Towards a new generation of IoT based on network coding for the Industry 4.0

by

Amir Abbas MODIR

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

MONTREAL, OCTOBER 12, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Pascal Giard, and my co-supervisor, Professor Georges Kaddoum, for their help during the learning process of this master thesis. Furthermore, I would like to express my gratitude to Dr. Lokman Sboui for his guidance in my work. Finally, thanks to my family, especially my sister which supports me and guides me through my studies.

Vers une nouvelle génération d’loT basée sur le codage de réseau pour l’Industrie 4.0

Amir Abbas MODIR

RÉSUMÉ

L’Internet des Objets (IoT) est une technologie révolutionnaire qui fournit des services intelligents en connectant aux réseaux des objets équipés de capteurs, d’actionneurs et de processeurs. Elle est utilisée dans différentes applications telles que les maisons intelligentes, la vie sociale et le divertissement, la santé et la forme physique, etc. De plus, l’IoT est utilisé dans des applications industrielles telles que le transport, la fabrication, l’énergie et l’agriculture. Dans ce cas, elle est connues sous le nom d’Internet Industriel des Objets (IIoT). Les usines utilisent les réseaux IIoT pour les services urgents, tels que la détection d’anomalies, le contrôle du système et le guidage des robots, qui requièrent de courts délais et des communications très fiables. Cependant, les réseaux IIoT utilisent les communications sans fil et sont fréquemment déployés dans des usines affectées par des niveaux élevés de bruits et d’interférences. En conséquence, la perte d’information est un problème inévitable dans les réseaux IIoT, augmentant les délais de communication et affectant la fiabilité du réseau. Dans ces réseaux, étant donné que les paquets doivent être livrés à la couche d’application dans l’ordre, un paquet ne peut être livré que si les précédents l’ont déjà été. Par conséquent, la perte d’informations est un problème pour les services critiques qui nécessitent des communications avec un faible délai. Récemment, le codage réseau a été identifié comme une technologie prometteuse pour gérer les pertes d’information. Le codage réseau peut récupérer les informations perdues en utilisant différentes méthodes. Dans cette thèse, nous étudions deux des méthodes de codage de réseau les plus importantes afin d’identifier celle qui est la mieux adaptée aux exigences strictes des réseaux IIoT. À cette fin, nous comparons le codage de réseau par blocs avec des méthodes de codage de réseau de type "sliding network" dans un scénario de type "one-hop multicast" et étudions les effets des différents paramètres sur leurs performances par le biais de simulations. Les simulations montrent que le codage réseau à fenêtre glissante décode plus de paquets avec un délai de livraison ordonnée inférieur à celui du RLNC basé sur les blocs. Cependant, le codage réseau à fenêtre glissante a un retard maximal de livraison ordonnée et une complexité de décodage plus élevés et nécessite une taille de tampon plus importante.

Mots-clés: Internet industriel des objets, codage réseau linéaire aléatoire, fenêtre coulissante RLNC, correction d’effacement avant au niveau de l’application, codage réseau.

Towards a new generation of IoT based on network coding for the Industry 4.0

Amir Abbas MODIR

ABSTRACT

Internet of Things (IoT) is considered to be a disruptive technology that provides intelligent services by connecting objects equipped with sensors, actuators, and processors to networks. It is used in different applications such as smart homes, social life, entertainment, health, fitness, etc. Furthermore, IoT is applied in industrial applications such as transportation, manufacturing, energy, and agriculture, known as the Industrial Internet of Things (IIoT). Factories use IIoT networks for time-critical services, such as anomaly detection, system control, and robot guidance, that require low delays and highly reliable communications. However, IIoT networks use wireless communication and are applied to factories with high levels of noises and interference. As a result, information loss is an inevitable issue in IIoT networks, increasing communication delay and affecting the reliability of the network. In these networks, since packets should be delivered in-order to the application layer, the next packets cannot be delivered to the application layer unless the previous packets are delivered. Thus, information loss is detrimental for the time-critical services that require low delay communications. Recently, Network Coding (NC) has been introduced as a promising technology to recover information loss. Network coding can recover information loss by different methods. In this thesis, we investigate two of the most important network coding methods in order to identify which would be the most suitable to the stringent requirements of IIoT networks. To this end, we compare block-based network coding with sliding network coding methods in a one-hop multicast scenario and study the effects of the various parameters based on their performance. Simulations show that sliding window network coding decodes more packets with a lower in-order delivery delay than that of block-based RLNC. However, sliding window network coding has a maximum in-order delivery delay and decoding complexity and also needs a larger buffer size.

