pose stringent constraints regarding data rate, latency and reliability. One of the main pillars of 5G networks is the support of Ultra-Reliable Low-Latency Communication (URLLC). The latter will cater to multiple advanced services where ultra-high reliability and low latency are pivotal requirements, such as Virtual (VR) and Augmented (AR) Reality systems or factory automation. Towards achieving these goals, the deployment of high-performance reliability mechanisms is essential in order to mitigate the impact of errors. Several legacy techniques have been utilized for that purpose. However, they usually fall short of complying with the hard specifications of URLLC. Random Linear Network Coding (RLNC) techniques incorporating a sliding window scheme have proved to be an enabler of URLLC.

While sliding window RLNC schemes have been broadly examined as an efficient reliability mechanism, little do we know so far about the impact of the coding window size on the coding scheme's efficiency. This issue is immensely important for channels with varying conditions, where the coding scheme should be appropriately adjusted to tackle the occurring errors efficiently. In this work, we examine the impact of the coding window size on the performance of sliding window RLNC protocols. To that end, we leverage the abstraction of coding depth to facilitate the definition of the coding window size. First, we observe that based on the concept of coding depth, sliding window RLNC schemes can achieve superior overall performance especially under varying channel conditions. We provide an analytical method to select an

optimal coding depth value for specific channel conditions while considering the performance and complexity constraints of the coding scheme. We also devise an efficient algorithm to dynamically adapt coding depth according to the dynamic channel conditions. Finally, we experimentally prove that the proposed adaptive scheme achieves comparable or even improved performance compared to previous sliding window RLNC models where the coding window size remains invariant.

***Keywords**:* Ultra-Reliable Low-Latency Communication (URLLC), Random Linear Network Coding (RLNC), sliding window RLNC, coding depth

Εκτεταμενη Περιληψη

Φωτεινή Καρέτση, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2022.
Η Επίδραση του Βάθους Κωδικοποίησης στα Πρωτόκολλα Κωδικοποίησης Δικτύου με Κυλιόμενο Παράθυρο.
Επιβλέπων: Ευάγγελος Παπαπέτρου, Επίκουρος Καθηγητής.

Τα ψηφιακά δίκτυα 5ης και μεταγενέστερης γενιάς αποβλέπουν στην παροχή υπηρεσιών για μια ποικιλία από ετερογενείς εφαρμογές που θέτουν αυστηρούς περιορισμούς όσον αφορά το ρυθμό μετάδοσης δεδομένων, τη συνολική καθυστέρηση και την αξιοπιστία. Μία από τις βασικές υπηρεσίες που σκοπεύουν να προσφέρουν αυτά τα δίκτυα είναι η εξαιρετικά αξιόπιστη, χαμηλής καθυστέρησης επικοινωνία (Ultra-Reliable Low-latency Communication, URLLC). Η υπηρεσία αυτή αφορά μια πληθώρα εφαρμογών για τις οποίες τόσο η εξαιρετικά υψηλή αξιοπιστία όσο και η χαμηλή καθυστέρηση είναι αναγκαίες απαιτήσεις, όπως είναι τα συστήματα εικονικής και επαυξημένης πραγματικότητας και οι διαδικασίες αυτοματοποίησης των εργοστασίων. Προκειμένου να επιτευχθούν αυτοί οι αυστηροί περιορισμοί, είναι αναγκαία η αξιοποίηση μηχανισμών που εξασφαλίζουν υψηλή απόδοση και μεγάλη αξιοπιστία περιορίζοντας, παράλληλα, την επίδραση των σφαλμάτων που συμβαίνουν στο σύνδεσμο μετάδοσης. Μέχρι στιγμής, διάφορες τεχνικές έχουν χρησιμοποιηθεί για αυτό το σκοπό. Ωστόσο, οι τεχνικές Κωδικοποίησης Δικτύου και, συγκεκριμένα, Τυχαίας Γραμμικής Κωδικοποίησης Δικτύου, που ενσωματώνουν ένα σχήμα κυλιόμενου παραθύρου (sliding window) έχουν τη δυναμική για την επίτευξη των υψηλών προδιαγραφών αξιοπιστίας και χαμηλής καθυστέρησης.

Παρόλο που τα πρωτόκολλα Κωδικοποίησης Δικτύου με κυλιόμενο παράθυρο έχουν εξεταστεί ευρέως ως αποδοτικοί μηχανισμοί που εξασφαλίζουν την αξιοπιστία, ελάχιστη αναφορά γίνεται στη βιβλιογραφία αναφορικά με την επίδραση του

μεγέθους του παραθύρου κωδικοποίησης στην αποδοτικότητα του μηχανισμού. Ο κατάλληλος ορισμός του παραθύρου είναι ιδιαίτερα σημαντικός σε κανάλια με μεταβαλλόμενες συνθήκες, όπου η προσαρμογή του σχήματος κωδικοποίησης είναι αναγκαία ώστε να επιτευχθεί η επιτυχής αντιμετώπιση των σφαλμάτων που προκύπτουν. Στην παρούσα εργασία εξετάζουμε την επίδραση του μεγέθους του παραθύρου κωδικοποίησης στην απόδοση των πρωτοκόλλων Κωδικοποίησης Δικτύου με κυλιόμενο παράθυρο, αξιοποιώντας την έννοια του βάθους κωδικοποίησης (coding depth) για τον ορισμό του παραθύρου κωδικοποίησης. Αρχικά, επισημαίνουμε τη σπουδαιότητα του ορθού καθορισμού του παραθύρου κωδικοποίησης για την αποδοτική λειτουργία των πρωτοκόλλων. Αυτό αφορά κανάλια που είτε οι συνθήκες μετάδοσης είναι σχετικά σταθερές είτε παρατηρούνται σημαντικές μεταβολές. Κατόπιν, παρέχουμε έναν αναλυτικό τρόπο υπολογισμού του βάθους κωδικοποίησης για συγκεκριμένες συνθήκες καναλιού που μπορεί να εξασφαλίσει σχεδόν βέλτιστη απόδοση. Επιπλέον, υλοποιούμε έναν αποτελεσματικό μηχανισμό για τη δυναμική προσαρμογή του βάθους κωδικοποίησης και, επομένως, του παραθύρου κωδικοποίησης λαμβάνοντας υπόψη τις μεταβαλλόμενες συνθήκες του καναλιού. Τέλος, αποδεικνύουμε ότι το προτεινόμενο δυναμικό σχήμα πετυχαίνει συγκρίσιμη ή και καλύτερη απόδοση από άλλα πρωτόκολλα Κωδικοποίησης Δικτύου με κυλιόμενο παράθυρο στα οποία το παράθυρο κωδικοποίησης παραμένει αμετάβλητο.

Λέξεις-κλειδιά: Εξαιρετικά Αξιόπιστη Χαμηλής Καθυστέρησης Επικοινωνία, Κωδικοποίηση Δικτύου, Κυλιόμενο Παράθυρο, Βάθος Κωδικοποίησης

Chapter 1

Introduction

1.1 The Goals of 5G and Beyond Networks

In a world of ubiquitous connectivity dictated by the Internet of Things (IoT) paradigm, the exponential growth of data traffic and the vast increase in mobile terminals are challenging issues for the next generation of communication networks. Apart from the significantly enhanced capacity to meet the growing demands of users, the unceasing emergence of new services as well as the heterogeneity across applications and devices pose additional constraints in terms of scalability, reliability and latency. 5G is envisioned to support this unprecedented demand for high-data rate connectivity, stringent low latency and reliability constraints. By leveraging the intrinsic merits of diverse technologies, such as satellite networks and unmanned aerial vehicles (UAVs), vehicular networking, edge and cloud computing as well as machine-to-machine communications (M2M) [1], 5G and beyond networks are expected to pave the way towards a fully connected digital society.

5G is completely reshaping our lives by enabling new technologies that enhance user experience and facilitate different aspects of life. Figure 1.1 illustrates various use

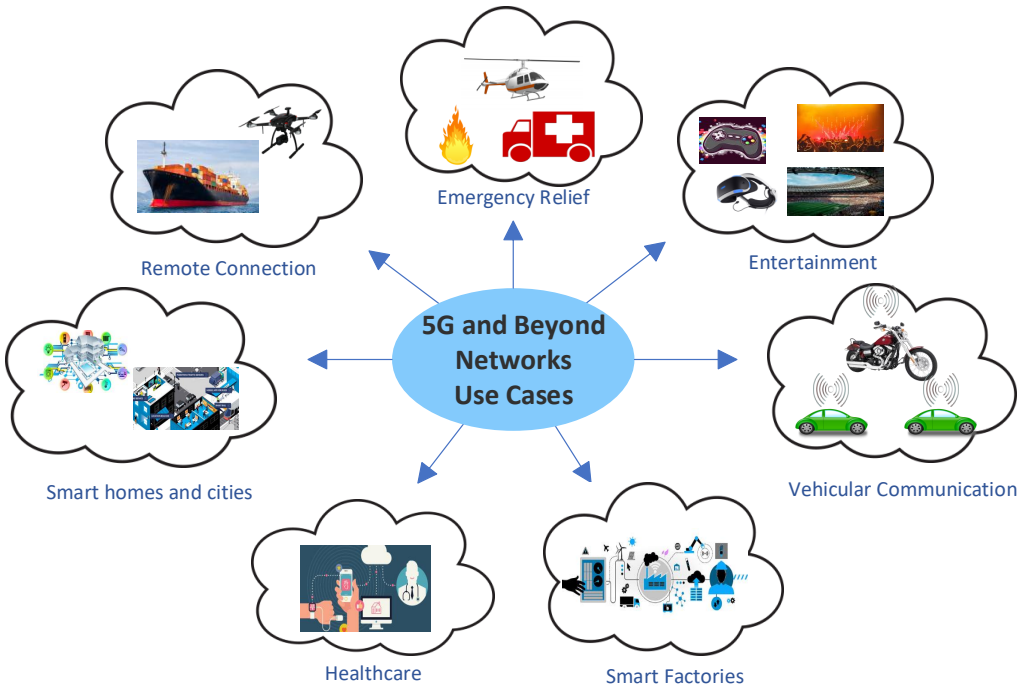


Figure 1.1: 5G and beyond networks: Use Cases.

cases which benefit from the advent of 5G and beyond networks. The emergence of smart homes and cities as well as ensuring remote connections worldwide could not be feasible with the support of legacy communications systems, due to scalability and coverage constraints. Reliability and timeliness are also critical parameters to various domains of applications, such as immersive entertainment platforms, e.g., Virtual Reality (VR) and Augmented Reality (AR) systems, autonomous vehicles, smart industries and healthcare. In particular, these applications demand ultra-reliability and low latency to perform efficiently, which are pivotal for Ultra-Reliable Low Latency Communication (URLLC) [2]. The latter will play a major role in 5G and future networks, since achieving these requirements is an extremely challenging task. In URLLC, the specification mandates a success probability of over 99.99% and targets for end-to-end latency up to $1ms$. The latter includes the transmission delay, queuing delay, processing delay and retransmission overhead when needed. Furthermore, regarding reliability, target block error rate ranges between 10^{-9} and 10^{-5} depending on the use case [3].

1.2 Achieving URLLC in 5G and Beyond

It is evident that the successful achievement of URLLC requirements will give rise to applications where the delay and reliability are critical. For instance, the targeted latency will allow efficient remote diagnosis and surgery as well as minimize hazards in manufacturing. An enhanced user experience will be offered via VR systems to sports or music fans that cannot attend various events in person. Similarly, ultra-high reliability will assist in industry automation or remote monitoring in cities and smart homes [3]. Towards achieving 5G and beyond goals, various sophisticated techniques are deployed to utilize more available spectrum and deliver information with high-data rates. In this direction, millimeter-wave (mmWave) communication is a key enabler for 5G networks while exploiting the terahertz band is also considered for 6G and beyond systems. Unfortunately, by leveraging these frequency bands, severe losses are observed from a physical-layer point of view, hence mechanisms for tackling packet losses should be devised in the data link or transport layer.

Several legacy techniques have been deployed to mitigate the impact of failures in wireless channels, such as Automatic Repeat ReQuest (ARQ), Forward Error Correction (FEC) or even Hybrid Automatic Repeat reQuest (HARQ) schemes. While these techniques consider the reliability-delay trade-off, the achieved performance falls short compared to the URLLC requirements, either due to increased overhead or prohibitive delay. To tackle this shortcoming, various coding approaches have been utilized over the last years but the most promising one entails the use of *Network Coding* (NC). Exploiting the combination of packets to enhance performance, NC and particularly Random Linear Network Coding (RLNC) has been successfully examined as an efficient reliability mechanism. Coding enhanced with the notion of sliding window has proved to be a promising candidate in order to manage delay and reliability constraints. To further enhance reliability, some sliding window (SW) RLNC schemes may employ retransmissions at the expense of increased delay. Hence, the presence of retransmissions is decided depending on the requirements of each application.

1.3 Objectives of Thesis

While sliding window RLNC has proved to be an enabler of URLLC, little attention has been paid to the optimization of the coding efficiency of such schemes. In particular,

the performance of SW RLNC protocols depends on two factors:

- the level of redundancy injected into the original data stream and,
- the number of packets involved in the encoding process, also known as the *coding window size*.

An appropriate definition of the coding window size is of high importance for the performance vs complexity trade-off, since the size determines the range of transmitted packets requiring additional protection through coding. However, the larger the range of packets, the larger the overhead induced due to the coding process. Therefore, we should be very cautious with the choice of the coding window size, otherwise we may undermine the overall performance of the coding scheme.

Despite its significance, typically, existing SW RLNC schemes do not examine the impact of the coding window size. Instead, they utilize an arbitrary window size, without specifying its suitability in the applied scheme. In case of retransmissions, SW RLNC schemes usually rest upon the performance of ARQ protocols, where an optimal window is determined based on the bandwidth-delay product. However, this approach bears inherent limitations, because the coding window should be optimized based on the link loss profile. Therefore, the choice of a suitable coding window size is still an open issue in literature, especially under varying channel conditions, where the coding paradigm should adjust to the occurring variations. In this work, our primary goal is to investigate the impact of the coding window size on the performance of SW RLNC protocols. To that end, we leverage the concept of *coding depth* to facilitate the discussion. The main contributions of this thesis are the following:

1. We investigate the impact of coding depth on the definition of the coding window size in SW RLNC protocols, by analyzing the strengths and limitations of this approach both theoretically and experimentally.
2. We provide an analytical method to optimally select coding depth for certain channel conditions in order to control the performance vs complexity trade-off of the coding scheme.
3. We devise an efficient algorithm for the dynamic adaptation of the coding depth based on the channel's varying conditions.

4. We experimentally confirm, through detailed simulations, that the performance of the adaptive algorithm is comparable to the one of SW RLNC protocols where the coding window size remains invariant throughput the simulation's execution. Indeed, we can observe a noticeable reduction in complexity owing to the efficient management of the coding window size.

1.4 Structure of Thesis

The rest of this thesis is organized as follows. Chapter 2 provides the essential background on RLNC protocols and a literature review on Sliding Window RLNC variants. In chapter 3, we analyze the concept of coding depth in sliding window RLNC protocols and highlight its impact on the protocols' performance especially under dynamic channel conditions. We back up our arguments through an extensive experimental analysis comparing the impact of coding depth against the applied level of redundancy. Chapter 4 focuses on the optimal selection of coding depth for specific channel conditions and presents an adaptive approach for the dynamic adaptation of coding depth based on channel's variations. We evaluate the performance of our approach through extensive experimental evaluation in chapter 5. Chapter 6 summarizes our findings and provides a list of future extensions of this work.

Chapter 2

Background and Related Work

URLLC is considered to play a pivotal role in providing novel services and applications in the context of 5G and beyond networks. Various techniques successfully deployed in the past to mitigate the impact of errors, such as ARQ and FEC schemes, have proved to be insufficient in terms of satisfying the stringent demands of URLLC, i.e., ultra-high reliability at a low delay and vice-versa. Network Coding (NC) [4] and particularly Random Linear Network Coding (RLNC) [5, 6], is a relatively recent technique that constitutes a promising candidate in this direction. RLNC variants have been successfully examined as an efficient reliability mechanism, though, there is still room for improvement regarding the coding process. In particular, Sliding Window (SW) RLNC schemes can bring major enhancements in the coding efficiency. Inspired by the main principles of ARQ protocols, SW RLNC schemes have the potential to tackle the shortcomings of existing coded mechanisms and close the gap between reliability protocols' performance and URLLC specifications.

In this chapter, we describe the fundamental principles of RLNC protocols and thoroughly review the related literature with an emphasis on SW RLNC protocols.

2.1 Sliding Window RLNC Primer

2.1.1 RLNC Fundamental Principles

RLNC relies on the concept of linearly combining multiple packets, based on the theory of finite fields (F_{2^s}) [5]. Practically, uncoded packets, called *native or source packets*, form groups which will be used to generate *coded* packets, using finite fields' arithmetic. Assume that we have a group of n source packets $P^1, P^2, \cdots, P^n$ stored at a node. Each source packet P^i is divided into symbols of s bits and then the j-th symbol of the coded packet $e(j)$ is computed as:

$$e(j) = \sum_{i=1}^{n} c_i P^i(j) \ , \forall j \tag{2.1}$$

where $P^i(j)$ is the j-th symbol of the i-th source packet and n is the total number of source packets in the group. The set of coefficients $c = (c_1, c_2, \cdots, c_n)$, called the *encoding vector*, is chosen uniformly at random over F_{2^s}. It is proved that the probability of creating linearly dependent packets depends on the field size and becomes negligible for large field sizes [7]. In this thesis, we consider a sufficiently large finite field, such as F_{2^8}, to avoid linear dependencies of the encoding vectors. Encoding can be performed recursively at intermediate nodes, thereby encoding already encoded packets.

As for the decoding process, the receiver stores the received packets (source or coded ones) in a *decoding matrix* D, which is populated by *innovative* packets, i.e., the ones that increase the rank of the matrix. The receiver needs to solve a linear system in order to retrieve the original packets P^i. Decoding is performed using Gauss-Jordan elimination when the matrix reaches full rank. Partial decoding [8], i.e., decoding when there exists a full rank sub-matrix of D, and other decoding optimizations also exist in literature [9, 10, 11]. Decoding is optional at intermediate nodes but it should be performed at the receiver.

