Packet Error Correction and Recovery based on Packet Combination and Cross-Layer Cooperation for Efficient Transmission Control Protocol Communications

by

Mona NAGHASHI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN INFORMATION TECHNOLOGY ENGINEERING
M.A.Sc.

MONTREAL, JULY 26, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

ACKNOWLEDGEMENTS

It has always been hard for me to express my emotions and my appreciation through manuscripts. But, I am taking this opportunity to say thanks to all the people who have been an inspiration to me and have supported me through this journey leading to this thesis.
First of all, I am extremely grateful to my parents for always being present, caring, loving, supporting and sacrificing for me and my life. Thanks to them for being true friends besides being my parents and for understanding and supporting me through my decisions. For the last 3 years, it was quite hard being far from you, but I was always delighted with your pure love. Mom and Dad, I'm deeply indebted to you for sacrificing your own dreams to help me towards mine. As always, I will try my best to fulfill your dreams about me.
I would like to express my deep and sincere gratitude and respect to my supervisor, Professor Stéphane Coulombe, a reliable leader, for supervising me professionally during my studies. This work would never have been a success without you. Dear Professor, many thanks for your knowledge, patience, advice and support. I was quite lucky and glad to work with you.
Thanks to my brother, Vahid, for his continuous encouragements and care to pursue my academic goals.
Thanks to École de technologie supérieure, for providing me with the opportunity of pursuing my studies as an international student in their professional engineering institution.
Thanks to my family, friends, and colleagues for being loyal and kind all the time.
Finally, I would like to thank the board of examiners that gave generously of their time for reviewing this thesis and for their participation in my defence examination, Professor Michel Kadoch and Professor Aris Leivadeas.

Correction et récupération de paquets basées sur la combinaison de paquets et la collaboration inter-couches pour des communications TCP efficaces

Mona NAGHASHI

RÉSUMÉ

Au cours des dernières décennies, nous avons assisté à une expansion sans cesse croissante des réseaux et des applications sans fil. Bien que ces réseaux soient sujets à des erreurs de transmission, nous parvenons à avoir une livraison de données fiable grâce à l'utilisation de protocoles tels que le Transmission Control Protocol (TCP) universellement utilisé. Ce dernier utilise des retransmissions, lorsque les données reçues sont corrompues, pour assurer des communications fiables au détriment d'une augmentation des délais de transmission et du surdébit. Pour éviter ces inconvénients, des méthodes améliorées de correction des erreurs et de récupération de paquets sont nécessaires.

De nombreuses approches ont été proposées pour faire face plus efficacement à la vulnérabilité des réseaux sans fil aux erreurs de transmission. Mais ils ont tous des avantages et des inconvénients spécifiques. Le Partial Packet Recovery (PPR) est parmi les plus répandus et les plus prometteurs. Il utilise les paquets partiellement reçus pour récupérer le paquet erroné et tente ainsi d'éviter autant que possible les retransmissions.

Dans cette thèse, nous visons à corriger et à récupérer les paquets corrompus au niveau du récepteur au lieu de les ignorer et de demander des retransmissions. En conséquence, nous introduisons trois méthodes du côté du récepteur: la reconstruction par majorité absolue (Majority Voting with Checksum Validation (MVCV)), la reconstruction par énumération des différences avec validation par somme de contrôle (Dissimilarity-based Enumeration Packet Recovery with Checksum Validation (DEPRCV)) et la reconstruction par approche inter-couches (Cross-Layer Packet Recovery (CLPR)). Elles tentent toutes de reconstruire les paquets en se basant sur une ou plusieurs copies reçues en erreur. MVCV repose sur la majorité des occurrences de valeur de bit parmi les paquets reçus à chaque position de bit. DEPRCV identifie les positions de bits où différentes valeurs sont observées dans chaque paquet reçu et construit une liste possible de paquets sans erreur. Dans les deux méthodes, les paquets candidats reconstruits sont validés avec la somme de contrôle Transmission Control Protocol (TCP). Enfin, CLPR combine une méthode CRC-based Error Correction (CRC-EC) récemment proposée avec DEPRCV et MVCV. Un avantage clé de ces méthodes est qu'elles ne nécessitent aucune modification de TCP, c'est-à-dire qu'elles sont compatible avec la norme TCP, et peuvent être facilement implémentées dans un client.

Les résultats de la simulation montrent la supériorité des méthodes proposées en termes de nombre de retransmissions et de délai de transmission par rapport à TCP et CRC-based Error Correction (CRC-EC). Ces méthodes améliorent également le débit efficace du réseau et l'efficacité énergétique. Par exemple, à un taux de bits en erreur de 0.0001, TCP et CRC-EC nécessitent tous deux 3 retransmissions pour livrer de manière fiable 99.9% des paquets tandis que MVCV, DEPRCV et CLPR nécessitent 2, 2 et 1 retransmissions pour atteindre les 99.9 % de livraison fiable des paquets. Pour un taux de bits en erreur plus élevé de 0.001, pour une livraison fiable de 99.9% des paquets, TCP et CRC-EC nécessitent au moins 20 et 8 retransmissions dans cet ordre, tandis que MVCV, DEPRCV et CLPR ne nécessitent que 3, 1 et 1 retransmissions, respectivement.

Mots-clés: correction d'erreurs, protocole de contrôle de transmission, combinaison de paquets, contrôle de redondance cyclique (CRC), correction d'erreurs basée sur le CRC, TCP.

Packet Error Correction and Recovery based on Packet Combination and Cross-Layer Cooperation for Efficient Transmission Control Protocol Communications

Mona NAGHASHI

ABSTRACT

For the last few decades, we have witnessed an ever-growing expansion of wireless networks and applications. Although these networks are subject to transmission errors, we manage to have reliable data delivery through the use of protocols such as the universally used Transmission Control Protocol (TCP). The latter uses retransmissions, when the received data is corrupted, to ensure reliable communications at the expense of an increase in transmission delays and overhead. To avoid these drawbacks, improved methods for error correction and packet recovery are needed.

Many approaches have been introduced to cope more efficiently with the vulnerability of wireless networks to transmission errors. But they all come with specific advantages and disadvantages. The Partial Packet Recovery (PPR) is and among the most widespread and promising ones. It utilizes the partially received packets to recover the erroneous packet and thus attempts to avoid retransmissions as much as possible.

In this thesis, we aim at correcting and recovering corrupted packets at the receiver instead of ignoring them and asking for retransmissions. Accordingly, we introduce three methods at the receiver's side : Majority Voting with Checksum Validation (MVCV), Dissimilarity-based Enumeration Packet Recovery with Checksum Validation (DEPRCV), and Cross-Layer Packet Recovery (CLPR). They all attempt to reconstruct the error-free packets relying on one or several erroneously received copies. MVCV relies on the majority of bit value occurrences among received packets at each bit position. DEPRCV identifies bit positions where different values are observed within each received packet and constructs a list of possible error-free packets. In both methods, the candidate reconstructed packets are validated with the Transmission Control Protocol (TCP) checksum. Finally, CLPR combines a recently proposed CRC-based Error Correction (CRC-EC) method with DEPRCV and MVCV. A key benefit of these methods is that it doesn't require any change to TCP, i.e., compatible with TCP, and can be easily implemented in a client.

Simulation results show the superiority of the proposed methods in terms of the number of retransmissions and transmission delay when compared to TCP and CRC-based Error Correction (CRC-EC). They also improve effective network throughput and energy efficiency. For instance, at a Bit Error Rate (BER) of 0.0001, TCP and CRC-EC both require 3 retransmissions to deliver reliably 99.9% of the packets while MVCV, DEPRCV and CLPR require 2, 2, and 1 retransmissions, respectively, in order to reach the same 99.9% reliability level. For a higher BER of 0.001, for the reliable delivery of 99.9% of the packets, TCP and CRC-EC require at least 20 and 8 retransmissions, respectively, while MVCV, DEPRCV and CLPR require 3, 1, and 1 retransmissions.

Keywords: error correction, transmission control protocol, packet combination, packet recovery, cyclic redundancy check (CRC), CRC-based error correction.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS AND ACRONYMS

BER	Bit Error Rate
CLPR	Cross-Layer Packet Recovery
CRC	Cyclic Redundancy Check
CRC-EC	CRC-based Error Correction
CV	Checksum Validation
DEPR	Dissimilarity-based Enumeration Packet Recovery
DEPRCV	Dissimilarity-based Enumeration Packet Recovery with Checksum Validation
DEPRCV-GEN	Dissimilarity-based Enumeration Packet Recovery with Checksum Validation Generation of Candidates
ECC	Error Correction Code
FEC	Forward Error Correction
MV	Majority Voting
MVCV	Majority Voting with Checksum Validation
PPR	Partial Packet Recovery
RTT	Round Trip Time
TCP	Transmission Control Protocol
WSN	Wireless Sensor Network

INTRODUCTION

0.1 Context

Nowadays, wireless networks play a key role in connecting various devices: personal computers, mobile phones, sensors, etc. (Nicopolitidis, Obaidat, Papadimitriou & Pomportsis, 2003). But compared to wired networks, wireless networks are more susceptible to errors and packet corruption. Since the early days of the Internet, the Transmission Control Protocol (TCP) has been used by the various networks for reliable data delivery. The main mechanism used by TCP to ensure reliable packet delivery is the *retransmission*. Over the past decades, researchers have shown that packet retransmission is not the most efficient way to salvage corrupted packets because it degrades the network throughput and bandwidth, and increases the delay and energy consumption. Lately, an assortment of error correction and packet recovery methods have been introduced to address this issue (Angelopoulos, Chandrakasan & Médard, 2014; Angelopoulos, Médard & Chandrakasan, 2017; Boussard, Coulombe, Coudoux & Corlay, 2020a; Boussard *et al.*, 2020a; Boussard, Golaghazadeh, Coulombe, Coudoux & Corlay, 2020b; Boussard, Coulombe, Coudoux & Corlay, 2021a,2; Dubois-Ferriere, Estrin & Vetterli, 2005; Fang, Schonfeld, Ansari & Leigh, 2000; Galluccio, Morabito & Palazzo, 2006; Han, Schulman, Gringoli, Spring, Bhattacharjee, Nava, Ji, Lee & Miller, 2010; Irianto, Nguyen, Salah & Fitzek, 2019; Jamieson & Balakrishnan, 2007; Khan, Moosa, Naeem, Alizai & Kim, 2016; Laurindo, Moraes & Montez, 2020; Lin, Kushman & Katabi, 2008; Miu, Balakrishnan & Koksal, 2005; Mohammadi, Zhang & Dutkiewicz, 2015; Qi, Wang, Wu & Tao, 2015; Sindhu, 1977; Tan & Zakhor, 1999; Xie, Hu & Zhang, 2011; Yavatkar & Bhagawat, 1994; Yli-Juuti, Chakraborty & Liinaharja, 1998; Zhang, Hu & Xie, 2012). Each of these methods has specific advantages and disadvantages which the more important ones are summarized as follows:

- Retransmissions cause network performance degradation (reduced throughput, high delay, high bandwidth, high energy consumption, etc.) (Irianto *et al.*, 2019).

- Methods based on Partial Packet Recovery (PPR) with PHY-layer information (PLI) approach are difficult to implement because they require modifying the current network and protocols (Irianto *et al.*, 2019).
- Cyclic redundancy check (CRC)-based error correction techniques are only able to perform packet recovery for low Bit Error Rate (BER) values.
- FEC brings redundant overhead and computational overhead (Irianto *et al.*, 2019).

Because of these issues with existing methods, there is a need to develop novel and practical error correction and packet recovery methods. By practical, we mean that can be implemented economically into communication devices. An important consideration is to maintain compatibility with existing TCP devices, i.e., develop a method compatible with the TCP standard. In this thesis, we make the assumption that the packets are lightly damaged, i.e., that the number of bits in error in a packet is relatively low. In such a situation, error correction methods such as CRC-based error correction can be exploited and the risk of miscorrection by the proposed methods is low since it will be followed by a checksum validation (Boussard *et al.*, 2021a).

0.2 Problem Statement

Our research problem is thus to develop new methods to perform error correction and packet recovery at the receiver side to ensure reliable and fast communications of TCP packets in an error-prone network environment where the impairments are not severe. More specifically, we are looking for TCP-compatible methods which do not add any overhead when the communications are reliable or slightly unreliable (e.g., one error per packet) while still being able to deliver all packets reliably regardless of the network conditions. **The focus of this research is on the efficient delivery of TCP packets over non congested but unreliable networks where packets are only be lightly affected by errors, i.e. where the number of erroneous bits in a packet is low. Furthermore, the solutions need to be fully compatible with the TCP standard**. In this thesis, we propose several new techniques for error correction and packet recovery for TCP communications. These techniques are combining or inspired from state-of-the-art methods in an effort to exploit their advantages while compensating for their weaknesses.

0.3 Contributions

The proposed error correction and packet recovery techniques in this thesis all combine the information of multiple erroneous transmissions of a packet to determine the associated error-free packet and are summarized as follows:

- Majority Voting with Checksum Validation (MVCV). For each bit position, the associated error-free packet is constructed by determining the bit value occurring the most often at that bit position among the previously received versions. The validity of this packet is determined using the TCP checksum before being declared the error-free packet.
- Dissimilarity-based Enumeration Packet Recovery with Checksum Validation (DEPRCV). The associated error-free packet is constructed by first identifying the bit positions where there are different values observed between versions. An enumeration of the potential error-free packets is established by creating a list of all possible packets obtained by the combination of bit values at these positions and the values that they all share at the other positions. The error-free packet is determined using the TCP checksum and the method is successful when the outcome is a single packet.
- Cross-Layer Packet Recovery (CLPR). The method first attempts to perform CRC-based error correction assuming a single error. If this step fails, it falls back to DEPRCV or MVCV methods.

It is important to note that all these methods are fully compatible with the TCP standard.

0.4 Thesis Structure

This manuscript is organized as follows. In Chapter 1, we provide the fundamental concepts related to the research. In Chapter 2, we review literature and state-of-the-art methods for error correction and packet recovery. We also compare these methods and analyze their advantages and drawbacks. Chapter 3 is dedicated to present the proposed methods for the research problem. In Chapter 4, we provide the experimental setup, simulation methodology, and experimental results of the proposed methods. We then conclude the work and make some recommendations.

CHAPTER 1

FUNDAMENTAL NETWORKING CONCEPTS

In this chapter, we present some fundamental networking concepts required to understand the methods presented in this thesis. The reader familiar with these topics may skip this chapter.

1.1 Introduction

Two computers are said to be interconnected if they can exchange information with each other. How this is done is crucial as networks come in many sizes, shapes, and forms to connect people and computer systems. Networks of computers are usually connected to make larger networks, with the Internet being the most well-known example of a network of networks (Comer, 2018). Between different kinds of networks, using a common language is the most important parameter of having effective communication between the members. The effectiveness of using a common language for having communication applies to people, animals, machines, and also computers. The merging of computers and communications has had a profound influence on the way computer systems are organized. Computers, as significant parts of computer networks, must use various protocols special in various situations. In other words, protocols are a way of ensuring that devices can talk to each other effectively and commonly (Fall & Stevens, 2011). In most cases, an individual protocol can describe how communication is accomplished between one particular software or hardware element in two or more devices. As an example, the TCP is responsible for the specific set of functions on the TCP/IP network. TCP/IP is a protocol suite that implements the Internet architecture and draws its origins from the ARPANET Reference Model (ARM) (Padlipsky, 1982). This type of protocol allows computers of all sizes, from many different computer vendors, running different operating systems, to communicate with each other. It forms the basis for what is called the worldwide Internet, or the Internet, a wide area network (WAN) of more than millions of computers that literally spans the globe (Comer, 2018). In this chapter, we will provide some basic knowledge about computers, networks, and protocols, which helps the reader to understand the rest of this thesis.

1.2 Network Layering

An architecture called Open Systems Interconnection (OSI) architecture is designed by the ISO organization to formally define a common way to connect computers (Zimmermann, 1980). This architecture is illustrated in the Figure 1.1, which defines a partitioning of network functionality into seven layers, where one or more protocols implement the functionality assigned to a given layer. This figure is a reference model for protocol graph (Hennessy & Patterson, 2011).

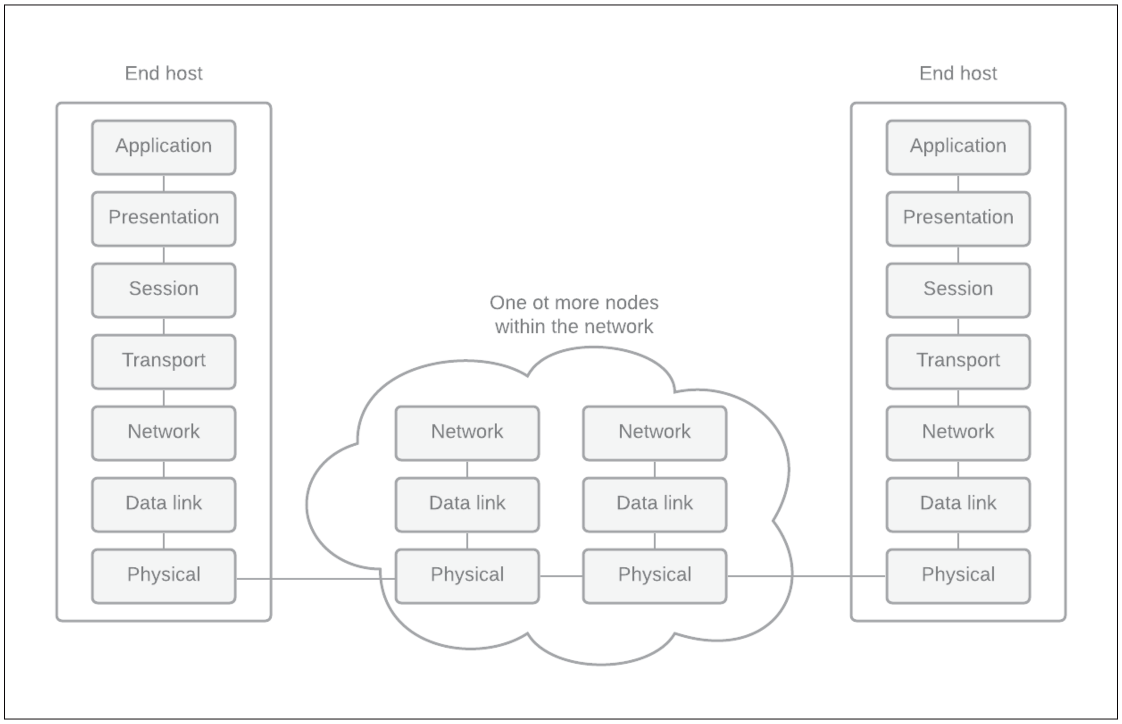


Figure 1.1 OSI network architecture
Taken from Hennessy & Patterson (2011)

The functionality of each layer is as follows (Hennessy & Patterson, 2011):

- Physical layer: handles the transmission of bits over a communication link.
- Data link layer: collects a stream of bits into a larger aggregate called a frame.
- Network layer: handles routing among nodes within a packet-switched network (exchanged data among nodes is called a packet instead of a frame).

- Transport layer: implements a process-to-process channel (the unit of exchanged data is called a message in this layer).
- Session layer: provides the mechanism for the opening, closing, and managing a session between end-user application processes.
- Presentation layer: is responsible for the delivery and formatting of information to the application layer for further processing or display.
- Application layer: is an abstraction layer that specifies the shared communication protocols and interface methods used by hosts in a communication network.

1.3 Internet Architecture

The Internet architecture, which is sometimes called TCP/IP architecture, is depicted in Figure 1.2. This architecture evolved out of experiences with an earlier packet-switched network called the ARPANET. A five-layer TCP/IP model for the Internet is usually used instead of the seven-layer model. This structure is simpler than the OSI model with 7 layers, but real implementations include a few specialized protocols that do not fit into the conventional layers (Fall & Stevens, 2011). According to Figure 1.2, there are protocols for each layer of the Internet model. There are a wide variety of network protocols in the link layer denoted NET_1, NET_2, and so on. The network layer consists of a single protocol, which is named the Internet Protocol (IP) (Hennessy & Patterson, 2011). This type of protocol supports the interconnection of multiple networking technologies into a single, logical internetwork. There are two main protocols for the fourth layer, which are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP and UDP provide alternative logical channels to application programs. TCP provides a reliable byte-stream channel, and UDP provides an unreliable datagram delivery channel. These protocols are called end-to-end protocols or transport protocols. There are varied protocols for the application layer as the fifth layer, which include: File Transport Protocol (FTP), Trivial File Transport Protocol (TFTP), Telnet (for remote login), and Simple Mail Transfer Protocol (SMTP) for electronic mail, which enable interoperation of popular applications. As an example, it is possible to access a particular site on the web by using any browser applications such as Firefox, Safari, Google Chrome, etc., because they all conform to the same application layer protocol: HTTP (HyperText Transport Protocol) (Hennessy & Patterson, 2011).

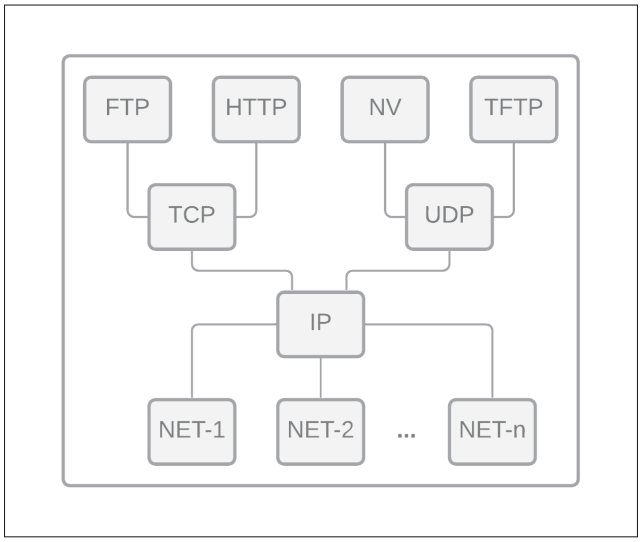


Figure 1.2 Internet protocol graph
Taken from Hennessy & Patterson (2011)

1.4 TCP/IP Protocol Overview

The TCP/IP protocol consists of various protocols, but only a few are the main ones, which define the core operations of the suite. Two of these main protocols are considered as the most important protocols, which are the IP protocol, the primary OSI network layer (layer 3) protocol that provides addressing, datagram routing, and other functions in an internetwork, and the Transmission Control Protocol (TCP), the primary transport layer (layer 4) protocol, which is responsible for connection establishment and management reliable data transport between applications on devices (Fall & Stevens, 2011). Since these two protocols are important, their abbreviations represent the entire TCP/IP suite. The TCP/IP suite has very critical functions that are implemented at layers three and four, which cause each of the transfer control and Internet protocols to be significant protocols. TCP/IP uses its four-layer architecture that corresponds to the OSI reference model (Kozierok, 2005).