Keywords: Industrial Internet of Things, Random Linear Network Coding, Sliding Window RLNC, Application-Level Forward Erasure Correction, Network Coding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AL-FEC	Application-Level Forward Erasure Correction.
ANC	Analog Network Coding.
ARQ	Automatic Repeat Request.
BEC	Binary Erasure Channel.
CRLNC	Caterpillar RLNC.
DNC	Digital Network Coding.
IDNC	Instantly Decodable Network Coding.
IIoT	Industrial Internet of Things.
IoT	Internet of Things.
KPIs	key performance indicators.
LDPC	low-density parity-check code.
LT	Luby Transform code.
NC	Network Coding.
ONC	Opportunistic Network Coding.
PEC	Packet Erasure Channel.
PER	Packet Erasure Rate.
QoS	Quality of Service.
RLNC	Random Linear Network Coding.

RS	Reed–Solomon code.
SWRLNC	Sliding Window RLNC.
WSNs	Wireless Sensor Networks.

INTRODUCTION

Industry 4.0 is the fourth industrial revolution that represents the newest generation of industries called smart factories. It extracts benefits from the IIoT that interconnects machines, humans, and smart devices and shares a large amount of data among them. IIoT networks require low latency and highly reliable communications to meet the requirements of time-critical services (Peralta, Cid-Fuentes, Bilbao & Crespo, 2018). However, IIoT networks most often use wireless communication in harsh environments, such as factories, with high levels of noises and interference (Zverev, Garrido, Agüero & Bilbao, 2019). Deploying sensors in these environments causes information loss in IIoT networks, since the impact of wireless channel and interference leads to information loss.

Different methods can recover information loss, including Automatic Repeat Request (ARQ) (Karzand, Leith, Cloud & Médard, 2017), error-correcting codes such as Reed–Solomon code (RS) (Roca, Cunche, Lacan, Bouabdallah & Matsuzono, 2013), fountain codes (Luby, 2002; Shokrollahi, 2006) , and network coding (Sundararajan, Shah, Médard & Sadeghi, 2017). The latter two are known as Application-Level Forward Erasure Correction (AL-FEC) codes (Roca, Teibi, Burdinat, Tran & Thienot, 2016). They are used in reliable transport protocols, such as TCP, to improve communication reliability (Roca *et al.*, 2016). ARQ uses feedback to recover information loss. It is an optimal solution for unicast scenarios. However, it cannot be used for broadcast and multicast scenarios, and where feedback is delayed or use of feedback is unavailable (Karzand *et al.*, 2017; Sundararajan *et al.*, 2017; Wunderlich, Gabriel, Pandi, Fitzek & Reisslein, 2017). Block-based codes such as RS solve these problems. However, because of their block-based structure, adaptation between block size and channel changes is a great challenge (Karzand *et al.*, 2017; Tournoux, Lochin, Lacan, Bouabdallah & Roca, 2011). Nonetheless, fountain codes are rateless codes that resolve the problem of RS. However, they decrease the throughput or increase the communication delays in the tandem network of broadcast links (Sundararajan *et al.*, 2017). Thus, they cannot be used in different network topologies. Recently, network coding is introduced as a promising method for the recovery of information loss in packet erasure channels (Karzand *et al.*, 2017; Sundararajan *et al.*, 2017; Wunderlich *et al.*, 2017). This technique has advantages over other methods. For example, it does not have a complex code structure in comparison to error-correcting codes such as RS and fountain codes; it can increase reliability by combining any number of coded or uncoded packets, it decreases communication delays; and it can be applied to different topologies (Fouli, (MIT) & Shroff, 2018). Network coding can recover information loss by different schemes, such as online network coding (Sundararajan *et al.*, 2017), Instantly Decodable Network Coding (IDNC) (Douik, Sorour, Al-Naffouri & Alouini, 2017), and sending redundant coded packets based on network coding (Karzand *et al.*, 2017; Wunderlich *et al.*, 2017). Concerning the limitations and advantages of these schemes, the last one has a higher potential for applying to IIoT networks in order to reduce communication delay. This scheme can be achieved using block-based and sliding window network coding methods (Wunderlich *et al.*, 2017). For example, (Karzand *et al.*, 2017) used sliding window network coding in a unicast scenario to recover information loss in satellite communications, where ARQ is not suitable given the high feedback delays.

In (Roca *et al.*, 2016; Wunderlich *et al.*, 2017), sliding window network coding is used without feedback, where results show that a low communication delay compared to block-based codes is achieved. However, in both schemes, packet loss rate is non-zero. In (Zverev *et al.*, 2019), the design parameters of the proposed scheme in (Wunderlich *et al.*, 2017) are used to achieve a zero packet loss rate in a fixed channel. However, in practice, the channel is varying with time, thus a zero packet loss rate cannot be attained without feedback. Moreover, (Zverev *et al.*, 2019) did not present analysis for the decoding complexity and the required buffer size at the receiver, which are the main constraints in IIoT devices.