2.1.2 RLNC Variants

In RLNC, a major performance factor is the complexity of coding operations. High complexity incurs large delays, hence a finite set of packets should be involved in the coding process. Apart from the group size, determining the way of populating the group is crucial in terms of the protocol's efficiency. RLNC schemes can be broadly

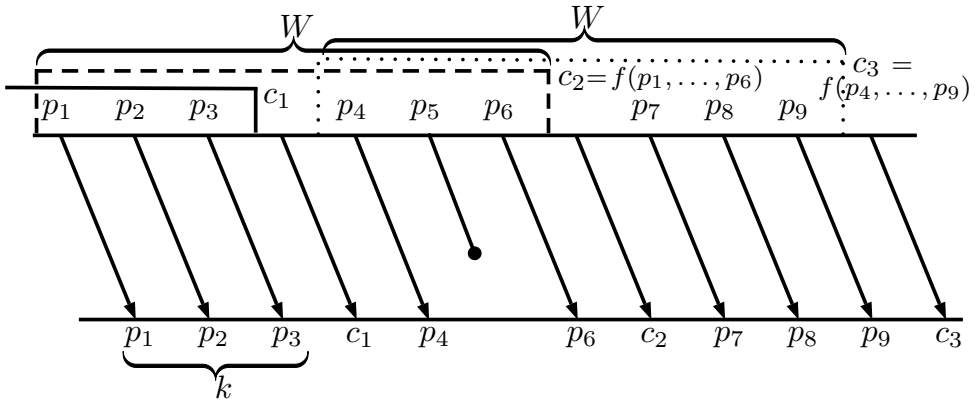


Figure 2.1: An illustration of the fixed-size sliding window RLNC concept ($W=6$ and $R=3/4\,(k=3)$), dashed and dotted lines indicate the sliding window contents when a coded packet is created).

categorized in *block-based* RLNC [8, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22] and *sliding-window* RLNC [11, 13, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]. In the first class, encoded packets are created using fixed-size groups of g packets, called $generations$. None subsequent packet can be involved in coding if the decoding of the previous generation is not complete. This, however, significantly increases delay and makes such schemes inappropriate for URLLC applications.

To address this problem, sliding window RLNC methods adopt the concept of the *sliding window* in the coding process. Coded packets are created as random linear combinations of a dynamically changing group of source packets known as the sliding (or coding) window. Here, instead of employing fixed-size sets of packets, as in block-based schemes, the management of the window's contents is quite flexible. New source packets are inserted whenever they are available at the source, whereas older ones are removed in a process known as *closing* the window. SW RLNC schemes usually adopt a systematic approach, that is, coding is applied only on the redundant packets while source ones are transmitted in an uncoded form. Figure 2.1 illustrates a typical example of a systematic SW RLNC protocol, borrowed from [23, 24]. Except for outlining the main functionality of such a scheme, this figure displays two major parameters that affect the coding performance of SW RLNC protocols. First, the number of source packets involved in the coding process delineates the coding window size (W), while the level of redundancy introduced into the original data stream is determined by the code rate R. The latter is defined as the ratio of the number of source packets of the coding window to the total number of packets (source and

redundant ones) sent by the transmitter. In practice, we express R as:

$$R = \frac{k}{k+r} \tag{2.2}$$

where k symbolizes the number of source packets and r the number of redundant ones. The coding window size and the code rate play a major role in determining the trade-off between coding efficiency and protocol's performance.

Returning back to figure 2.1, we can observe that the arrival of new packets forces the window to move forward and drop older packets so that the window size never exceeds the predetermined size W. In this example, only one coded packet is injected in the stream every k source packets, thereby defining the code rate equal to $R = \frac{k}{k+1}$. The coded packets are destined for recovering any lost source packets. Recovery is indeed possible when the receiver collects a number of linearly independent coded packets equal to the number of lost source packets. For instance, the recovery of lost packet p_5 can be achieved either using c_2 or c_3.

2.2 Related Work on Sliding Window RLNC

To date, extensive research has been conducted on the impact of applying SW RLNC protocols in various fields, such as multimedia [30, 31, 32], IoT scenarios [38] and opportunistic routing [28]. Additionally, SW RLNC serves as the basis for designing and investigating more complex coding approaches which rely on its main principles. For instance, [40] examines the performance of Fulcrum SW codes while in [41], the authors investigate SW BATS codes. SW RLNC-based reliability mechanisms have also been examined in the context of transport-layer protocols, initially regarding the TCP protocol [29] and, more recently, QUIC [44].

The proposed SW RLNC schemes [11, 13, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43] share some common concepts regarding the coding process which could be used to categorize existing works according to these features. Figure 2.2 illustrates a classification of existing SW RLNC schemes. First of all, SW RLNC protocols can be classified into *full-vector* [35, 36] and *systematic* [45], based on the "per-packet" coding approach. As for the first class, each transmission packet is obtained by coding, i.e., linearly combining all source packets currently existing in the sliding window. On the contrary, systematic approaches transmit a

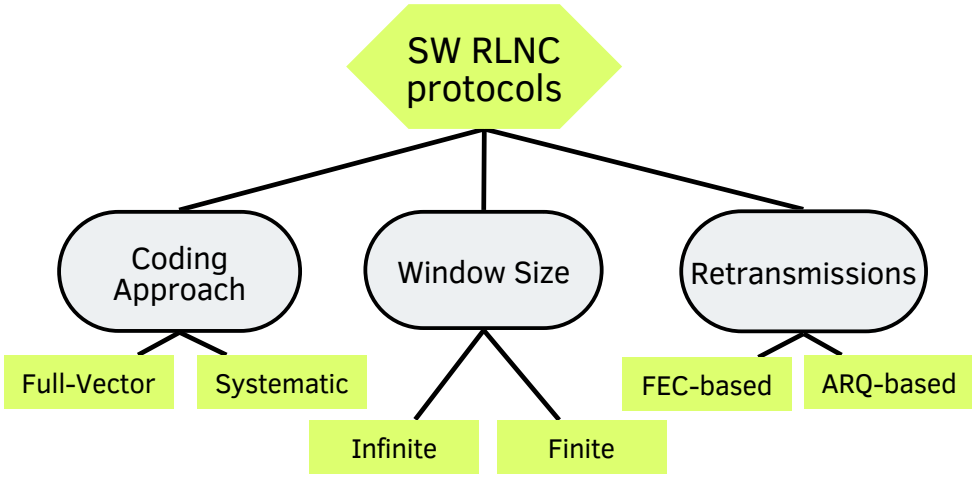


Figure 2.2: A classification of existing SW RLNC protocols.

mixture of source and coded packets and coding is applied only on the redundant packets. Actually, coded packets are interleaved within the data stream typically by transmitting a coded packet after sending k source packets. In this way, packet losses can be recovered more quickly, hence achieving lower mean packet delivery delay compared to full-vector codes [24]. Since achieving low delay is a common target for URLLC applications, the majority of the existing SW RLNC schemes adopt systematic codes.

As mentioned in Section 2.1.2, the performance of SW RLNC schemes heavily depends on R and the SW size. Devising an efficient strategy to manage the SW is quite important because the SW size determines which source packets are included in the generation of a coded one and therefore "demand" protection through coding. Although inserting new source packets is straightforward, various policies for removing packets, i.e., closing the window, have been discussed. Some algorithms adopt an *infinite window* approach [13] where source packets are never removed from the window, that is, all previously transmitted source packets contribute to the generation of a new coded one. This strategy, though, cannot be applied in practice because of the prohibitive computational complexity of the coding processes as well as the increased memory requirements for buffering source packets at both sides. Most algorithms follow a *finite* window approach [11, 23, 24, 30, 32, 33], where the number of source packets in the SW is upper bounded by the coding window size. Two policies can be adopted in order to ensure that SW never exceeds this size. In the first one, old packets are removed from the window only when the addition of new

packets would result in exceeding W [33]. On the other hand, employing feedback from the receiver would allow the sender to remove delivered packets from its buffer, and hence from the decoding process. This is achieved, for instance, through the use of periodic acknowledgements [30, 31, 34], as demonstrated in Tetrys protocol [30]. There, periodic acknowledgements are used in order to "close" the window, while loss recovery is achieved only through coded redundancy.

Utilizing feedback in SW RLNC schemes is not solely exploited in closing the SW, though. A common strategy entails the use of feedback to trigger re-transmissions in a similar fashion as in ARQ protocols [11, 23, 24, 10, 25, 26, 27, 28, 29, 32, 35, 36]. RLNC serves the following purpose: coded redundancy is injected into data stream to overcome packet losses without inducing additional delay because of re-transmissions. Unless redundant packets suffice for loss recovery, feedback activates the transmission of additional packets (coded or not) to facilitate the decoding process and increase the probability of packet recovery. Caterpillar RLNC-FB [11] is a representative ARQ-based SW RLNC protocol employing systematic coding, re-transmissions and feedback. On the contrary, SW RLNC protocols that do not incorporate feedback fall into the category of *FEC-based* schemes. The main difference of ARQ-based and FEC-based SW RLNC lies in the use of re-transmissions in the presence of feedback. FEC-based schemes rely exclusively on coded redundancy to recover lost packets [33] and they leverage feedback, if available, to solely update the coding window's contents [30, 31].

According to figure 2.2, the coding window size is a critical parameter for the performance of the coding schemes. Nevertheless, existing finite window schemes tend to become complacent about the superiority of the finite window against the infinite one and they do not elaborate further on the reasoning behind choosing the examining coding window sizes. Instead, in FEC-based schemes, coding window size is arbitrarily defined whereas, in ARQ-based SW RLNC [11, 10, 25, 26, 27, 29, 32, 35, 36], a single window is associated with both the data flow process and the coding one, i.e., they use the sliding window to maximize data flow and generate redundant packets at the same time. In our previous work [23, 24] we identified this issue regarding ARQ-based SW RLNC protocols as a practice that poses limitations in the performance of SW RLNC protocols due to the conflict of interest regarding the SW size. On the one hand, flow maximization relies on the bandwidth-delay product whereas coding parameters are chosen based on the link loss profile. Contradictory

requirements are more evident in links with high bandwidth-delay product, e.g., satellite links, where a large window size is optimal for maximizing data flow, but the link loss profile may dictate a different configuration for the SW size. In particular, a large window size increases the coding complexity at both sides which also incurs larger delay, hence the performance of SW RLNC protocols significantly degrades. We addressed this issue by utilizing two distinct windows; the ARQ-like sliding window ($\mathbb{S}_w$) for maximizing data flow and the coding window ($\mathbb{C}_w$) used at the coding process. However, despite our initial effort to formulate an appropriate coding window size, the problem of choosing a suitable value for the coding window remains an open issue in literature, both in ARQ-based and FEC-based schemes. It is evident that utilizing some seemingly appropriate coding window sizes is a restrictive factor regarding the efficient performance of a SW RLNC coding scheme and we should not rest on our laurels. Au contraire, it is essential that a systematic approach for determining the coding window size be devised in order to maximize protocols' efficiency. In the following chapters, we are going to discuss about the importance of properly defining the coding window size and highlight the benefits of choosing an optimal coding window under dynamic channel conditions. Especially for the occasions of arguing for ARQ-based schemes, we will utilize the notation introduced in rapidARQ [23, 24], our protocol that embodies the novelty of decoupling the coding and the sliding window.

Chapter 3

Addressing the Limitations of SW RLNC with the Use of Coding Depth

---

While the idea of differentiating between the sliding and the coding window seems to be an efficient solution, little do we know so far about the determination of the coding window size. In [23, 24], we facilitate the discussion about the coding window by introducing the concept of *coding depth (d)*. The latter refers to the protection a coded packet offers to a specific number of k-packet groups constituting the redundant packet. By means of coding depth, we propose a practical way to articulate the factors affecting the performance of SW RLNC protocols and the envisioned benefits of optimally choosing the coding window size. Since little attention has been paid to the coding window size in SW RLNC schemes, there is plausibly no prior work dwelling upon the advantages of an optimal coding window. Note that this problem is especially important under dynamic channel conditions. Thus, our aim is to analyze the appropriate selection of the coding window size via the use of d as well as the strengths and limitations of the online adaptation of d. Our motivation derives from the fact that static configuration of RLNC coding parameters is usually insufficient

when the packet loss of the channel varies. A common approach proposed in literature for coping with loss variations and increase robustness focuses on dynamically adapting the code rate R. While rate adaptation seems to be an appropriate solution, it comes at the high cost of bandwidth consumption required for transmitting more redundant information. On the contrary, we make the observation that the dynamic adaptation of coding depth outweighs rate adaptation in that it does not require additional bandwidth resources. Therefore, it looks like a good prospect in tackling packet losses in error-prone channels.

In this chapter, we first elaborate on the concept of the coding depth and its impact on the coding window size. We also provide a brief review of the existing literature regarding rate adaptation and argue for the benefits of manipulating d both in static and dynamic configuration of RLNC protocols in order to cope with packet losses. We back up our arguments with experimental data suggesting that an appropriate selection of the coding window size (equivalently of d) may outperform the strategy of adapting the coding rate in terms of throughput-delay.

3.1 Analyzing the Concept and the Benefits of Coding Depth

An effort to formally describe the coding window size has been made in our previous work [23, 24], where we first utilize the concept of coding depth in the context of ARQ-based SW RLNC protocols. We define the coding window as:

Definition 3.1. Coding Window ($\mathbb{C}_w$): The subset of $\mathbb{S}_w$ containing the most recent not-yet acknowledged source packets which are used in the encoding process.

The range of source packets comprising $\mathbb{C}_w$ depends on the level of protection we would like to offer to them, i.e., the number of redundant packets "covering" the source ones in case of losses. For a specific code rate R the level of redundancy is specified by the coding window size. *The coding depth (d) incarnates that level of protection, as it describes the number of k-packet tuples constructing a coded packet.* Hence, we define the:

Definition 3.2. Coding Window size (CW): The maximum number of source packets in $\mathbb{C}_w$ which is equal to

$$CW = d \times k \tag{3.1}$$

CW is chosen as an integer multiple of k, as $\mathbb{C}_w$ should provide equal protection to all source packets comprising it. However, so far the choice of d (or CW) is empirical and there is no previous work on how to optimally decide it. Nor has its impact been discussed on the coding process under varying channel conditions. A randomly chosen CW can cause various implications in the performance of the coding scheme. In particular, a small CW leads to reduced redundancy, which, in turn, offers fewer opportunities for decoding. On the contrary, overestimating the window increases coding complexity and may interfere with the other sliding window parameters, leading to performance degradation [23, 24].

By selecting the optimal d, we are able to minimize coding complexity and delay without sacrificing coding efficiency. We should bear in mind that by fine-tuning $\mathbb{C}_w$, we can obtain performance improvements both in the encoding and the decoding operation. Indeed, the maximum number of source packets involved in the encoding process affects not only the encoding process but also the size of the encoding vector appended in a transmitted coded packet. During decoding, an optimal $\mathbb{C}_w$ results in a smaller decoding matrix size and therefore in reduced computational complexity during Gaussian elimination. The proper adjustment of d and, by extension $\mathbb{C}_w$, signifies that the redundant coded packets have higher probability of compensating the resultant packets losses, hence less redundancy is stored at the decoding matrix. Since the size of the matrix is reduced and that size is the only factor affecting the Gaussian elimination, the optimal configuration of CW using d demonstrates lower decoding complexity and, hence, significantly reduced delay.

3.2 Rate Adaptation in RLNC Schemes

Although the size of the coding window is a powerful tool for mitigating the impact of packet losses, the most common approach in the related literature is to increase the level of redundancy, i.e., the code rate R, e.g., [14, 16, 17, 19, 20, 21, 22, 35, 36, 42]. This is typically done by changing k, the number of source packets between two successive coded ones. The rationale is clear; by increasing k, the probability of packet losses is expected to be smaller. Even though this practice is effective, its drawback is that the increased redundancy compromises the protocol's throughput.

3.2.1 Existing Literature on Rate Adaptation

A significant research effort has been made in exploring adaptations of various coding parameters based on feedback from the receivers. We could use various factors to classify existing research in this direction, such as the RLNC variant, e.g., block-based or SW-based RLNC, or the objective of rate adaptation, i.e., modify k or r based on equation 2.2. We choose to review these works following the categorization based on the coding paradigm. Although only SW RLNC schemes belong to the scope of this thesis, we examine both block-based and SW schemes in this section for the sake of completeness.

In *block-based* RLNC schemes [14, 15, 16, 17, 18, 19, 20, 21, 22, 46, 47, 48, 49, 50], the code rate R is shaped based on the generation size g. Hence, the majority of adaptive coding schemes focus on modifying g to enhance performance. In particular, [14] uses a heuristic method to adaptively determine the maximum generation size after measuring the average input load in the system for a predetermined time period, while [46] and [50] rely on feedback to adjust coding redundancy by estimating the link loss rate. [22] examines the dynamic adaptation of g in the context of the TCP protocol and utilizes the Generation Round Trip Time (GRTT), i.e., the time period between the transmission of the first packet of a generation and the reception of its last ACK, to estimate the next generation size. A time-variant optimization of the generation size is examined in [15], where the authors develop an adaptive scheme that sequentially adjusts the block size in polynomial time in order to maximize throughput under hard transmission deadlines. [47] presents an adaptive redundancy scheme that utilizes MAC acknowledgements as a feedback regarding the link quality. By exploiting these acknowledgements, the scheme decides when is the appropriate time to pump redundancy in the network to tackle packet losses. Also oriented towards data link layer protocols, [16] designs an adaptive coding scheme that utilizes two encoding methods, where the constraints imposed by the MAC and physical layers determine which encoding method and generation size can ensure a specific QoS level. The latter is also the objective of [17], where a feedback control mechanism adjusts g according to the network load variations for multicast flows with the aim of achieving efficient throughput-delay performance.

Dynamic adaptation of g has also been investigated in the context of wireless broadcast channels [18, 21, 48]. Actually, [21] shows that the optimal generation size

can be expressed as a function of the network size, while [48] formulates a dynamic rate adaptation scheme that uses performance prediction models and feedback from the receivers to regulate the sender's transmission rate. Since this scheme affects which packets currently exist in the transmission queue and the coding scheme employs only packets from that queue, the proposed dynamic adaptation scheme implicitly determines the generation size. An interesting variation of systematic block-based RLNC is Pace [20], in which the transmission of redundant coded packets is interspersed throughout the generation, thus creating a form of sub-generations. This implies that redundant packets transmitted at the beginning of the generation protect fewer source packets, while the subsequent ones include all the sub-generations transmitted until then. While this strategy allows the receiver to decode earlier in case of a packet loss at the beginning of the generation, this approach offers more protection to the first source packets of the generation than to the subsequent ones, whilst a coding scheme should ensure equal protection to all transmitted packets. Lastly, adaptive redundancy has been examined even in the context of SDN [49], where the SDN controller receives as input the actual packet losses monitored by SDN switches and calculates the amount of redundancy required to compensate for the losses given a certain decoding probability.

As for *sliding-window* RLNC schemes, a number of studies examine the dynamic adaptation of rate [31, 35, 36, 37, 42, 43, 51]. In [31], a redundancy adaptation algorithm for real-time video transmission is designed based on Tetrys with the aim of minimizing packet losses. The receiver incorporates a monitoring agent, which observes packet losses and induced delay. The information collected by the monitoring agent is forwarded to the redundancy adaptation module, which is reponsible for making the receiver send a feedback message requiring a redundancy increment or decrement. Another approach studies a tiny coding scheme ($W = 2$) with cumulative feedback, where its contents regulate the code rate [51], while in [43] a scheme for jointly selecting link rate and adaptive redundancy is proposed. Adaptive coding has also been studied concerning the recoding process at intermediate nodes [37, 52]. Finally, [35] proposes a SW scheme that monitors the channel conditions in order to adjust its coding window size in a causal fashion. The proposed model learns the loss pattern through feedback and adaptively adjusts the level of redundancy via two mechanisms. The first one acts proactively, as it sends redundant packets in advance to prevent decoding failure based on the model's estimation of channel behavior.

The second mechanism is reactive, since it triggers the transmission of additional redundancy to compensate for packet losses identified by feedback information. The aforementioned proposal has also been generalized for multipath multi-hop communications [36].