1.4.1 TCP/IP Protocols

The Internet network, which connects all kinds of computer systems from supercomputers, costing millions of dollars, to personal computers varies from wireless to wired or from copper to fiber. All of these variations are supported and enabled by protocols and software collectively known as the TCP/IP Internet protocol suite or TCP/IP (Fall & Stevens, 2011). There are some specific protocols for each layer of the TCP/IP protocol suite. These protocols in each layer of the TCP/IP architecture model work and gather together to allow TCP/IP as a whole to operate (Fall & Stevens, 2011; Kozierok, 2005). The most important protocols of the TCP/IP protocol suite are Internet Protocol (IP), TCP, and UDP, which are called core protocols and they support many other protocols to perform a variety of functions at each layer of the TCP/IP model layers (Kozierok, 2005).

1.4.2 The Transmission Control Protocol

The transmission control protocol is one of the main protocols of the Internet protocol suite. Since TCP originated in initial network implementations and complemented the IP protocol, therefore the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and error-checked delivery of a stream of data between applications, which are running on hosts communicating via an IP network (Fall & Stevens, 2011). TCP/IP uses its five-layer architecture that corresponds roughly to the OSI reference model and provides a framework for the various protocols that comprise the suite. This protocol also includes some high-level and well-known applications by Internet users, who may not realize that they are parts of TCP/IP protocol, such as HTTP and FTP Kozierok (2005).

1.4.2.1 TCP Timeout and Retransmission

TCP provides reliable delivery of data between too end hosts by using an underlying network layer IP, which may cause losing, duplication, or reordering of transmitted packets. TCP uses some techniques to detect which packets need retransmission. It depends on a continuous flow of acknowledgments from the receiver to the sender, which acknowledges the successful arrival of packets. When packets or acknowledgments are lost, a retransmission process is initiated for the related packet which has not been acknowledged or for the related acknowledgment which has not arrived. There are two separate mechanisms for accomplishing retransmission. One is based on time and one is based on the structure of acknowledgments (Fall & Stevens, 2011). When a packet is sent over TCP, TCP sets a timer for that. When the timer expires, if the packet is not acknowledged by the receiver side, a timeout or timer-based retransmission of data occurs. The timeout occurs after an interval, which is called the retransmission timeout (RTO).

There is another retransmission type, which is named fast retransmission or fast retransmit that happens quickly and without any delay if the receiver receives an out-of-order segment. It works as follows. In TCP, upon the reception of an each new packet, the receiver sends an ACK indicating the highest packet number it received without any loss. For instance, if packets 1, 2, 3 and 4 are transmitted but packet 2 is lost, the receiver will send an ACK 1 after the reception of the packets 1, 3, and 4 respectively since packet 2 breaks the sequence of consecutively received packets. Therefore, in the case of lost packets, the receiver sends duplicate ACKs to the sender. With fast retransmission, if the sender receives three duplicate ACKs, it assumes that a segment is lost and immediately performs fast retransmissions without waiting for the timeout (Fall & Stevens, 2011). As we discussed before, TCP sends an acknowledgment after receiving a packet at the receiver side. So, it is possible to send a byte with a particular sequence number and measure the time required to receive an acknowledgment, which covers that sequence number. Such measurement is called an RTT sample. One of the challenges for TCP is to estimate accurately the RTT values for transmissions in appropriate value ranges given a set of samples that vary over time. A key TCP operation is to set the RTO based on these values. Performing this process in the right way is very important for TCP performance (Fall & Stevens, 2011).

In the scope of our work, it is sufficient to know that when a packet has been received with errors, the retransmission process will be initiated.

1.5 Lower Layer Technologies

1.5.1 Wired Networks - Ethernet (IEEE 802.3)

The Ethernet is a data link layer Local Area Network (LAN) technology defined in the IEEE 802.3 standard Spurgeon (2000), which uses Carrier Sense Multiple Access with Collision Detection (CSMA/CD) as the media access control method. Ethernet simply refers to the most common type of LAN used today. Ethernet variants are known as “A” Base-“B” networks, where A stands for the speed, and “B” identifies the type of physical medium used (e.g., 10 Base-T for 10 Mbps Ethernet) (Spurgeon, 2000). There are three different Ethernet protocols, which provide data transmitting at various speeds. If maximum speed is needed, it is possible to go with the fastest Gigabit Ethernet using both fiber optic and twisted-pair cabling. Today’s Gigabit Ethernet supports speeds of up to 1,000 Mbps (Miller, 2013; Spurgeon, 2000).

1.5.2 Wireless Networks

The world has become mobile. Consequently, traditional ways of networking were not adequate to meet the challenges posed by people’s new collective lifestyle. Therefore, it was clear that people needed another type of networking, which made it possible for them to be connected to networks from everywhere they are. Because being connected to a network by physical cables, reduces users’ movement dramatically. Wireless telephony has been successful because it enabled people to connect regardless of location. Wireless networks provide some advantages, which are not deniable. The first advantage of wireless networking is mobility, which lets people connect to existing networks and are then allowed to roam freely. Another advantage of wireless networking is its flexibility which is an important attribute for service providers (Gast, 2005).

1.5.2.1 The IEEE 802.11 Networks

802.11 is a set of Wireless Local Area Network standards developed by the Institute of Electrical and Electronic Engineers (IEEE). The IEEE 802.11 standard is quite similar to Ethernet but it adapts traditional Ethernet technology to a wireless world. The 802.11 is a member of the IEEE 802 family, which are a series of specifications for LAN technologies. The IEEE 802 specifications are focused on the two lowest layers of the OSI reference model because they incorporate both physical and data link components. All of the 802 networks have both a Medium Access control (MAC) and a Physical (PHY) component. The MAC comprises a set of rules to determine how to access the medium and send data, but the details of transmission and reception are left to the PHY component (Gast, 2005). By using a second number, individual specifications are determined. For example, 802.3 is the specification for a Carrier Sense Multiple Access networks with Collision Detection (CSMA/CD), 802.5 is the token ring specification, 802.1 specifies the management features for 802 networks, and 802.2 specifies a common link layer, the Logical Link Control (LLC), which can be used by any lower-layer LAN technology. The 802.11 is another link layer that can use the 802.2/LLC encapsulation. The 802.11b specifies a high-rate direct-sequence layer (HR/DSSS). The 802.11a describes a physical layer based on orthogonal frequency division multiplexing (OFDM). According to Figure 1.3, the 802.11 networks consist of four major components as follows (Gast, 2005):

- Distribution system: a logical component of the 802.11 networks, which forward the frames to their destination.
- Access point: performs the wireless-to-wired bridging function, which converts the frames to another type of frame for delivery to the rest of the world.
- Wireless medium: is used for moving frames from station to station[1].
- Station (endpoint): a computing device with a wireless network interface such as a laptop or hand-held computer.

This standard led to the Wi-Fi commercial deployments with 802.11b, 802.11a, 802.11g, 802.11n, 802.11ac, and 802.11ax protocols. During the last decades, Wireless Fidelity or Wi-Fi has quickly grown to become the dominant wireless LAN standard. Because it operates in unlicensed frequency bands, anyone can set up a Wi-Fi network and cover an area of typically 100-500 feet

[1] In this context, stations are endpoints

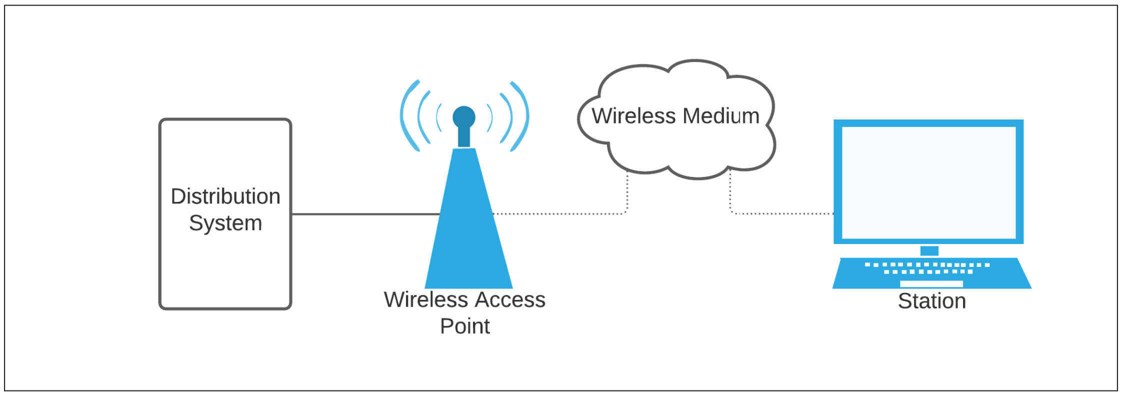


Figure 1.3 Components of 802.11 LANs
Taken from Gast (2005)

with high-speed wireless access to a LAN and hence to the Internet (Al-Alawi, 2006). Wi-Fi is a family of wireless network protocols, based on the IEEE 802.11 family of standards, which are commonly used for local area networking of devices and Internet access, allowing nearby digital devices to exchange data by radio waves. This family is almost the most widely used computer networks in the world, used globally in home and small office networks, in public places like coffee shops, hotels, libraries, and airports to provide the public Internet access for mobile devices to link them together and to a wireless router to connect them to the Internet, and in wireless access points.

1.5.2.2 Choosing a Wireless Service

There are different types of wireless services, which are appropriate for the specific combination of cost, coverage areas, reliability, ease of use, and security situations (Ross, 2008). People's choices will depend on their need and availability of signals in the locations where they need wireless Internet access. As an example, if people need to use their devices in just a few locations, which are within the range of Wi-Fi (Wireless-Ethernet) hotspots, the built-in Wi-Fi adapter (or an inexpensive plug-in adapter) is probably their best choice. The Wi-Fi hotspots, which use spread-spectrum radio signalling to distribute computer data through local area networks, are available in most general and public locations. Because Wi-Fi access points are relatively inexpensive to install, they can be found in numerous locations. Else, if people need a constant wireless Internet access wherever they go, the cellular data and WiMAX metropolitan area network services are better choices for them. Both of these systems provide large geographic regions coverage of Internet access and both allow people to maintain a connection while they're moving from one place to another (Ross, 2008). However, it's important to make sure that there's a usable cellular or WiMAX signal in all the places where they expect to use them before they commit to a long-term contract. Most devices operate with both types of wireless services. When they detect a high-speed Wi-Fi signal, they will automatically try to establish a connection to that network. But, when there is no local Wi-Fi signal, they will automatically shift over to WiMAX or cellular data account and use that service to connect to the Internet (Ross, 2008).

1.5.2.3 Wireless Sensor Networks

Wireless Sensor Network (WSN) is a category of wireless networks, in which sensors are used to monitor and record the physical conditions of the environment (i.e., temperature, sound, pollution levels, humidity, wind, and etc.) to gather and analyze the collected data in a central station in order to cooperatively pass the data through the networks based on the current network condition (Raghavendra, Sivalingam & Znati, 2006). Therefore the quality of the link layer communication depends on the condition of the channel underneath and the internode distance. These networks are more reliable than WiFi or mobile networks because of the provided feedback by the sensor and its collaborative structure, which make them suitable for the military, healthcare, environmental and industrial applications (Raghavendra *et al.*, 2006). Since wireless sensor networks have been utilized widely in specific domains and there are significant applications for them, improving the performance of TCP over this type of wireless networks is an important goal of the proposed methods in this thesis.

1.5.2.4 Cooperative Communication in Wireless Sensor Networks

Cooperative communication is a technique used in wireless sensor networks, in which, single antenna mobiles share their antennas and take the benefits of multiple antennas communications.

Each wireless end-node will be assumed as a cooperative agent for another user besides being able to transmit its own data. This cooperative communication leads to generate a virtual multiple-antenna transmitter, which provides a more reliable wireless communication (Laurindo *et al.*, 2020).

CHAPTER 2

LITERATURE REVIEW

In this chapter, we review the state-of-the-art error correction and packet recovery methods relevant to this thesis.

2.1 Protocols and Mechanisms to Recover Erroneous Packets in Wired and Wireless Networks

Compared to wired networks, wireless networks presents challenging conditions because they are more susceptible to packet failure. Packet failure can take two forms: 1) packet corruption, which means the packet is received albeit with bit errors, and 2) packet loss, which means the packet is not received at all and is lost at the physical layer. As mentioned, some techniques are developed to recover the corrupted or lost packets such as retransmission. But retransmitting the packets degrades the network's performance metrics such as throughput, delay, and energy consumption. To cope with this problem, some packet recovery techniques are proposed, which try to discern packet corruption from packet loss. These techniques try to recover packets on the receiver side, without or with the least number of retransmissions. The protocols and mechanisms to recover failed packets are grouped into three main groups including (Khan *et al.*, 2016):

1. TCP retransmission. It is the most traditional method of wired and wireless networks. In this method, the sender has to retransmit a packet if it has not been acknowledged within a certain period. This method is more commonly known as automatic repeat request (ARQ), which is a fundamental method employed in many protocols. It employs a checksum algorithm to verify the integrity of a received packet. A correctly received packet is positively acknowledged (ACK) whereas packet failure is either negatively acknowledged by the receiver (NACK) or assumed by the sender upon the expiry of a retransmission timer when no ACK/NACK is received (when a timeout occurs). It is very important to note that real-time services and applications requiring low latency cannot use ARQ schemes

when the data would arrive too late to be useful by the time the retransmission arrives at the receiver because of the round trip time.

2. Error correction: When retransmission is not possible or too expensive in terms of energy and bandwidth, error correction methods may be deployed at different layers of the network stack. According to our explanations in the previous chapter, error correction code (ECC) also known as Forward Error Correction (FEC) or channel coding is a technique, which adds redundant data to a packet that can be used by the receiver to correct errors introduced during transmission. Besides ECCs, some other methods have emerged for the recovery of corrupted packets, which range from the simple ones that combine multiple corrupted copies of a packet to the more complex ones that use confidence values from the physical layer. We will discuss these methods in the next section.

3. Error tolerance: This approach is accepting erroneous packets while disregarding errors and is used mostly in real-time applications.

Our main focus in this thesis will be on retransmission and error correction approaches as only a few applications can tolerate errors.

2.1.1 TCP Retransmission

TCP is a reliable protocol for transferring data, which utilizes a classical approach to recover failed packets. The standard method of handling errors or losses in TCP is by using retransmissions. The TCP protocol specifies a checksum field in its header presented in the Figure 2.1 to validate the integrity of received packets. It is computed by the sender using the whole packet and the remaining parts of the TCP header. After receiving a packet, the recipient computes the checksum using the entire packet and TCP header. If the computed checksum is zero then the packet is error-free (intact). Otherwise, the packet is erroneous and the receiving client requests retransmissions until it receives the packet without any error. In Figure 2.2, we illustrate this behaviour for a single retransmission. However, this approach leads to a high overhead on the network especially when the error rate is high. This leads to a decrease in effective network bit rate and performance.

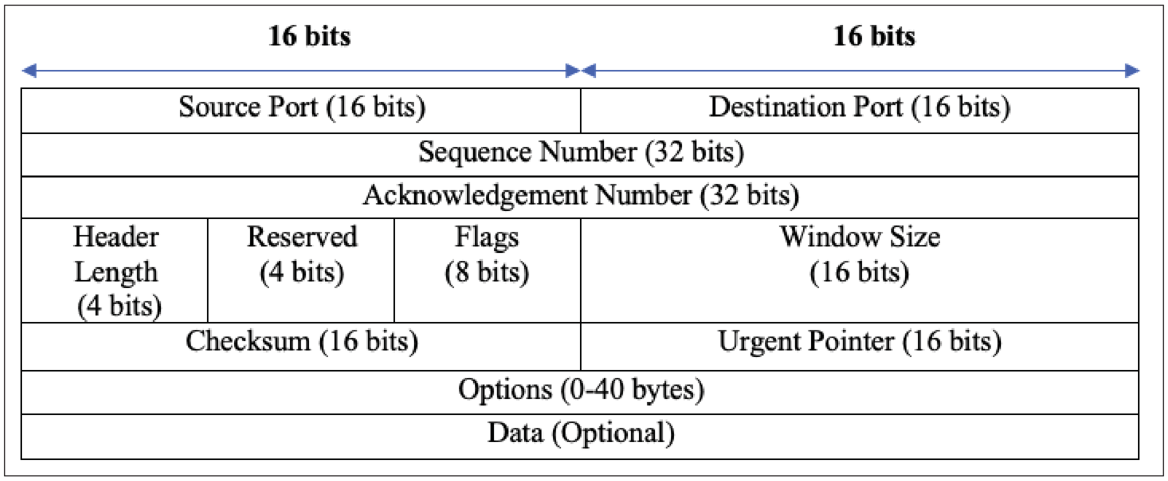


Figure 2.1 TCP header details

Table 2.1 presents an example of the original and received packets for single retransmission. In this example, the first column presents various packets, columns 2 to 9 present the packets' bits, and the last column, called "EF (error flag)", presents if the packet is erroneous (EF=1) or error-free (EF=0). Rows 2-4 present the received packets for different transmissions, row 5 presents the recovered packet, and the last row, called status, provides the status of the recovery process (T if the value is true and W if wrong). In this example and those in this thesis, the packets contain 8 bits for simplicity but the concepts extend to larger packets. As shown in the 2nd, 3rd and 4th rows representing respectively the original data, the received transmission and the received retransmission packets, after receiving the Xmit-1 packet (first retransmission), the TCP checksum will indicate to the receiver that the received packet is erroneous (EF = 1). Therefore, the first retransmission will be requested and provided by the sender. If the first received packet retransmission is still erroneous, the checksum will indicate it to the receiver and another retransmission will be requested. Otherwise, as illustrated in Table 2.1, the checksum will indicate that the received retransmitted packet is error-free (EF = 0) and it will be transferred to the application layer.

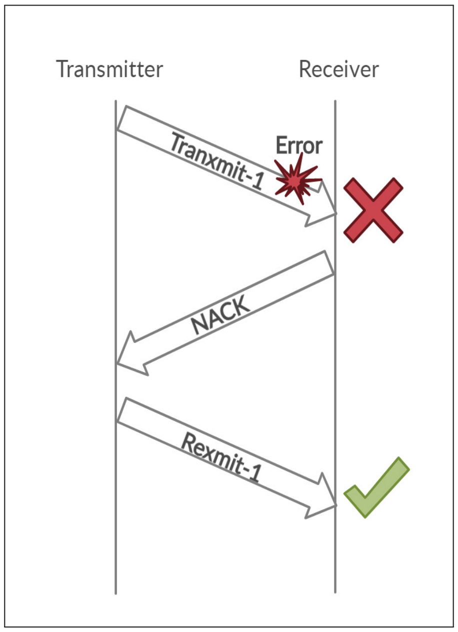


Figure 2.2 TCP data transfer
(With single retransmission)

Table 2.1 TPC protocol performing single retransmission of a packet

Packet (bits)	8th	7th	6th	5th	4th	3rd	2nd	1st	EF
Original	1	0	1	1	1	1	1	1	-
Xmit-1	1	0	1	1	0	1	1	1	1
Rexmit-1	1	0	1	1	1	1	1	1	0
Recovered	1	0	1	1	1	1	1	1	-
Status	T	T	T	T	T	T	T	T	-

Table 2.2 TPC protocol performing double retransmissions of a packet

Packet (bits)	8th	7th	6th	5th	4th	3rd	2nd	1st	EF
Original	1	0	1	1	1	1	1	1	-
Xmit-1	1	0	1	1	0	1	1	1	1
Rexmit-1	1	0	1	1	0	1	1	1	1
Rexmit-2	1	0	1	1	1	1	1	1	0
Recovered	1	0	1	1	1	1	1	1	-
Status	T	T	T	T	T	T	T	T	-

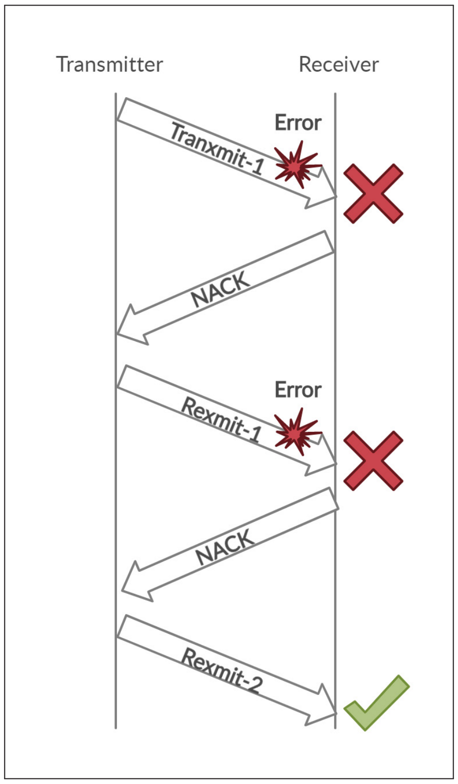


Figure 2.3 TCP data transfer
(With double retransmissions)

Illustrated in figure 2.3 and Table 2.2, if there is an error on n-th received retransmission, the (n+1)-th retransmission will be required until the checksum confirms receiving the correct and error-free packet. In such a case, asking for retransmission may take a long time because of a bad network situation, which causes errors in the received packets. As we are aware, this scenario slows down the communication and transmission process and is not appropriate for real-time applications over wireless networks. Even for non-realtime applications, it degrades the network throughput and also increases delays.

2.2 Error Correction Techniques

As discussed, although the classical retransmission method is the most reliable packet failure recovery method, it is not a proper technique in terms of bandwidth and energy. Therefore, error correction and packet recovery methods have been introduced to help decrease the negative effect of retransmission on wired and especially wireless networks. Error correction and packet recovery methods can be divided into four main groups as:

1. Error Correction Code (ECC).
2. PPR.
3. Correlation-based partial packet recovery.
4. Hybrid techniques.

We summarize the methods of these four main groups in the next subsections.