We need network coding methods that decrease communication delay and consider existing limitations in IIoT networks, such as small buffer size and low computational power. Block-based RLNC and sliding-window RLNC with feedback have never been evaluated for a multicast scenario, so we don't know which one is the best for IIoT networks. In this work, we investigate the performance of block-based and sliding window network coding in terms of in-order delivery delay, the number of feedbacks and coded packets used in the network, the required buffer size for receivers, and the decoding complexity in a multicast scenario over packet erasure channels. To this end, we examine the effects of the various parameters on in-order delivery delay and other terms for sliding window and block-based network coding schemes and determines the best network coding scheme that meets the requirements of IIoT networks.

The rest of this thesis is structured as follows: In chapter one, we explain the requirements and problems of next-generation IIoT. In chapter two, we conduct a literature review and present existing solutions for next-generation IIoT. In chapter three, we present the necessary background in network coding, including random linear network coding, network coding approaches, and parameters of random linear network coding. Furthermore, we describe block-based RLNC and sliding window RLNC schemes in more detail. Finally, in chapter four, we investigate the impact of two specific network coding methods on IIoT networks as enabling technology in next-generation IIoT communications and present related results.

CHAPTER 1

OVERVIEW OF INDUSTRIAL INTERNET OF THINGS

IoT is a technology that connects things to a network through communication protocols (Hassan, 2018). It provides connectivity between physical and virtual objects to present intelligent services with or without human intervention anytime and anywhere (Hassan, 2018; Čolaković & Hadzialic, 2018). Things have the ability of sensing, actuating and communicating among themselves and with the environment. They capture data and send them through the internet network to one or multiple data processing and decision making centers (cloud or fog). This communication between different IoT devices or nodes of the IoT network needs a specific network architecture to meet application requirements. In this chapter, the network architecture of IoT will be investigated. Then the next generation of IIoT for industries along with its architectures will be discussed and at the end of the chapter, requirements of IoT networks are explained.

1.1 Architecture

In IoT, all of the devices are connected to the internet network to provide intelligent services. These devices are equipped with embedded sensors, transceivers, actuators, and processors. Sensors collect data from the physical environment and then send data to edge of the network or remote servers. The edge of the network performs an early data analysis or data preprocessing and then sends processed data to remote servers for further analysis (Sethi & Sarangi, 2017). The remote servers store and interpret processed data and make the required decisions. The decisions are sent to actuators to modify the physical world. This information process needs a specific network architecture. Fig. 1.1 shows the five layers model for architecture of IoT , including perception, transport, processing, application, and business layers as follows:

(a) The perception layer includes sensors that sense and collect information about the environment. They sense physical parameters such as temperature, humidity or distinguish other smart objects in the environment. The perception layer acts in the role of the physical layer in the OSI model.

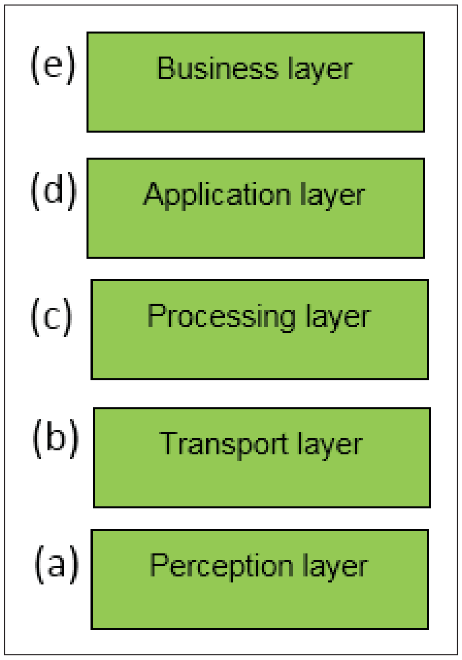

Figure 1.1 Five layer IoT architecture
Adapted from Sethi & Sarangi (2017)

(b) The transport layer communicates between the perception layer and the processing layer. Data collected by sensors is transferred from the perception layer to the upper layer and vice versa by the transport layer. It includes various technologies such as wireless networks, 2G, 3G, 4G, LAN, Bluetooth, RFID, and NFC.

(c) The processing layer which is also known as the middleware layer does a collection of activities on data collected by the sensors including storing, filtering, analyzing, and processing. It utilizes different technologies such as information discovery, machine learning, predictive modeling, databases, cloud computing, and big data processing to provide services for lower layers.

(d) The application layer presents various services to users. It defines different applications using IoT technology such as smart homes.

(e) The business layer is responsible for managing of IoT system including applications, business, profit models, and users' privacy. For example, different objects, users, and scenarios require various networks, different information processes, and different application methods (Xiaocong & Jidong, 2010). A business operation support platform (BOSP) in the business layer can classify different demands into different types such as network maintenance, device monitoring, and application maintenance (Xiaocong & Jidong, 2010).