3.3 Rate adaptation vs Coding depth: Strengths and Limitations

As we explained in Section 3.2, adapting the coding rate comes at the cost of bandwidth consumption in order to cope with increased errors. This is because more redundant packets need to be transmitted. On the contrary, adjusting the coding depth d does not pose additional overhead in throughput as we continue sending out the same number of redundant packets at the expense of a slight overhead in the coding complexity. Should an adequate coding depth be in use, the adjustment to short-term loss variations can be more efficient than in conventional SW RLNC schemes.

The aforementioned arguments manifest that the proper adjustment of CW (or equivalently d) can assist in tackling the inherent deficiencies of existing SW RLNC schemes and make an important step towards achieving URLLC. So far, we have shown that, in the context of ARQ-based SW RLNC protocols, rapidARQ, which allows adjusting the coding depth, demonstrates superior throughput-delay and coding performance compared to traditional ARQ-based SW RLNC schemes [24]. A concept similar to ours is proposed in [38] destined only for FEC-based schemes, where the authors introduce the notion of *overlap*, as the number of coding windows/groups to which a source packet belongs. However, that work does not highlight the impact of the novelty on the decoding process (they assume infinite decoding window at the receiver) nor do they examine the scheme's performance with adaptive overlap values under varying channel conditions. We expect that the importance of the optimal adaptation of CW is even greater under dynamic channel conditions, particularly when source packets require increased protection. We claim that the dynamic adaptation of coding depth outweighs the strategy of rate adaptation.

To validate our observation, we evaluate the performance of varying d in contrast to variable code rate (R) using rapidARQ [23] and Caterpillar RLNC-FB [11], two of the most representative ARQ-based SW RLNC protocols. We use two different scenarios

Table 3.1: Experiment's Main Parameters.

Parameter	*5G link*	*Satellite link*
Propagation delay	66.7μs	4ms
Bandwidth	500Mpbs	1Gbps
Packet error rate (p_l) (one-way)	5%	10%
Sliding Window Size (W)	43	668
Packet Size	200 bytes	1500 bytes
Traffic Type	CBR	CBR

for this comparison, based on the evaluation settings in [24]. We are interested in the throughput-delay performance, i.e., the set of throughput values and delivery delays achievable by the system. The first scenario refers to a typical 5G terrestrial wireless link whereas in the second one, we examine the performance over a high-speed link to a low-earth orbiting satellite. Table 3.1 summarizes the parameters used in this experiment for each test scenario. We consider a uniform model for channel's losses, i.e., errors are uniformly distributed in time. Regarding rapidARQ, R is set according to the link's loss rate (p_l), so that, on average, the coding redundancy is sufficient for coping with the average number of expected packet losses. More specifically, we define $R = 9/10$ in the 5G scenario and $R = 4/5$ in the satellite case. The coding depth can range between $[1, \lfloor \frac{W}{k} \rfloor]$. In the 5G scenario, we depict all possible values, namely $d \in [1, 4]$, whereas in satellite scenario, we choose to illustrate indicative values, mainly covering the smallest and the largest possible ones, as well as the configurations resulting in the best throughput-delay performance. In particular, we depict the performance of rapidARQ when d takes one of the following values: $2, 3, 5, 7, 20, 85, 167$. As for Caterpillar RLNC-FB, apart from the optimal R configuration (based on p_l as in rapidARQ), we examine the performance in cases where R is smaller or greater than the optimal value. In other words, we study the cases where additional coded packets are sent out every fewer coded ones or more sparsely ($k' \geq$ optimal k). We examine the performance of both protocols under the existence of re-transmissions and without them.

Figure 3.1 illustrates the throughput-delay performance of Caterpillar RLNC-FB and rapidARQ in the 5G scenario. At first sight, rapidARQ outweighs Caterpillar for every value tested. Regarding Caterpillar, we can observe that increased redundancy

($R = 7/8, R = 8/9$) results in poorer throughput performance, as we expect theoretically. Additionally, the larger the redundancy, the smaller the average delay is, since redundant coded packets are transmitted more often and this allows the receiver to quickly recover any missing source packets. In fact, in case of no re-transmissions, when $R = 7/8$, the average delay is at least 25% smaller than the rest of the values tested, at the expense of slightly reduced throughput. The overhead in throughput runs up to 3.5%, compared to $R = 10/11$. The advantage in average delay dissipates, though, when re-transmissions are included, since they mostly account for packet recovery at the receiver side. By accepting a slight overhead in average delay, we can apply a scarcer coding rate, e.g., $R = 9/10$, which results in improved throughput performance.

On the contrary, rapidARQ proves that it performs optimally simply by adjusting d and without modifying R. While in Caterpillar increased redundancy aims to enhance protection of source packets, we manage to combine increased protection and superior overall performance by only adapting d. Not only do we accomplish comparable average delay with $R = 7/8$ with and without re-transmissions, but also we outperform Caterpillar massively regarding throughput. In particular, we notice a net throughput benefit of ∼30Mbps in $RTX = 0$ and ∼25Mbps in $RTX = 3$, which corresponds to a 10% and 7% improvement, respectively, compared to Caterpillar RLNC-FB.

Similar results are obtained in the satellite scenario, too. Figure 3.2 highlights the superior throughput-delay performance of rapidARQ against Caterpillar RLNC-FB. Here, the throughput benefits of rapidARQ are quite more evident, depending on the applied value of R in Caterpillar. In case of high redundancy ($R = 2/3$), the throughput benefits can reach up to 69% and 75% for $RTX = 0$ and $RTX = 3$, respectively. For the default code rate ($R = 4/5$), the net increase in throughput runs to $\sim 169Mpbs$ with and without re-transmissions, which corresponds to around 40% increase with the same introduced redundancy. As for the average delay, rapidARQ achieves minimum delay equivalent to the one noticed when $R = 2/3$ in Caterpillar. RapidARQ's performance can be considered poor only in very small values of d, such as $d = 2$, where the achieved delay is double as the minimum one and throughput performance depends on the existence of re-transmissions. In case of the latter, the importance of the decoding process is neutralized by the impact of re-transmissions, hence the average delay remains small in Caterpillar regardless of the code rate.

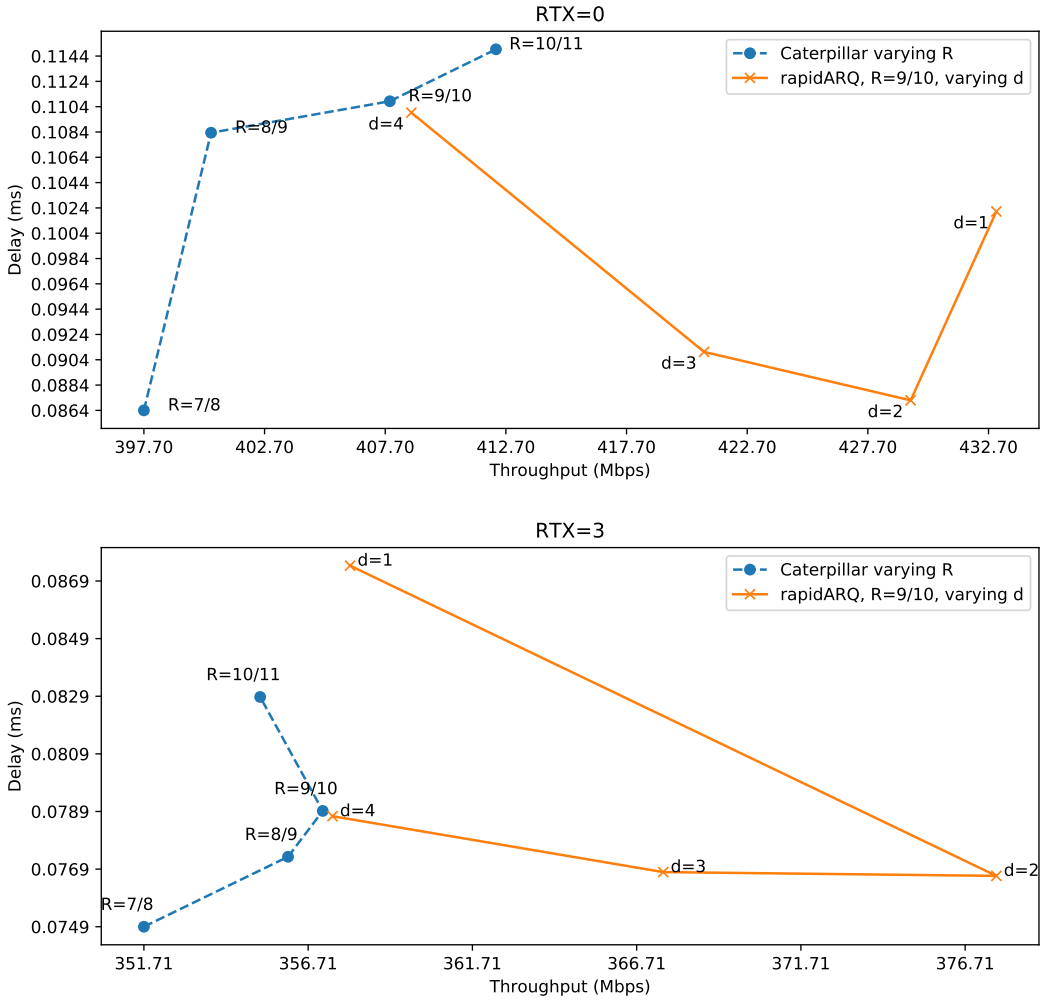


Figure 3.1: Throughput-delay performance of Caterpillar RLNC-FB [11] and rapidARQ [23] in 5G test scenario. The above plot corresponds to the protocols' performance when $RTX = 0$ while the below one to $RTX = 3$. Labels indicate the various values of R and d examined in this test scenario.

Without re-transmissions, though, Caterpillar RLNC-FB substantially falls short of the expected delay, since it introduces an overhead of around 70% with the default code rate R. Therefore, Caterpillar performs poorly either regarding delay or throughput depending on the level of protection it offers to the source packets.

The results obtained from the uniform error model can be generalized in bursty models as well. The previous comparison indicates that the advantages of rate adap-

tation come at a high throughput cost. On the contrary, by altering only d, rapidARQ demonstrates an improved performance. Equally noteworthy is the fact that a wide range of d values can perform efficiently. Actually, rapidARQ's performance degrades only for very small or large d values. In figure 3.2, when d ranges in $[5, 20]$, rapidARQ achieves almost optimal performance. The improvement is notable against Caterpillar even when $d = 85$. However, the optimal range of values for d depends on the test scenario and has yet to be determined.

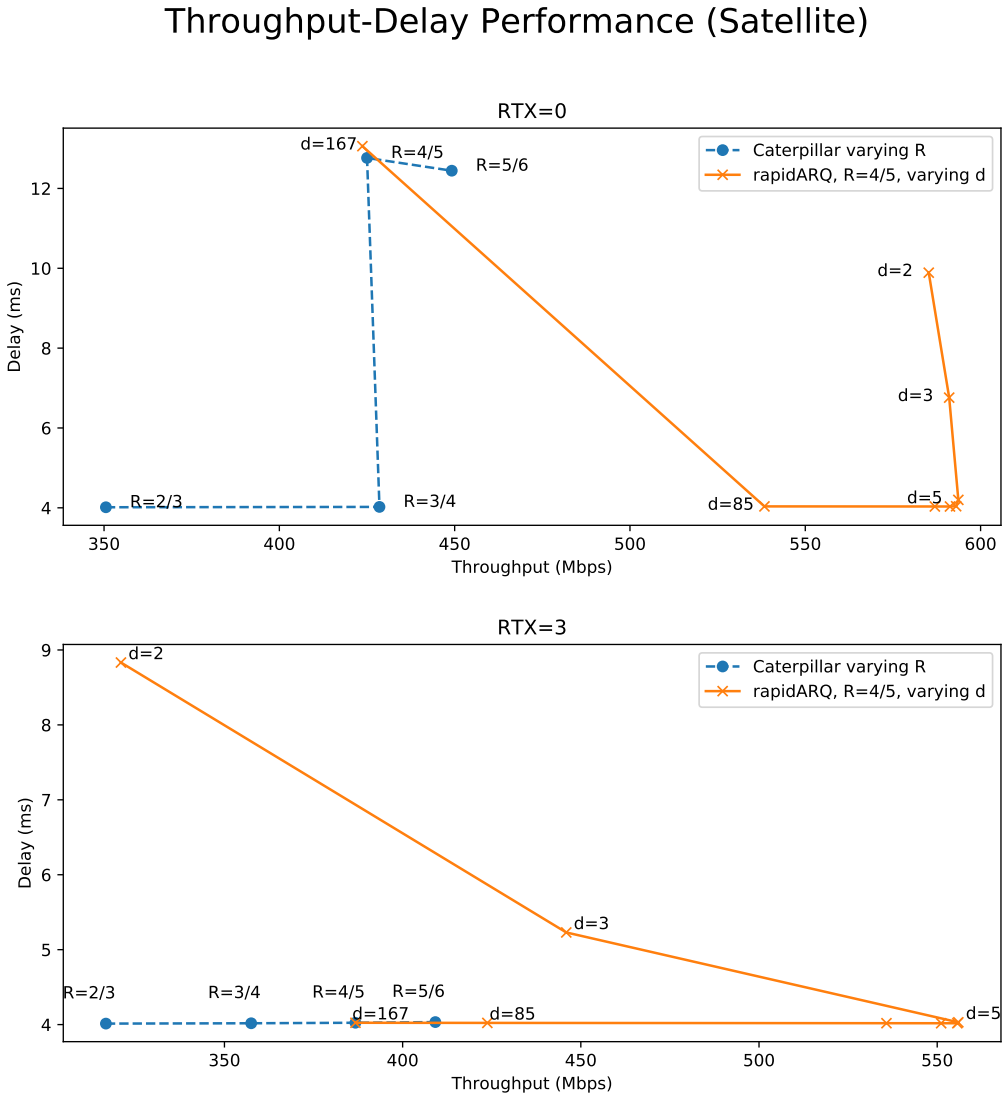


Figure 3.2: Throughput-delay performance of Caterpillar RLNC-FB [11] and rapidARQ [23] in satellite test scenario. The above plot corresponds to the protocols' performance when $RTX = 0$ while the below one to $RTX = 3$. Labels indicate the various values of R and d examined in this test scenario. Labels for values $d = 7$ and $d = 11$ are omitted for clarity reasons.

Chapter 4

Dynamic Adaptation of Coding Depth

In chapter 3 we experimentally proved that the use of coding depth does result in superior performance under dynamic channel conditions. However, an efficient mechanism for dynamic adaptation of d under varying channel conditions has yet to be examined. In the following sections, we outline the properties of d for achieving near-optimal performance by providing a theoretical analysis to support these arguments. Subsequently, we discuss an efficient algorithm for adaptive configuration of d under variable channel loss rate.

4.1 Exploring the Vicinity of the Optimal d

As we observed in section 3.3, rapidARQ exhibits increased robustness in conjunction with enhanced throughput-delay performance under variable packet loss rates. This superior performance is attributed to the concept of coding depth that makes possible the maximization of coding benefits. Interestingly enough, rapidARQ performs near-optimally not for a single d value but for a wide range of values (see, for example, fig. 3.2). This is a reasonable result since increasing d also increases the offered

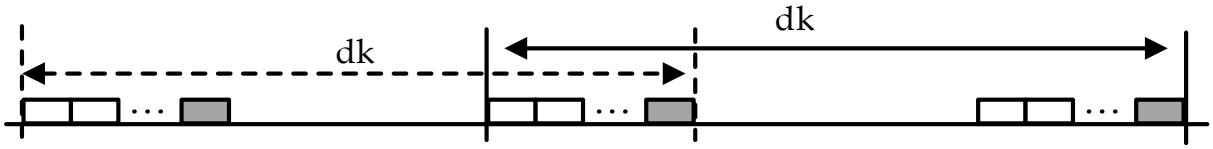


Figure 4.1: Successive coding cycles observed at the sender side in a SW RLNC protocol. The product $d \cdot k$ describes the coding window size.

protection. When d goes beyond a threshold value (d_{th}), which corresponds to the required protection, no significant improvement is witnessed. However, depending on the channel properties and loss profile, it is not always the case that a wide range of d values can produce a near optimal performance. For example, this is the case of the 5G scenario discussed in the previous chapter. As can be seen in fig. 3.1, the range of d values that optimize performance is rather narrow (i.e., $d = 2, 3$). The reason is that, while d should be greater than d_{th}, it also should not exceed an upper value d_{up}. In other words,

$$d_{th} \leq d \leq d_{up} \tag{4.1}$$

We discussed the existence of d_{up} in our previous work [24]. This upper limit is imposed by the limited sliding window size at the receiver side. To elaborate on this, let us examine fig. 4.1 where a series of coding cycles at the sender is presented. As can be seen, when a coded packet arrives at the destination, the decoding process involves all the native packets within the coding window, i.e., the $d \cdot k$ last native packets, and the corresponding coded ones in this range. However, note that the oldest coded packet in this group of packets also "contains" the previous $d \cdot k$ native packets. As a result, in the worst case, in any given moment, the last $(2d - 1) \cdot k$ packets are involved in the decoding process at the receiver. However, the receiver has a limited sliding window size W. Therefore, $(2d - 1) \cdot k \leq W$, otherwise the receiver will start dropping packets that are outside the sliding window although those packets may be necessary for the decoding process to complete. The previous limitation results in

$$d \leq \frac{1}{2}(\lfloor \frac{W}{k} \rfloor + 1) = d_{up} \tag{4.2}$$

Clearly, d_{up} depends on the channel characteristics since W is set based on the channel's bandwidth-delay product and k is chosen based on the packet loss rate.

It is straightforward from the previous discussion that, depending on the channel characteristics, it may either be that $d_{up} \leq d_{th}$ or $d_{th} \leq d_{up}$. It is reasonable that the

former inequality holds in channels with small bandwidth-delay products and/or relatively large loss rates while the latter is valid in channels with large bandwidth-delay products and rather small loss rates. In general, the optimal value for d can be determined as

$$d_{opt} = min\{d_{th}, d_{up}\} \tag{4.3}$$

This equation can explain the behavior of rapidARQ witnessed in the experiments of Chapter 3. In the case of the 5G scenario, the bandwidth delay product is rather small therefore the best value for d is dominated by d_{up}. The opposite is true for the satellite scenario where the large bandwidth-delay product indicates that $d_{opt} = d_{th}$. Keep in mind that, in this case, rapidARQ's throughput-delay performance is near optimal for all values in the range $[d_{th}, d_{up}]$ since all those values provide the required protection.

While calculating d_{up} is straightforward, determining d_{th} is a daunting challenge because it depends on a variety of coding/decoding design choices. Not surprisingly, to the best of our knowledge, such an analytical calculation does not exist in literature. Our approach is to derive an analytical framework (see Section 4.2) that will allow us to obtain a rough estimation ($\tilde{d_{th}}$) of d_{th}. In the case that $\tilde{d_{th}} > d_{up}$ clearly the best choice for d is d_{up}. However, when $\tilde{d_{th}} < d_{up}$, our estimation will produce a near optimal performance if $\tilde{d_{th}} \in [d_{th}, d_{up}]$.