2.2.1 Error Correction Codes

The error correction codes, also known as FEC or channel coding, add redundant data to packets to be used by the receiver to retrieve the corrupted packets and correct the errors that occurred. The method can be further divided into two groups (Irianto *et al.*, 2019; Khan *et al.*, 2016): simple codes, which have a low computational overhead along with higher communication overhead that makes them proper for wired networks, and efficient codes, which include complex algorithms, bringing high computational overhead and less communication overhead, therefore well suited for wireless networks (Khan *et al.*, 2016). The FEC method causes a considerable computational overhead and also increases the bandwidth requirements. Hence, since the bandwidth overhead may cause an overall decrease in wireless resource efficiency, the amount of redundancy to be introduced must be set very carefully so as to maximize the resource efficiency.

2.2.1.1 Simple Codes

The simplest error correction code is the repetition code that repeats the message or bits multiple times over the transmission channel. By using repetition code, the receiver can reconstruct the correct message either by using majority voting over each bit or simply selecting a message that occurs most often. It is very simple and straightforward to implement but it causes high communication overhead (Irianto *et al.*, 2019; Khan *et al.*, 2016). Another simple code is the multidimensional parity-check code (MDPC), which is more robust and provides higher data rates at almost no extra computational cost in comparison to the repetition code. The idea is to arrange the data bits into a matrix and calculating the parity information for each row and column of the data matrix (Irianto *et al.*, 2019; Khan *et al.*, 2016). An example of a simple code is hamming code, which can correct a single-bit error and also detect two-bit errors. It works by creating a set of 3 parity bits for 4 bits of data. Those 3 redundant parity bits help in detecting and correcting single-bit errors in the data (Irianto *et al.*, 2019; Khan *et al.*, 2016). The repetition codes, MDPC, and Hamming codes are very simple computationally but since they cause overhead, they are not scalable and thus not a feasible option in wireless communications (Khan *et al.*, 2016).

2.2.1.2 Efficient Codes

Since wireless networks are more subject to interferences and other impairments than their wired counterparts, they incur higher error bit rates and necessitate using more complex error correction codes than simple codes. These codes include the following (Khan *et al.*, 2016): Reed-Solomon codes, Low-Density Parity-Check codes (LDPC), Raptor Codes (Khan *et al.*, 2016).

Reed-Solomon code is one of the first and most notable codes among efficient codes. Its encoding scheme, which enabled it to recover data subject to both multiple random errors in symbols/blocks and burst errors, where a sequence of symbols/blocks is lost, is unique. The Reed-Solomon code views the data to be encoded as a polynomial (Khan *et al.*, 2016). Lately, LDPC has been introduced as new efficient code, which is replacing Reed-Solomon codes in many applications because of its improved performance. Mostly used raptor code is a method belonging to LDPC (Khan *et al.*, 2016).

Since error correction codes have been extensively studied in the literature and since they lead to a constant overhead which is undesirable in reliable environments while not guaranteeing the delivery of the packet in more severe conditions, we will not consider them further in this thesis. Indeed, as mentioned earlier, we are looking for a solution which doesn't lead to overhead in reliable networks or slightly unreliable networks (packets with very few errors) while being able to always deliver packets reliably. Furthermore, such solutions require another processing layer between TCP and the data-generating application to manage the error correction codes. To some extent, it breaks the compatibility with the TCP standard communication as both client and server would have to implement such modifications. Nevertheless, some of the methods developed in this thesis could be combined with error correction codes.

2.2.2 Partial Packet Recovery

In wireless networks utilizing the TCP protocol, when a packet is received with corruption at the receiver side, it is usually discarded and retransmission is being required. This might lead to frequent retransmissions based on the unreliable channel conditions of wireless networks compared to the wired networks. A method called Partial Packet Recovery (PPR) has been introduced to help the described situation by utilizing the corrupted packets instead of ignoring them and correlating them with the other copies of the corrupted packet, delivered as retransmissions (if retransmissions are also corrupted) to correct the occurred errors (Irianto *et al.*, 2019). The first PPR method was proposed in 1977 and since then, many researchers studied proper PPR methods for error correction and packet recovery applications. The PPR method performs better in wireless networks compared to other proposed methods (i.e., automatic repeat request (ARQ), forward error correction (FEC), and Hybrid-ARQ (HARQ)) because wireless networks have unstable link condition and packet losses are the common case in these networks.

The partial packet recovery approaches are divided into two classes including as follows (Irianto *et al.*, 2019):

- PPR with PHY-layer information (PLI). The main concept of this method is based on using the soft values from the physical layer. In other words, upper layers use the physical layer hints (confidence measure of 0-1 computed on the received data by physical layer) for the received packets to decide which bits should be retransmitted. This process is called soft-decoding and it can distinguish among an intact or erroneous bit of the received data stream in the physical layer and sends its confidence values to the upper layers for sending a retransmission request to sender only for the erroneous parts of a packet (Irianto *et al.*, 2019). These methods have been proposed but they are not compatible with the real-world wireless network deployments because they need to modify the physical layer (and the hardware) as well as other higher layers to achieve the aforementioned functionality (Jamieson & Balakrishnan, 2007).

- PPR without PHY-layer information (PLI). The main concept of this category is optimizing the link layer to recover the partial packets and skip from relying on soft values of the physical layer because a high computation is required to get the soft values and also currently, it is impossible to get the soft data from the physical layer without modifying the standard protocol stack. In contrast, implementing the PPR with PLI is easily possible by utilizing some of the commercial wireless cards. Most of the proposed methods under this category have been implemented in actual 802.11 networks with standard commercial Wi-Fi hardware (Irianto *et al.*, 2019).

Consequently, the PPR without PLI is the most realistic approach for error correction and packet recovery. We will discuss, in the following subsections, some of the PPR methods without PLI, which are most relevant and can be counted as inspirational methods for our proposed methods in this thesis.

2.2.2.1 Maranello and ZipTx Techniques for Partial Packet Recovery

In Han *et al.* (2010), when a transmitted packet is received by a Maranello supporting device, it is divided into 64-byte blocks[2] by the receiver. Then, a separate checksum for each block is computed by the receiver. After computing these checksums, if the received packet contains erroneous blocks (checksum value is not 0), the checksums of the blocks are added to a NACK and sent back to the sender. The received packet, which has errors, is buffered on the receiver

[2] The last block may be smaller.

side waiting for the sender to transmit correct blocks. While the sender is waiting to receive an ACK for the transmitted packet, it receives the negative acknowledgment (Han *et al.*, 2010). Therefore, the sender computes the Fletcher-32 checksum, which is more efficiently computed on the wireless card's microprocessor, for the original packet and compares it with the received checksum (associated with the NACK) from the receiver side. Then, the sender sends a repair packet containing only the blocks of the original transmission that were corrupted. Once the repair packet is received correctly, the receiver informs the sender by sending a normal 802.11 ACK. The sender devices, which do not support the Maranello strategy, will not recognize the negative acknowledgment and retransmit the corrupted packet as normal TCP (Han *et al.*, 2010). Also, the receiver devices, which do not support the Maranello strategy, cannot inform the corrupted packet by sending a NACK and causes a Maranello sender to retransmit the full packet after the timeout. If NACK is lost in the Maranello strategy, the sender will retransmit the full packet as in 802.11 and if the retransmission has errors, the receiver will send another NACK. The characteristics of the proposed Maranello strategy are as follows (Han *et al.*, 2010):

- Requires no extra bits for intact packets. No additional error checking information, beyond the existing CRC-32, is added to normal packets.

- Reduces recovery latency. By using the time reserved for positive acknowledgments, Maranello ensures that recovery latency is smaller than the retransmission time.

- Compatibility with existing 802.11 standard.

- Incremental deployability on existing hardware. The Maranello protocol can be deployed for users just by updating their firmware.

The size of a normal ACK frame is 14 bytes but, a Maranello NACK frame, based on 64-byte blocks, is at most 96 bytes longer than an ACK frame (4-byte checksum for each block, 24 blocks maximum). Compared to other partial packet recovery protocols, Maranello does not need to send significantly larger repair packets. For repairing corrupted bits, all of the repair protocols must send significantly more repair bits. For the traces associated with low BER, Maranello needs more repair bits than other repair protocols. For low BER, the ZipTx method in Lin *et al.* (2008) can retransmit fewer bits because Reed Solomon works well when there are few bit errors.

2.2.2.2 Unite Technique for Partial Packet Recovery

Unite is a software-only partial packet recovery framework (Xie *et al.*, 2011). Since the EC-based and block-based PPR approaches are not mutually exclusive and can complement each other in many ways, a combined approach may achieve a better performance than each of the individual approaches. The proposed Unite method supports both the EC-based and block-based methods. The key features of this method are as follows:

- It supports three repair methods according to the number of occurred errors in each packet.
- It employs an error ratio estimator, called AMPS (Automated Mounting and Positioning System), and estimates the number of occurred errors in each partial packet. The obtained information of AMPS guides the sender to select the appropriate repair method according to the number of errors.
- It may operate under configurable CPU and power constraints.

It supports three repair methods according to various conditions. When the number of errors is low, it uses Targeted Error Correction (TEC), which sends parity bytes only for the targeted erroneous blocks. If the number of occurred errors is high, it uses Holistic Error Correction (HEC), which spreads occurred errors in the packet using interleaving (spreads the error evenly across all codewords and copes with burst errors because, without interleaving, there might be many errors in one received code block while very few in others). Then, it sends enough parity bytes to correct errors in each codeword. If the error correction code decoding exceeds the computational capacity of the CPU, Unite may resort to block retransmission. The packet transmission procedure is as follows (Xie *et al.*, 2011)[3]:

[3] The sender sends only the data packet in the initial transmission

- If the transmitted packet is received and passes the checksum test, it is delivered to the upper layers immediately and the transmission process of that packet finishes.

- If the transmitted packet is erased (no information is received or the header is corrupted), detected by the receiver according to the absent sequence number, the sender will be informed and may retransmit the packet up to three times until the packet is received correctly or becomes a partial packet.

- If the packet is received partially, the receiver runs the error estimator to determine the number of errors in the packet. It also divides the packet into blocks and calculates the checksum for these blocks. Then, the receiver sends feedback to the sender containing the sequence number, the AMPS samples, and the block checksums for the partial packets. After receiving feedback from the sender, it runs the error estimator to find the number of errors by comparing local block checksums with the received block checksums to find the corrupted blocks. The sender selects a repair method and sends the repair data for the partial packet.

When the receiver receives the repair data for a partial packet, it attempts to repair the packet as follows (Xie *et al.*, 2011):

- If the repair succeeds, the packet is delivered to the upper layers, and transmission of this packet finishes.

- If the repair fails, if the packet is repaired with block retransmission, it will be repaired with block retransmission again for a maximum of two-times or if the packet is repaired with HEC or TEC, it will be repaired with block retransmission method for a maximum of three times.

In wireless communications, since errors tend to be clustered in some locations, there can be many errors in one received code block while a few in others. So, an interleaving procedure is used, in which each byte is relocated to a random location based on a random permutation. This causes clustered errors to be spread evenly across all codewords. On the receiver side, a packet undergoes a de-interleaving procedure, the reverse of the interleaving, in which the bytes are mapped to their original locations. According to the estimated number of errors by the error estimator, a repair method can be selected among three repair methods. The simplest repair method is blocking retransmission, in which the sender retransmits the corrupted checksum blocks and all corrupted packets can be repaired with the block retransmission method. In HEC repairing method, the sender sends parity bytes for the code blocks. This repairing method is used for packets with a high number of errors (the max threshold of the number of errors is 100). The TEC repairing method is targeted for packets with very few errors clustered in a few blocks (The max threshold of the number of errors is 15 and the number of corrupted checksum blocks should be fewer than three blocks) (Xie *et al.*, 2011). According to their experiment results, the proposed Unite scheme outperforms existing methods (PPR, SOFT, ZipTx, MIXIT, Maranello) for PPR.

2.2.2.3 The PRAC Partial Packet Recovery Scheme

The packetized rate-less algebraic consistency (PRAC), a partial packet recovery technique, leverages the information contained in partial packets and reduces the number of retransmissions, which results in higher throughput and energy efficiency (Angelopoulos *et al.*, 2017). The idea is to use an encoding and decoding process to protect the packets. Therefore, the sender encodes a packet using a matrix of coefficients and sends the encoded packet to the receiver side along with the coefficient matrix (coefficient matrix can also be locally produced at the receiver side). At the receiver side, the receiver tries to retrieve the original packet by decoding them using the coefficient matrix. This method does not use any physical soft information, multiple CRCs and pilot bits within a packet, detailed feedback information, and additional redundancy enabling error estimation codes (Angelopoulos *et al.*, 2017). Instead of that, in the PRAC technique, the transmitted information is encoded using a rate-less linear cross-packet code and correct information is harnessed from partial packets making use of PRAC's algebraic consistency rule (ACR) check. The main features of the PRAC scheme are as follows (Angelopoulos *et al.*, 2017):

- It does not require physical soft information on the receiver side to be exposed to higher layers for the packet recovery. So, it can be easily deployed in current wireless networks.
- PRAC's encoding process incurs no transmission overhead for correctly received packets.

- PRAC requires minimal feedback information since it can operate with only a notification of completion.

- The computational requirements of PRAC's recovery algorithm can be easily adapted to the available resources, balancing the recovery performance with the algorithmic complexity.

PRAC sits between the data link and network layers. Its core component is a cross-packet random linear rate-less code, which enables its optimal operation without the knowledge of the channel quality. Since, in mobile networks, a rate adaptation mechanism might be inefficient to track changes of a wireless fast-varying channel, the rate-less code, which does not require prior information of the channel quality, is used in the PRAC mechanism (Angelopoulos *et al.*, 2017). Its encoding scheme exposes no fixed overhead to any transmitted packets and doesn't modify or increase the feedback information. On the transmitter side, because the encoding process is performed on groups of packets when a packet arrives from the network layer to the data link layer, it is buffered and processed in batches before being transmitted. The approaches mainly based on erasure codes are applied to packets to encode them. To detect a suitable encoding code for the PRAC scheme, some features are noticeable including being rate-less, performing well over small block lengths, and having zero coding overhead, which are the features of random linear codes. So, random linear codes are selected as PRAC's core coding components. The encoding process transfers k initial packets into k coded ones by a matrix multiplication, performed over finite field operations. Every packet is associated with a set of coefficients (used for the encoding process), which can be conveyed with the packet transmission or locally produced at the receiver side by a random generator in sync with one of the transmitter. In PRAC, it is assumed that the sets of coefficients are locally produced at the receiver side and are error-free. Consequently, when the channel condition is good, only k packets are transmitted and no fixed overhead is introduced by PRAC. But when the channel quality drops and additional packets are requested by the receiver, a new set of coefficients is generated and a new coded packet (coded by using a new set of coefficients) can be created and transmitted by the physical layer (Angelopoulos *et al.*, 2014; Angelopoulos *et al.*, 2017). At the receiver side, while receiving a new coded packet from the sender, its CRC status is checked and it is properly buffered as valid or partial, along with the remaining packets of its batch. PRAC's recovery process at the receiver side is column-based and there are two main steps in this process:

- Identifying correct or erroneous packets.
- Running correction process for erroneous packets.

After receiving a packet on the receiver side, while ACR detects an erroneous column, PRAC's correction process will be initialized to correct its erroneous columns. To recover the erroneous packets, the recovered packet is generated by using the received packet and their inverted coefficient matrix. The recovered packet is re-encoded by using coefficients. Then, the re-encoded packet is compared to the received packet. If they are equal then no error has occurred and the packet is consistent with high probability. If not, PRAC's correction process is triggered. In this process, a reduced-complexity search algorithm attempts to identify the correct symbols by examining first symbols with minimum hamming distance from the received ones and setting a limit on the maximum number of trials. This is performed until the ACR check for the specific column is satisfied. At the end of the correction process, the CRC status of the partial packets with corrected symbols is updated before the recovery algorithm processes the next column. This causes a reduction in the average searching time and a trade-off between recovery performance and processing time, respectively (Angelopoulos *et al.*, 2014; Angelopoulos *et al.*, 2017).

In Angelopoulos *et al.* (2014); Angelopoulos *et al.* (2017), the researchers compared their proposed PRAC scheme with the baseline ARQ scheme, which discards partial packets and retransmits the erroneous packets, and a genie-aided ideal HARQ scheme called iHARQ, which exploits partial packets by combining them with their retransmitted copies. Their experimental results demonstrate an average throughput improvement of 35% compared to a baseline ARQ scheme discarding partial packets and achieving an average throughput gain of 13% and 34% in high PER links against the iHARQ scheme.

2.2.2.4 Random Linear Codes for Exploiting Partial Packets

In Mohammadi *et al.* (2015), the authors proposed a method, which exploits the partial packets by solving a set of standard sparse recovery (SR) problems. There are some advantages for their proposed scheme including (Mohammadi *et al.*, 2015):

1. In a broadcast channel, it can directly be applied as a random linear network code (RLNC), which has been used to improve the resource utilization in wireless networks.
2. The main computational complexity is at the receiver side.
3. No cross-layer information or bit-level soft information is required.
4. No extra overhead or loss of bandwidth is imposed on the system in comparison with the conventional random linear code (RLC).
5. The receiver can be tuned to achieve a specific trade-off between computational load and performance in a systematic and tractable way.

In this method, the transmitter sends a packet at each time instance t. When the packet is received successfully at the receiver side, the receiver notifies the transmitter by an acknowledgment message. All of the packets are encoded by an error detection code such as a CRC. When receiving a packet at the receiver side, after performing FEC recovery at the physical layer, the receiver decodes the received packets and looks at their check sequence to verify the integrity of the packets. The packets with inconsistent checksum contain bit errors and are referred to as partial packets. Since there might exist several correct segments in a partial packet, they can be used to recover the whole packet. A transmitter can perform cross packet RLC on the groups of g packets, which is referred to as a generation size of the rate-less code. At each time instance t = i of RLC process, g packets of length l are combined by a random coding vector c containing g random elements to form a new packet of the same length. The encoded packets are transmitted on a wireless channel. Depending on the channel quality, some parts of the received packets are corrupted by channel errors. The receiver will be able to recover the corrupted packets only when it receives sufficient error-free packets Mohammadi *et al.* (2015).

In Mohammadi *et al.* (2015), the authors design a systematic RLC, which means that the first g packets are identical to the original packets. Using a systematic RLC facilitates the procedure. While using a systematic code, when the channel quality is desirable, the packets can be correctly received with high probability. Therefore, the receiver doesn't have to wait for receiving enough independent packets and it can notify the transmitter when it receives g packets. This causes saving resources such as energy and throughput.

2.2.2.5 Other Partial Packet Recovery Approaches

There are many introduced approaches for partial packet recovery and improving wireless network performance. The discussed approaches in the previous sections are the most important ones in this area. There are also other approaches, but since most of them use confidence information (hints) of the physical layer, we do not mention them in separate sections. As an example, a method by the name of a smart pilot approach, which aim is avoiding redundant transmissions, is proposed (Qi *et al.*, 2015). This approach incorporates two novel ideas including an adaptive hard pilot and a reliable soft pilot. Adaptive hard pilots refer to the bits, which are predefined and inserted into the information block before encoding. They are devised to be pre-known at both sender and receiver side and reliable soft pilots are the extracted confidence information of the physical layer with the high confidence level. The incorporation of hard and soft pilots provides recovery of more partial packets, reduction of the number of the retransmissions, reduction of the number of the retransmitted bits, and improving the throughput of the network. The drawback of this method, which causes it not to be applicable in real wireless networks, is using physical layer hints (Qi *et al.*, 2015).

2.2.2.6 Correlation-based Partial Packet Recovery

Using the multiple corrupt copies of the same packet is the key algorithm for the correlation-based partial packet recovery approach. In this section we provide some proposed techniques under the category of correlation-based partial packet recovery.

2.2.2.6.1 Extended-ARQ Mechanism

The proposed extension mechanism for the ARQ scheme, known as EARQ, presents the idea of combining multiple copies of a unique packet in order to reconstruct it without asking for extra retransmissions. Therefore, upon receiving the erroneous retransmission of an erroneous packet, the EARQ technique uses XOR to locate the bit errors and if the errors are not in the same locations in all copies, another retransmission will be required and if the second retransmission is still erroneous, the copies will be combined to reconstruct the main packet and remove the bit errors. Otherwise, other retransmissions will be required until being able to recover the intact packet (Yli-Juuti *et al.*, 1998). The proposed method in Sindhu (1977) proposes to use a block-by-block mechanism for error correction in EARQ in the way that the error correction and packet recovery process only apply on blocks instead of the whole packet, which results in retransmitting only the blocks with errors instead of retransmitting the entire packet.

2.2.2.6.2 Multi-Radio Diversity Mechanism

In Miu *et al.* (2005), the author proposes to utilize the spatial diversity of multiple receivers and packet combining mechanism for error correction and packet recovery. Therefore, there is a need to have multiple APs (access points) for uplink and multiple antennas for downlink on the receiver side to utilize the immunity of independent antennas to reduce the risk of error occurrence. Therefore, there are multiple APs including overlapping coverage and listening on the same radio frequency, which provides alternative communication paths for data delivery. The introduced method gathers the packets and combines them using the frame combining algorithm by providing a list of different possible combinations of the received packets and performing CRC check to detect the correct packet among the combination candidates (Miu *et al.*, 2005).

2.2.2.6.3 Simple Packet Combining (SPaC)

In Dubois-Ferriere *et al.* (2005), the proposed SPaC method relies on combining two or multiple copies of an erroneous packet to build the original one. According to Dubois-Ferriere *et al.*

(2005), once a received packet is erroneous, according to the receiver's feedback (ACK), the sender alternates the plain and parity packets as retransmitted versions. It transforms some packets into parity packets (with the same length as the plain packets) with a systematic, invertible block code before transmission (the original bits can be recovered from an error-free parity packet). Once the receiver receives an erroneous packet, it first buffers it and waits for the next coming packet (the corrupt packets will be discarded after a dedicated timeout). After receiving the next packet, they use a merging scheme and CRC check for retrieving the erroneous packet. For multiple corrupted copies, based on the type of the first copy (plain or parity), it performs a merging or decoding processes over the received corrupted packet copies. Then it checks the CRC for the combined result and if it passes the CRC check, the packet will be delivered to the upper layers as the recovered packet. Otherwise, the packet receiving and combining process will continue until reaching N_{max} number of copies for an erroneous packet (Dubois-Ferriere *et al.*, 2005). This technique is trying to avoid increasing the number of transmitted bits by a factor of two by the FEC method. The idea of combining multiple packets was an inspiration to one of our proposed methods and we aimed to develop this idea and provide a novel method in partial packet recovery, which also brings a higher error correction and packet recovery performance.