In this architecture, data collected by the sensors is transferred to the processing layer which consists of cloud computers. Cloud computing has several advantages, for example, a large amount of data can be processed in cloud computers, and it provides scalability and flexibility which quickly meet application and business requirements. However, the edge of the network such as the sensors and network gateways can also process part of the data. This new approach in data processing is called fog computing.

In this section, we state a short explanation for IoT architecture. Given the importance of IoT technology in industrial applications, we explain it in more detail in the next section.

1.2 Next-Generation IoT for Industry 4.0

Industry 4.0 is the fourth industrial revolution to represent the newest generation of industries which is called smart factories. It benefits from the IoT through different industrial applications. The industrial operation uses cloud computing, fog computing, and artificial intelligence available in IoT architecture to get more improvement. However, there is a trade-off between latency and computation for time-critical decision-making IIoT applications that need low latency and large computation. While, using fog nodes on the edge of the network decreases latency, so it can only do an early analysis and computation. Thus fog computing can be considered as a complementary solution cloud computing. The requirements of industrial applications such as low-latency and highly reliable communications can be met somewhat by this architecture of next-generation IIoT (Peralta *et al.*, 2018). In this section, we will investigate next-generation IIoT architecture and then explain the requirements and problem of next-generation IIoT.

1.2.1 Next-generation IIoT architectures

Today, Industry 4.0 applications use cloud-based architectures because they can process a large amount of data and are scalable. However, using IoT in industrial environments causes new challenges which command an architectural adaptation. As we mentioned, IIoT applications need less communication delay and fast decision-making. Thus, fog nodes are integrated into industrial systems to meet these requirements. They offer early analysis and closed-loop control (Peralta *et al.*, 2018). Furthermore, such systems should be robust against faults, therefore, multi-cloud deployments are proposed to provide a fault-tolerant architecture and ensure the reliability of the system. In addition to the latter, industrial systems can profit from the connections with the best conditions. As a result, it causes a decrease in delays. Also, this new architecture can avoid dependencies on a single cloud.

Fig. 1.2 shows the next-generation IIoT architecture. It comprises of three layers including smart devices, fog nodes, and multiple clouds. Description and comparison of the layers are as follows (Peralta *et al.*, 2018):

(1) **Wireless Sensor Networks (WSNs):** Wireless sensor networks contain the main part of IIoT. They employ intelligent devices that are equipped with actuators, sensors, and smart things to gather measurements from the environment. These devices have the storage, battery and processing power limited. They collect data such as machine temperature or vibration measurements and transfer them to the upper layers. The upper layers then make decisions and send instructions to wireless sensor networks. Finally, wireless sensor networks perform a corresponding action or task according to the received instructions.

(2) **Fog:** The fog is an intermediate layer which is placed between the cloud and IIoT devices. The fog includes computing capabilities similar to clouds but deployed at the edge of the network (Sethi & Sarangi, 2017; Peralta *et al.*, 2018). The fog is close to the end-nodes, so it reduces communication latency and can support real-time services. It decreases the amount of data sent to the cloud by early data processing. Moreover, mobility and location-awareness are two important advantages of the fog for supporting moving devices.

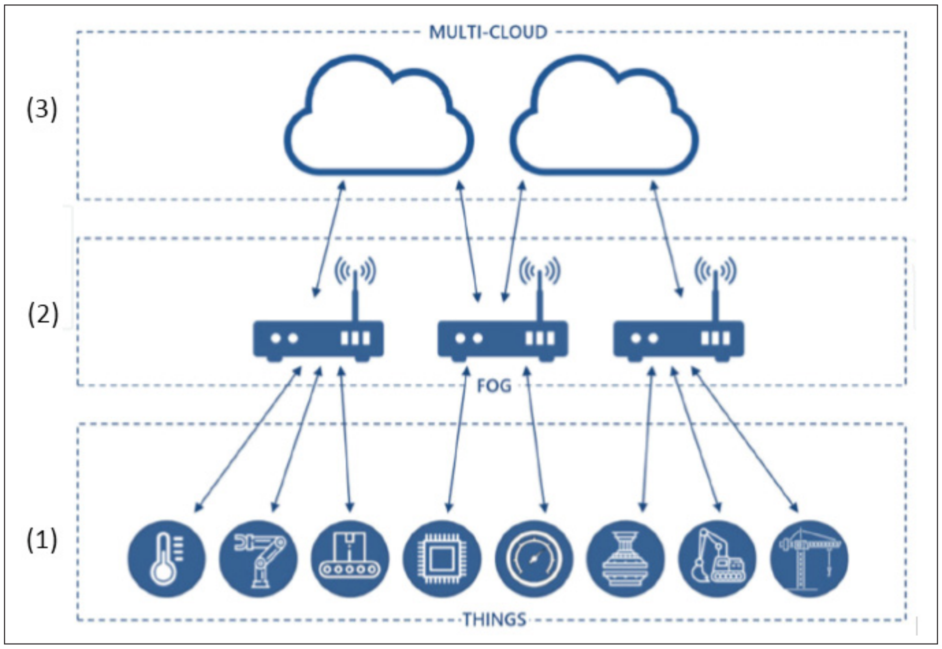

Figure 1.2 Next-generation IIoT architecture
Adapted from Peralta *et al.* (2018)