4.2 Calculating an Efficient d Estimation

In this section we provide the theoretical analysis for computing a rough estimation of d_{th}. As previously explained, in rapidARQ, the receiver side can perceive the packet reception/decoding process as consisting of a repetition of *coding cycles*. Each coding cycle consists of k native (or uncoded) packets and a coded one. The coded packet in a specific coding cycle is built based on the native packets comprising the last d coding cycles, including the current one. If the receiver misses native packets (because they are lost) then, upon the reception of a coded packet, checks whether decoding the lost packets is possible. Let us assume that at time $t = 0$ the decoder is not missing any packet and a new coding cycle starts where at least one native packet is missing. The decoder will try to recover the lost packet(s) at the end of the decoding cycle, i.e., after receiving redundant information in the form of the coded packet. If this

not possible the process will continue with another chance to recover the packet(s) in the second coding cycle. Note that, in the meanwhile, the number of missing packets may increase if native packets belonging to the second coding cycle are missing. The process will continue in the following coding cycles until decoding (the entirety of the missing packets) is possible or is confidently deemed unrecoverable. For the latter, we assume in this work, that this happens after d cycles because beyond that point no more coded packets containing the lost packets in round 1 will be received[1].

We wish to calculate the probability that the receiver will not to be able to recover (through decoding) the missing packets after i coding cycles. Let $\widetilde{P}_i$ denote this probability. We assume that X_i denotes the random variable corresponding to the number of lost packets (either native or coded) during coding cycle i. Note that X_i is binomially distributed, i.e., $X_i \sim B(k+1, p)$ where $k+1$ is the number of packets (native and coded ones) in the coding cycle and p is the channel's loss rate. Let also W_i denote the opposite of the decoding matrix's rank deficiency, i.e., the difference between the decoding matrix's rank and its number of rows (number of linearly independent packets), at the end of cycle i. In other words, W_i represents the number of linearly independent packets needed by the receiver to perform a decoding. Note that:

$$W_i = max(0, W_{i-1} + X_i - 1), \;\; \forall\, i > 0,\; W_0 = 0 \tag{4.4}$$

The previous formula is easy to understand if we keep in mind that there are two possibilities for the X_i packets missing during cycle i. The first is that all X_i packets are native and the one coded packet offers a linearly independent combination containing those packets therefore it can increase the rank of the decoding matrix. The second possibility is that only $X_i - 1$ packets are native and the one coded packet is lost. In this case, again, W increases by $X_i - 1$. It is clear that $W_i = 0$ implies that decoding is possible at round i. Similarly, no decoding is possible after i coding cycles iff $W_i > 0$ and $W_j > 0 \,\forall\, j < i$. Therefore, we can write $\widetilde{P}_i$ as:

$$\widetilde{P}_i = \sum_{x=1}^{ik} P\{W_i = x \mid W_j > 0, \forall\, j < i\} \tag{4.5}$$

[1]Whether there is a possibility of decoding or not after this point heavily depends on the policies implemented at the decoder for keeping the decoding matrix at a reasonable size. Our assumption is based on the reasonable strategy not to wait for a possible decoding beyond that point in time. This is based on the rationale that the probability of such an event is small and the trade-off in terms of delay and decoding matrix size is not negligible.

Note that the maximum value of W_i is $i \cdot k$. This can be derived by (4.4) if $X_j = k+1,\ \forall\, j$, which corresponds to the worst case where all native and all coded packets are lost in every coding cycle. Given $\tilde{P}_i$, we can write the probability of decoding in any of the cycles up to i as:

$$\hat{P}_i = 1 - \widetilde{P}_i \tag{4.6}$$

Moreover, we can prove that the probability of decoding at cycle i, i.e., without any decoding at a previous round, is:

$$P_i = 1 - \widetilde{P}_i - \hat{P}_{i-1} \tag{4.7}$$

Proof. By its definition, P_i can be written as $P_i = P\{W_i = 0 \mid W_j > 0, \forall\, j < i\}$. Thus, we can derive P_i by examining all sequences of $W_1, W_2, \ldots W_{i-1}, W_i(=0)$ and eliminating those containing at least one 0 in any place $j \leq i-1$ because those latter sequences correspond to chains of events where a decoding occurs before cycle i. Equivalently, we first examine all possible sequences of $W_1, W_2, \ldots W_{i-1}, W_i$. The probability of any such sequence occurrence is clearly 1. Then we exclude those sequences that contain at least 0 in any place $j \leq i-1$. Note that, by definition, the probability that any such sequence takes place is $\hat{P}_i$. Finally, from the remaining set of sequences, each of which does not contain any 0 in any position $j \leq i-1$, we exclude the ones that end with $W_i > 0$. In other words, we exclude that sequences that end in 0 but do not not contain any other 0s. The probability of any such sequence taking place is, by definition, $\widetilde{P}_i$. □

Going back to (4.5), clearly, it is easy to derive $\widetilde{P}_i$ if we can analytically compute $P_x^{(i)} = P\{W_i = x \mid W_j > 0, \forall\, j < i\}$, i.e., the probability that the receiver's decoding matrix requires x linearly independent packets at cycle i while no decoding has occurred in the previous rounds. To this end, as a first step, we use (4.4) recursively to re-write $P_x^{(i)}$. More specifically, when $W_j = 0\, \forall\, j < i$, we can conclude from (4.4) that $W_i = X_1 + \cdots + X_i - i$. As a result, $P_x^{(i)}$ can be re-written as:

$$P_x^{(i)} = P\{X_1 + \cdots + X_i = x + i \mid W_j > 0, \forall\, j < i\}$$

This makes the computation of $P_x^{(i)}$ easier since $X_1, X_2, \cdots, X_i$ are all binomial random variables. Therefore, the sum of those variables is also binomially distributed with $X_1 + \ldots + X_i \sim B(i \cdot (k+1), p)$ and as a result $P\{X_1 + \cdots + X_i = x + i\}$, i.e., the probability of any combination that sums up to $x + i$, is ease to compute. Note

that this latter probability is a good starting point for the computation of $P_x^{(i)}$. However, we need to eliminate from this calculation the probability of any combination of $X_1, X_2, \cdots, X_i$ that results in a decoding in a previous round j. Since a decoding in round $j < i$ requires that $X_1 + \cdots + X_j = j$, this means that $X_{j+1} + \cdots + X_i = x + i - j$. The probability of such a combination happening is clearly $P_i \cdot P\{X_{j+1} + \cdots + X_i = x + i - j\}$ where again $P\{X_{j+1} + \cdots + X_i = x + i - j\}$ is easy to calculate using the Binomial distribution. As a result, $P_x^{(i)}$ can be written as:

$$\begin{aligned} P_x^{(i)} =& P\{X_1 + \ldots + X_i = x\} - \sum_{j=2}^{i-1} P_j \cdot P\{\sum_{l=j+1}^{i} X_l = x+i-j\} \\ & - P\{X_1 = 0\} \cdot P\{X_2 + \ldots + X_i = x+i\} - P\{X_1 = 1\} \cdot P\{X_2 + \ldots + X_i = x+i-1\} \end{aligned} \tag{4.8}$$

In the previous equation we also take into account the special case where in round 1 there is no lost packet ($X_1 = 0$), so decoding is not required. Note that (4.8) can be used recursively to compute $P_x^{(i)}$ and therefore $\widetilde{P}_i$ through (4.5).

Finally, we can compute the average number of packets lost during a decoding failure, $\widetilde{L}$. As we mentioned previously, $W_i = x$ means that $X_1 + X_2 + \ldots + X_i = i + x$,i.e., the total number of lost packets is $i + x$. However, only a percentage of those packets are native. More specifically, it is expected that, on average, $\frac{k}{k+1}(x + i)$ native packets will be lost. Based on this, we can now calculate $\tilde{L}$ as:

$$\widetilde{L} = \sum_{x=1}^{i \cdot k} P\{W_i = x \mid W_j > 0,\ \forall\, j < i\} \cdot \frac{k}{k+1} \cdot (x + i) \tag{4.9}$$

By setting a requirement for $\widetilde{L}$ (e.g., such as 10^{-6} described in URLLC), one can obtain i, i.e., the required coding depth.

4.3 An Efficient Algorithm for the Dynamic Adaptation of d

Up to now we assumed that a static value of d can provide the desirable protection from channel losses. However, this is true only if the channel's packet loss rate is rather static because a static value of d results in a fixed coding performance. If the loss rate varies then there may exist time periods when the introduced redundancy may not be useful, i.e., transmitted coded packets do not assist in compensating packet losses. Similarly, there may exist periods of time that the provided protection is not sufficient to cope with bursts of errors. Given the fact that bursty channels are

Table 4.1: Adaptive d: Symbol Definitions.

Parameter	**Definition**
P	Set containing the source packets *positively* acknowledged by a cumulative ACK during a loss estimation period.
N	Set containing the source packets *negatively* acknowledged by a cumulative ACK during a loss estimation period.
sle	The total number of packets that should be positively or negatively acknowledged before activating the adaptation of d.
$previouslr$	Loss estimation ratio calculated in the previous period.
pcd	Number of periods before decreasing d.
d_{max}	Maximum d for achieving a specific packet loss ratio, derived from theoretical analysis.
RTX	Per-packet re-transmission limit.

common, it is clear that the coding protocol should be able to realize the emergence of such idle or burst time periods and appropriately adjust its coding window to reduce/increase its coding efficiency. Clearly, this can be achieved by adjusting the coding depth d. In the effort to strike a balance between the coding complexity and the protocol's performance, the dynamic adaptation of d would be a promising approach. Our aim is to offer the essential level of protection through coding and at the same time minimize the coding complexity by taking advantage of the idle periods.

In practice, the main idea behind the adaptation of d relies on the exploitation of rapidARQ's cumulative acknowledgements [23, 24], so that we can keep track of the total number of source packets successfully delivered to the receiver. Apart from the main notations regarding $\mathbb{S}_w$, $\mathbb{C}_w$ and their sizes (see glossary), the definition of additional parameters used in the following adaptive algorithm is provided in Table 4.1. Coding depth is upper bounded by the rough estimation of optimal d derived from equation 4.3 (see section 4.1). This estimation is given as input to the algorithm. We distinguish three main components regarding the efficiency of our adaptive algorithm which will be discussed in the following sections.

Algorithm 4.1 Loss Ratio Estimation

Require: P, N, sle

1: $s \leftarrow |P| + |N|$
2: **if** $s \geq sle$ **then**
3: $lr \leftarrow |N|/s$
4: **Adapt(**lr**)**
5: $previouslr \leftarrow lr$
6: $P.clear()$
7: $N.clear()$
8: **end if**

4.3.1 Loss Ratio Estimation

In order to decide whether an adjustment to the $\mathbb{C}_w$ is essential, the transmitter should track the channel behavior, i.e., the erasure probability, its variations and the burst pattern. This can be accomplished using the cumulative acknowledgements sent by the receiver. This feedback incorporates both positive acknowledgements for successfully delivered source packets and negative ones for the lost packets that cannot be decoded yet. Apart from the cumulative acknowledgements, a timeout event at the transmitter side indicates the occurrence of a loss. This event is useful in case feedback has not provided updated information for a particular source packet. A critical parameter is the total number of acknowledgements required from the transmitter in order to activate the process of loss ratio estimation. This is illustrated through variable sle and remains invariant throughput the data transmission. A typical yet efficient approach sets sle equal to the sliding window size W, since a cumulative acknowledgement already contains information for that range of packets.

Algorithm 4.1 describes in detail the loss ratio estimation process activated either by the reception of a new cumulative acknowledgement or a timeout occurrence. These events are responsible for populating the sets P and N given as input in the loss ratio estimation procedure. P describes the collection of positively acknowledged source packets, while N contains the negatively acknowledged ones. Of course the insertion of a new positively acknowledged packet excludes any obsolete information regarding the loss of the particular packet, i.e., a source packet cannot be included in both sets concurrently. If the transmitter has collected feedback for at least sle unique source packets (line 2), it calculates the current loss as the ratio of the number of

estimated lost packets to the total number of packets acknowledged so far (line 3). The calculated loss ratio is passed as an argument to function *Adapt*, which implements the actual adjustment of the coding depth (see Algorithm 4.2). The calculation of the loss ratio and the adaptation of d signify the end of a loss estimation period and hence we should update the value of variable $previouslr$, which will be used as a comparison to the subsequent loss estimation, as well as clear the contents of sets P and N (resp. to lines 5- 7).

4.3.2 Triggering Change in d

Two major parameters should be taken into account when discussing the dynamic adaptation of coding depth; that is, the *moment* and the *level* of change of d. The former one denotes at which point in time a modification in d is essential whereas the second parameter refers to how much we should alter d in order to adapt to the varying channel conditions. The adaptive algorithm should follow the principle of *fast-increase-slow-decrease*, which signifies an instant reaction in case of increased loss estimations in contrast to a moderate approach in shortening coding depth when the impact of losses seems to decline. In this way, we avoid dropping unackowledged packets that are still essential for the decoding process. Nevertheless, we should keep in mind that re-transmissions also affect the coding window size. In particular, under the presence of re-transmissions ($RTX > 0$), the actual size of $\mathbb{C}_w$ may be smaller than CW. Hence, the process of dynamically adaptating d should conform to the prevailing channel conditions. On the contrary, in a re-transmission-free scheme, $\mathbb{C}_w$ constantly attains the maximum coding window size. This is because the sender does not wait to receive any feedback for the transmitted packets; instead, it drops the oldest ones whenever a new source packet arrives. Therefore, it is critical for the coding scheme to ensure that all source packets receive the required protection. Otherwise, the risk of dropping source packets that are still necessary for the decoding process looms. To that end, it is essential that we specify additional safety nets apart from the triggering conditions to reduce that risk.

In algorithm 4.2, lines 2, 5, 14 and 18- 20 depict the conditions which activate the adaptation of d. In particular, we choose to enhance protection offered by coding depth every time the loss estimation is greater than the one in the previous period (lines 2 and 14). In this way, our algorithm reacts immediately to the effect of loss

variations. On the contrary, this is not the case when decreasing d. While one would consider that a complementary-to-the increase strategy would be a reasonable solution (i.e., decreasing d whenever $lr < previouslr$), the performance obtained by adopting this strategy is instead quite poor. Instant decrease acts as a counter-measure to any attempts of increasing d, as it results in needless fluctuations of the coding window size, hence it ends up canceling the desired enhanced protection and amplifying performance issues, e.g., increased delay. This happens due to the sensitivity even in small variations of loss and the fact that loss estimation is not always accurate. Therefore, a slightly smaller new estimation would trigger a reduction in d, although this modification is not essential. A possible solution to surpass the aforementioned issue would be the introduction of a *threshold* value to enhance tolerance against small variations between two successive loss estimation periods. However, the appropriate choice of such a threshold depends on various parameters concerning the channel conditions, hence we consider it as a future extension of this work.

To tackle any unnecessary reductions in d, we incorporate the following strategy (lines 5, 18- 20). Instead of taking into account only the smaller new loss estimations ($lr < previouslr$), we regard any invariable zero estimations ($lr == previouslr == 0$) as evidence of potential decrease in d (lines 5, 18). The absence of losses indicates that the level of protection provided by the coding process is more than adequate and therefore we could mitigate the redundancy in aid of improved throughput performance. Nevertheless, before proceeding to any alteration, we should specify an additional constraint regarding re-transmission-free schemes. Since $|\mathbb{C}_w| = CW$ is typically true in such scheme, we should take precautions to prevent dropping useful source packets from the coding window. Hence, instead of immediately modifying d every time the condition in line 18 is true, we monitor the loss variations for a number of periods, and advance to the reduction of d after noticing pcd successive reductions or zero loss estimations (line 19) In this way, we implicitly presume that the protocol enters an idle period, hence the risk of removing useful source packets from the coding window is smaller. In practice, the algorithm performs well under any channel conditions assuming $pcd = CW/2$.

4.3.3 The Actual Adaptation of Coding Depth

In addition to the triggering conditions, the actual adjustment of coding depth and, by extension, the coding window, is of great significance. We implement two modification policies to tackle the different requirements of the scheme depending on the presence of re-transmissions. While the increase of d seems trivial, we should lay more emphasis on the decrease process, since it is critical to prevent dropping packets that are still necessary for the coding procedure. More specifically, lines 3 and 4 illustrate the updates regarding the increase in d when the protocol incorporates re-transmissions. As we can observe, an increment of 1 takes place whenever the algorithm opts for increasing d (line 3). An upper bound for d is given through the theoretical analysis provided in section 4.2. Subsequently, we update the maximum CW, which can run up to W, i.e., the size of the sliding window (line 4). As for the reduction of d, we monitor the difference in size between the maximum coding window size CW and the actual contents of the $\mathbb{C}_w$ (line 6). We downscale d by normalizing the aforementioned difference with the number of source packets transmitted between two successive coded ones, i.e., k (line 3). This strategy allows us to reduce the size of d and, hence CW, by conforming to the current channel conditions.

On the other hand, we choose to keep the adjustment steps simple, yet efficient, when re-transmissions are not used. Lines 16 and 17 describe the occurring updates when we increase d. It is evident that we follow the same approach here as in case of $RTX > 0$. Nevertheless, the major difference lies in the decrement of d (lines 24- 28). First, we reduce the size of coding depth by 1 following by the assignment of the updated value to CW (lines 25 and 26 respectively). The minimum value of coding depth is equal to 1. An additional step is required in this case regarding the update of the $\mathbb{C}_w$ contents. In particular, if the current size of $\mathbb{C}_w$ exceeds the new CW, we remove the $|\mathbb{C}_w| - CW$ oldest source packets from $\mathbb{C}_w$ to ensure consistency (line 28).

Algorithm 4.2 Adapt(lr)

Require: $previouslr, pcd, d_{max}, W, CW, \mathbb{C}_w, d, RTX, k$

1: **if** $RTX > 0$ **then**
2: **if** $lr > previouslr$ **then**
3: **if** $d + 1 < d_{max}$ **then** $d \leftarrow d + 1$ **else** $d \leftarrow d_{max}$
4: **if** $d \times k \geq W$ **then** $CW \leftarrow W$ **else** $CW \leftarrow d \times k$
5: **else if** $(lr < previouslr)$ **or** $(lr == 0$ **and** $previouslr == 0)$ **then**
6: $diff = CW - |\mathbb{C}_w|$
7: **if** $diff > 0$ **then**
8: $depth\_decr = diff \;/\; k$
9: **if** $d - depth\_decr > 1$ **then** $d \leftarrow d - depth\_decr$ **else** $d \leftarrow 1$
10: $CW \leftarrow d \times k$
11: **end if**
12: **end if**
13: **else**
14: **if** $lr > previouslr$ **then**
15: $pcd \leftarrow 0$
16: **if** $d + 1 < d_{max}$ **then** $d \leftarrow d + 1$ **else** $d \leftarrow d_{max}$
17: **if** $d \times k \geq W$ **then** $CW \leftarrow W$ **else** $CW \leftarrow d \times k$
18: **else if** $(lr < previouslr)$ **or** $(lr == 0$ **and** $previouslr == 0)$ **then**
19: $pcd \leftarrow pcd + 1$
20: **if** $pcd < CW/2$ **then**
21: **return**
22: **end if**
23: $pcd \leftarrow 0$
24: **if** $d > 1$ **then**
25: $d \leftarrow d - 1$
26: $CW \leftarrow d \times k$
27: **if** $|\mathbb{C}_w| > CW$ **then**
28: remove $|\mathbb{C}_w| - CW$ oldest packets from $\mathbb{C}_w$
29: **end if**
30: **end if**
31: **else**
32: $pcd \leftarrow 0$
33: **end if**
34: **end if**

Chapter 5

Experimental Evaluation

5.1 Simulation Setup

In this chapter, we evaluate the performance of the proposed adaptive algorithm using ns2 simulator [53]. Since the dynamic adaptation of coding depth is implemented on top of rapidARQ [23, 24], we refer to the adaptive scheme as *Adaptive rapidARQ* from here on. We provide a performance comparison of adaptive rapidARQ with multiple sliding window reliability mechanisms. In particular, we consider Caterpillar RLNC-FB [11] as a prominent ARQ-based systematic SW RLNC protocol, Tetrys [30] as the most representative FEC-based SW RLNC scheme and Selective Repeat ARQ (SR-ARQ) [54]. Of course, we could not overlook the performance comparison of adaptive rapidARQ with its static counterpart, which we symbolize as *rapidARQ* henceforward.