2.2.3 Hybrid Techniques

In poor channel conditions, the error correction codes are not sufficient to recover corrupted packets. So, a mechanism is introduced to improve transmission performance while using error correction codes that apply error corrections combined with retransmission as a fallback. This mechanism is called Hybrid-ARQ or simply HARQ. HARQ mechanism combines FEC and ARQ, in which FEC works as the primary mechanism to correct errors, while ARQ is used as a fallback mechanism. We know that in normal ARQ, the parity bits are added to the original data to detect errors, but HARQ can omit parity bits (CRC) if error correction code is used by FEC (Reed-Solomon as an example) mechanism can detect errors. There are three versions of the HARQ method (Type I, Type II and Type III), which have their advantages and disadvantages depending on channel conditions. Different types of HARQ are used in widely adopted technologies (Khan *et al.*, 2016).

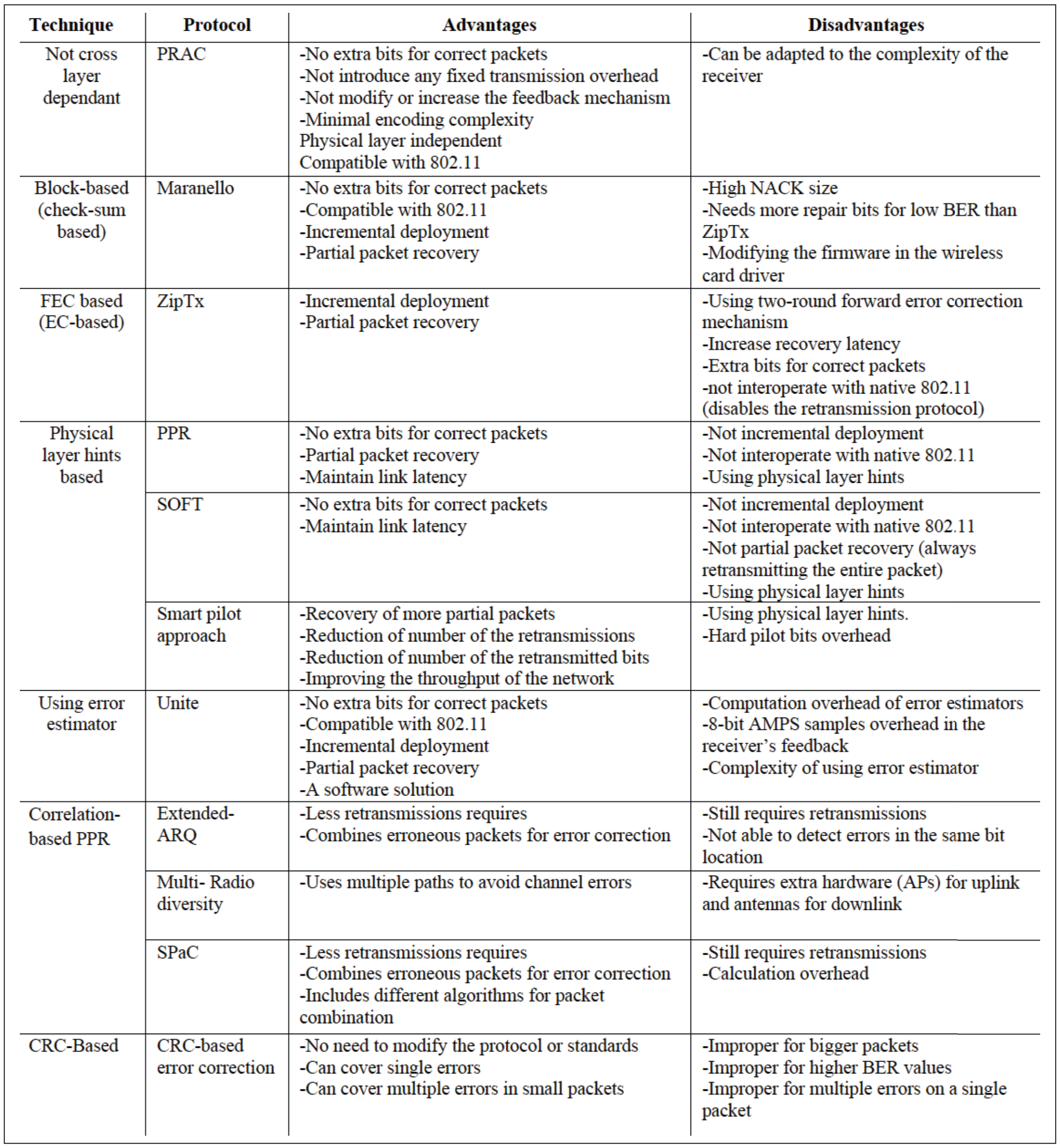
2.2.4 The State-of-the-Art CRC-based Error Correction (CRC-EC)

In Boussard *et al.* (2020a), a novel error correction and packet recovery method to correct multiple errors in data packets utilizing the Cyclic Redundancy Check (CRC) syndrome present in low layers of protocol stacks have been proposed. In error-prone networks, while applying the proposed method, a list of all possible error patterns, leading to the computed CRC syndrome containing up to a given maximum number of errors, will be generated. There might be one or several entries in the generated list, which represent the positions of the bits to be flipped to recover a CRC-valid packet. This method is the newest in this domain. Therefore, we have been chosen for this method as the state-of-the-art method for our work. According to the experiments, this method can instantly correct single errors under the condition that the protected data length does not exceed the period of the generator polynomial. It is also able to correct multiple errors in small-sized packets. The drawback of this method is that it is only trustworthy for a lower number of errors and in deleterious networking applications, it takes a lot of time and computations to recover the corrupted packets. Therefore, we must think of a method, which can be powerful enough to cover higher BER values as well.

2.3 Summary and Conclusion

In this chapter, we have presented and discussed available and existing methodologies for improving the performance of data transmissions over wireless networks using TCP while transmission errors occurred on the transmitted packets and without relying on TCP. Various approaches with specific features are proposed, which are appropriate for specific conditions with the aim of error correction and packet recovery. The methods proposed under the category of correlations-based PPR were inspirations for our proposed methods and the CRC-EC method is the latest introduced technique for error correction and packet recovery. We provide Table 2.3, which gathers the advantages and disadvantages of all introduced approaches.

Table 2.3 Comparison of the proposed approaches for error correction and packet recovery



Technique	Protocol	Advantages	Disadvantages
Not cross layer dependant	PRAC	-No extra bits for correct packets -Not introduce any fixed transmission overhead -Not modify or increase the feedback mechanism -Minimal encoding complexity Physical layer independent Compatible with 802.11	-Can be adapted to the complexity of the receiver
Block-based (check-sum based)	Maranello	-No extra bits for correct packets -Compatible with 802.11 -Incremental deployment -Partial packet recovery	-High NACK size -Needs more repair bits for low BER than ZipTx -Modifying the firmware in the wireless card driver
FEC based (EC-based)	ZipTx	-Incremental deployment -Partial packet recovery	-Using two-round forward error correction mechanism -Increase recovery latency -Extra bits for correct packets -not interoperate with native 802.11 (disables the retransmission protocol)
Physical layer hints based	PPR	-No extra bits for correct packets -Partial packet recovery -Maintain link latency	-Not incremental deployment -Not interoperate with native 802.11 -Using physical layer hints
	SOFT	-No extra bits for correct packets -Maintain link latency	-Not incremental deployment -Not interoperate with native 802.11 -Not partial packet recovery (always retransmitting the entire packet) -Using physical layer hints
	Smart pilot approach	-Recovery of more partial packets -Reduction of number of the retransmissions -Reduction of number of the retransmitted bits -Improving the throughput of the network	-Using physical layer hints. -Hard pilot bits overhead
Using error estimator	Unite	-No extra bits for correct packets -Compatible with 802.11 -Incremental deployment -Partial packet recovery -A software solution	-Computation overhead of error estimators -8-bit AMPS samples overhead in the receiver's feedback -Complexity of using error estimator
Correlation-based PPR	Extended-ARQ	-Less retransmissions requires -Combines erroneous packets for error correction	-Still requires retransmissions -Not able to detect errors in the same bit location
	Multi- Radio diversity	-Uses multiple paths to avoid channel errors	-Requires extra hardware (APs) for uplink and antennas for downlink
	SPaC	-Less retransmissions requires -Combines erroneous packets for error correction -Includes different algorithms for packet combination	-Still requires retransmissions -Calculation overhead
CRC-Based	CRC-based error correction	-No need to modify the protocol or standards -Can cover single errors -Can cover multiple errors in small packets	-Improper for bigger packets -Improper for higher BER values -Improper for multiple errors on a single packet

According to Table 2.3, the CRC-EC method is the most efficient and newest among other proposed approaches, which has been chosen as the state-of-the-art method for this thesis. Therefore, we have implemented the CRC-EC method and repeated the simulations and experiments for this method presented in the experimental results chapter.

CHAPTER 3

PROPOSED ERROR CORRECTION AND PACKET RECOVERY METHODS

In this chapter, we describe our proposed methods to handle erroneous packets. First, we introduce a method called Majority Voting (MV), which mainly relies on the majority voting process to correct the erroneous packets. The method is then enhanced by the use of the checksum to validate whether the corrected packet (or candidate packets) is valid or not. It is referred as Majority Voting with Checksum Validation (MVCV). As the next method, we introduce the Dissimilarity-based Enumeration Packet Recovery with Checksum Validation (DEPRCV), which tries to recover the packet based on an enumeration process of possible packets and validates each of them with the checksum to determine the recovered packet. Finally, we introduce a more sophisticated packet recovery method called Cross-Layer Packet Recovery (CLPR), which is a combination of multiple error correction and packet recovery strategies to provide a high-performance packet recovery solution.

3.1 Majority Voting Algorithm for Error Correction and Packet Recovery

As in elections, if the majority rule is applied to a simple choice between two alternatives x and y, then x wins if it gets more votes than y. There is a tie if they get the same number of votes. We believe in the possibility of utilizing the Majority Voting (MV) algorithm for error correction and packet recovery purposes. Accordingly, while having multiple versions of a single erroneous packet (transmission and retransmissions for a unique data packet to deliver) the proposed MV method at the receiver always chooses the value that occurs the most often among the packets for each bit position. More specifically, the method counts the number of 0-s and 1-s for each same bit position among the received transmitted and retransmitted packet versions and selects the value based on the greatest number of occurrences of a bit value. The MV algorithm has been used in Dubois-Ferriere *et al.* (2005) for the aim of error correction and the packet recovery for the first time.

An example, provided in the figure 3.1, illustrates the use of the MV algorithm for the packet recovery purposes. In Figure 3.1, the value of N and L, representing respectively the number of packets and the number of bits in a packet, are 4 and 8. There are four received erroneous versions of the original packet called transmission, retransmission-1, retransmission-2, and retransmission-3. If we perform MV on the first bit position of the received packets, starting from the leftmost bit, since there are three with a bit value 1 and one with a bit value 0, the computed value by MV for this bit position is 1. We perform MV for each bit position (8 such positions in the example) as illustrated. The bitstream on the right side (green colour) presents the output packet of the MV process. It is important to note that since a single packet is not sufficient to cast a vote and two packets cannot allow discriminating when there is a tie, the MV algorithm requires at least 3 transmissions (packet versions) to operate. Ideally, MV should be performed on an odd number of transmissions to avoid possible ties.

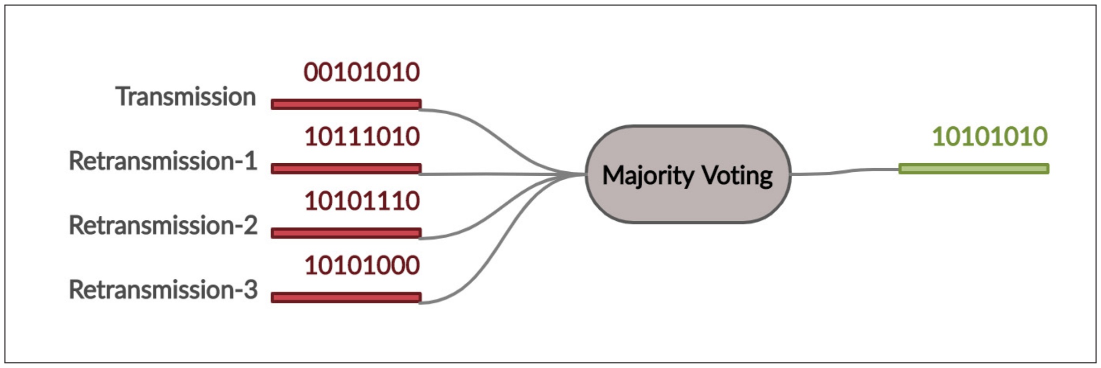


Figure 3.1 An example of the MV process over 4 packet versions of 8 bits each

Equation 3.1 specifies the decision cases for the MV process over multiple packets. If there are N transmitted packet versions of an original packet (containing the first transmissions and several retransmissions) with a length of L bits each, the MV process considers the bit value at each of the L bit positions on each of the N packets. It then chooses the repaired bit value, for each specific bit position, as the value (0 or 1), which has the highest number of repetitions among packets.

Let $\mathrm{b}_\mathrm{i}^\mathrm{t}$ be the value of the bit position i of the packet version t and MV_i be the bit value at position i for the packet generated by the MV process. The value of each MV_i can be calculated as follows:

$$\mathrm{MV}_i = \begin{cases} 1, & \text{if } \sum_{t=1}^{N} b_i^t > \frac{N}{2} \\ 0, & \text{if } \sum_{t=1}^{N} b_i^t < \frac{N}{2} \\ \text{Undefined}, & \text{if } \sum_{t=1}^{N} b_i^t = \frac{N}{2} \end{cases}, \ \forall i = 1, 2, \ldots, L \tag{3.1}$$

The chosen bit value (0 or 1) for each bit position is obtained by the votes for that bit value (The number of bit values 0 and 1 for the bit position i will be counted among all the packet versions, and the value of the highest vote wins). Therefore, the summation in the equation is dedicated to counting the number of votes for each bit value. As an example, if we consider having 4 versions of an erroneous packet (1 transmission and 3 retransmissions) with erroneous bit position k, if the value of the bit position k is equal to 1 in 3 versions, and it equals 0 in only 1 version; therefore, there will be 3 votes for the bit value of 1 and only 1 vote for the bit value of 0. In this case, the MV will successfully choose the bit value of 1 as the correct value for the bit position k. In another scenario, if the value of the bit position k is equal to 1 in 2 versions, and it equals 0 in the other 2 versions, therefore, the number of votes for both bit values 0 and 1 is equal. In such a case, the basic MV method is not able to decide the correct bit value for the position k and it gives up the packet correction scenario, which is observable in the last case in the equation 3.1. Then the basic MV process will ask for retransmissions until succeeding in recovering the erroneous packet. Note that the undefined state can only be reached when there is an even number of transmissions.

Some examples are provided in the Tables 3.1, 3.2, 3.3, 3.4 to help illustrate the application of the MV algorithm in the packet recovery process and the cases in which the MV algorithm fails to decide the value at a specific bit position (undefined case in the provided equation 3.1).

Table 3.1 An example of the MV method (one-bit error per packet and errors have occurred in different positions for each packet)

Packet (bits)	8th	7th	6th	5th	4th	3rd	2nd	1st	EF
Original	1	0	1	1	1	1	1	1	-
Xmit-1	1	0	1	1	0	1	1	1	1
Rexmit-1	1	0	1	1	1	0	1	1	1
Rexmit-2	1	1	1	1	1	1	1	1	1
MV	1	0	1	1	1	1	1	1	0
Recovered	1	0	1	1	1	1	1	1	0
Status	T	T	T	T	T	T	T	T	

Table 3.2 An example of the MV method (one-bit error per packet and errors have occurred in the same positions for each packet)

Packet (bits)	8th	7th	6th	5th	4th	3rd	2nd	1st	EF
Original	1	0	1	1	1	1	1	1	-
Xmit-1	1	0	1	1	0	1	1	1	1
Rexmit-1	1	0	1	1	1	1	1	0	1
Rexmit-2	1	0	1	1	0	1	1	1	1
MV	1	0	1	1	0	1	1	1	1
Recovered	1	0	1	1	0	1	1	1	1
Status	T	T	T	T	W	T	T	T	

The example in Table 3.1 illustrates the MV process for the case of three transmissions where errors occur at different positions in the various transmissions. The method succeeds in providing an error-free packet by exploiting data in erroneous packets to decrease the number of required retransmissions for packet recovery. In the examples provided in Tables 3.1 and 3.2, the eight-bit values for each of the packets (the original one, the first transmission (Xmit-1), the first and second retransmissions (Rexmit-1 and Rexmit-2)) are shown. Also, the output packet returned by the MV method and the recovered packet is presented in rows 6 and 7 respectively. The last row presents the status, or accurateness, of each recovered bit when compared to the original bit, i.e., if true (T) or wrong (W). The last column indicates an error flag (EF) which is 1 when the packet is tagged as erroneous by the checksum.

As mentioned, the MV method has some drawbacks and weaknesses. In the example of Table 3.2, the 4th bit of 2 versions (xmit-1 and rexmit-2) out of the 3 received versions is erroneous. Therefore, the MV process will select the wrong bit value for this bit position, which leads to a wrong packet recovery process. Therefore, when, among the received packet versions, there are more than 50% of the bit values that are wrong in a bit position, the MV process will lead to a wrong correction at that position. An additional validation is required to detect such situations and will be presented in the next subsection.

Another case of interest is illustrated in Table 3.3 where there is an even number of transmissions and exactly 50% of the bit values are wrong in various positions for the received packets. In this case, the MV result will be undefined. In such a case, the MV will fail to correct the errors and will ask for another retransmission.

Table 3.3 An example of the MV method (one-bit error per packet split equally into two positions, leading to a tie at these two positions)

Packet (bits)	8th	7th	6th	5th	4th	3rd	2nd	1st	EF
Original	1	0	1	1	1	1	1	1	-
Xmit-1	1	0	1	1	0	1	1	1	1
Rexmit-1	1	0	1	1	1	1	1	0	1
Rexmit-2	1	0	1	1	0	1	1	1	1
Rexmit-3	1	0	1	1	1	1	1	0	1
MV	1	0	1	1	0/1	1	1	0/1	×
Recovered	1	0	1	1	0/1	1	1	0/1	×
Status	T	T	T	T	ND	T	T	ND	

Table 3.4 An example of the MV method (one-bit error in the same position for all the packets)

Packet (bits)	8th	7th	6th	5th	4th	3rd	2nd	1st	EF
Original	1	0	1	1	1	1	1	1	-
Xmit-1	1	0	1	1	0	1	1	1	1
Rexmit-1	1	0	1	1	0	1	1	1	1
Rexmit-2	1	0	1	1	0	1	1	1	1
MV	1	0	1	1	0	1	1	1	1
Recovered	1	0	1	1	0	1	1	1	1
Status	T	T	T	T	W	T	T	T	

Observable in the example of Table 3.4, where the error position is the same for all the received versions of a packet (bit 4 in the provided example), MV is not able to recover the original data by itself and again, it will fail to recover the original packet. Worse, the method by itself will not even be aware that the recovered packet is wrong, leading to an unreliable communication.

In summary, as observed in the provided examples, the MV algorithm is not very efficient not reliable for correcting the erroneous packets because of the following reasons:

- First, it needs at least three transmissions of the same packet to be able to operate.
- Second, it may fail to recover the original packet when there is an even number of packet transmissions since ties may occur at some bit positions.
- Third, it cannot recover the original packet if, for a bit position, more than half of the received values are erroneous.
- Lastly, the method cannot even detect that it provided a wrongly corrected packet (previous case).

Although the first two reasons only affect the number of retransmissions, i.e., the efficiency of the communication, the last two compromises the reliability of the entire communication. It is therefore crucial to detect when the correction has failed. This will be addressed in the next subsection.

3.2 Majority Voting with Checksum Validation for Error Correction and Packet Recovery

In this section, we introduce the use of MV along with Checksum Validation (CV) to enhance the robustness of the basic MV algorithm for the purpose of error correction and the packet recovery. The addition of CV addresses the main drawbacks of using the basic MV algorithm regarding the reliability of the result. As before, we propose to use the MV method, where at least 3 versions of an erroneous packet are used to generate a correct packet. For each bit position, we select the bit value which is the most frequent for that position. The recovered packet is then validated using the TCP checksum. When there are ties, we propose to generate a list of all possible candidates for the MV algorithm and to eliminate the candidates that do not pass the CV process. If at the end of the process there is a single candidate, it becomes the corrected packet. Otherwise further retransmissions are requested.

It is important to note that the applied checksum is obtained using the same MV method and that any corrected packet candidate that doesn't pass CV must be eliminated as it cannot be the corrected packet. However, it is possible that a wrongly corrected packet will pass CV. This event is not very likely but may still happen. Even under normal TCP communications, it is possible, although not very likely, that an erroneous packet will pass TCP checksum validation.

The flowchart for the Majority Voting with Checksum Validation (MVCV) method is provided in the Figure 3.2. According to the flowchart whenever the packet arrives at the transport layer, the checksum will be checked to detect whether the packet is erroneous or not. If the packet is error-free, it will be passed to the upper layers. Otherwise, the packet will pass through the recovery phase. After performing the error detection by the checksum, each packet containing errors will be stored in a dedicated buffer along with its transmitted version number. It will allow the recovery method to have access to all versions (transmission and retransmissions) of the erroneous packets for the recovery process. For erroneous packets, retransmissions will be required until a correct packet is received or the number of received versions is higher than two (i.e., at least three versions are required). Then, the buffer containing erroneous packets will be passed to the majority voting part of the process. This MV process may return a single candidate packet or a list of candidates in the case of ties. Then, the checksum will be checked for each of the candidates to eliminate those that cannot represent the corrected packet. Finally, if at the end of the validation a single candidate remains, it is established as the corrected packet and passed to the upper layers. Otherwise, an additional retransmission is requested and the MVCV process is performed again with this additional packet. Algorithm 3.1 presents the method in the form of an algorithm.