(3) **Multicloud:** The cloud can be considered as several distributed remote servers which can store and manage huge data (Peralta *et al.*, 2018). Cloud computing provides virtualized, elastic, controllable services and also powerful computational capabilities, enabling complex application systems (Peralta *et al.*, 2018). It includes different advantages such as fault tolerance against service outages and increased the system's security level. The system's security level can be improved if we store the information distributed into different clouds. Furthermore, applications can select the best available cloud resources with connections under the best conditions.

Although a large amount of data can be processed in the clouds, cloud computing has an inherent latency which is detrimental, especially for IIoT time-critical applications. On the other hand, we need to make autonomous decisions to prevent failures and manufacturing line downtimes, or optimize production. Thus, we employ fog computing. The fog is only useful for applications which are sensitive to latency and quick in decision making. It can process data directly on the edge of the network to reduce latency and jitter. However, the fog can only perform early processing due to its limitations. Therefore advanced processing and analysis should be delivered into clouds. Table 1.1 explains significant differences in WSNs, fog computing, and cloud computing .

Table 1.1 Comparison between WSNs, fog computing, and cloud computing
Adapted from Peralta *et al.* (2018)

Feature	**WSNs**	**Fog computing**	**Cloud Computing**
Latency	Very low	Low	High
Delay jitter	Very low	Low	High
Server location	—	Local	Internet
Client–server distance	—	One hop	Multiple hops
Location awareness	Yes	Yes	No
Distribution	Highly distributed	Distributed	Centralized
Mobility awareness	Guaranteed	Supported	Limited
Real-time interactions	Guaranteed	Supported	Limited

The efficiency and scalability of the fog, whereas the powerful storage and computing resources of the cloud are advantages of using the three-layer architecture (Peralta *et al.*, 2018). These advantages help industrial applications to meet their requirements. For example, smart manufacturing can benefit from both fog and cloud computing to increase product quality by real-time sensor analysis or perform predictive maintenance of their machines, since fog computing can meet the requirements of real-time applications, and cloud computing can provide greater computing power and system management (Peralta *et al.*, 2018).

1.3 Requirements of next-generation IIoT and problem description

Factories employ IIoT networks in time-critical services that require a reliable and low delay communication (Peralta *et al.*, 2018). However, IIoT networks cannot achieve these requirements readily because noise or multipath interferences existing in WSNs cause packet loss (Zverev *et al.*, 2019), this packet loss increases communication delay. Since the next packet cannot be delivered unless the previous packets are delivered, a lost packet leads to a retransmission which causes a communication delay. Furthermore, sensors deployed in IIoT networks have constraints, such as the small buffer size and low computational power, which add more requirements for designing reliable IIoT networks.

Thus, we need to design a reliable IIoT network with a low delay communication and considers the existing limitations in sensors. There are multiple methods to design a high-performance IIoT network and it's difficult to select the best. Our literature review will provide an overview of these methods. This thesis takes a closer look at two specific methods that are identified in the literature review, to determine which is most suitable for time-critical services.

CHAPTER 2

LITTERATURE REVIEW OF PACKET-LOSS RECOVERY METHODS

Firstly, in this chapter, we explain the concept of erasure networks. Then, we investigate packet loss recovery methods in an erasure network and discuss their advantages and disadvantages.

2.1 Erasure Networks

To define the concept of erasure networks, first, we need to explain two basic definitions of the Binary Erasure Channel (BEC) and the Packet Erasure Channel (PEC).

Binary erasure channel (BEC) : A binary erasure channel is a communication channel model, where the source X sends a bit for a receiver Y. The receiver receives the same bit or a message that shows the sent bit is not received because it is erased in the communication channel with a probability of ϵ (Peled, Sabag & Permuter, 2019). Fig. 2.1 represents this model.