We exploit the same evaluation scenarios described in section 3.3, i.e., a high-speed link to a low-earth orbiting satellite, which is a suitable scenario for 6G and beyond networks, and a typical 5G wireless link. The main simulation parameters are summarized in table 3.1. The choice of code rate R and coding depth d follows

the same principles explained in section 3.3. That is, R is set accordingly so that, on average, the introduced redundancy can compensate a single packet loss every $k+1$ packets while the maximum coding depth value d_{max} ranges in $[1, \lfloor \frac{W}{k} \rfloor]$. We examine the performance of the aforementioned protocols by allowing at most $RTX = 3$ re-transmissions in ARQ-based sliding window schemes. For a fair comparison with FEC-based schemes, i.e., Tetrys, we perform additional comparisons without the presence of re-transmissions ($RTX = 0$). As for the channel's error model, we consider both a uniform and a bursty model. If burstiness is not introduced in the discussion, we consider by default a uniform error model. To ensure that a constant flow of traffic is always available, we use constant bit rate (CBR) traffic generators in both simulation scenarios. For each experiment, we carry out a set of 5 trials considering a sufficiently large simulation time of $50s$ and report the average values.

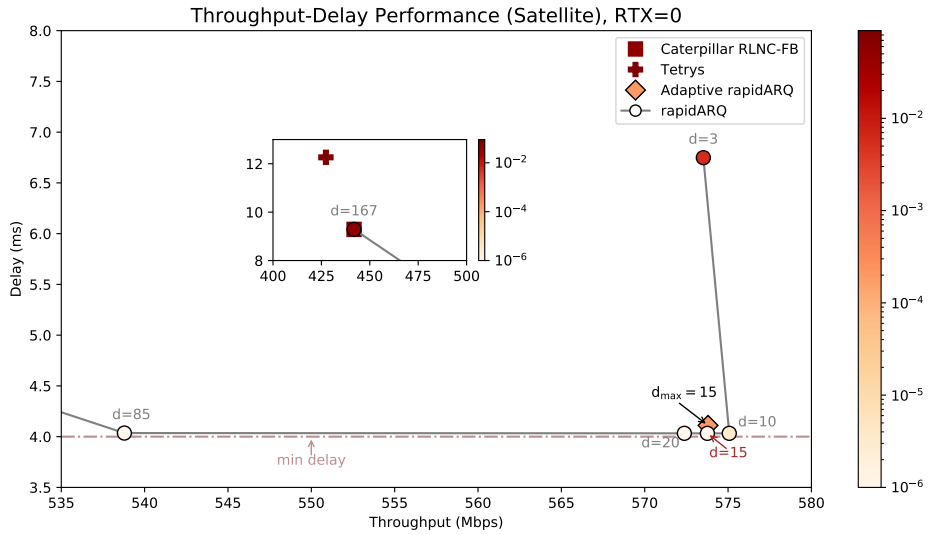
We are interested in the in-order packet delivery at the receiver. To that end, we make use of the following evaluation metrics:

- *Throughput*: The average bit-rate recorded at the receiver side.
- *Average Delay*: The average in-order source packet delay, defined as the time elapsed from the moment when the source packet becomes available at the sender for transmission to the instant when it is successfully delivered in-order to the receiver's upper layer.
- *Average Decoding Matrix Size*: The average number of linearly independent packets in the decoding matrix whenever a decoding operation is performed.
- *Packet Drop Rate*: The percentage of source packets that could not be successfully received or decoded even after re-transmissions.

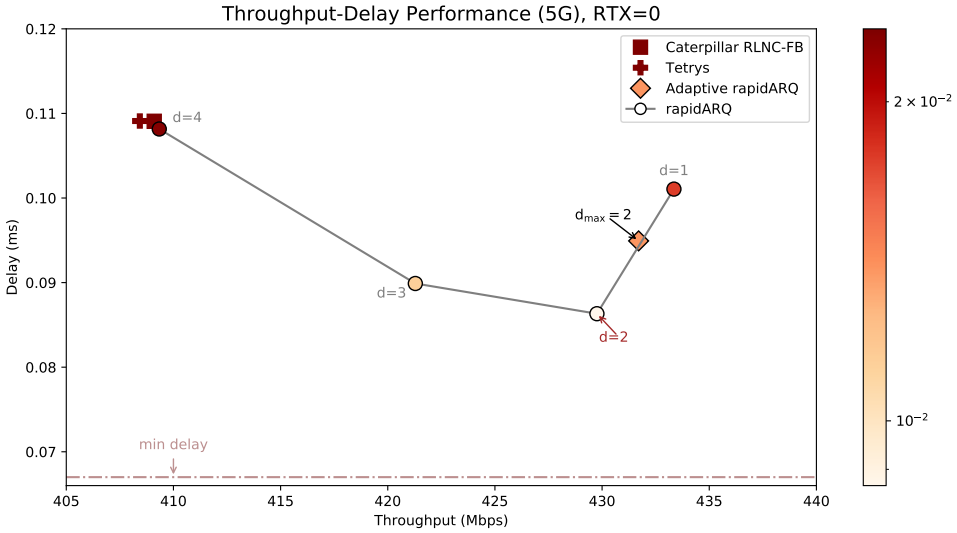
5.2 Performance Evaluation in Diverse Scenarios

5.2.1 Throughput - Average Delay Performance

In the first experiment we assess the throughput-delay performance of all protocols in the two scenarios, i.e., (a) the satellite link and (b) the 5G wireless link, with or without the utilization of re-transmissions. We assume a relatively stable transmission link, where packet losses are uniformly distributed in time. Figure 5.1 illustrates the



(a)



(b)

Figure 5.1: Throughput-Delay Performance of Adaptive and static rapidARQ, Caterpillar-FB and Tetrys without re-transmissions ($RTX = 0$) in (a) the satellite scenario, and (b) the 5G wireless scenario. The color in this figure indicates the packet drop rate of each protocol.

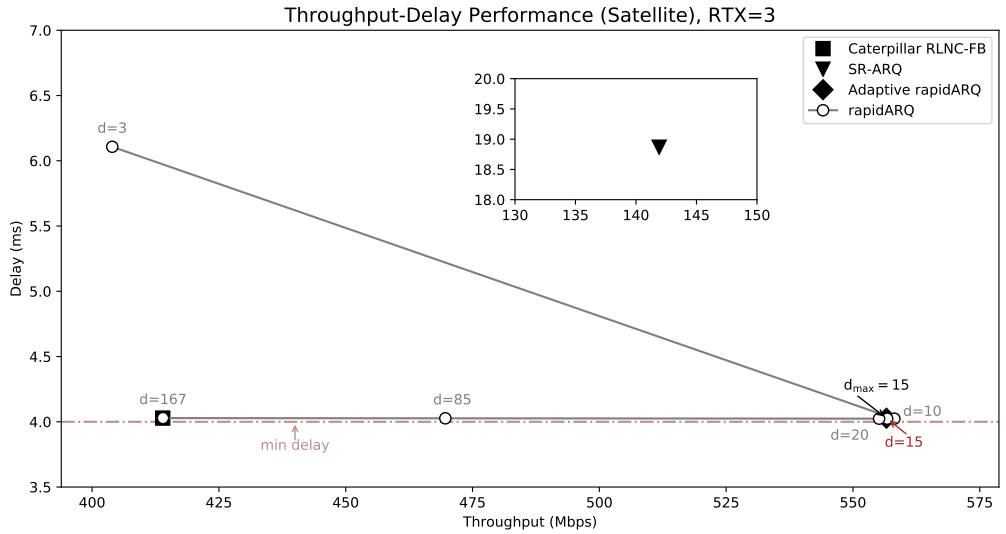
throughput-delay performance of all re-transmission-free mechanisms ($RTX = 0$) considered in this experiment, that is Caterpillar RLNC-FB, Tetrys, rapidARQ and the proposed adaptive rapidARQ. Figure 5.1a refers to the satellite scenario while

Figure 5.1b to the 5G link. We display the performance of rapidARQ under multiple coding depth values, hence it is plotted as a line, whereas the rest of the protocols are illustrated as single points. We leverage the analytical framework explained in chapter 4 and, based on equation 4.3, we estimate the appropriate d_{opt} for each test scenario given that the coded scheme should be able to achieve a drop rate in the order of 10^{-6}. The near optimal estimation for the satellite scenario is $d_{opt} = 15$, whereas for the 5G link $d_{opt} = 2$. We conventionally describe d_{opt} as d_{max} in adaptive rapidARQ, so that we maintain consistency with the symbol definition in table 4.1. Unless stated differently, these estimations will be used throughout this chapter to evaluate the performance of adaptive rapidARQ. We also denote the performance of rapidARQ when these rough estimations are assigned to d as a reference for the resulting performance-complexity trade-off. Since coding is the only utilized practice for counteracting channel's loss rate, we present the packet drop rate performance of each protocol with color (the darker the color, the larger the drop rate is).

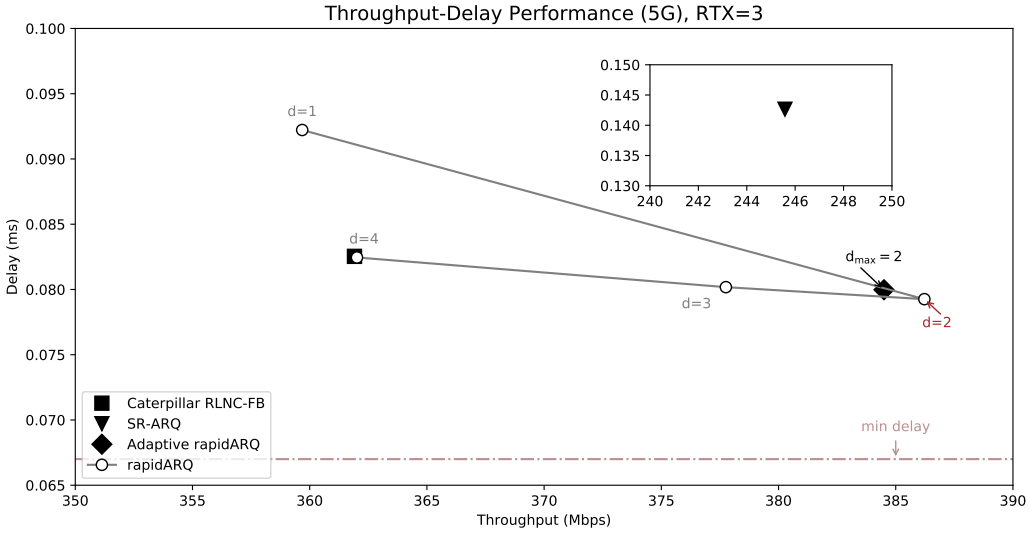
As expected, both variants of rapidARQ massively outweigh Caterpillar RLNC-FB and Tetrys protocol. This is due to the appropriate definition of the coding window size according to the channel's loss profile. The performance gains are more salient in links with larger bandwidth-delay products, i.e., the satellite link, since, in the latter, d can be chosen among a wider range of possible values, hence its optimal choice is crucial. While we notice a significant improvement in throughput, the most notable gains pertain to average delay, where the reduction in the satellite scenario runs to nearly 55% compared to Caterpillar and 66% against Tetrys, and packet drop rate (up to two orders of magnitude reduction in the 5G case).

Regarding rapidARQ, both the adaptive and the static scheme perform near optimally by utilizing the theoretical estimation ($d = 15$ in the satellite scenario, $d = 2$ in 5G). This proves that the estimated threshold value for d can adequately provide the required protection without compromising the overall performance. Meanwhile, it is quite impressive that we can derive a near optimal estimation of d without an exhaustive experimental search of the optimal value. While the effort of the experimental assessment would not be critical for channels with relatively small bandwidth-delay product, e.g., the 5G link, this is not true for channels with large bandwidth-delay products, such as the assumed satellite link, since a wide range of values can be assigned to d. In the satellite scenario, it is equally noteworthy that the proposed analytical methodology converges to a relatively small near optimal d, hence the cod-

ing complexity remains low as well. In the 5G scenario, the small bandwidth-delay product substantially restrains the available options for d. However, the proposed methodology successfully identifies the best d value even under such restrictions.



(a)



(b)

Figure 5.2: Throughput-Delay Performance of Adaptive and static rapidARQ, Caterpillar-FB and SR-ARQ with re-transmissions ($RTX = 3$) in (a) the satellite scenario, and (b) the 5G wireless scenario.

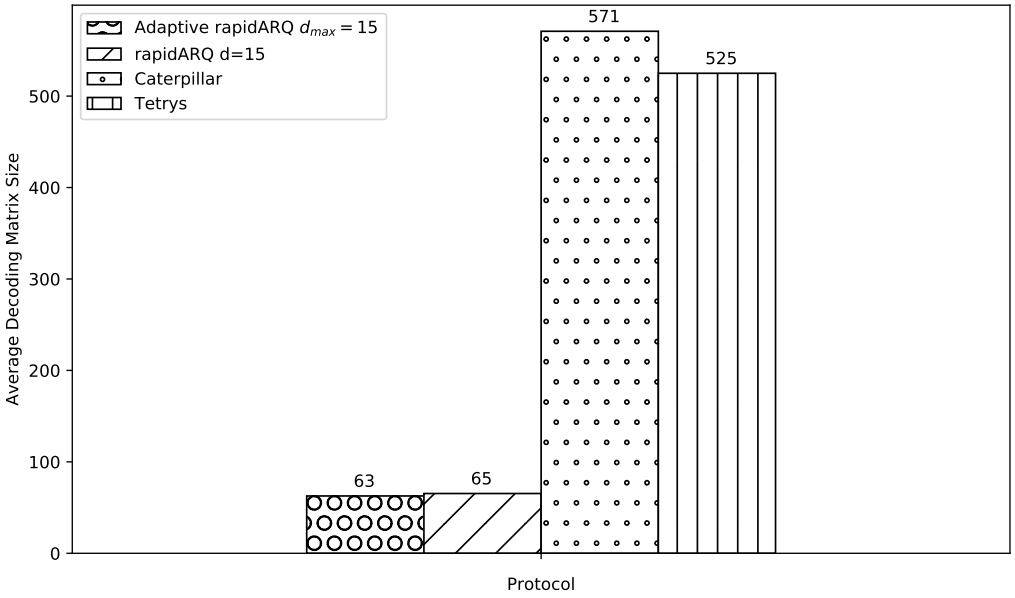
Compared to its static counterpart, adaptive rapidARQ's performance is compa-

rable to the near-optimal one, although the channel loss rate is rather static and, therefore, no significant improvements are expected. More specifically, adaptive rapidARQ performs nearly as efficiently as the static protocol regarding throughput and delay with a single exception in the 5G link, where we can observe a slight increase in the average delay. This can be attributed to the dynamic adaptation of the coding window size, which delays the convergence to the desired operating point. However, the adaptive scheme may affect the smoothness of the coding process due to the constant updates in d and, as a consequence, the packet drop rate is increased compared to the near optimal performance. Still, the packet drop rate difference between the static and the adaptive scheme is very small (e.g., in the order of 10^{-4}).

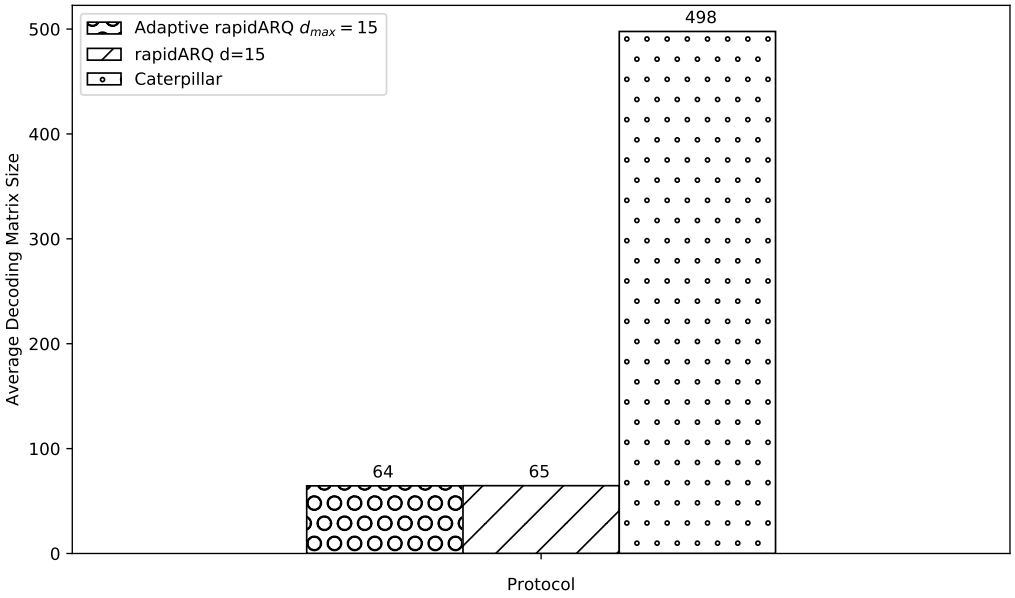
When re-transmissions are allowed (figure 5.2), similar observations can be made regarding the throughput-delay performance of the examined protocols. Here, apart from the network-coded schemes, we also evaluate the performance of SR-ARQ. It is evident that both Caterpillar and rapidARQ variants enormously outperform SR-ARQ in both test scenarios. In fact, rapidARQ achieves more impressive performance gains in the satellite scenario, where it provides a vast net increase in throughput of up to 415 Mbps while simultaneously decreasing delay up to 79%. As for Caterpillar, although its delay performance is equivalent to the one of rapidARQ, it is considerably inferior to rapidARQ regarding throughput, especially in the satellite scenario, where we can observe a throughput difference of $\sim$ 150 Mbps ($\sim$ 30%). As for rapidARQ, while the theoretical framework does not take into account the effect of re-transmissions, the exploitation of the proposed methodology can benefit the overall performance even under the presence of re-transmissions. By utilizing the calculated estimation of d, we can minimize the probability of packet losses, by avoiding decoding failures. Therefore, minimum loss probability entails the minimization of re-transmissions, which induce prohibitive delay overhead in the context of URLLC. The aforementioned arguments are valid even for the adaptive scheme, which performs close to optimal under re-transmissions as well.