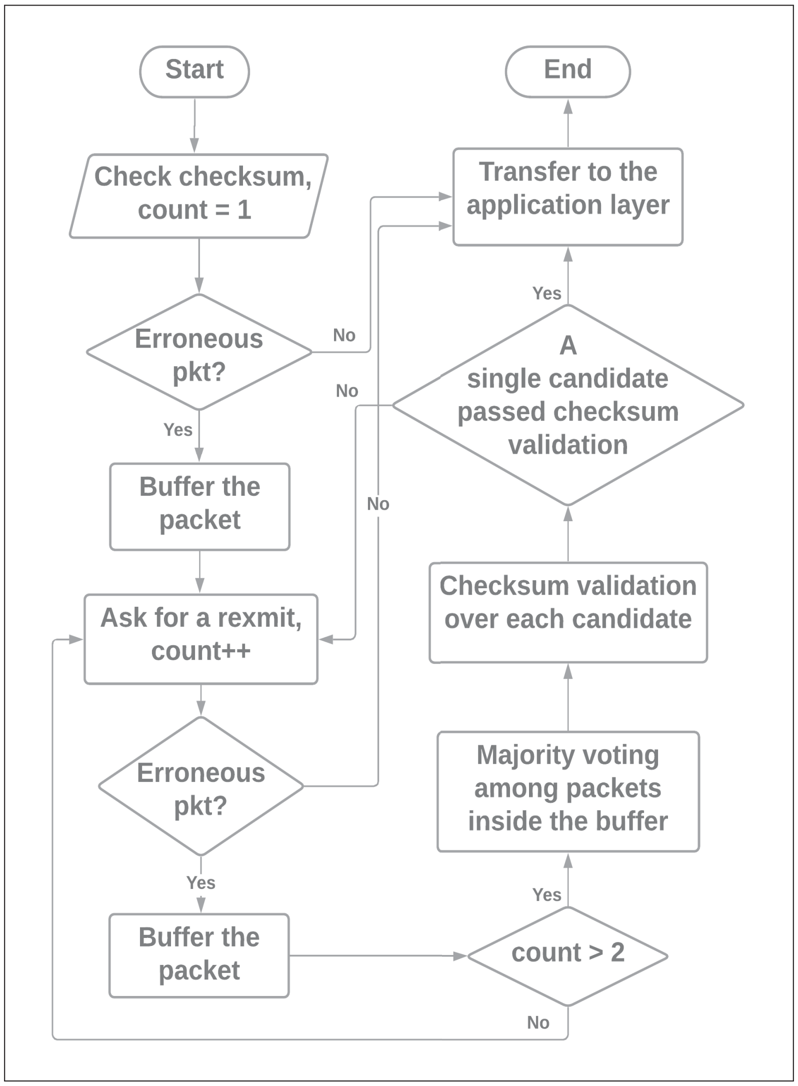

Start
Check checksum, count = 1
Erroneous pkt?
No
Yes
Buffer the packet
Ask for a rexmit, count++
Erroneous pkt?
No
Yes
Buffer the packet
count > 2
Yes
No
Majority voting among packets inside the buffer
Checksum validation over each candidate
A single candidate passed checksum validation
Yes
No
Transfer to the application layer
End

Figure 3.2 MV along with CV

This proposed method thus combines several ideas. It combines the basic MV method with that of creating a candidates' list from the ambiguity raised by ties when we have an even number of transmissions. It then uses CV to eliminate the candidates that cannot be solutions to the correction problem. Even in the case of a single candidate, the use of CV is important to validate that the solution is valid; a major problem of the MV method used alone. Thus, this proposed method is expected to not only have a dramatically lower rate of invalid solutions but to achieve this with fewer retransmissions than MV.

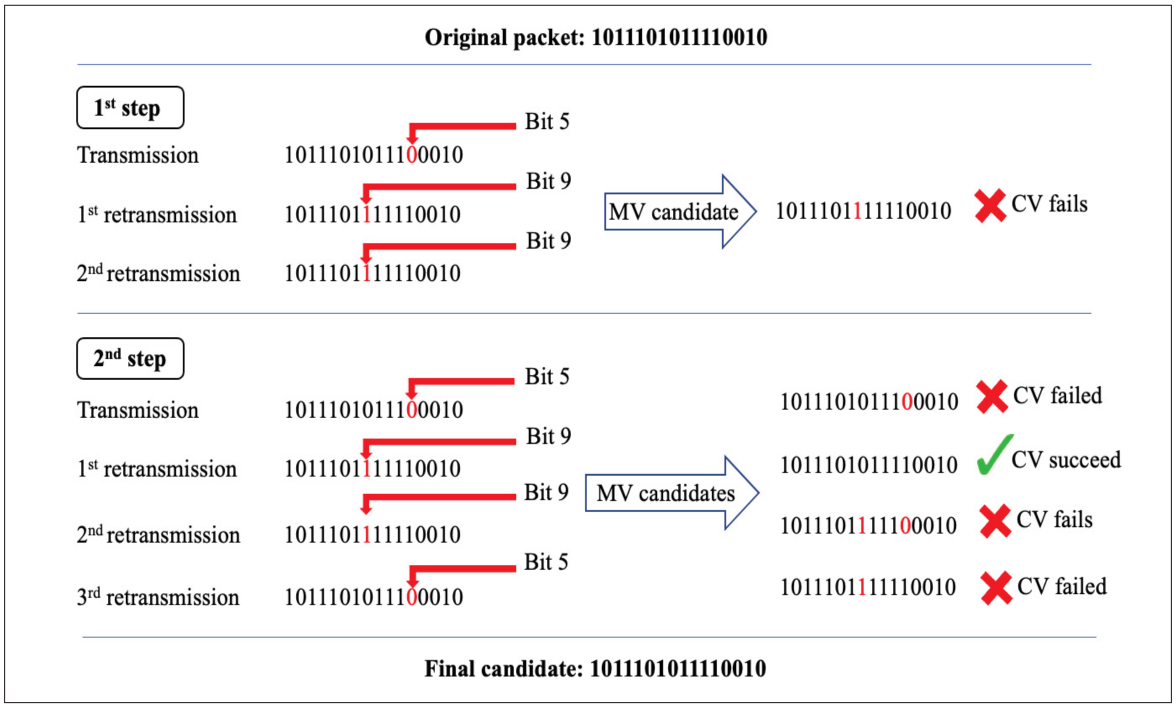

Original packet: 1011101011110010
1st step
Transmission 1011101011100010 Bit 5
1st retransmission 1011101111110010 Bit 9
2nd retransmission 1011101111110010 Bit 9
MV candidate
1011101111110010 CV fails
2nd step
Transmission 1011101011100010 Bit 5
1st retransmission 1011101111110010 Bit 9
2nd retransmission 1011101111110010 Bit 9
3rd retransmission 1011101011100010 Bit 5
MV candidates
1011101011100010 CV failed
1011101011110010 CV succeed
1011101111100010 CV fails
1011101111110010 CV failed
Final candidate: 1011101011110010

Figure 3.3 An example of MV along with CV

In the Figure 3.3, an example is provided to illustrate the introduced MVCV method. In step one, three erroneous versions of a packet are received and are passed through the MV process. Since bits at position 9 are erroneous in two versions (1st and 2nd retransmissions), the result of MV is also erroneous and it fails the CV step. As step two, another retransmission is received for the erroneous packet and again MV is performed over four versions of the packet. Since bits at position 5 are erroneous in two versions and bits at position 9 are erroneous in the other two versions, the MV method cannot decide on the bit value for these positions and it outputs a list of candidates out of these four packets. The list of candidates provided by MV will be passed to the CV process. As shown in the right part of the second step of the figure, three of the candidates fail the CV process (two of them are the same as packet versions, which failed the CV process before) and only one candidate passes the CV process. Therefore, the CV succeeded candidate will be selected as the final candidate and will be passed to the application layer as the recovered packet without any need for more retransmissions. Note the situation provided in this example occurs less often than the case where errors occur in different columns, especially for large packets. But we provide it to illustrate one of the severe cases which can happen.

The example clearly shows the benefit of generating candidates' list and then passing the candidates through the CV process in order to reduce the number of retransmissions.

3.3 Dissimilarity-based Enumeration Packet Recovery with Checksum Validation

According to the previous subsection, the improved MVCV method starts the packet recovery process only after 3 erroneous transmissions. Although this method is expected to decrease significantly the number of retransmissions compared to normal TCP, especially over the networks with relatively high BERs, it is still desirable to further reduce the number of required retransmissions before it can start produce results. Indeed, as mentioned previously, the goal of this thesis is to reduce the number of retransmissions as much as possible.

In this section, we propose another method, in which, the packet recovery process can be starting after only 2 erroneous versions of a packet (1 transmission and 1 retransmission). This packet recovery method is called Dissimilarity-based Enumeration Packet Recovery with Checksum Validation (DEPRCV). As its name suggests, it is based on comparing multiple versions of the packet and identify positions where bit values differ to generate a list of possible corrected candidates. In this method, once the receiver realizes that the received packet is erroneous, the first retransmission is requested. After receiving the first retransmission, the packet recovery process can be performed relying on these two available versions of the packet. In DEPRCV, we identify the bit positions where there are different values in the received versions (i.e., bit positions where not all the received versions have the same value). We then assume that each of these positions the value could either be 0 or 1. We enumerate a list of all possible corrected packets considering all the possibilities generated by such consideration. For instance, if there are p positions where bit values among the received packets differ, then the list comprises 2^p candidates to consider all permutations of values. The candidates in the list are then validated

Algorithm 3.1 Majority Voting with Checksum Validation

```
Input: Received packet
Output: Recovered packet
CS-check ← 0 // Transport layer checksum of the received packet initialized to 0
count ← 1
recovered ← 0
CS-check ← CV performed on Received packet
while (count < 3 and recovered = 0) do
    if CS-check = 1 then
        recovered = 1
        Output the last received packet        // Transfer to the application layer
    end
    if CS-check = 0 then
        save last received packet as number "count" in the buffer
        count ← count + 1
        ask for a retransmission
        CS-check ← CV performed on last received packet
    end
end
while (recovered = 0) do
    Call the MVCV method for the buffer including packet transmissions
    Forward the MV results to the CV process
    if CV = 1 for a single candidate then
        recovered = 1
        Output the candidate passing CV        // Transfer to the application layer
    end
    ask for a retransmission
    CS-check ← CV performed on last received packet
    if CS-check = 1 then
        recovered ← 1
        Output last received packet        // Transfer to the application layer
    end
    if CS-check = 0 then
        save last received packet as number "count" in the buffer
        count ← count + 1
    end
end
```

using the checksum as in the previous method. This method generalizes easily to any number of received packet.

The main difference between the DEPRCV and MVCV methods lies in the way the corrected bit values are established. This has an impact on the number of transmissions required to start the correction process. In MVCV, having a majority requires at least 3 transmissions while in DEPRCV, identifying different bit values only requires 2 transmissions. This permits the recovery method to start the correction process with fewer transmissions. Still the candidates are validated by CV to avoid correction errors. In the case where all the received versions contain errors at the same position, we count on the CV to detect that it did not succeed in correcting the packet. We thus expect DEPRCV to perform better than MVCV.

The figure 3.4 shows a flowchart of the DEPRCV packet recovery approach. If the first received packet is erroneous, a retransmission is requested. Assuming this packet is also erroneous, having two versions of a packet, the DEPRCV approach then compares the bit values of the two available packets. It detects the bit positions, which are different among the two packets. It then generates a list of all possible corrected packets by using all the combinations of bit values in these positions. These candidates go through a CV process where invalid candidates are eliminated. If a single candidate remains then it is transferred to the application layers. Otherwise, a retransmission is requested and the process is repeated with an additional packet.

According to the provided example of figure 3.5, if there is only one error in each version (position 5 in the first and position 9 in the second packets), the method provides all possible combinations of these two bits as {[5], [9], [5, 9]}. Then it flips the bits of these positions and it provides a list of the candidates (4 different candidates in this example). The number of candidates is 2^P, while P is the number of various positions among packets. However, in practice, N of these candidates have already been tested since they correspond to transmitted packets. Therefore, only 2^P-N of these candidates really need to be validated[4]. Then it performs the CV process on each of these candidates. As we observe, only one candidate can pass the CV step (dedicated to the green check boxes in the figure). Therefore, this candidate will be delivered to the application layer as the recovered packet. The biggest advantage of this method

[4] Here we considered that each received packet is unique. If multiple received packets are identical then we need to test 2^P-N_U candidates, where N_U is the number of received packets which are unique.

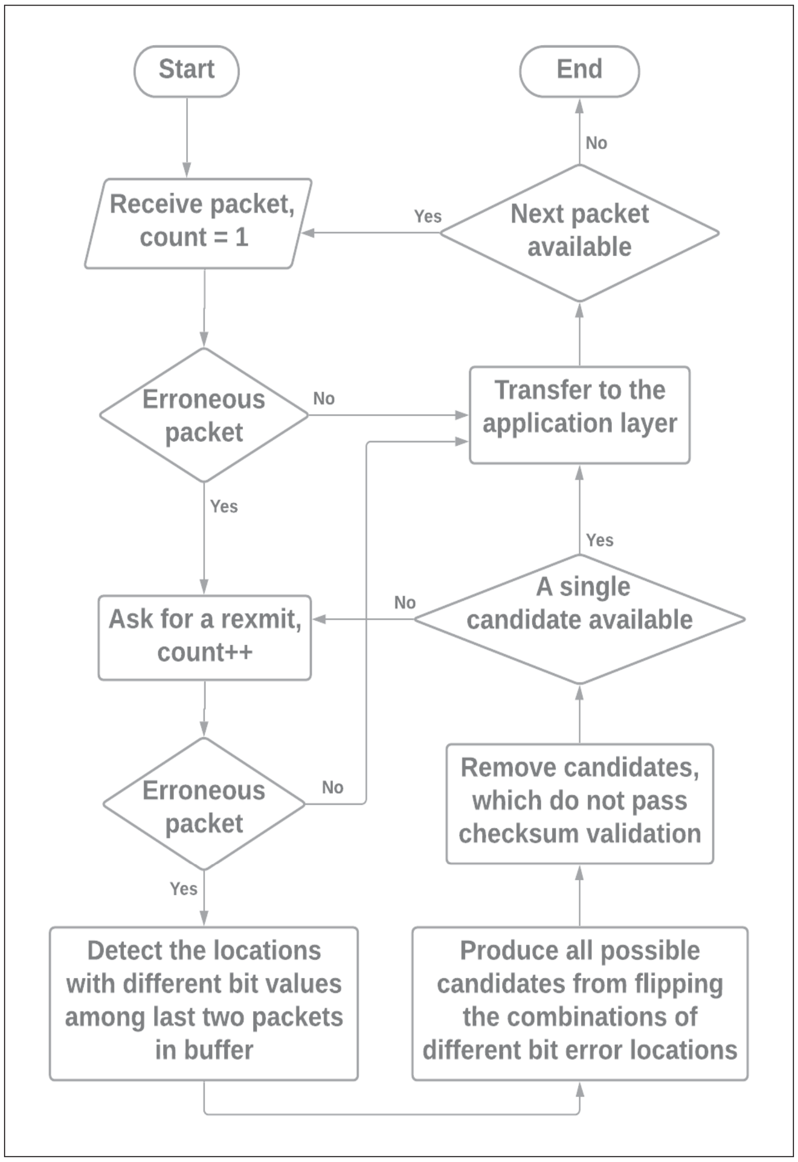

Start
End
No
Receive packet, count = 1
Yes
Next packet available
Erroneous packet
No
Transfer to the application layer
Yes
Yes
A single candidate available
No
Ask for a rexmit, count++
Erroneous packet
No
Remove candidates, which do not pass checksum validation
Yes
Detect the locations with different bit values among last two packets in buffer
Produce all possible candidates from flipping the combinations of different bit error locations

Figure 3.4 Dissimilarity-based enumeration packet recovery with checksum validation

is that it can reduce the number of required retransmissions for the packet recovery process to one. This method also does not contain complex computations. It only requires a small amount of memory to keep the candidates and to analyze them.

Algorithm 3.2 details the steps of the Dissimilarity-based Enumeration Packet Recovery with Checksum Validation method. As mentioned before, the inputs for this method are the erroneous

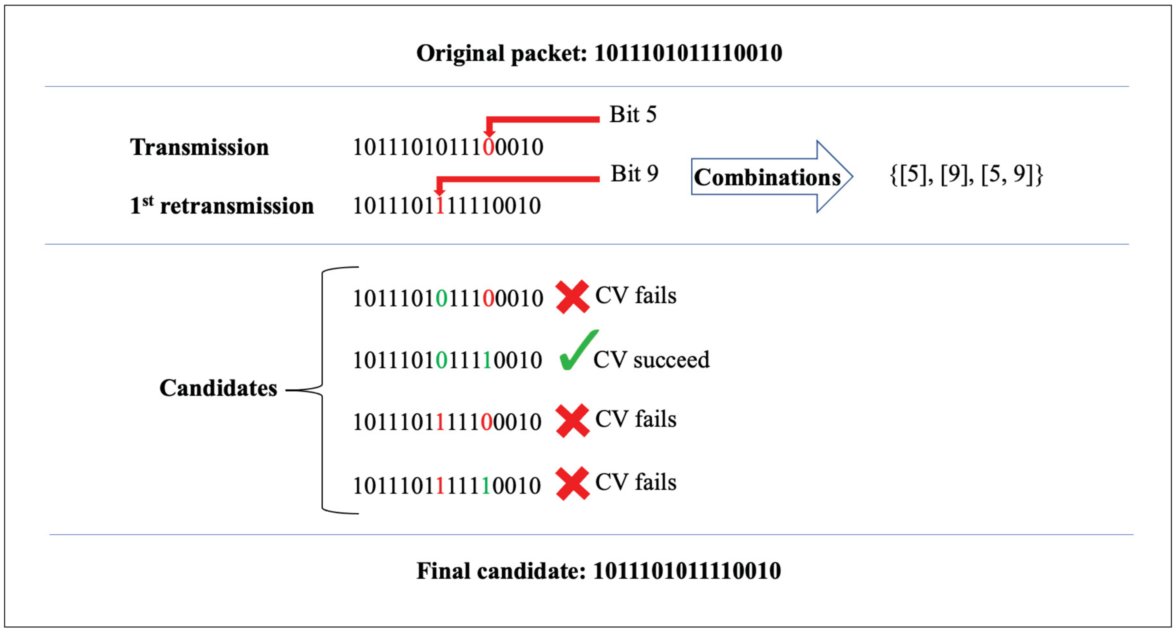


Figure 3.5 Dissimilarity-based enumeration packet recovery

packet buffer and the output is the recovered packet. Once the packet is detected as an erroneous one, after receiving its first retransmission, two versions will be going through the Dissimilarity-based Enumeration Packet Recovery with Checksum Validation algorithm to recover the original packet. It is important to note that DEPRCV-GEN in the algorithm represents the steps in figure 3.5 where, from the buffer of packets, we detect the locations with different bit values among the last two packets in the buffer. Then produce all possible candidates from flipping the combinations of different bit error locations.

3.4 Cross-Layer Packet Recovery

In this section, we propose a novel method which combines advantageously numerous error recovery methods which we call Cross-Layer Packet Recovery (CLPR). Specifically, we combine the proposed error correction and packet recovery methods presented in the previous sections with the state-of-the-art CRC-Error-Correction method presented in Boussard *et al.* (2020a) to take advantage of the strengths of each. The proposed CLPR method consists of four different

Algorithm 3.2 Dissimilarity-based Enumeration Packet Recovery with CV

```
Input: Received packet
Output: Recovered packet
Initialize an empty buffer of packets
CS-check ← 0 // Transport layer checksum of the received packet initialized to 0
count ← 1
CS-check ← CV performed on Received packet // CV is checksum validation
if CS-check = 1 then
    Output the Received packet        // Transfer to the application layer
else
    Save Received packet as number "count" in the buffer of packets
    Ask for a retransmission
    count ← count + 1
end
while (1) do
    CS-check ← CV performed on last received packet
    if CS-check = 1 then
        Output last received packet        // Transfer to the application layer
    else
        Save last received packet as number "count" in the buffer of packets
    end
    Call the DEPRCV-GEN(Input: Two latest packets in buffer of packets, Output: List
      of candidates)
    if List of candidates not empty then
        for each candidate in List of candidates do
            CS-check(candidate) ← CV performed on candidate
        end
    end
    if CS-check(candidate) = 1 for a single candidate then
        Output candidate for which CS-check(candidate) = 1        // Transfer to the
          app-layer
    end
    Ask for a retransmission
    count ← count + 1
end
```

packet corrections and recovery methods, which can recover the erroneous packets independently. The methods are as follows:

- TCP retransmissions

- CRC-based error correction (Boussard *et al.*, 2020a)

- Dissimilarity-based Enumeration Packet Recovery with Checksum Validation

- Majority Voting with Checksum Validation

Each of the mentioned sub-method has specific advantages and disadvantages and each performs well in some specific situations. Therefore, combining them is a simple yet very effective way to design a very efficient packet recovery method where the number of retransmissions and the delay are reduced in every networking situation.

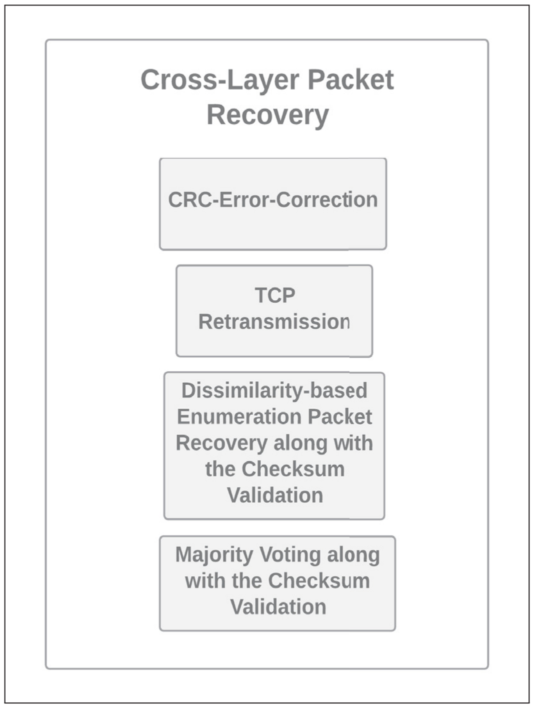

Cross-Layer Packet Recovery
CRC-Error-Correction
TCP Retransmission
Dissimilarity-based Enumeration Packet Recovery along with the Checksum Validation
Majority Voting along with the Checksum Validation

Figure 3.6 High-level cross-layer method description

The figure 3.6 illustrates a high-level view of the proposed CLPR method. At a high-level, the CLPR combines TCP retransmissions, CRC-EC, DEPRCV, and MVCV methods. The order in which the mentioned methods are applied is based on the number of transmissions they need to start operating. Since CRC-EC can start the recovery process using the transmitted packet and without any retransmission, it is the first step in the proposed CLPR. If it fails to correct the packet, a retransmission is requested and DEPRCV can stand as the second step of the packet recovery. Finally, if after a single retransmission the packet has still not been corrected, MVCV is then involved.