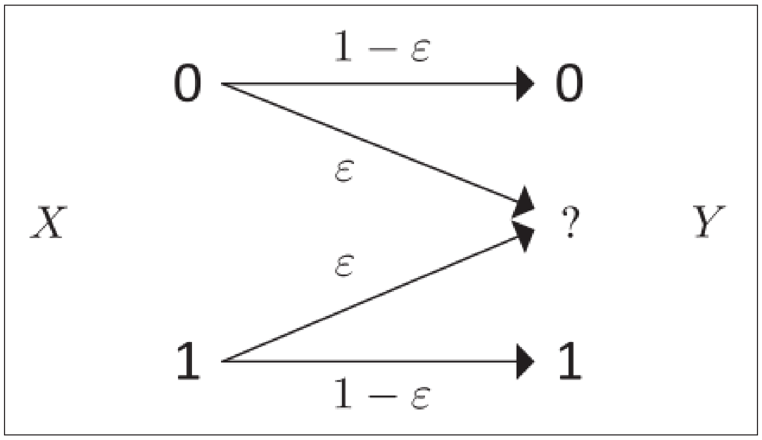

Figure 2.1 Binary erasure channel with erasure parameter ϵ
Adapted from Peled *et al.* (2019)

Packet erasure channel : Packet erasure channel is a communication channel model in which packets are received or erased. Each packet contains some data.

Regarding these definitions, the concept of erasure networks is explained as follows:

Erasure networks: In erasure networks, communication links consist of packet erasure channels.

2.2 Packet loss Recovery Methods

There are three main methods for recovery of packet losses in an erasure network, including ARQ (Karzand *et al.*, 2017), fountain codes (Luby, 2002; Shokrollahi, 2006), and network coding (Sundararajan *et al.*, 2017). In this section, we explain each features.

2.2.1 ARQ

ARQ method is considered as the simplest method for dealing with lost packets in an erasure network. In this method, the receiver informs the source about the status of the sent packets by sending feedback. Then, the source retransmits lost packets for the receiver. ARQ can be used as link-by-link or end-to-end in source and destination. However, this method has a drawback which is that this method requires a reliable feedback. If feedback is unavailable in the network, or feedback suffers from lots of errors, ARQ cannot be implied. In addition, in a multicast or broadcast scenario, using ARQ creates a large number of feedback packets and some repetitive packets for some receivers (Gabriel, Wunderlich, Pandi, Fitzek & Reisslein, 2018). Overall, ARQ is an optimal method in terms of throughput and delay in unicast scenarios with no delay between the source and receiver (Gabriel *et al.*, 2018). It ensures optimal throughput in the network since after receiving each packet, the next packet is sent. It is also an optimal solution in terms of packet delay since this next packet is a new unknown packet for the receiver and satisfies the lowest possible packet delay (Sundararajan *et al.*, 2017).

2.2.2 Fountain codes

Fountain codes is an another solution for the recovery of lost packets. However, before going into the technical details, let's investigate it from a historical point of view.

Block codes such as the RS (Roca *et al.*, 2013) can be used for recovery of lost packets, however, they have a rigid structure. For example, in a multicast scenario, the source adjusts the code rate based on the received channel estimation feedback from users. In practice, the channel's situation changes during communication. Thus, if the channel situation is worst than the channel estimation of the user, the block code cannot recover lost packets. On the other hand, if the channel situation is better than the channel estimation of the user, additional data is transmitted.

Fountain codes are block codes that present another solution for the recovery of lost packets. In a fountain code such as Luby Transform code (LT) (Luby, 2002), a file is divided into K symbols, that are combined linearly by coefficients selected from a binary field. Raptor codes, a modified version of LT codes, are another class of fountain codes (Shokrollahi, 2006). They encode packets in two steps. In the first step, packets are encoded using RS or low-density parity-check code (LDPC), and re-encoded using an LT code. Fountain codes are rate-less because the rate of coded packets can change readily, and the source can produce an infinite number of coded packets. The rate-less property creates a flexible structure. These codes are called fountain codes because they behave like a fountain. Indeed coded packets are similar to water drops. Users can fill their containers with water drops that fall from a fountain. If users can collect a sufficient number of the coded packets (water drops), they can start the decoding process. Fig. 2.2 demonstrates the concept behind fountain codes. Following that, we will explain the disadvantages of fountain codes and introduce network coding.

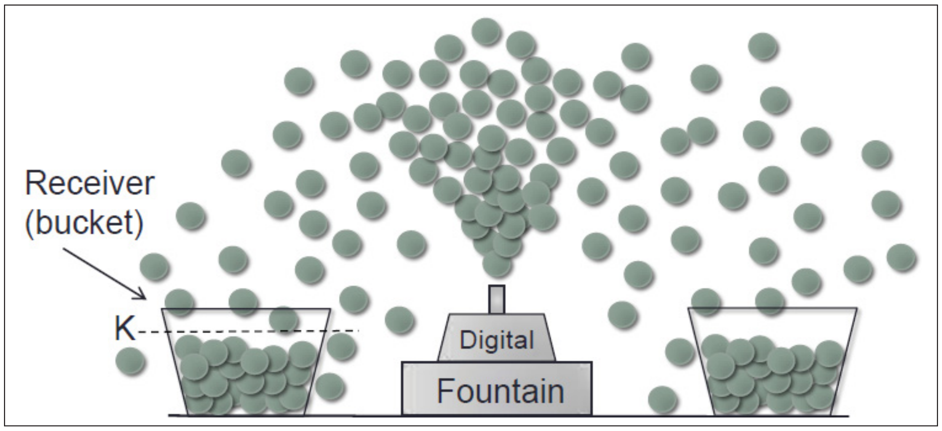

Figure 2.2 Fountain code
Adapted from Lopes (2014)