5.2.2 Investigating the Average Decoding Matrix Size

A critical aspect on the coding scheme's efficiency is the computational cost induced by the coding operations. In [24] we proved that rapidARQ has indeed lower decoding complexity than the other SW RLNC schemes. Figures 5.3 and 5.4 illustrate the

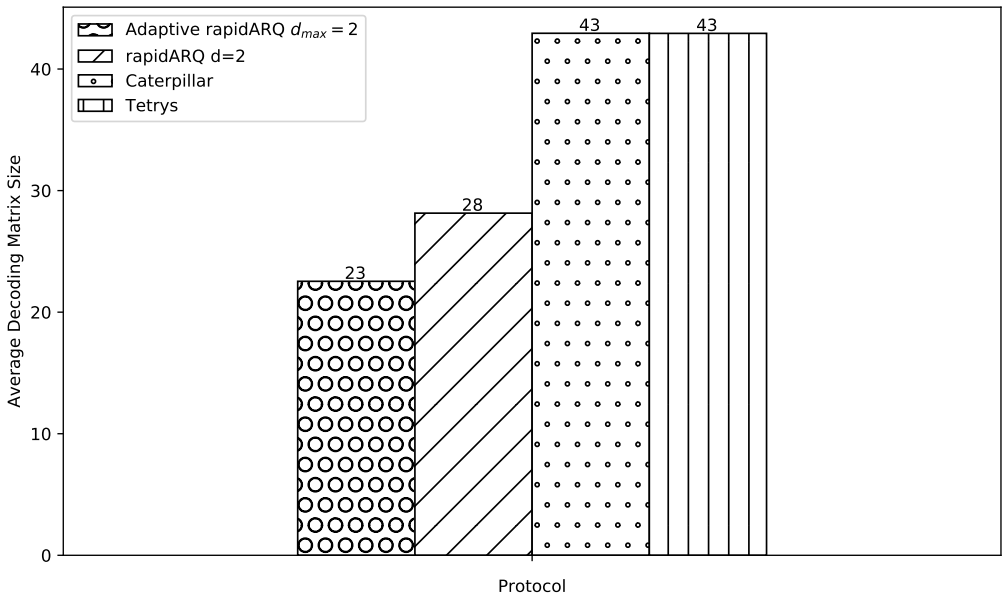


(a)

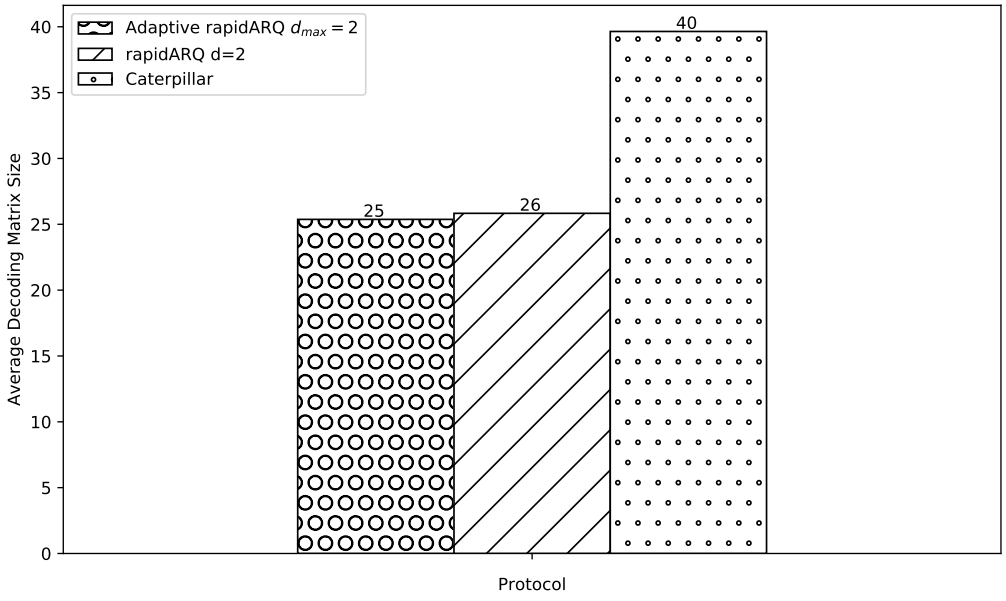


(b)

Figure 5.3: Average Decoding Matrix Size of network-coded protocols in satellite scenario (a) without re-transmissions ($RTX = 0$) and (b) with re-transmissions ($RTX = 3$).



(a)



(b)

Figure 5.4: Average Decoding Matrix Size of network-coded protocols in 5G wireless scenario (a) without re-transmissions ($RTX = 0$) and (b) with re-transmissions ($RTX = 3$).

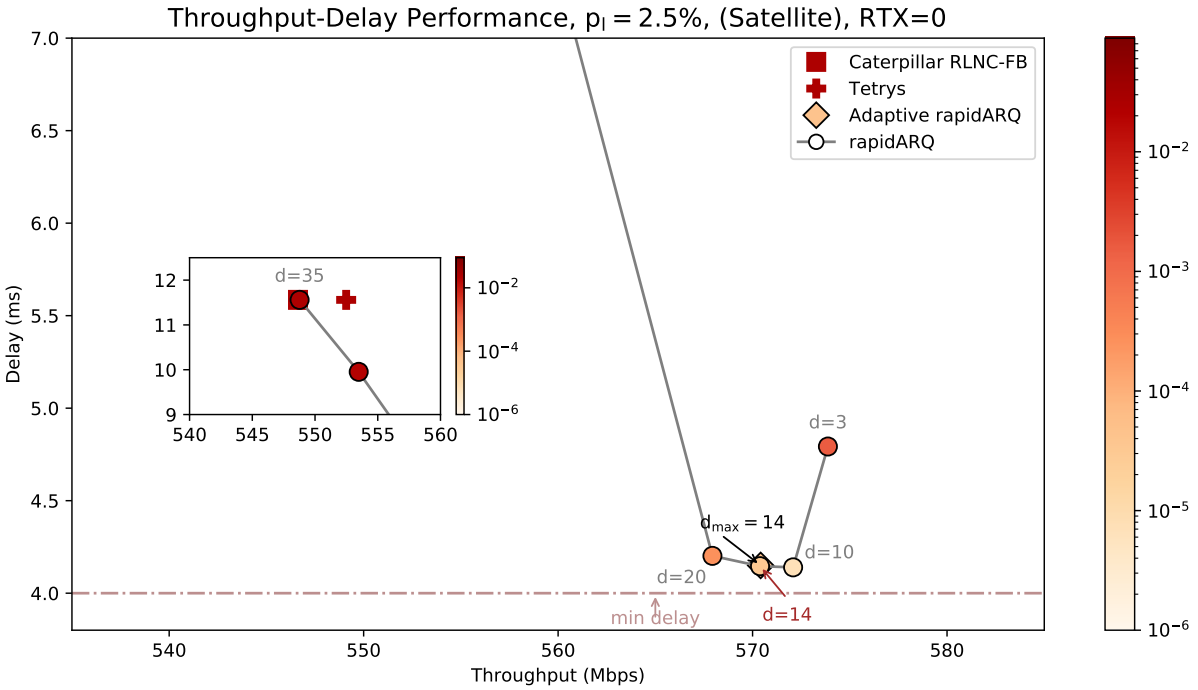
average decoding matrix size of the coded schemes in the default satellite and 5G scenarios, respectively. Again, we examine the performance of the protocols both with and without the presence of re-transmissions. To avoid cluttering, we choose to display only the performance of $d = 15$ regarding rapidARQ. We refer any interested readers to [23, 24] for further experimental results on the impact of d on the average decoding matrix size. Previously, we mentioned that rapidARQ achieves significant performance gains compared to Caterpillar and Tetrys. These gains are accompanied by a huge decrease in the decoding matrix size, as reflected in the experimental results, where we can clearly perceive an order of magnitude reduction. The benefits are more obvious in the satellite scenario (figure 5.3).

As for adaptive rapidARQ, the proposed scheme aims to further improve coding complexity by adjusting the coding window size according to the current channel conditions. However, due to the uniform distribution of losses, the performance gains, as expected, are minimal. Even so, the resulting complexity of the adaptive scheme is very close to the minimum complexity achieved by the estimated near optimal d when applied in static rapidARQ. In fact, we can note further reduction in the decoding matrix size, which is typically rather small. For instance, in the satellite case (figure 5.3), the recorded improvement compared to rapidARQ is negligible (63 vs 65 when $RTX = 0$ and 64 vs 65 when $RTX = 3$), whereas in the 5G scenario, we can discern a larger improvement of $\sim 18\%$ when no re-transmissions are allowed ($RTX = 0$). However, as previously illustrated, this comes at the expense of slightly poorer throughput, delay and loss performance (see figure 5.2b), signifying that the fluctuations in d have affected the overall performance. Based on the obtained results, we can conclude that adaptive rapidARQ achieves comparable throughput, delay and loss performance with its static counterpart in relatively stable channel conditions. The achieved small reduction in the decoding matrix size is a consequence of the absence of bursts of errors. We will show in the following sections that under such bursts, the adaptive scheme performs more efficiently.

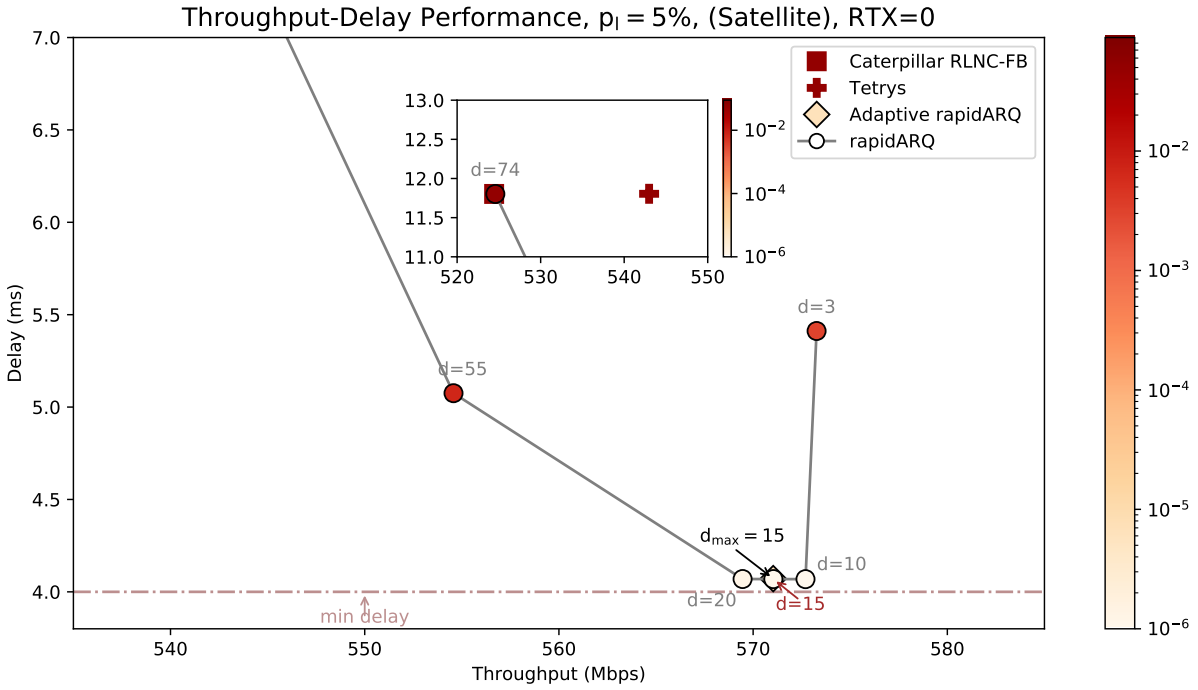
5.3 Performance under Various Channel Error Rates

Another interesting experiment refers to the performance evaluation of rapidARQ under various channel loss rates. Here, we aim to assess the suitability of the analytical

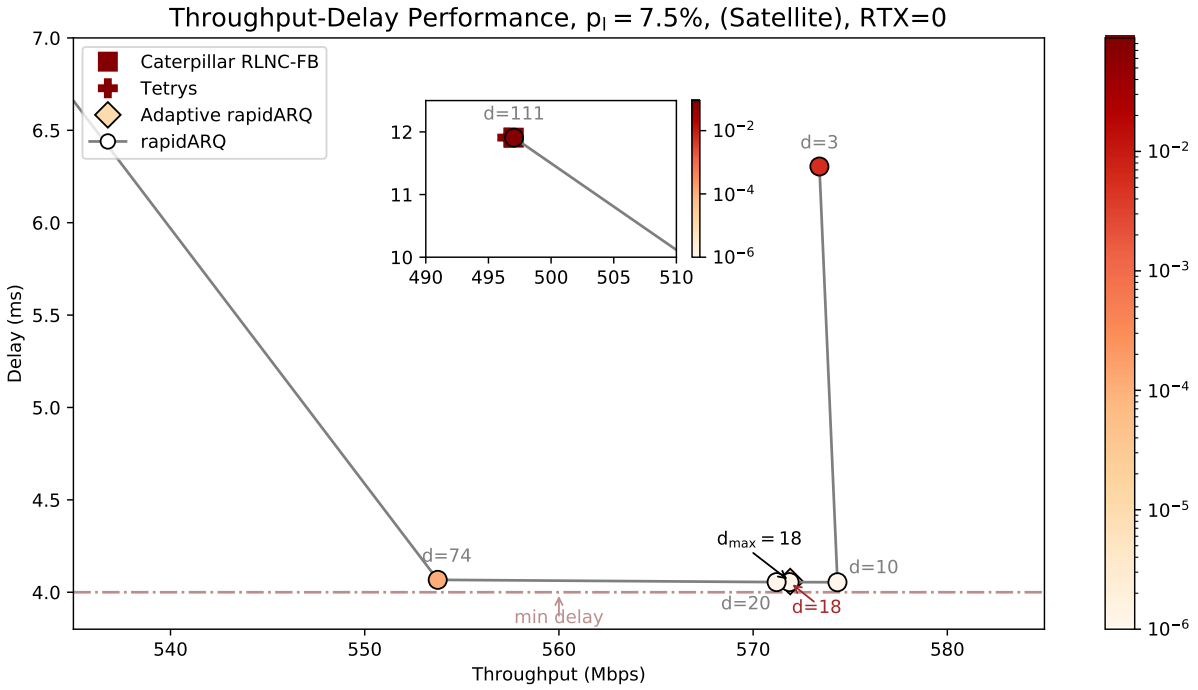
framework in deriving efficient rough estimations for different loss rates. Note that the distribution of errors remains uniform; only the level of occurring errors changes. In particular, we examine the throughput-delay performance when the channel loss rate p_l takes one of the following values: $2.5\%, 5\%, 7.5\%$. Since p_l varies, we should



(a)



(b)



(c)

Figure 5.5: Throughput-Delay Performance of Adaptive and static rapidARQ, Caterpillar-FB and Tetrys without re-transmissions ($RTX = 0$) in the satellite scenario for varying channel error rate, (a) $p_l = 2.5\%$, (b) $p_l = 5\%$ and (c) $p_l = 7.5\%$. The color in this figure indicates the packet drop rate of each protocol.

adjust the code rate R so that, on average, the coding redundancy is sufficient for coping with the average number of expected packet losses. Based on the updated channel conditions and without modifying the prescribed drop rate target, i.e., 10^{-6}, it is essential that we reexamine the estimation of the optimal d_{opt} value, based on the framework described in section 4.2. Table 5.1 contains the updated simulation parameters accordingly adjusted to the varying channel conditions using the aforementioned framework.

Table 5.1: Coding parameters used under variable channel loss rates.

Link loss rate (p_i)	Code Rate R	d_{opt}
2.5%	19/20	14
5%	9/10	15
7.5%	6/7	18

Figure 5.5 illustrates the throughput-delay performance of the examined protocols for different link loss rates. For simplicity, we choose to investigate the protocols' performance only in the satellite scenario and without employing re-transmissions. This is because a re-transmission-free scenario portrays the maximum potential of the coding scheme, whereas the satellite scenario offers more useful lessons regarding the impact of coding depth on rapidARQ's performance (d may take a wide range of values compared to the 5G scenario). Again, rapidARQ variants significantly outperform Caterpillar and Tetrys. Furthermore, it is impressive that both rapidARQ variants perform close to optimal when d is determined by the theoretical estimation. We can also observe that a larger p_l results in a larger range of values for d, which can achieve near optimal performance. Nevertheless, through the analytical framework, we obtain an estimation of d_{opt} close to the lower limit of this range of values. Once again, the adaptive scheme works quite well, even though the uniformly distributed error model does not point out its contribution.

5.4 Resilience to Bursts of Errors

So far, we have only examined the protocol's performance in channels with relatively stable loss rate. However, many real-world channels exhibit a bursty nature. In such channels, there may exist idle time periods, i.e., periods of time that errors are absent, where the introduced redundancy may not be useful. Similarly, there may exist time periods where an increased level of protection is required to deal with bursts of errors. In view of the fact that channels with loss variations are quite common, it is essential that the coding protocol be able to realize the emergence of such idle or burst time periods and appropriately adjust its coding window and, hence, its coding efficiency. Therefore, it is necessary to investigate how the adaptive scheme handles bursts of errors. To that end, we introduce a bursty error profile using an *on-off* model. Assume that a channel with a uniform error model has an error rate p_l. In the bursty model, the channel error rate p_l' for the "on" period is set accordingly so that during a complete on-off cycle, the error rate is on average equal to p_l, while no errors occur during the "off" period. Supposing the duration of "on" and "off" periods is expressed with T_{on} and T_{off}, respectively, we define *burstiness* B as $B = \frac{(\bar{T}_{on}+\bar{T}_{off})}{\bar{T}_{on}}$, where $\bar{T}_{on}$ and $\bar{T}_{off}$ express the average values of T_{on} and T_{off}. Therefore, the packet

Table 5.2: Performance difference of Adaptive rapidARQ ($d_{max} = 15$) and rapidARQ ($d = 15$) under variable channel conditions in the satellite scenario.