While applying the proposed CLPR method, the combined methods support each other to recover the packet with the least number of retransmissions. The packet recovery is performed as follows:

- In the first phase, the CLPR method relies on the state-of-the-art CRC-EC method. This method is performed first because it does not require any retransmission to perform packet error correction and the recovery process applies to the erroneous packet itself. This method works well for low BERs values. It is not suitable for high BERs because it would take a long time to generate candidates as its computational complexity increases exponentially with the number of errors considered. Furthermore, it would generate too many candidates among which we would not be able to identify the corrected one, even with CV. If this method fails to recover the packet, a retransmission is requested and again, this method is first applied to the received retransmitted version as well. If this method fails to correct the retransmitted packet, the next phase begins.
- In the second phase, the CLPR relies on the DEPRCV method for the recovery. Therefore, it forwards the buffer including two versions of the erroneous packet (first transmissions and the first retransmission) to the DEPRCV method. If it succeeds in recovering the packet, the recovery process ends. In the case that the applied methods cannot recover the erroneous packet, the second retransmission is requested and if the received retransmitted packet cannot be recovered applying the previous method or this one, the third step of the packet recovery scenario is initiated.
- As the third step, since none of the CRC-EC and DEPRCV methods could recover the erroneous packet, the buffer including the erroneous packet along with two retransmissions pass to the MVCV method as the last phase of CLPR. Again, if the packet can be recovered successfully, the packet recovery process ends. Otherwise, it restarts by requiring another

retransmission. If the retranssmitted copy also contains errors, CRC-EC, DEPRCV, and MVCV approaches try to fix it respectively. Otherwise, another retransmission will be sent and the process will continue until being able to recover the erroneous packet. We are placing the MVCV method after the CRC-EC method in order to discriminate between multiple candidates when CRC-EC yields more than one valid candidate. When this is the case, we want to select the candidate with the most frequent bit values at each position. Indeed, it is important to note that the candidate returned by MVCV is always included in the list of candidates generated by DEPRCV.

According to the presented flowchart of the CLPR method in Figure 3.7, when an erroneous packet is received on the receiver side and the CRC check fails, as the first step of the CLPR method, it passes through the CRC-EC method to be recovered using this state-of-the-art method. If the CRC-EC method succeeds to recover the erroneous packet, the recovered packet is passed to the application layer and the recovery process ends. Otherwise, the erroneous packet is stored in a buffer and a retransmission is requested. When receiving the retransmission, the CRC check is performed on the retransmitted version as well to check the errors over it. If the correction succeeds, the corrected packet passes to the application layer and the recovery process ends. Otherwise, the CRC-EC method will be performed over the received retransmission. If the CRC-EC succeeds to recover the packet, the latter is passed to the application layer and the recovery process ends. Otherwise, the first received retransmission is inserted into the buffer and the buffer, including two versions of the erroneous packet (transmissions and the first retransmission), is passed to the DEPRCV method. As mentioned in the section 3.3, the DEPRCV method tries to recover the erroneous packet relying on only two transmitted versions. If this method succeeds to recover the packet with outputting a single candidate, the recovery process ends and the recovered packet (single outputted candidates) will be passed to the application layer. Otherwise, if the DEPRCV method cannot output a single candidate, another retransmission will be requested. After receiving the second retransmission, if it contains errors, both CRC-EC and DEPRCV methods try to recover the packet. If both of them fail to recover it, as the third try, the buffer including three versions of the erroneous packet (one

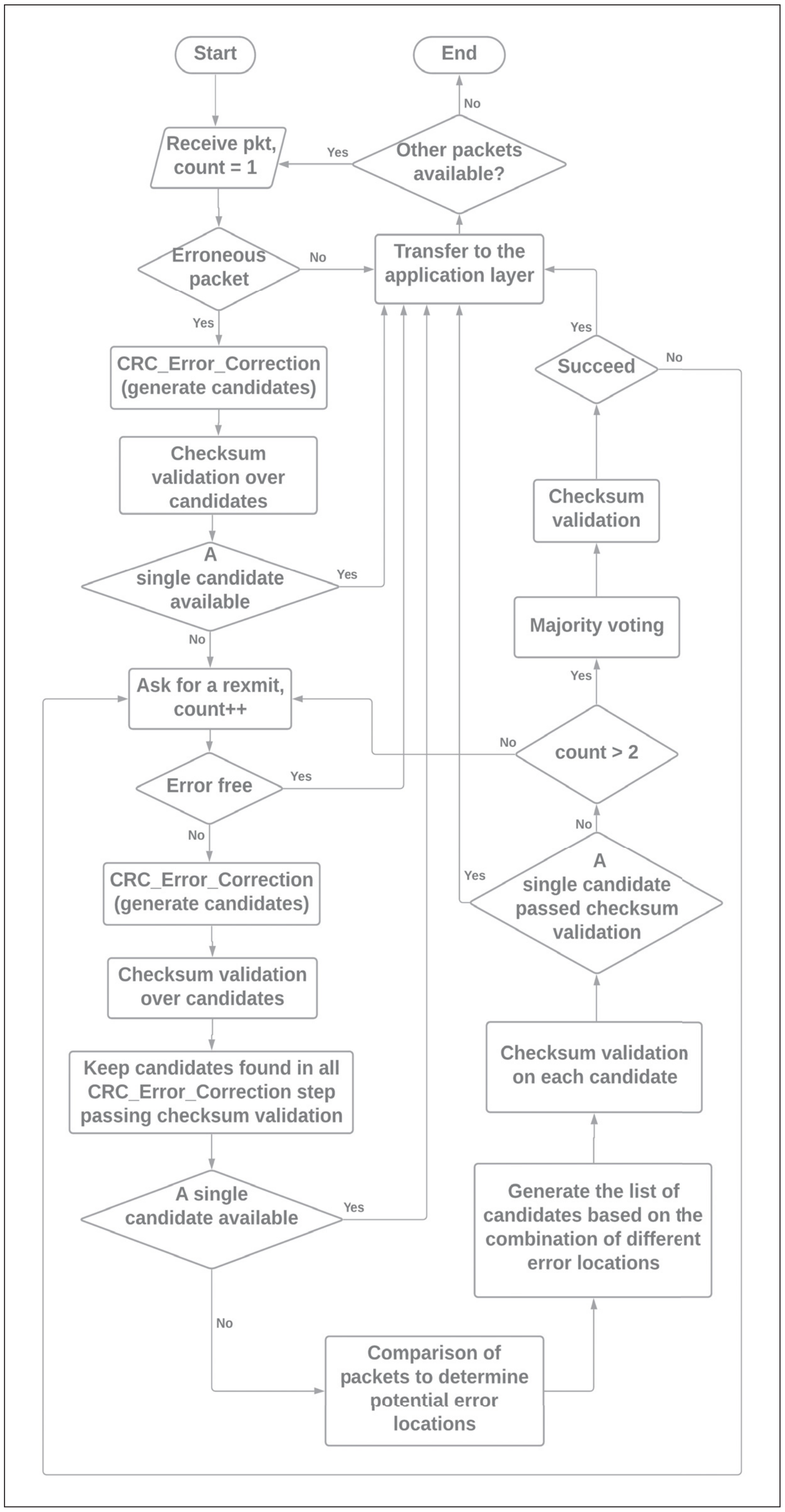

Start
End
No
Receive pkt, count = 1
Yes
Other packets available?
Erroneous packet
No
Transfer to the application layer
Yes
CRC_Error_Correction (generate candidates)
Yes
Succeed
No
Checksum validation over candidates
Checksum validation
A single candidate available
Yes
Majority voting
No
Yes
Ask for a rexmit, count++
No
count > 2
Error free
Yes
No
No
A single candidate passed checksum validation
Yes
CRC_Error_Correction (generate candidates)
Checksum validation over candidates
Checksum validation on each candidate
Keep candidates found in all CRC_Error_Correction step passing checksum validation
Generate the list of candidates based on the combination of different error locations
A single candidate available
Yes
No
Comparison of packets to determine potential error locations

Figure 3.7 Flowchart of the Cross-Layer Packet Recovery method

transmission and two retransmissions) is passed to the MVCV method in order to output a single candidate, which is passed to the checksum validation step. If the MVCV method succeeds to recover the packet, it is passed to the application layer and the packet recovery process ends. Otherwise, another retransmission will be requested and the described process will be repeated (CRC-EC, DEPRCV, and MVCV in order). This process will continue until recovering the erroneous packet. All the steps discussed, are provided in the algorithm 3.3 and algorithm 3.4 as the second part of the algorithm of the CLPR method.

In the next chapter, we present our experimental methodology and evaluate and compare the performance of the methods presented in this chapter.

Algorithm 3.3 Cross-layer packet recovery (Part 1/2)

```
Input: Received erroneous packet Output: Recovered packet
CS-check ← 0 // Transport layer checksum of the received packet initialized to 0
count ← 1, recovered ← 0, create empty buffer of packets
Call CRC-EC(Input: Received erroneous packet, Output: List of candidates)
if List of candidates not empty then
    for each candidate in List of candidates do
        CS-check(candidate) ← CV performed on candidate
    end
    if CS-check(candidate) = 1 for a single candidate then
        Output the candidate for which CS-check(candidate) = 1  // Xfer to app-layer
    end
end
Save last received packet as number "count" in the buffer of packets
Ask for a retransmission
count ← count + 1
Save last received packet as number "count" in the buffer of packets
CS-check ← CV performed on last received packet
if CS-check = 0 then
    Call CRC-EC(Input: Received erroneous packet, Output: List of candidates)
    if List of candidates not empty then
        for each candidate in List of candidates do
            CS-check(candidate) ← CV performed on candidate
        end
        if CS-check(candidate) = 1 for a single candidate then
            Output candidate for which CS-check(candidate) = 1  // Xfer to app-layer
        end
    end
    Call the DEPRCV-GEN(In: 2 latest packets in buffer of packets, Out: List cand.)
    if List of candidates not empty then
        for each candidate in List of candidates do
            CS-check(candidate) ← CV performed on candidate
        end
        if CS-check(candidate) = 1 for a single candidate then
            Output candidate for which CS-check(candidate) = 1  // Xfer to app-layer
        end
    end
else
    Output the last received packet  // Xfer to app-layer
end
...
```

Algorithm 3.4 Cross-layer packet recovery (Part 2/2)

```
...
while (1) do
    Ask for a retransmission
    CS-check ← CV performed on last received packet
    if CS-check = 1 then
        Output the last received packet        // Transfer to the app-layer
    end
    count ← count + 1
    Save last received packet as number "count" in the buffer of packets
    Call CRC-EC(Input: last received erroneous packet, Output: List of candidates)
    if List of candidates not empty then
        for each candidate in List of candidates do
            CS-check(candidate) ← CV performed on candidate
        end
        if CS-check(candidate) = 1 for a single candidate then
            Output candidate for which CS-check(candidate) = 1  // Xfer to app-layer
        end
    end
    Call the DEPRCV-GEN(Input: Two latest packets in buffer of packets, Output: List
      of candidates)
    if List of candidates not empty then
        for each candidate in List of candidates do
            CS-check(candidate) ← CV performed on candidate
        end
        if CS-check(candidate) = 1 for a single candidate then
            Output candidate for which CS-check(candidate) = 1  // Xfer to app-layer
        end
    end
    Call the MVCV(Input: buffer of packets, Output: MV candidate)
    CS-check ← CV performed on MV candidate
    if CS-check = 1 then
        Output MV candidate  // Xfer to app-layer
    end
end
```

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we present the performance of the methods proposed in the previous chapter. We first present the experimental setup for simulating the proposed methods. All the details regarding the testbed are covered in the first section. Then, we present the results of simulations with various figures and tables and analyze them.

4.1 Experiment Setup

The experiments for this thesis are performed on a desktop computer running Windows 10 at 64 bits equipped with an 8-core i7-3770 processor running at 3.40 GHz, with 12 GB of RAM. MATLAB is chosen as the experimental environment where we implement the functions dedicated to server endpoint, client endpoint, TCP, CRC-EC, MVCV, DEPRCV, and CLPR packet recovery methods.

4.1.1 The Testbed

In this section, we provide the details regarding the dedicated testbed to simulate our proposed methods.

4.1.1.1 Workstation

According to the figure 4.1, in our testing workstation, we consider having two endpoints : a server and a client. For each packet in our simulations, the server generates the packet and sends it over the network (dedicated as error injecter in our simulations). Then the client receives the packet and checks for the possible errors that may have occurred during the transmission. If the packet contains errors, the packet recovery process starts on the client side. If the packet can be recovered, an ACK is sent to the server and the next packet is processed and sent by the server.

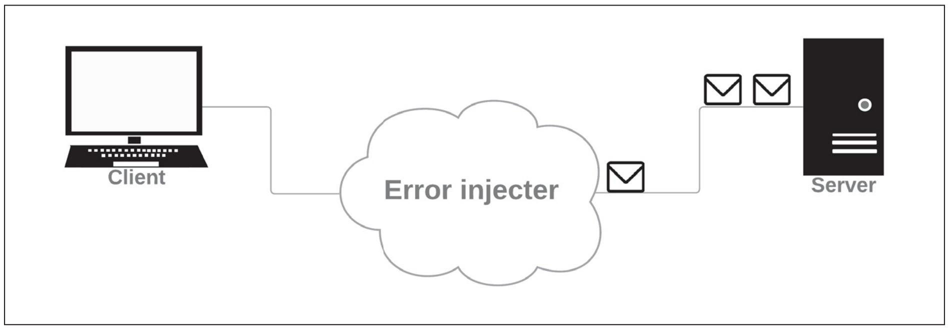


Figure 4.1 Testing workstation

Otherwise, the client informs the server with a NACK that the packet was not received and it requests retransmissions until it is able to recover the original packet. Once the recovery process ends up successfully, the client informs the server with an ACK. The connection between the server and the client stays active until the last scheduled packet has been received without error. Once the ACK for the last packet is received by the server, the connection is terminated.

4.1.1.2 Structure of the Packets

The information sent from the server to the client consists of realistic Ethernet data packets including the headers of each networking layer and the payload along with the CRC calculated over the packet. The figure 4.2 presents the content of each packet in detail.

The assigned header is fixed to 54 bytes, including the Ethernet header type II with the length of 14 bytes, the IP header with a length of 20 bytes, and the Transport header (TCP) with a length of 20 bytes. The length of the payload can be variable. In our case, the dedicated payload length occupies 125 bytes of the packet. Other packet sizes could be tested but the selected one is sufficient to illustrate how the performance of the various packet recovery strategies changes as the bit error rate is varied. At the end of the packet, the CRC part occupies 4 bytes of the total length of the packet. We do not modify the packet structure during our simulations and as illustrated in the Figure 4.2. In the scope of this thesis, we use a packet with the size of 1464 bits, which does not require to be split into smaller parts for transmission over the network. But in general, TCP splits larger packets (larger than 536 bytes) Postel (1983) into chunks and delivers

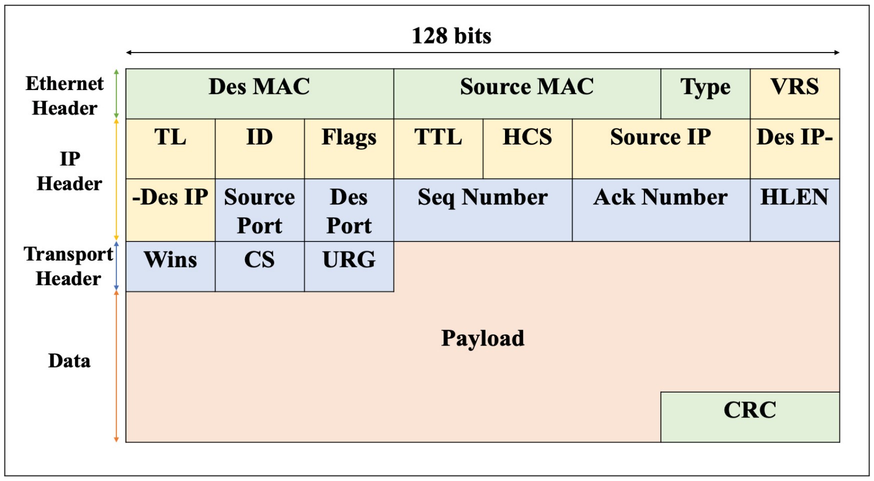


Figure 4.2 Ethernet II packet structure
Taken from Hartpence (2011)

them as segments using the underlying IP protocol. The approach we have adopted simplifies the experimental implementation and analysis of the results of various proposed solutions. But the results of this thesis can be applied to the more general case of larger packets.

4.1.1.3 The Network Model and Error Injection Process

Since there are many internal and external factors affecting the actual network used for data transmission and, most importantly, since the error occurrence over the network is unpredictable and out of our control, having a networking environment with control over the BER parameter is crucial for testing our proposed methods. Therefore, the network is simulated using the MATLAB environment. We use the functions which carry the data between the server and the client with the possibility of having control over the error occurrence through the BER parameter. To simulate a network with poor reception (e.g., high interference), we increase the value of the BER parameter, and the error injection over the packets occurs based on this parameter. The error injection scenario affects the value of some random bit positions of each packet according to the BER parameter and the size of the packet. As an example, if the size of the packet is 1000 bytes and the BER parameter value is 0.001, the average number of errors observed in this packet will be approximately 8 and therefore bit values will be changed at 8 different bits positions on average.

4.1.2 Simulation Parameters

The most significant parameters in this research are the following:

- Bit Error Rate (BER): the number of bit errors divided by the total number of transferred bits during a studied time interval (it is a probability without any unit). This parameter is used for error injection on the packets commuting between the server and the client.
- Packet size L: is the number of bits per packet commuting between the server and the client. This parameter can be variable but in our case, it is fixed and equal to 1464 bits.
- Number of data packets in the simulation N_{sim}: indicates the number of data packets to transfer between the server and the client during our simulations. It is different from the number of packets actually transmitted due to errors and required retransmissions.

To perform the various tests, we modify the value of BER and N_{sim} parameters. We keep the other parameters constant.

4.2 Goal of the Experiments

The goal of performing the experiments is to validate the performance if the proposed approaches for error correction and packet recovery presented in the previous chapter and compare them with the current and state-of-the-art error correction and packet recovery methods.

We expect that our proposed methods will perform better than the basic TCP and the state-of-the-art CRC-EC methods. To validate this, we performed a set of experiments. Using MATLAB, we developed separate functions for each of the methods to calculate their throughput. Our main methods' functions are as follows:

- Basic TCP

- State-of-the-art CRC-EC method
- Proposed MVCV method
- Proposed DEPRCV method
- Proposed CLPR method

All the mentioned methods have been tested under the same experimental circumstances. Different testing scenarios have been provided for each method by modifying the BER parameter and the number of data packets considered for the simulation to ensure that enough packets have been exchanged to obtain consistent results.

4.3 Data Transmission Over Various Error Conditions

Not all the networks exhibit the same error rate. We control the quality of the network with the parameter BER. The lower the parameter BER, the fewer errors occur on the transferred data (i.e., packets) and the opposite. We consider the value range of 0.0001-0.001 for the BER parameter as the lower range and the value range of 0.001-0.002 as the higher range.

4.3.1 Experimental Results

In this section, we present the results of the performed experiments. First, we provide the results through tables. Each table is dedicated to a unique test with the specified packet size of 1464 bits, the number of transferred packets (N_{sim}), and the BER parameters. We also provide a plot view of the results for better visualization. The theoretical and experimental probability tables, such as tables 4.1, 4.2, and 4.3, are organized into two parts. The top part presents, in each column, the probability of having j retransmissions, $P(\text{Rexmit}=j)$. The bottom part presents the cumulative probabilities of having at most j retransmissions, $P(\text{Rexmit} \leq j)$. In each part of the tables, after the title, the 1st row provides the title for each column which specifies the number of retransmissions considered. The 2nd row provides the theoretical probability values for the usual TCP denoted P_{TCP} computed using Eq. 4.1.

$$P_i = p^i \times (1 - p) \tag{4.1}$$

In the provided equation 4.1, P_i is the probability that “i" retransmissions are required before a good packet is received. This is computed as the probability of receiving i consecutive damaged packets multiplied by the probability of receiving 1 good packet. The p is the packet loss rate, which has been calculated utilizing the following equation, where the BER is the bit error rate parameter and L is length of packets:

$$p = 1 - ((1 - \text{BER})^L) \tag{4.2}$$

These values are used to validate the simulation results for TCP, presented in the 3rd row of the tables. Rows four, five, six and seven present the probability values, obtained through simulations, for the state-of-the-art CRC-EC method and the proposed MVCV, DEPRCV and CLPR methods, in that order. The last column of each table presents the average ratio of the Round Trip Time (RTT) delay for each method. The details and equations are provided in section 4.3.8.

Tables 4.1, 4.2, and 4.3 present the simulation results for 1 million packets (i.e., N_{sim} = 1M). As mentioned, the values presented as P_{TCP} in the 2nd row of each table, are probability values of having j retransmissions. As an example, in Table 4.1, in the row dedicated to P_{TCP}, the value P(Rexmit = 0) = 0.86381, indicates that 86.38% of the packets (out of 1M packets) do not require any retransmission as these packets are error-free. In the same row, the value P(Rexmit = 1) = 0.11765 indicates that the 11.76% of the packets require only 1 retransmission to be recovered. The value P(Rexmit = 2) = 0.01602 indicates that only 1.60% of packets require 2 retransmissions to be recovered and so on. Figures 4.3, 4.4, and 4.5 present the same results in the form of plots for better visualization.