2.2.3 Network coding schemes

As mentioned earlier, ARQ performs well in terms of throughput and delay in a unicast scenario (Gabriel *et al.*, 2018). However, its performance in multicast and broadcast scenarios is limited. On the other hand, fountain codes can resolve the problem of block codes in multicast and broadcast scenarios. Fountain codes were proposed for point-to-point erasure channels (Sundararajan *et al.*, 2017). These codes are end-to-end codes where encoding and decoding are performed only at the source and destination, respectively. However, they aren't useful for scenarios in which intermediate nodes can re-encode packets. We clarify this issue by using the two-link tandem network in Fig. 2.3.

Here, node A sends packets to node B, then node B sends them to node C. The loss probabilities of the $A \rightarrow B$ and $B \rightarrow C$ links are α and β, respectively. If we use fountain codes in this scenario, the maximum capacity of the network is (1-α)(1-β), because node B cannot perform recoding and generate more coded packets. However, if node B decodes and recodes, the maximum capacity is $\min\{(1-\alpha),(1-\beta)\}$, which is greater than (1-α)(1-β). However, decoding and recoding in the intermediate nodes increases the communication delay (Pakzad, Fragouli & Shokrollahi, 2005).

Network coding can resolve this issue as it can generate any number of coded packets in nodes corresponding to the loss probability of each link, without performing additional operations in intermediate nodes. Recoding in intermediate nodes is one of the main properties of network coding (Fouli *et al.*, 2018). Furthermore, each node can generate any number of coded packets by combining existing uncoded/coded packets for each block of code before each block of code is received completely. This property of network coding is known as on-the-fly coding (Fouli *et al.*, 2018).

So far, we have explained the advantages of network coding over ARQ and fountain codes for packet loss recovery. Network coding can be applied using several techniques to recover lost packets in erasure networks. The following are the three most widely used, including Online network coding (Sundararajan *et al.*, 2017), IDNC (Douik *et al.*, 2017), and sending redundant

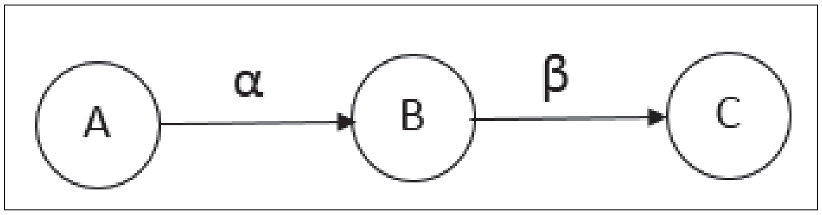

Figure 2.3 Two-links tandem network
Adapted from Sundararajan *et al.* (2017)

coded packets based on network coding (Karzand *et al.*, 2017; Wunderlich *et al.*, 2017). In the next subsections, we explain each of them in details.

2.2.3.1 Online Network Coding

ARQ is a simple scheme that can achieve optimality in terms of throughput and delay in a unicast scenario (Sundararajan *et al.*, 2017). However, network coding is an optimal solution for multicast and broadcast scenarios, where ARQ does not have a good performance. In ARQ, no packets are stored at the source to create a block. Therefore, coding is performed in an online manner(Sundararajan *et al.*, 2017), and this advantage cannot be ignored. Thus, the question of how to combine network coding with ARQ comes in notice. The combination of network coding with ARQ is not easy. In a network which uses ARQ, the source deletes the received packets based on ACK received from the user and sends a new one. However, in a coded system, receiving an ACK for a coded packet would not be useful since the source would not be able to determine which packet should be combined in the next transmission. On the other hand, if the size of the block of code is increased, the receivers should collect a large number of coded packets to start the decoding process, leading to an increase in the decoding delay (Sundararajan *et al.*, 2017). Moreover, the source would have to store a large number of source packets, which is inefficient. In an ordinary network coding scheme, no packets can be deleted by the source, unless all the packets are decoded. This scheme is known as drop-when-decoded (Sundararajan *et al.*, 2017) and is not optimal. To resolve this problem, we consider the following example (Sundararajan *et al.*, 2017). Assuming that we have n packets p_1, p_2,... , p_n at the source. $(n-1)$ linear combinations $(p_1 + p_2)$, $(p_2 + p_3)$, ... , $(p_{n-1} + p_n)$ are received at the receivers.