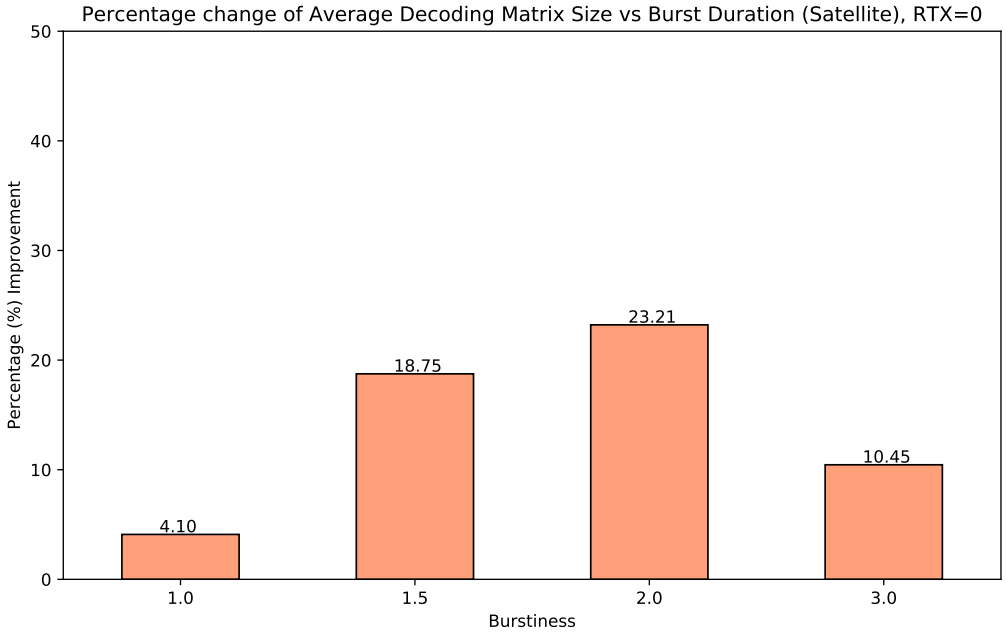
	T_{diff} (Mbps)		D_{diff} (ms)		L_{diff}	
B	$RTX = 0$	$RTX = 3$	$RTX = 0$	$RTX = 3$	$RTX = 0$	$RTX = 3$
1	0.0389748	-0.1467068	0.0784574	0.0036118	0.0002024	0.0000006
1.5	-0.9113904	0.0364908	1.5515418	0.0035814	0.0032904	0.0000018
2	-0.1337934	-1.6107328	1.0156504	0.034613	0.0019448	0.0000066
3	0.8316014	2.220455	0.0048696	1.1064776	-0.0006404	0.0009002

Table 5.3: Performance difference of Adaptive rapidARQ ($d_{max} = 2$) and rapidARQ ($d = 2$) under variable channel conditions in the 5G wireless scenario.

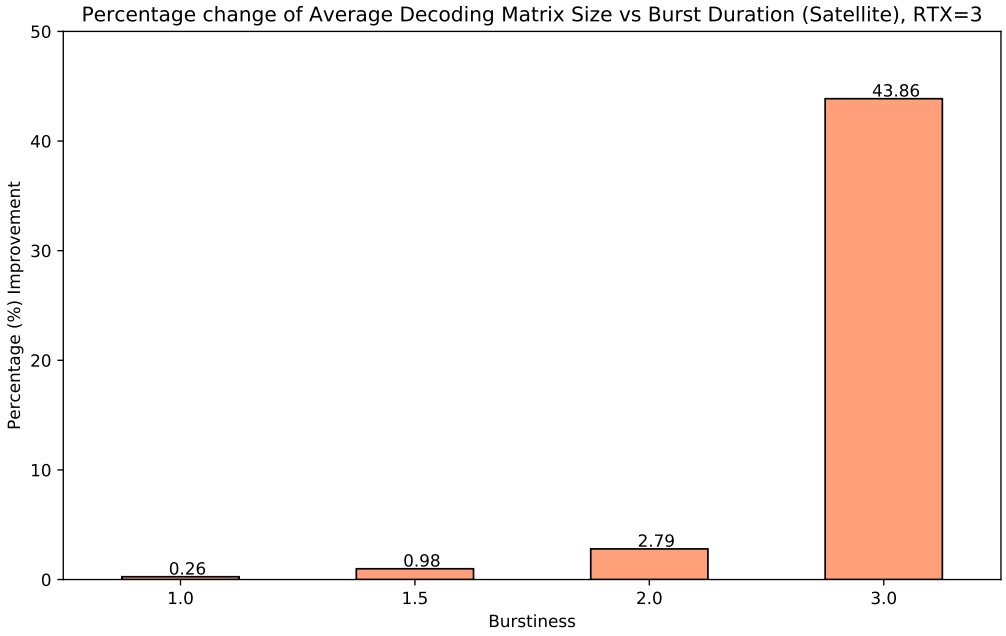
	T_{diff} (Mbps)		D_{diff} (ms)		L_{diff}	
B	$RTX = 0$	$RTX = 3$	$RTX = 0$	$RTX = 3$	$RTX = 0$	$RTX = 3$
1	1.9448608	-1.722714	0.0086084	0.0007228	0.0053012	0.0000154
1.5	1.4134774	-2.4772956	0.0093132	0.0011138	0.0060306	0.0000366
2	1.053732	-3.1950436	0.0095582	0.001507	0.006386	0.0000564
3	1.2508618	-4.7282036	0.0084766	0.00263	0.005376	0.0001602

error rate observed during an "on" period is equal to $p_l' = B \cdot p_l$. When $\bar{T}_{off} = 0$, the error model degenerates into the uniform error model, i.e., $B = 1$. On the other hand, larger B values indicate that $\bar{T}_{on}$ is very small and p_l' very high.

We are interested in the comparison of adaptive and static rapidARQ regarding not only performance metrics, i.e., throughput, delay and loss, but also the complexity improvements. This multi-faceted comparison is crucial in order to determine the efficacy of the proposed adaptive scheme under variable burstiness. Once again, we assess the performance of the examined protocols in both the satellite and the 5G scenario. Tables 5.2 and 5.3 display the performance difference of adaptive and static rapidARQ in the satellite and 5G scenarios, respectively, with and without considering re-transmissions. The experiment is repeated for different levels of burstiness. We define the performance difference by subtracting the obtained rapidARQ metric from the corresponding result of adaptive rapidARQ, i.e. $M_{diff} = M_a - M_s$, where, M_{diff} is the resulting difference for metric M, M_a refers to the performance metric of adaptive rapidARQ and M_s the corresponding metric of the static protocol. Assume T_{diff},

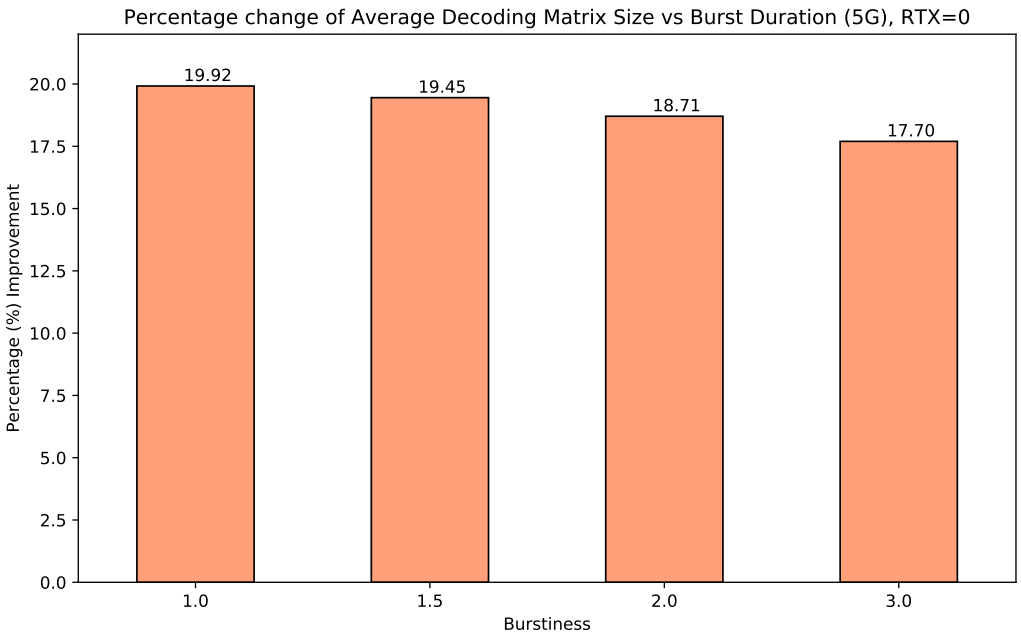


(a)

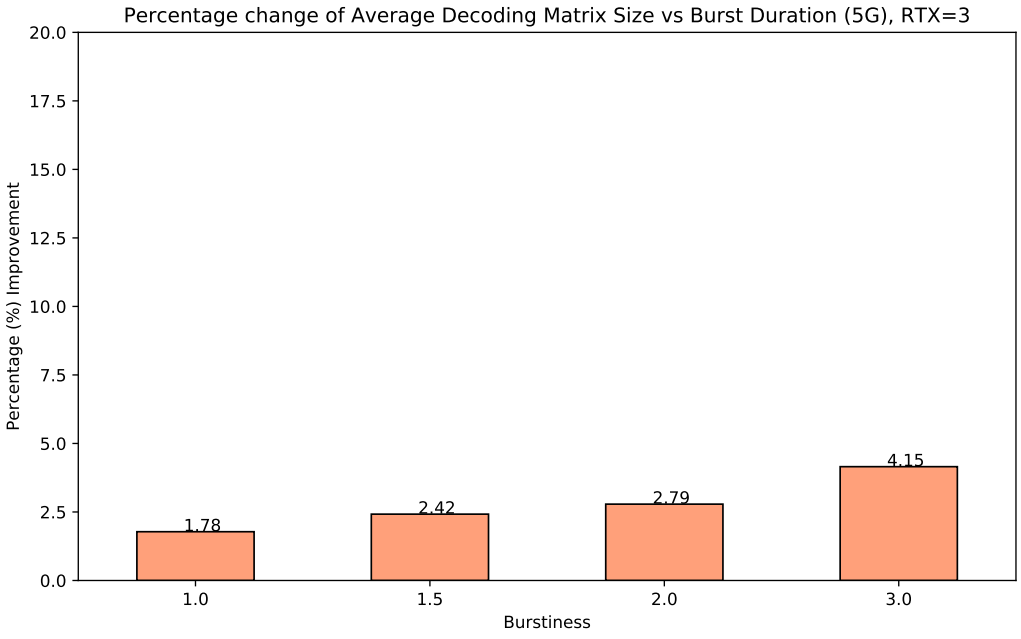


(b)

Figure 5.6: Percentage improvement of average decoding matrix size vs burstiness in the satellite scenario when (a) $RTX = 0$ and (b) $RTX = 3$.



(a)



(b)

Figure 5.7: Percentage improvement of average decoding matrix size vs burstiness in the 5G test scenario when (a) $RTX = 0$ and (b) $RTX = 3$.

D_{diff} and L_{diff} indicate the difference in throughput, delay and loss, respectively. By examining the table contents regarding both test scenarios, we can observe that adaptive rapidARQ achieves comparable performance in all evaluation metrics. Ideally, we would like to obtain increased throughput ($T_{diff} > 0$), reduced delay ($D_{diff} < 0$) and smaller packet loss rate ($L_{diff} < 0$) with the adaptive scheme. However, we cannot distinctly deduce that the adaptive algorithm ameliorates or deteriorates the overall protocol's performance. Although we can distinguish slight throughput gains, both the average delay and the packet loss rate are slightly worse in the adaptive scheme compared to its static counterpart.

As for the decoding matrix size, the adaptive scheme achieves superior performance compared to static rapidARQ. This can be verified by figures 5.6 and 5.7, which capture the percentage change of the average decoding matrix size of adaptive rapidARQ against static rapidARQ for the estimated d_{opt} value both in satellite (fig. 5.6) and in the 5G scenario (fig. 5.7). In the former scenario, figure 5.6a indicates a re-transmission free mechanism, while figure 5.6b refers to the case where re-transmissions are allowed. Similarly, figures 5.7a and 5.7b portray the same percentage improvements but in the context of 5G. At first sight, we observe that adaptive rapidARQ consistently yields improvements in the decoding matrix size. This observation is rational, because the adaptive scheme is designed in such a way so that it takes advantage of the idle time periods and reduces the decoding matrix size. Since increased burstiness indicates frequent idle time periods, the decoding matrix size benefits of the adaptive scheme should increase. However, the level of improvement varies depending on the experiment. In particular, when re-transmissions are not allowed in the satellite scenario (figure 5.6a), the most notable improvements pertain to intermediate levels of burstiness, that is $B = 1.5$ and $B = 2$. The percentage improvement in these cases exceeds 18% and 23%, respectively. This is attributed to the fact that the dynamic adaptation of d responds well to the channel variations. When the intensity of bursts is mediocre, the oscillations of the coding window size are not that extreme, hence the algorithm has enough opportunities to detect the absence of losses during "off" periods and gradually decrease the coding window size. This is not exactly the case, though, when bursts are very intense, because the benefits are reduced compared to smaller burstiness values. Typically, the algorithm increases d to a larger extent in order to provide enough protection to tackle any subsequent packet losses. Although the adaptive scheme is able to detect the loss-free periods,

any attempts to reduce the coding window size seem to be less efficient than one would anticipate, because the rapid increase during the update period ends up to a very large d_{max} value. In other words, in case of intense bursts, the reduction process still works but the decrease steps do not suffice to cancel the preceding increments in d, as it happens when burstiness is lower.

Different conclusions can be deduced, though, under the impact of re-transmissions (see figure 5.6b). Here, the performance improvements are rather small but they show an incremental trend while increasing burstiness. In fact, the percentage improvement when $B = 3$ exceeds 43%, which means that we can conserve around half of the storage requirements dictated by rapidARQ when $d = 15$. As a consequence, we expect that the decoding delay will be significantly reduced. The demonstrated results are rather reasonable; by employing re-transmissions the usefulness of coding is restricted to a certain extent. Further attempts to adjust the coding window size do not have a tremendous impact on coding complexity. On the other hand, when bursts are quite intense, e.g., when $B = 3$, the coding process plays a major role in coping with bursts of errors. Therefore, being able to identify the idle or burst time periods and appropriately adjust the coding window yields significant complexity improvements.

Similar observations can be made in the context of 5G. Under the effect of re-transmissions (fig. 5.7b), adaptive rapidARQ exhibits increasing complexity improvements in accordance with the increase in burstiness. Nevertheless, the gains are relatively small (up to 4% when $B = 3$), since the best value of d is dominated by the upper bound dictated by the bandwidth-delay product (see section 4.1), unlike the satellite scenario. On the other hand (fig. 5.7a), the complexity improvement without re-transmissions runs to $\sim 20\%$ with a decreasing inclination for larger burstiness values.

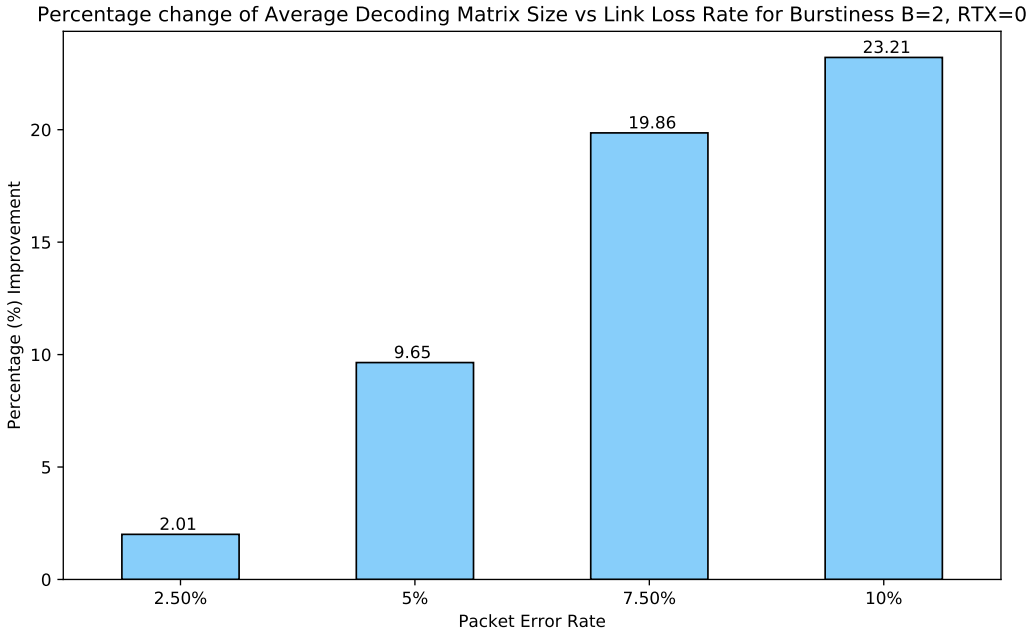
5.5 The Impact of Error Burst for Various Loss Rates

Previously, we examined the impact of burstiness in channels with a certain average packet loss rate. In this experiment, we keep burstiness fixed and vary the packet loss rate in order to investigate the performance benefits as well as the complexity ones. Here, we focus on the satellite test scenario without re-transmissions. We also study the impact of variable loss rates in channels with relatively heavy bursts, that

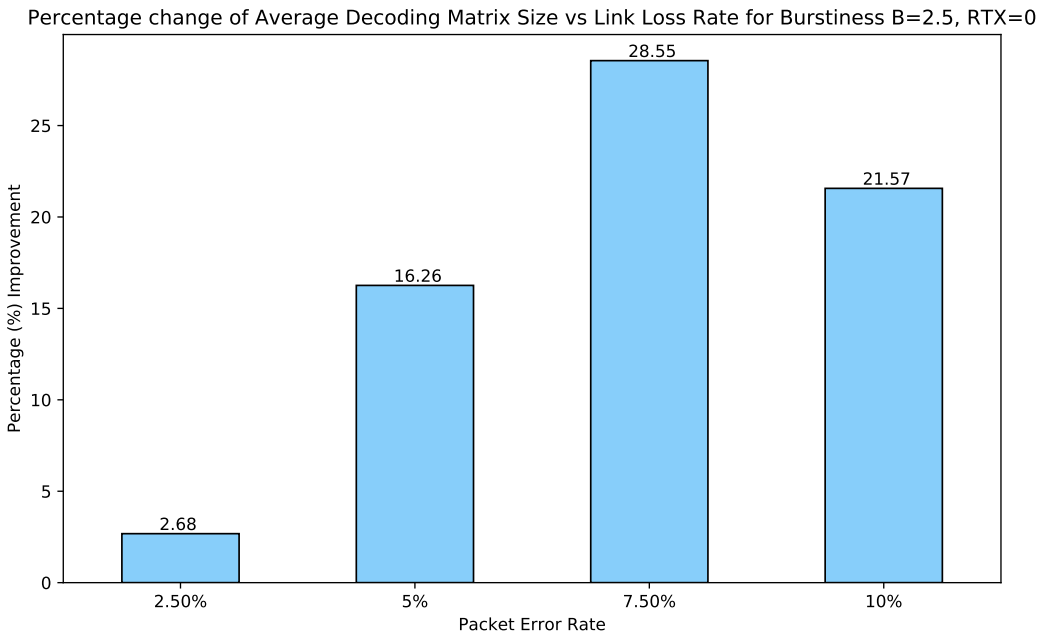
is, channels where the maximum packet loss rate during an "on" period is at least double than that of average p_l. In particular, we investigate the cases of $B = 2$, $B = 2.5$ and $B = 3$. Note that since we examine various average packet loss rates (p_l), the optimal d_{opt} estimation provided by the theoretical framework should be recalculated. Since the examined p_l values have been already considered in previous experiments, we exploit the estimated optimal d values illustrated in table 5.1. For $p_l = 10\%$, we consider $d_{opt} = 15$.