Table 4.1 Theoretical and experimental probability results for a packet size of 1464 bits, N_{sim} = 1M, and BER = 0.0001

Probability/Results						
Methods	P(Rexmit=0)	P(Rexmit=1)	P(Rexmit=2)	P(Rexmit=3)	P(Rexmit=4)	D_{RTT}
P_{TCP}	0.86381	0.11765	0.01602	0.00218	0.00030	N/A
TCP	0.86380	0.11767	0.01601	0.00217	0.00031	0.6577
CRC-EC	0.98766	0.01219	0.00014	0.00001	0.00000	0.5125
MVCV	0.86366	0.11782	0.01853	0.00000	0.00000	0.6549
DEPRCV	0.86366	0.13633	0.00001	0.00000	0.00000	0.6364
CLPR	0.98746	0.012540	0.00000	0.00000	0.00000	0.5125
Cumulative Probability/Results						
Methods	P(Rexmit=0)	P(Rexmit≤1)	P(Rexmit≤2)	P(Rexmit≤3)	P(Rexmit≤4)	D_{RTT}
P_{TCP}	0.86380	0.98145	0.99747	0.99966	0.99995	N/A
TCP	0.86380	0.98147	0.99748	0.99965	0.99996	0.6577
CRC-EC	0.98770	0.99985	1.00000	1.00000	1.00000	0.5125
MVCV	0.86370	0.98147	1.00000	1.00000	1.00000	0.6549
DEPRCV	0.86370	0.99999	1.00000	1.00000	1.00000	0.6364
CLPR	0.98750	1.00000	1.00000	1.00000	1.00000	0.5125

Table 4.2 Theoretical and experimental probability results for a packet size of 1464 bits, N_{sim} = 1M, and BER = 0.001

Probability/Results						
Methods	P(Rexmit=0)	P(Rexmit=1)	P(Rexmit=2)	P(Rexmit=3)	P(Rexmit=4)	D_{RTT}
P_{TCP}	0.23114	0.17771	0.13664	0.10506	0.08077	N/A
TCP	0.23138	0.17772	0.13590	0.10541	0.08082	3.7271
CRC-EC	0.56185	0.24609	0.10811	0.04713	0.02075	1.2796
MVCV	0.23095	0.17832	0.58810	0.00059	0.00205	1.8644
DEPRCV	0.23162	0.76721	0.00118	0.00000	0.00000	1.2695
CLPR	0.56257	0.43676	0.00038	0.00007	0.00013	0.9388
Cumulative Probability/Results						
Methods	P(Rexmit=0)	P(Rexmit≤1)	P(Rexmit≤2)	P(Rexmit≤3)	P(Rexmit≤4)	D_{RTT}
P_{TCP}	0.23114	0.40885	0.54549	0.65055	0.73132	N/A
TCP	0.23138	0.40910	0.54500	0.65041	0.73123	3.7271
CRC-EC	0.56185	0.80794	0.91605	0.96318	0.98392	1.2796
MVCV	0.23095	0.40926	0.99736	0.99795	1.00000	1.8644
DEPRCV	0.23162	0.99882	1.00000	1.00000	1.00000	1.2695
CLPR	0.56257	0.99933	0.99971	0.99978	0.99990	0.9388

Table 4.3 Theoretical and experimental probability results for a packet size of 1464 bits, N_{sim} = 1M, and BER = 0.002

Probability/Results						
Methods	P(Rexmit=0)	P(Rexmit=1)	P(Rexmit=2)	P(Rexmit=3)	P(Rexmit=4)	D_{RTT}
P_{TCP}	0.05335	0.05050	0.04781	0.04526	0.04284	N/A
TCP	0.05303	0.05050	0.04776	0.04547	0.04272	5.8378
CRC-EC	0.20597	0.16424	0.12974	0.10340	0.08174	4.1485
MVCV	0.05345	0.05037	0.88314	0.00069	0.01230	2.3682
DEPRCV	0.05343	0.94203	0.00451	0.00000	0.00000	1.4511
CLPR	0.20592	0.79016	0.00081	0.00020	0.00058	1.3103
Cumulative Probability/Results						
Methods	P(Rexmit=0)	P(Rexmit≤1)	P(Rexmit≤2)	P(Rexmit≤3)	P(Rexmit≤4)	D_{RTT}
P_{TCP}	0.05335	0.10385	0.15166	0.19691	0.23976	N/A
TCP	0.05303	0.10352	0.15128	0.19674	0.23946	5.8378
CRC-EC	0.20597	0.37021	0.49995	0.60335	0.68509	4.1485
MVCV	0.05345	0.10382	0.98695	0.98764	0.99994	2.3682
DEPRCV	0.05343	0.99546	0.99997	0.99998	0.99998	1.4511
CLPR	0.20592	0.99608	0.99689	0.99708	0.99766	1.3103

4.3.2 Analyzing TCP Results

In each table, the probability values presented for the P_{TCP} and the TCP methods help us to determine the percentage of error-free and erroneous packets. Therefore, these packets are all error-free and all other methods recover at least those. In all the tables, the theoretical probability values for TCP, P_{TCP}, are almost equal to the TCP probability values from simulations. This validates the simulations and the fact that the experimental results are compatible with theoretical calculations. Considering the lower BER value in the Table 4.1 and higher BER value in Table 4.3 for N_{sim} of 1M, while the value of BER increases, the probability values for zero retransmissions decrease, which means by increasing the BER value, more packets become erroneous and they require more retransmissions to be recovered. Note that in some cases, the number of required retransmissions to obtain an intact packet can exceed 100 for higher BER values. We have decided to show only the first retransmissions as they are the most relevant for this thesis.

4.3.3 Analyzing CRC-EC Results

According to all the tables, the CRC-EC method performs better than the normal TCP, i.e., it requires fewer retransmissions than the normal TCP and the probability values for the lower number of retransmissions (P(Rexmit = 0), etc.) are higher than TCP meaning that even for the high BER values, this method can recover the packets with fewer retransmissions than TCP. However, because the method can only correct packets containing a single error, it is most effective when the BER is low enough to generate at most one error per packet. Otherwise, more retransmissions are required, which reduces its performance.

4.3.4 Analyzing MVCV Results

Since the MVCV method requires at least 3 transmissions (2 retransmissions) to start the recovery process, the probability values for this method are very close TCP for P(Rexmit = 0) and P(Rexmit = 1). After receiving a third version of an erroneous packet, it starts the packet recovery process and performs better than TCP, i.e., the probability values for P(Rexmit = 2) are higher in the MVCV method compared to TCP. After the third transmission, we observe that it can recover most of the packets without asking for extra retransmissions as shown by the cumulative probability which is close to 1. Again, in all the tables, we observe the same repercussion for MVCV, which starts performing better than the TCP from P(Rexmit = 2). The benefit over TCP is increasing with increasing BERs. Indeed, for low BER values, retransmitting a packet provides a very good chance that the newly received packet will be without error and the retransmission process suffices to obtain an error-free packet. But at low BER values, this is not the case and the recovery process performed by MVCV makes a tremendous difference. As long at the errors occur in different positions in the retransmitted packets, the correction is ensured after 3 transmissions. Comparing MVCV with the state-of-the-art CRC-EC method, we observe that MVCV is not able to compete with CRC-EC when the BER is low or when we target a very low number of transmissions. However, as the BER increases and we have to accept more retransmissions, the proposed MVCV method outperforms CRC-EC. For any BER, starting with 2 retransmissions, the cumulative distribution of MVCV is always higher than that of CRC-EC and the gap increases with increasing BER. For instance, we have 1.0000 versus 0.99748 respectively for BER=0.0001 and 0.98695 versus 0.49995 for BER=0.002.

4.3.5 Analyzing DEPRCV Results

Improving over the MVCV method, the proposed DEPRCV method only requires two versions of the erroneous packet to start the recovery process. Therefore, the probability values for the P(Rexmit = 0) of this method are almost equal to TCP and the MVCV method. But after receiving the first retransmission, this method starts to perform better than TCP and MVCV for P(Rexmit = 1), P(Rexmit = 2), etc. Comparing DEPRCV to the state-of-the-art CRC-EC method, we observe that DEPRCV performs better than CRC-EC after receiving the first retransmission. In most of the cases, after receiving the first retransmissions, DEPRCV can recover the erroneous packet as shown by the cumulative probability which is close to 1 (0.99999 for BER=0.0001 and 0.99546 for BER=0.002). These values are better than those of CRC-EC starting at 2 transmission for any BER. But the weakness of DEPRCV compared to CRC-EC is that it requires at least 1 retransmission to start the recovery process, while the state-of-the-art CRC-EC can be applied to the erroneous packet without any retransmission required. Before that first retransmission, the proposed DEPRCV method is no better than TCP or MVCV.

4.3.6 Analyzing CLPR Results

The proposed CLPR method, which combines the other proposed methods with state-of-the-art CRC-EC, takes advantage of all of them to compensate for their disadvantages. Doing so, it outperforms all the other methods overall for any BER. For every number of retransmissions taken individually, it either has the same performance of any other methods or surpasses it. But taking all the numbers of retransmissions together, it outperforms every method. As an example, in Table 4.2, CLPR has the highest throughput for P(Rexmit = 0) = 0.5625, meaning 56.25% of the packets do not need any retransmission. Not all of the 56.25% of the packets were erroneous, i.e., based on the probability of TCP, we observe that 23.11% of the packets were error-free. But the other 56.25% − 23.11% = 33.14% of the packets, which were erroneous, could be recovered with no retransmission using the CLPR method, which has applied state-of-the-art CRC-EC as the first step. The probability P(Rexmit = 1) = 0.4367 of CLPR means that the 43.67% − 17.77% = 25.90% of the erroneous packets could be recovered only using one retransmission, while CRC-EC could recover 24.61% − 17.77% = 6.84% of the erroneous packets, which shows the effect of utilizing the DEPRCV method as the second step. Therefore, according to the Cumulative(P(Rexmit = 1)) = 0.9993, 99.93% of the packets can be delivered to the application layer with 1 retransmission. We can observe that with a single transmission, CLPR and CRC-EC are the two best methods for any BER and offer with equal performance. Considering two transmissions or more, CLPR and DEPRCV are the two best methods for any BER and offer equal performance.

4.3.7 Cumulative Probability Results

In this subsection, we take a closer look at the second part of each table exposing the cumulative probability values for each method as it eases the understanding of their ability to quickly deliver an error-free packet. Figures 4.6, 4.7, and 4.8 illustrate visually the cumulative experimental probabilities shown in the tables. Obviously, for each method, starting from the column dedicated to P(Rexmit = 0) and moving to the right towards the column dedicated to P(Rexmit = 4), we observe that the probability values increase. This helps show the progress of each method in recovering the erroneous packets and delivering them to the application layer. For instance, in Table 4.1, looking at CLPR, P(Rexmit ≤ 1) = 1.0000 demonstrates that it delivers 100% of the packets with at most 1 retransmission. Whenever the probability value of a method reaches 1 (or 100%), it means that the method recovered all the erroneous packets. In the same table, the method DEPRCV reaches a probability of 1 when Rexmit =2. For MVCV, this recovery process ends when Rexmit = 3, and the state-of-the-art CRC-EC method reaches 100% probability when Rexmit = 3. But for TCP, the probability value reaches 1 only for higher values of Rexmit.

Tables 4.1 to 4.3 are provided in order where the BER values are increasing from one table to the next. Although the cumulative probability values increase for all methods in all tables, this increase towards probability 1 happens faster as we increase the number of retransmissions for the lower BER values. Nevertheless, a cumulative probability of more than 0.999 is reached with a single retransmission for CLPR and DEPRCV for all shown BER values, while that level requires three retransmissions for MVCV at low BERs and even more at high BERs.

4.3.8 Delay Results

In this thesis, we considered the average delay as a function of RTT, which is equal to "ratio × RTT". The considered delay is the time from the departure of a packet at the server to the reception of that packet (without error) at the client, which may include one or several retransmissions. The considered average delay has been calculated based on the following equation 4.3:

$$\text{Average-Delay} = \sum_{j=0}^{\infty} \frac{\text{RTT}}{2} \times (2j+1) \times \text{P}_\text{j}\,. \tag{4.3}$$

In the provided equation, the counter variable j presents the required number of retransmissions, RTT is the round trip time, and P_j is the experimental probability of having j retransmissions. In the provided tables 4.1 to 4.3, we use a ratio for convenience as the result can be applied to any network's RTT to evaluate the expected average delay for each method. As an example, the delay ratio for the method TCP in Table 4.1, is equal to 0.6577. Therefore the delay is 0.6577 × RTT, where the RTT can be measured for each network of interest. As mentioned before, since our packet size is limited to 1464 bits, we did not consider breaking them into chunks and therefore, the RTT measurements are dedicated to each packet in our case. But in general, since the TCP splits large packets into chunks, the RTT measurements of the chunks are much more complicated because intact chunks will not be retransmitted and only the erroneous chunks will be asked to be retransmitted.

Now, we analyze the RTT results provided in the last column of the tables 4.1 to 4.3. Their visual equivalents are shown in figures 4.9, 4.10, and 4.11 respectively. We can observe that for low BER values, the CLPR and CRC-EC methods have slightly better performance than the other ones. This is because packets are rarely lost. As the BER increases, we can observe that the RTT increases for all methods but less for CLPR and DEPRCV than the others. Indeed, at BER=0.002, they exhibit an average RTT ratio of 1.31 and 1.45 respectively while MVCV, CRC-EC and TCP exhibit ratios of 2.37, 4.15 and 5.84 respectively. The first two methods are thus very effective at keeping the delay required to provide an error-free packet very low regardless of the BER value (within a reasonable range of values). The ratio value for the CLPR method is always the smallest.

The least required number of retransmissions makes the plots of CLPR, DEPRCV, and MVCV methods to have an extremely concentrated distribution because they reach the 100% of their performance after receiving only 1 retransmission for CLPR and DEPRCV and 2 retransmissions by the MVCV. Considering the CLPR and TCP methods in Figure 4.5, TCP has a nearly uniform distribution while CLPR has a very concentrated distribution. Again, the reason for TCP having a uniform distribution is the number of retransmissions it requires to recover the erroneous packets. We observe that CLPR has the least average delay among all other methods (shown in the plots of the delay).

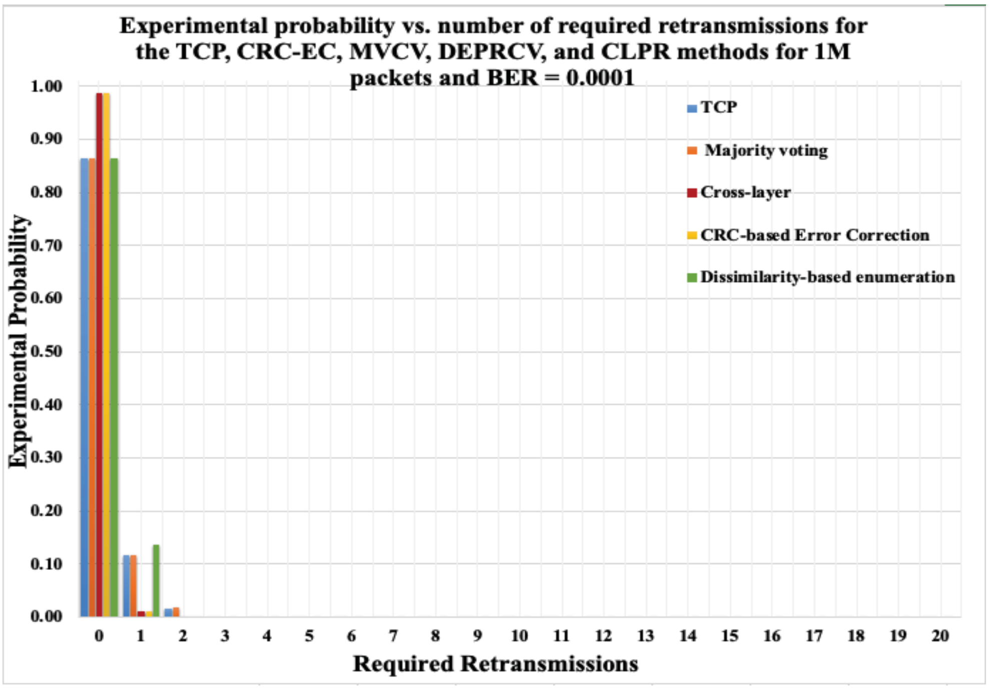


Figure 4.3 Experimental probability vs. number of required retransmissions for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.0001

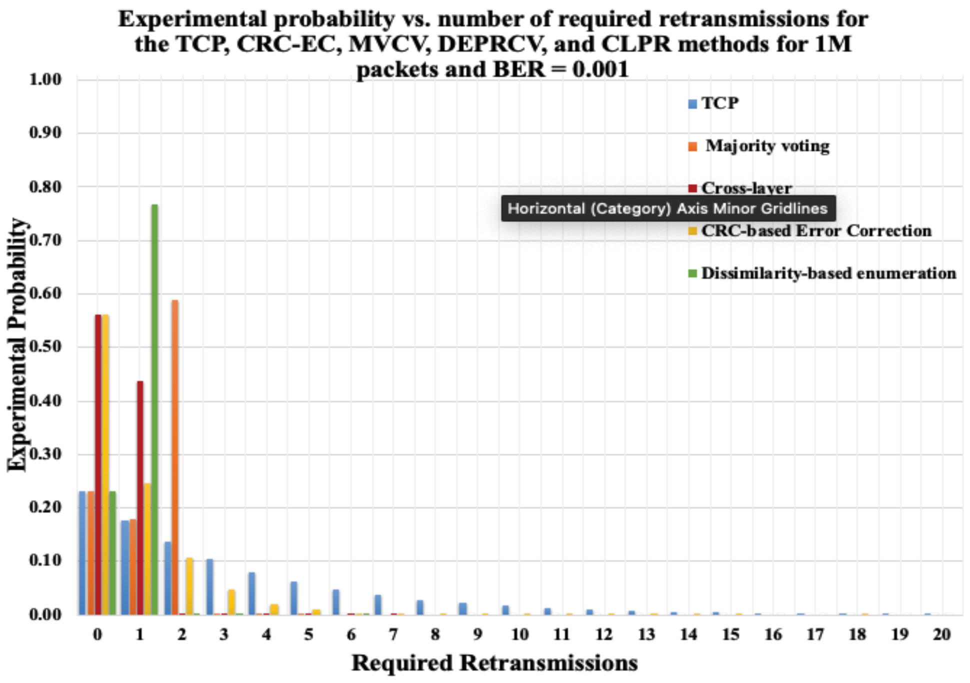


Figure 4.4 Experimental probability vs. number of required retransmissions for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.001

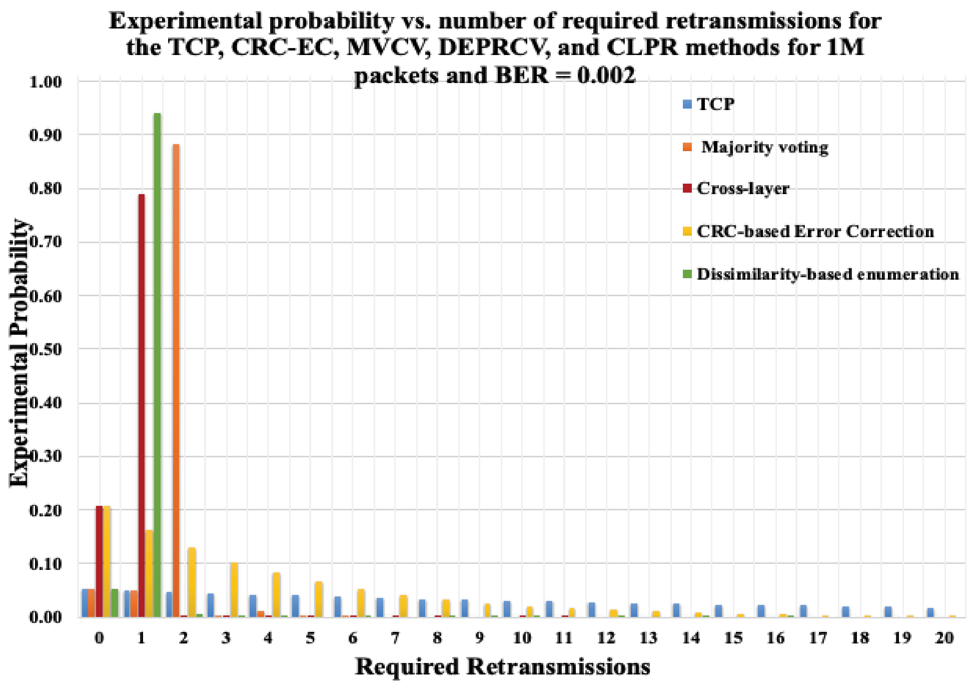


Figure 4.5 Experimental probability vs. number of required retransmissions for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.002

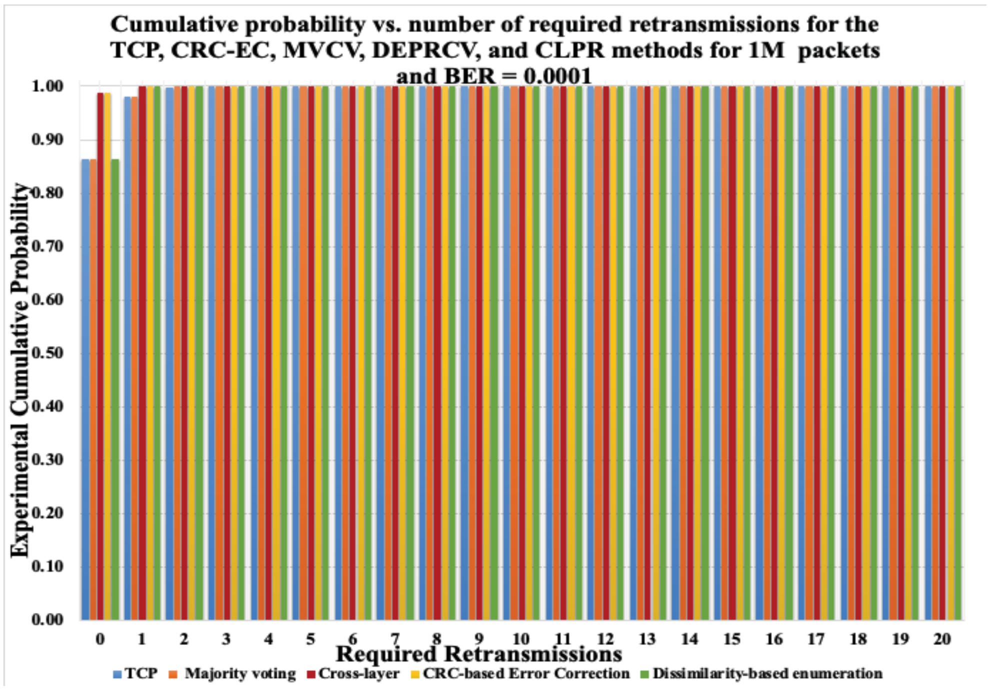


Figure 4.6 Cumulative probability vs. number of required retransmissions for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.0001

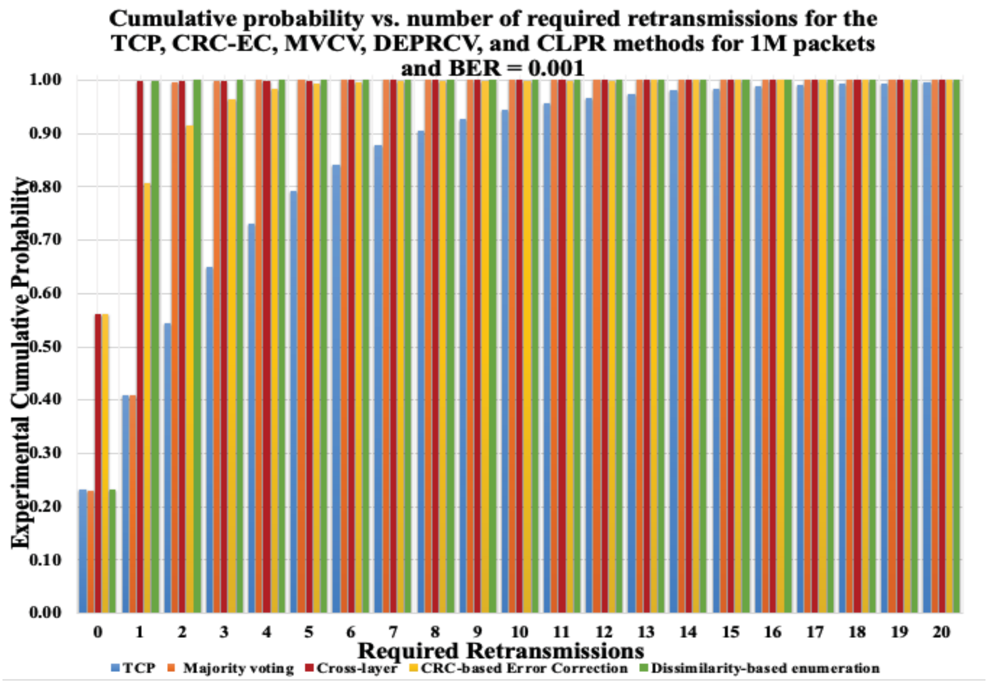


Figure 4.7 Cumulative probability vs. number of required retransmissions for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.001

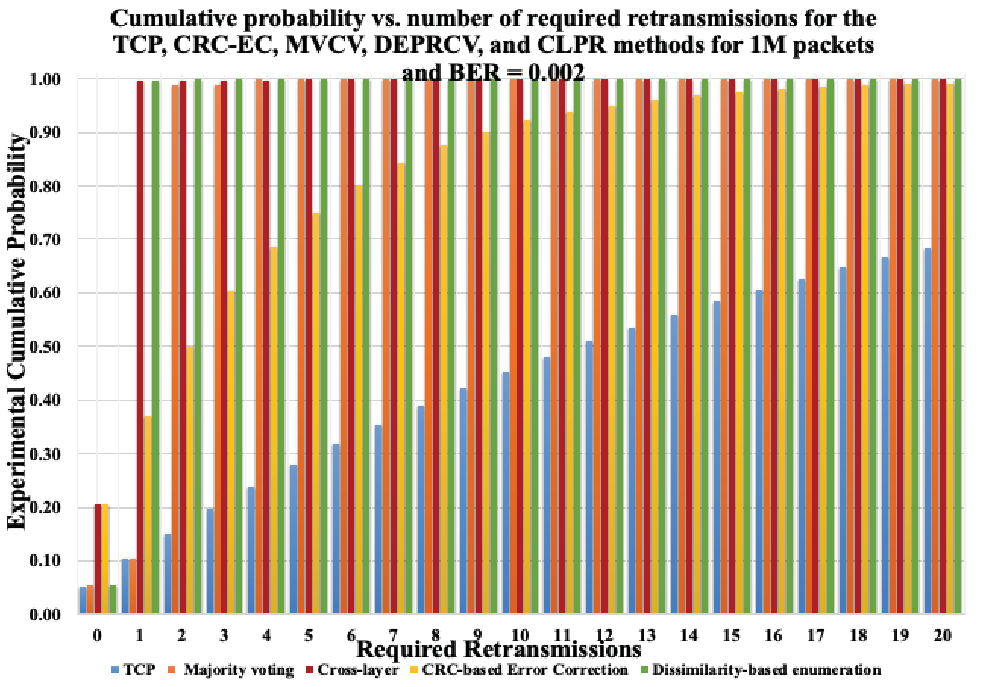


Figure 4.8 Cumulative probability vs. number of required retransmissions for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.002

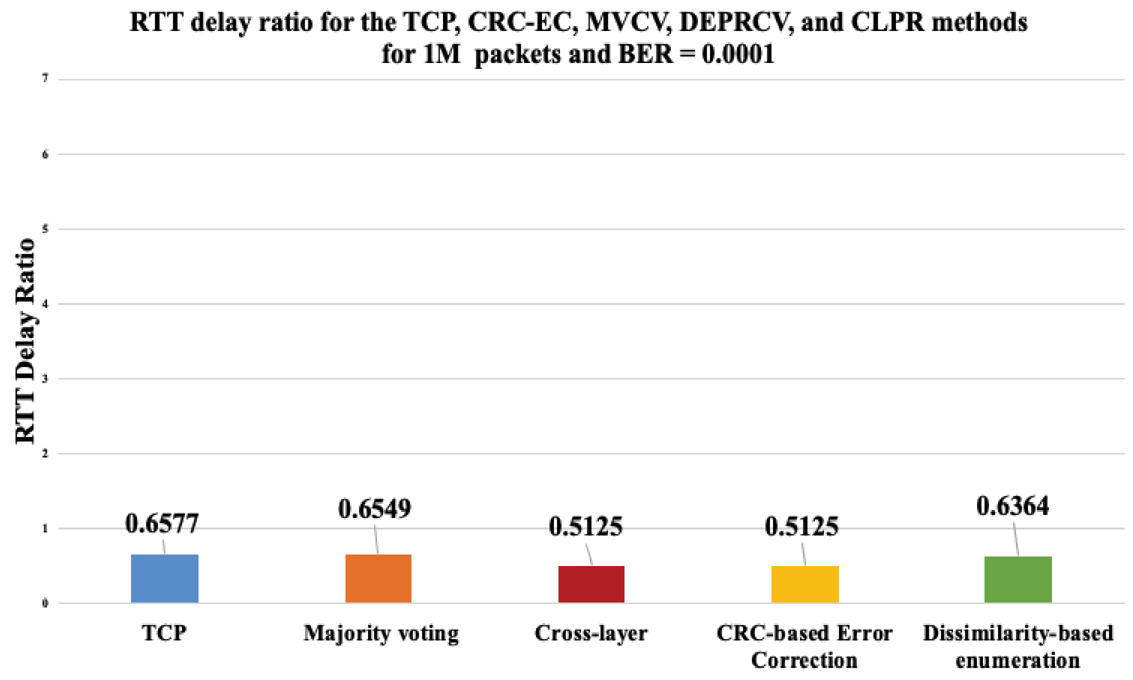


Figure 4.9 RTT delay ratio for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.0001

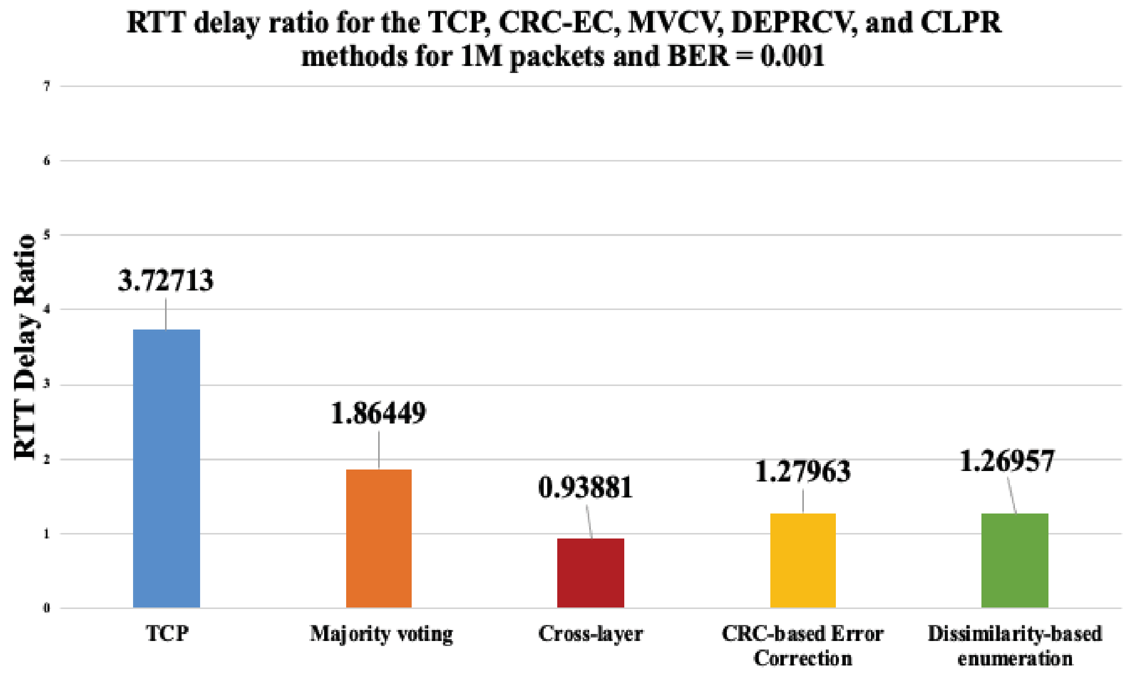


Figure 4.10 RTT delay ratio for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.001

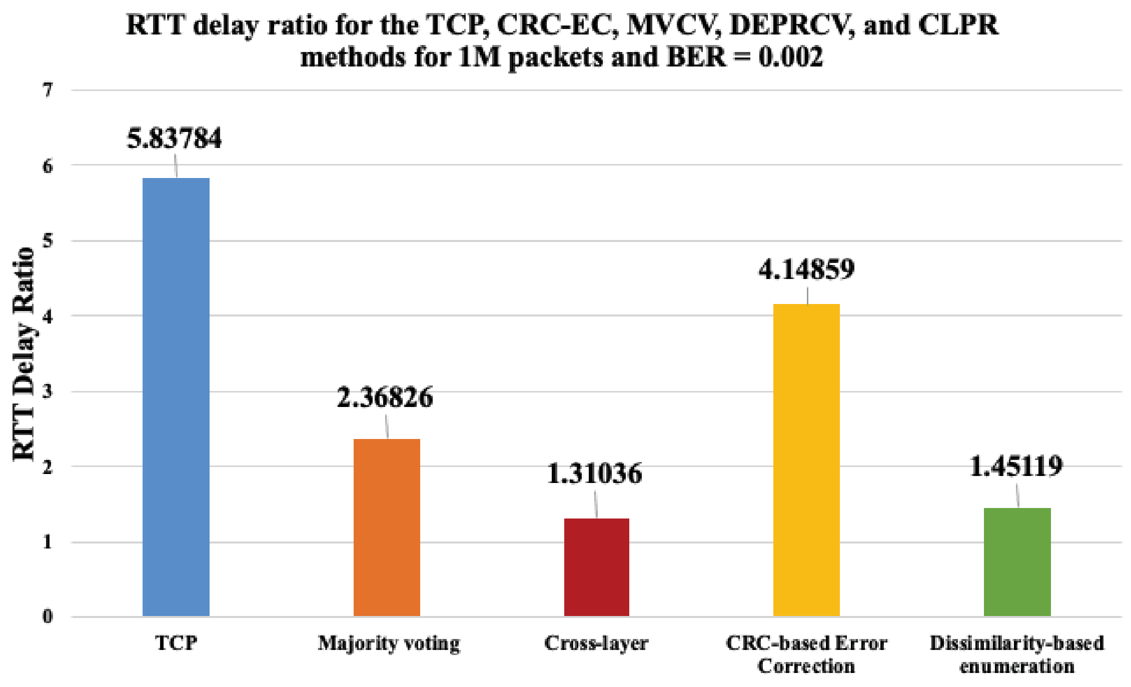


Figure 4.11 RTT delay ratio for the TCP, CRC-EC, MVCV, DEPRCV, and CLPR methods for 1M packets and BER = 0.002

CONCLUSION AND RECOMMENDATIONS

In this thesis, we addressed the problem of packet error correction and recovery in the context of TCP over networks lightly affected by bit errors. We proposed three different mechanisms which are compatible with the TCP standard to solve this problem. The methods, their performance and our contributions can be summarized as follows:

- We have proposed to use the majority voting process along with the checksum verification. After implementing and simulating the MVCV method, we observed that this method performs better than TCP as it limits the number of required retransmissions to 2 in most cases.
- We have proposed the DEPRCV method, which performs better than MVCV and TCP since it requires fewer retransmissions than them. We observed that it limits the number of required retransmissions to 1 in most cases. This method can compete with the state-of-the-art CRC-EC method when the BER is high-enough.
- We have introduced the CLPR method, which combines multiple error correction and packet recovery methods and provides higher performance and lower delay compared to the state-of-the-art and other proposed methods.

We thus recommend the use of CLPR as it provides the best performance.

5.1 Summary of Advantages and Drawback of the Various Methods

Each of the existing and proposed methods for error correction and packet recovery have advantages and disadvantages when they are used as a single method. Here we provide a table to compare the methods we have studied in this chapter.

Table 5.1 Comparison table for the proposed and state-of-the-art methods

Methods	TCP	CRC-EC	MVCV	DEPRCV	CLPR
High performance				✓	✓✓
Low delay				✓	✓✓
Fewer required retransmissions		✓		✓	✓✓
Low computational complexity				✓	
Low memory	✓	✓			
Proper for high BER				✓	✓
Proper for low BER		✓			✓

According to Table 5.1, we observe that the CLPR method is the pioneer among all. Therefore, this method can be introduced as the most powerful error correction and packet recovery method among all.

5.2 Future Works

Future work may be carried to evaluate the performance of CLPR when the CRC-EC it contains is configured to correct more than one error (at the expense of more computational complexity). It would also be valuable to evaluate the proposed methods in a complete communication scenario with complete TCP protocol and actual wireless network conditions. Another proposition is to combine FEC approach along with the introduced CLPR method to take advantage of error correction codes to improve the performance of CLPR approach for error correction and packet recovery. Finally, congestion detection and control combined with the proposed error correction approaches should be investigated.

REFERENCES

Al-Alawi, A. I. (2006). WiFi technology: Future market challenges and opportunities. *Journal of computer science*, 2(1), 13–18.

Angelopoulos, G., Chandrakasan, A. P. & Médard, M. (2014). PRAC: Exploiting partial packets without cross-layer or feedback information. *2014 IEEE International Conference on Communications (ICC)*, pp. 5802-5807. doi: 10.1109/ICC.2014.6884247.

Angelopoulos, G., Médard, M. & Chandrakasan, A. P. (2017). Harnessing partial packets in wireless networks: throughput and energy benefits. *IEEE Transactions on Wireless Communications*, 16(2), 694–704.

Balakrishnan, H., Padmanabhan, V. N., Seshan, S. & Katz, R. H. (1997). A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM transactions on networking*, 5(6), 756–769.

Boussard, V., Coulombe, S., Coudoux, F.-X. & Corlay, P. (2020a). Table-Free Multiple Bit-Error Correction Using the CRC Syndrome. *IEEE Access*, 8, 102357–102372.

Boussard, V., Golaghazadeh, F., Coulombe, S., Coudoux, F.-X. & Corlay, P. (2020b). Robust H.264 Video Decoding Using CRC-Based Single Error Correction And Non-Desynchronizing Bits Validation. *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 1098–1102.

Boussard, V., Coulombe, S., Coudoux, F.-X. & Corlay, P. (2021a). *Enhanced CRC-Based Correction of Multiple Errors with Candidate Validation*. Submitted to Signal Processing: Image Communications.

Boussard, V., Coulombe, S., Coudoux, F.-X. & Corlay, P. (2021b). *CRC-Based Correction of Multiple Errors Using an Optimized Lookup Table*. Submitted to Digital Communications and Networks.

Chen, Y.-C., Lim, Y.-s., Gibbens, R. J., Nahum, E. M., Khalili, R. & Towsley, D. (2013). A measurement-based study of multipath TCP performance over wireless networks. *Proceedings of the 2013 conference on Internet measurement conference*, pp. 455–468.

Comer, D. E. (2018). *The Internet book: everything you need to know about computer networking and how the Internet works*. CRC Press.

Dubois-Ferriere, H., Estrin, D. & Vetterli, M. (2005). Packet combining in sensor networks. *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 102–115.

Fall, K. R. & Stevens, W. R. (2011). *TCP/IP illustrated, volume 1: The protocols*. Addison-Wesley.

Fang, R., Schonfeld, D., Ansari, R. & Leigh, J. (2000). Forward error correction for multimedia and teleimmersion data streams. *FEC*, 1(P2), P3.

Galluccio, L., Morabito, G. & Palazzo, S. (2003). An analytical study of a tradeoff between transmission power and FEC for TCP optimization in wireless networks. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, 3, 1765–1773.

Galluccio, L., Morabito, G. & Palazzo, S. (2006). Achieving TCP optimization over wireless links through joint FEC and power management: an analytical study. *IEEE transactions on wireless communications*, 5(10), 2956–2966.

Gast, M. (2005). *802.11 wireless networks: the definitive guide*. " O'Reilly Media, Inc.".

Han, B., Schulman, A., Gringoli, F., Spring, N., Bhattacharjee, B., Nava, L., Ji, L., Lee, S. & Miller, R. R. (2010). Maranello: Practical Partial Packet Recovery for 802.11. *NSDI*, pp. 205–218.

Handley, M., Bonaventure, O., Raiciu, C. & Ford, A. (2013). TCP extensions for multipath operation with multiple addresses. 1–64.

Hartpence, B. (2011). *Packet guide to core network protocols*. " O'Reilly Media, Inc.".

Hennessy, J. L. & Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.

Irianto, K. D., Nguyen, G. T., Salah, H. & Fitzek, F. H. (2019). Partial packet in wireless networks: a review of error recovery approaches. *IET Communications*, 14(2), 186–192.

Jamieson, K. & Balakrishnan, H. (2007). PPR: Partial packet recovery for wireless networks. *ACM SIGCOMM Computer Communication Review*, 37(4), 409–420.

Khan, S. A., Moosa, M., Naeem, F., Alizai, M. H. & Kim, J.-M. (2016). Protocols and mechanisms to recover failed packets in wireless networks: History and evolution. *IEEE Access*, 4, 4207–4224.

Kozierok, C. M. (2005). *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. No Starch Press.

Laurindo, S., Moraes, R. & Montez, C. (2020). Cooperative Communication Mechanisms Applied to Wireless Sensor Network. *Doctoral Conference on Computing, Electrical and Industrial Systems*, pp. 121–128.

Lin, K. C.-J., Kushman, N. & Katabi, D. (2008). ZipTx: Harnessing partial packets in 802.11 networks. *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pp. 351–362.

Miller, M. (2013). *Wireless Networking Absolute Beginner's Guide*. Que Publishing.

Miu, A., Balakrishnan, H. & Koksal, C. E. (2005). Improving loss resilience with multi-radio diversity in wireless networks. *Proceedings of the 11th annual international conference on Mobile computing and networking*, pp. 16–30.

Mohammadi, M. S., Zhang, Q. & Dutkiewicz, E. (2015). Exploiting partial packets in random linear codes using sparse error recovery. *Communications (ICC), 2015 IEEE International Conference on*, pp. 2577–2582.

Nicopolitidis, P., Obaidat, M. S., Papadimitriou, G. I. & Pomportsis, A. S. (2003). *Wireless networks*. Wiley Online Library.

Padlipsky, M. (1982). Perspective on the ARPANET reference model. 1–24.

Postel, J. (1983). *The TCP maximum segment size and related topics*.

Qi, X., Wang, L., Wu, K. & Tao, J. (2015). Exploring smart pilot for partial packet recovery in super dense wireless networks. *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pp. 2145–2150.

Rackley, S. A. (2011). *Wireless networking technology: From principles to successful implementation*. Elsevier.

Raghavendra, C. S., Sivalingam, K. M. & Znati, T. (2006). *Wireless sensor networks*. Springer.

Ross, J. (2008). *The book of wireless: A painless guide to Wi-Fi and broadband wireless*. No Starch Press.

Sindhu, P. (1977). Retransmission error control with memory. *IEEE Transactions on Communications*, 25(5), 473–479.

Spurgeon, C. E. (2000). *Ethernet: the definitive guide*. " O'Reilly Media, Inc.".

Tan, W.-T. & Zakhor, A. (1999). Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol. *IEEE Transactions on Multimedia*, 1(2), 172–186.

Xie, J., Hu, W. & Zhang, Z. (2011). Revisiting partial packet recovery in 802.11 wireless LANs. *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp. 281–292.

Yavatkar, R. & Bhagawat, N. (1994). Improving end-to-end performance of TCP over mobile internetworks. *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pp. 146–152.

Yli-Juuti, E., Chakraborty, S. & Liinaharja, M. (1998). An adaptive ARQ scheme with packet combining. *IEEE Communication Letters*, 2, 200–202.

Zhang, Z., Hu, W. & Xie, J. (2012). Employing coded relay in multi-hop wireless networks. *Global Communications Conference (GLOBECOM), 2012 IEEE*, pp. 5650–5656.

Zimmermann, H. (1980). OSI reference model-the ISO model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4), 425–432.