Figure 5.8 displays the average decoding matrix size percentage change for different link loss rate. Each subfigure pertains to a different burstiness level. The maximum complexity improvement exceeds 20% in each case. More specifically, the obtained improvements when $B = 2$ differ compared to higher burstiness values. Here, there is an increasing trend in complexity gains which follows the increase of the average packet loss rate. In other words, the higher p_l is, the higher the complexity improvements noticed. This is sensible, since small packet loss rates signify lower complexity at the receiver side and the coding window size converges to a relatively small value without further significant improvement margins. While the average p_l increases, more protection is required from the coding scheme, hence larger fluctuations are noticed in the coding window size. Adaptive rapidARQ is able to identify idle periods, which results in a significant complexity improvement. In more intense bursts of errors (e.g., fig. 5.8b, 5.8c), the main principles remain the same. Nevertheless, we can observe a notable difference when $p_l = 10\%$. Instead of further reducing complexity, adaptive rapidARQ cannot improve the computational cost beyond a certain extent. This behavior reminds us of the observation we made in section 5.4 for very intense bursts, which is attributed to the large fluctuations in the coding window size.

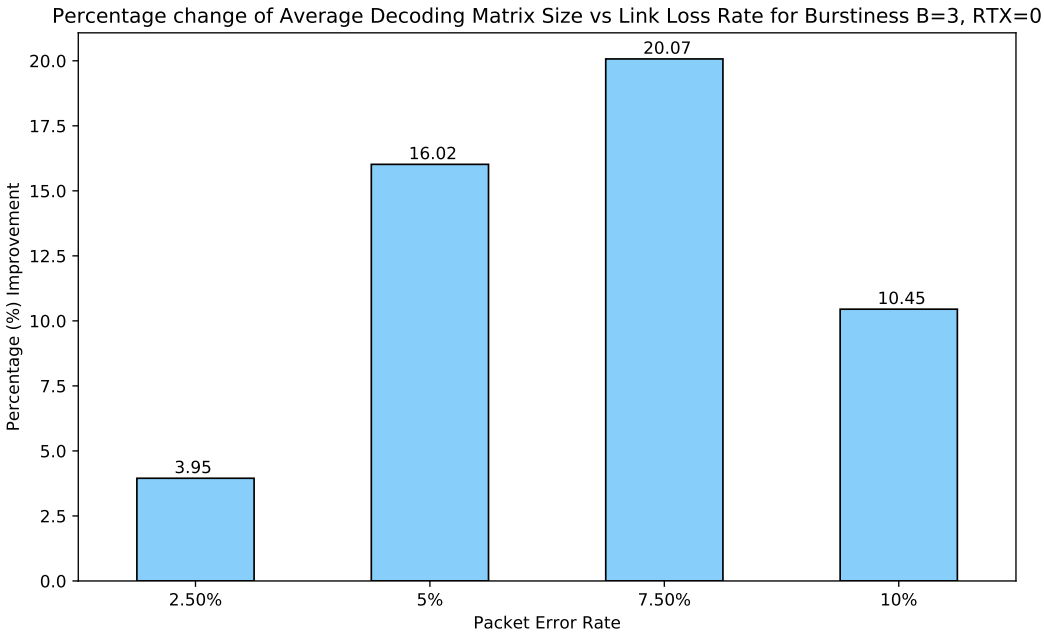
Besides complexity, we also evaluate the throughput, average delay and drop rate performance by calculating the difference of adaptive and static rapidARQ, under the assumption that both variants operate based on the optimal d estimation for each applied channel loss rate. Once again, the achieved performance is comparable to the one of the static scheme. Tables 5.4, 5.5 and 5.6 depict these results for $B = 2$, $B = 2.5$ and $B = 3$, respectively. Even though the adaptive scheme does not result in superior performance, the achieved results are close to the near-optimal one, which renders the applicability of adaptive rapidARQ quite successful in this context.


Percentage change of Average Decoding Matrix Size vs Link Loss Rate for Burstiness B=2, RTX=0
Percentage (%) Improvement
Packet Error Rate
2.01
9.65
19.86
23.21
2.50%
5%
7.50%
10%
0
5
10
15
20

(a)


Percentage change of Average Decoding Matrix Size vs Link Loss Rate for Burstiness B=2.5, RTX=0
Percentage (%) Improvement
Packet Error Rate
2.68
16.26
28.55
21.57
2.50%
5%
7.50%
10%
0
5
10
15
20
25

(b)



(c)

Figure 5.8: Percentage improvement of average decoding matrix size vs burstiness in the satellite scenario without re-transmissions: (a) $B = 2$, (b) $B = 2.5$ and (c) $B = 3$. For the applied link loss rates, the estimated optimal d values are: 2.5% - $d_{opt} = 14$, 5% - $d_{opt} = 15$, 7.5% - $d_{opt} = 18$ and 10% - $d_{opt} = 15$.

Table 5.4: Performance difference of Adaptive rapidARQ and rapidARQ in channels with variable packet error rate when $B = 2$ in the satellite scenario without re-transmissions ($RTX = 0$).

Packet Error Rate p_l	T_{diff} (Mbps)	D_{diff} (ms)	L_{diff}
2.50%	0.7531078	-0.096753	-0.0005332
5%	0.2326052	0.0129936	0.0002836
7.50%	0.5585636	0.0132738	0.0002524
10%	-0.1337934	1.0156504	0.0019448

Table 5.5: Performance difference of Adaptive rapidARQ and rapidARQ in channels with variable packet error rate when $B = 2.5$ in the satellite scenario without retransmissions ($RTX = 0$).

Packet Error Rate p_l	T_{diff} (Mbps)	D_{diff} (ms)	L_{diff}
2.50%	1.1946938	-0.1503742	-0.0009306
5%	0.1473676	0.0662248	0.000446
7.50%	1.990353	0.000889	-0.0015282
10%	1.840873	0.17727	-0.001595

Table 5.6: Performance difference of Adaptive rapidARQ and rapidARQ in channels with variable packet error rate when $B = 3$ in the satellite scenario without retransmissions ($RTX = 0$).

Packet Error Rate p_l	T_{diff} (Mbps)	D_{diff} (ms)	L_{diff}
2.50%	1.5487776	-0.2306058	-0.0014246
5%	0.5679624	-0.0346076	-0.0005262
7.50%	1.6705072	0.0120326	-0.001021
10%	0.8316014	0.0048696	-0.0006404

Chapter 6

Conclusion

6.1 Closing Remarks

5G and beyond networks are envisioned to completely alter our lifestyle by enabling a plethora of heterogeneous applications with different requirements in data rate, delay and reliability. In fact, they are considered to support URLLC services to a broad class of novel applications, where ultra-high reliability and very low latency are of critical concern. To satisfy such demands, it is essential that appropriate reliability mechanisms be devised in the data link layer and foremost in transport and application layers in order to cope with the occurring packet errors.

There are strong indications that the synergy between RLNC and the concept of sliding window can be a game changer in achieving ultra reliable data transmission over a link with minimum delay. The increasing dynamics of sliding window RLNC is reflected on the rising popularity of such schemes and their in-depth study in literature. However, existing SW RLNC schemes tend to neglect the discussion about the importance of an appropriate coding window size, although the latter plays a major role in the coding scheme's efficiency. In this thesis, we investigate the impact of the coding window size on the performance of SW RLNC protocols. To that end, we leverage the abstraction of coding depth to facilitate the discussion about the coding window size. By introducing coding depth (d), we create the required degrees of

freedom so that RLNC optimizes the level of protection offered to each source packet without degrading the goodput of the protocol, i.e., by modifying the amount of injected redundancy. It is still an open issue, though, how to determine the optimal coding window size so that SW RLNC schemes can achieve superior overall performance. This is extremely important under varying channel conditions, where the coding scheme should be able to detect the level of packet errors and appropriately adjust its coding parameters in order to achieve an optimal performance-complexity trade-off.

Motivated by the aforementioned observation, we attempt to devise an analytical framework for calculating the optimal d value for specific channel conditions. Since the precise approximation of coding depth is a daunting challenge, we notice that even a rough estimation can result in near-optimal performance. This relies on the observation that when coding depth exceeds a threshold value for a prescribed level of protection, no significant performance improvements are noticed. Therefore, we provide an analytical method to estimate this threshold d value by taking into account the channel conditions. Apart from that, we explore the dynamic adaptation of coding depth under varying channel conditions by implementing an efficient adaptive scheme for this purpose. We exploit the theoretical result to upper bound the range of values which can be assigned to d during the protocol's execution. Finally, we assess the performance of the introduced novelties through extensive experimental evaluation in test scenarios suitable for 5G and beyond networks. We confirm that the SW RLNC scheme which exploits the rough estimation of the optimal d value performs near optimally in all test cases. Similarly, the adaptive scheme achieves comparable performance to the near optimal one. We should further stress that the adaptive scheme can provide substantial complexity improvements through the efficient management of the coding window size.

6.2 Future Extensions

Clearly, the proposed analytical methodology as well as the adaptive scheme can benefit the performance of SW RLNC protocols. During this work, we implemented our reliability mechanism as a link-layer protocol. However, there is a tendency towards transferring such responsibilities in the transport or even in the application layer. This

is attributed to the convenience in introducing modifications by overlooking network routers and the restrictions posed by the host. By integrating the proposed novelties, e.g., in a transport-layer protocol, such as QUIC, we anticipate that the benefits of our scheme will be more intense since the bandwidth-delay product may grow very large over multiple transmission links. However, the real challenge is to integrate the ability of recoding in multi-hop wireless networks, where intermediate nodes also take part in the coding process, as the principles of network coding dictate.

Bibliography

[1] Y. Xu, G. Gui, H. Gacanin, and F. Adachi, "A survey on resource allocation for 5g heterogeneous networks: Current research, future trends and challenges," *IEEE Communications Surveys & Tutorials*, 2021.

[2] "Study on scenarios and requirements for next generation access technologies," 3GPP TSG RAN TR38.913 R14, Tech. Rep., 05-2017.

[3] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[5] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[6] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network coding: an instant primer," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 63–68, 2006.

[7] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," 2003.

[8] N. Papanikos and E. Papapetrou, "Deterministic broadcasting and random linear network coding in mobile ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1540–1554, 2017.

[9] J. K. Sundararajan, D. Shah, and M. Médard, "Arq for network coding," in *2008 IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 1651–1655.

[10] J. K. Sundararajan, D. Shah, M. Médard, and P. Sadeghi, "Feedback-based online network coding," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6628–6649, 2017.

[11] F. Gabriel, S. Wunderlich, S. Pandi, F. H. Fitzek, and M. Reisslein, "Caterpillar rlnc with feedback (crlnc-fb): Reducing delay in selective repeat arq through coding," *IEEE Access*, vol. 6, pp. 44 787–44 802, 2018.

[12] J. Cloud, D. Leith, and M. Médard, "A coded generalization of selective repeat arq," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2155–2163.

[13] J. Cloud and M. Médard, "Network coding over satcom: Lessons learned," in *International Conference on Wireless and Satellite Systems*. Springer, 2015, pp. 272–285.

[14] W.-L. Yeow, A. T. Hoang, and C.-K. Tham, "Minimizing delay for multicast-streaming in wireless networks with network coding," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 190–198.

[15] L. Yang, Y. E. Sagduyu, and J. H. Li, "Adaptive network coding for scheduling real-time traffic with hard deadlines," in *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*, 2012, pp. 105–114.

[16] H.-T. Lin, Y.-Y. Lin, and H.-J. Kang, "Adaptive network coding for broadband wireless access networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 4–18, 2012.

[17] Y. Benfattoum, S. Martin, and K. Al Agha, "Qos for real-time reliable multicasting in wireless multi-hop networks using a generation-based network coding," *Computer Networks*, vol. 57, no. 6, pp. 1488–1502, 2013.

[18] 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*, vol. 2010, pp. 1–14, 2010.

[19] W. Zeng, C. T. Ng, and M. Médard, "Joint coding and scheduling optimization in wireless systems with varying delay sensitivities," in *2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. IEEE, 2012, pp. 416–424.

[20] S. Pandi, F. Gabriel, J. A. Cabrera, S. Wunderlich, M. Reisslein, and F. H. Fitzek, "Pace: Redundancy engineering in rlnc for low-latency communication," *IEEE Access*, vol. 5, pp. 20 477–20 493, 2017.

[21] B. Swapna, A. Eryilmaz, and N. B. Shroff, "Throughput-delay analysis of random linear network coding for wireless broadcasting," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6328–6341, 2013.

[22] X. Zhong, Y. Qin, and L. Li, "Tcpnc-dgsa: Efficient network coding scheme for tcp in multi-hop cognitive radio networks," *Wireless Personal Communications*, vol. 84, no. 2, pp. 1243–1263, 2015.

[23] F. Karetsi and E. Papapetrou, "A low complexity network-coded arq protocol for ultra-reliable low latency communication," in *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2021, pp. 11–20.

[24] ——, "Lightweight network-coded arq: An approach for ultra-reliable low latency communication," *Computer Communications*, 2022.

[25] M. Karzand, D. J. Leith, J. Cloud, and M. Medard, "Design of fec for low delay in 5g," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 8, pp. 1783–1793, 2017.

[26] J. Cloud and M. Médard, "Multi-path low delay network codes," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[27] M. Karzand and D. J. Leith, "Low delay random linear coding over a stream," in *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2014, pp. 521–528.

[28] Y. Lin, B. Liang, and B. Li, "Slideor: Online opportunistic network coding in wireless mesh networks," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–5.

[29] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets tcp: Theory and implementation," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, 2011.

[30] P. U. Tournoux, E. Lochin, J. Lacan, A. Bouabdallah, and V. Roca, "On-the-fly erasure coding for real-time video applications," *IEEE Transactions on Multimedia*, vol. 13, no. 4, pp. 797–812, 2011.

[31] T. T. Thai, J. Lacan, and E. Lochin, "Joint on-the-fly network coding/video quality adaptation for real-time delivery," *Signal Processing: Image Communication*, vol. 29, no. 4, pp. 449–461, 2014.

[32] P. J. Braun, D. Malak, M. Medard, and P. Ekler, "Multi-source coded downloads," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[33] S. Wunderlich, F. Gabriel, S. Pandi, F. H. Fitzek, and M. Reisslein, "Caterpillar rlnc (crlnc): A practical finite sliding window rlnc approach," *IEEE Access*, vol. 5, pp. 20 183–20 197, 2017.

[34] M. Brulatout, H. Khalifé, V. Conan, J. Leguay, E. Lochin, and J. Lacan, "Reliable streaming protocol for lossy networks," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2015, pp. 1486–1491.

[35] A. Cohen, D. Malak, V. B. Bracha, and M. Médard, "Adaptive causal network coding with feedback," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 4325–4341, 2020.

[36] A. Cohen, G. Thiran, V. B. Bracha, and M. Médard, "Adaptive causal network coding with feedback for multipath multi-hop communications," *IEEE Transactions on Communications*, vol. 69, no. 2, pp. 766–785, 2020.

[37] E. Tasdemir, M. Tömösközi, J. A. Cabrera, F. Gabriel, D. You, F. H. Fitzek, and M. Reisslein, "Sparec: Sparse systematic rlnc recoding in multi-hop networks," *IEEE Access*, vol. 9, pp. 168 567–168 586, 2021.

[38] M. Zverev, P. Garrido, R. Agüero, and J. Bilbao, "Systematic network coding with overlap for iot scenarios," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2019, pp. 1–6.

[39] P. Garrido, D. Gómez, J. Lanza, and R. Agüero, "Exploiting sparse coding: A sliding window enhancement of a random linear network coding scheme," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[40] E. Tasdemir, V. Nguyen, G. T. Nguyen, F. H. Fitzek, and M. Reisslein, "Fsw: Fulcrum sliding window coding for low-latency communication," *IEEE Access*, 2022.

[41] J. Yang, Z.-P. Shi, C.-X. Wang, and J.-B. Ji, "Design of optimized sliding-window bats codes," *IEEE Communications Letters*, vol. 23, no. 3, pp. 410–413, 2019.

[42] P. Garrido, D. J. Leith, and R. Agüero, "Joint scheduling and coding for low in-order delivery delay over lossy paths with delayed feedback," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1987–2000, 2019.

[43] T. Geithner, F. Sivrikaya, and S. Albayrak, "Joint link rate selection and adaptive forward error correction for high-rate wireless multicast," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 830–846, 2022.

[44] M. Zverev, P. Garrido, F. Fernández, J. Bilbao, Ö. Alay, S. Ferlin, A. Brunstrom, and R. Agüero, "Robust quic: Integrating practical coding in a low latency transport protocol," *IEEE Access*, vol. 9, pp. 138 225–138 244, 2021.

[45] J. Heide, M. V. Pedersen, F. H. Fitzek, and T. Larsen, "Network coding for mobile devices-systematic binary random rateless codes," in *2009 IEEE International Conference on Communications Workshops*. IEEE, 2009, pp. 1–6.

[46] C.-C. Chen, C. Chen, J.-S. Park, S. Y. Oh, M. Gerla, and M. Sanadidi, "Multiple network coded tcp sessions in disruptive wireless scenarios," in *2011-MILCOM 2011 Military Communications Conference*. IEEE, 2011, pp. 754–759.

[47] T. Van Vu, N. Boukhatem, T. M. T. Nguyen, and G. Pujolle, "Adaptive redundancy control with network coding in multi-hop wireless networks," in *2013*

IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2013, pp. 1510–1515.

[48] A. Fu, P. Sadeghi, and M. Médard, "Dynamic rate adaptation for improved throughput and delay in wireless network coded broadcast," *IEEE/ACM Transactions on Networking*, vol. 22, no. 6, pp. 1715–1728, 2013.

[49] J. Rischke, F. Gabriel, S. Pandi, G. Nguyen, H. Salah, and F. H. Fitzek, "Improving communication reliability efficiently: Adaptive redundancy for rlnc in sdn," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 291–295.

[50] Y. Shi, Y. E. Sagduyu, J. Zhang, and J. H. Li, "Adaptive coding optimization in wireless networks: Design and implementation aspects," *IEEE Transactions on Wireless Communications*, vol. 14, no. 10, pp. 5672–5680, 2015.

[51] D. Malak, M. Médard, and E. M. Yeh, "Tiny codes for guaranteeable delay," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 809–825, 2019.

[52] H. H. Yin, K. H. Ng, A. Z. Zhong, R. W. Yeung, S. Yang, and I. Y. Chan, "Intrablock interleaving for batched network coding with blockwise adaptive recoding," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 4, pp. 1135–1149, 2021.

[53] S. McCanne and S. Floyd. ns Network Simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/

[54] E. Papapetrou and F. Karetsi, "Selective repeat arq implementation for ns2," [Online]. Available: https://github.com/epapapet/SR-ARQ, 2020.

Author's Publications

1. Karetsi, Foteini, and Evangelos Papapetrou. "A low complexity network-coded ARQ protocol for ultra-reliable low latency communication." 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). IEEE, 2021.

2. Karetsi, Foteini, and Evangelos Papapetrou. "Lightweight network-coded ARQ: An approach for Ultra-Reliable Low Latency Communication." Computer Communications (2022).

Short Biography

Foteini Karetsi was born in Ioannina in 1997. She obtained a Diploma in Computer Science and Engineering with honours from the Department of Computer Science and Engineering at the University of Ioannina, Greece, in 2020. She is currently an M.Sc. student at the same department. She has received several scholarships and awards for academic excellence. She was also a research intern in Prof. Katerina Argyraki's Network Architecture Laboratory (NAL) at École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, under Summer@EPFL 2021 program. Her research interests include network coding, routing and performance optimization for next generation wireless networks.