Upon receiving feedback, one of the questions remains that which packets can be deleted in the next linear combination? Based on the drop-when-decoded scheme, the source cannot delete any packets because the receivers have not decoded any packets. However, the receivers need only one packet to start the decoding process. Thus, the source can store an arbitrary packet to guarantee reliable transmission (Sundararajan *et al.*, 2017).

Online network coding was proposed based on the above example. It takes advantage of the feedback used in ARQ, and creates a chain of coded and uncoded packets. This is known as online network coding, since the source adjusts the coding decisions based on the erasure patterns observed at the receivers (Sundararajan, Shah & Médard, 2008). To explain the online network coding method, we start by defining the concept of seeing a packet in the following (Sundararajan *et al.*, 2017).

Seeing a Packet : A receiver is assumed to have seen a packet p_n if it has collected enough packets to calculate a linear combination of the form $(p_n + q)$, where q is itself a linear combination including only packets with an index greater than that of p_n. Decoding can be started, when $q = 0$.

Online network coding acknowledges received packets based on seeing packets rather than decoded ones such as in conventional ARQ (Sundararajan *et al.*, 2017). Thus, the source drops a packet when it is seen by all the receivers. This scheme is known as drop-when-seen (Sundararajan *et al.*, 2017).

Table 2.1 shows an online network coding scheme based on the drop-when-seen algorithm for a broadcast scenario in an erasure network. There are two receivers: A and B. The status of the source's queue is shown after and before the transmission. At the beginning of any transmission, the source determines which packets should be combined based on received ACKs from A and B. If both of them need the same packet, the source transmits it. Otherwise, it sends their XOR. This scheme guarantees that all of the receivers get their next unseen packets. From the example in Table 2.1, in time slot 1, $p1$ is received by A but not by B. In time slot 2, the XOR of $p1$ and $p2$ is received by A and B. Receiver A decodes $p2$, and receiver B calculates $p1$ as seeing packet. In time slot 2, the source drops packet $p1$ as it is seen by both receivers. Receiver B decodes $p1$, $p2$, and, $p3$ in time slot 4. Receiver A decodes packets $p3$ and $p4$ in time slot 6. The source drops packets $p2$, $p3$, and $p4$ in time slots 3, 5, and 6, respectively.

Table 2.1 Drop-when-seen algorithm
Adapted from Sundararajan *et al.* (2017)

Time	**Source's queue**	**Transmitted packet**	**Channel state**	**Decoded(A)**	**Seen but not decoded**	**Decoded(B)**	**Seen but not decoded**
1	P1	P1	A(OK), B(NO)	P1	-	-	-
2	P1, P2	P1⊕P2	A(OK), B(OK)	P1, P2	-	-	P1
3	P2, P3	P2⊕P3	A(NO), B(OK)	P1, P2	-	-	P1, P2
4	P3	P3	A(NO), B(OK)	P1, P2	-	P1, P2, P3	-
5	P3, P4	P3⊕P4	A(OK), B(NO)	P1, P2	P3	P1, P2, P3	-
6	P4	P4	A(OK), B(OK)	P1, P2, P3, P4	-	P1, P2, P3, P4	-

Online network coding has several advantages: It reduces the queue size at the source (Sundararajan *et al.*, 2017); it benefits from simple queue management since the source does not need to save linear combinations of packets, and instead it can store the original uncoded packets and create a service based on a simple first-in-first-out queue (Sundararajan *et al.*, 2017); each coded packet is sparse, which decreases the decoding complexity (Sundararajan *et al.*, 2017); can be used in a tandem network of broadcast links provided, as some modifications are done (Sundararajan *et al.*, 2017). However, despite all these benefits, this technique cannot guarantee immediate decoding of the seen packets, since this requires the receiver to collect a sufficient number of unknown packets before starting decoding process (Sundararajan *et al.*, 2017).

2.2.3.2 Opportunistic and Instantly Decodable Network Coding

Opportunistic Network Coding (ONC) is a network coding scheme that recovers lost packets. In an ONC scheme, encoding is performed wherever the opportunity occurs, and there are no mechanisms for increasing the number of encoding opportunities (Alic & Svigelj, 2018). Encoding is performed only in the source, which can be a base station or a device, and decoding is also performed only in the sinks (devices). There is no intermediate node. In general, encoding can be done on different streams or the same stream, though it is usually performed on different streams (Alic & Svigelj, 2018).

Fig. 2.4 shows an ONC scheme. There are four packets in the queue of node *B*. Nodes *A*, *C*, and *D* are its neighbors. Each of them has saved some packets in their buffer. Node *B* can make a different encoding decision and send a coded packet to its neighbors. Fig. 2.4 shows these decisions. For the first decision, node *C* can decode, but node *A* cannot. For the second decision, nodes *C* and *A* can decode. For the third decision, all of them can decode. Thus, the third decision is the best in this scenario.

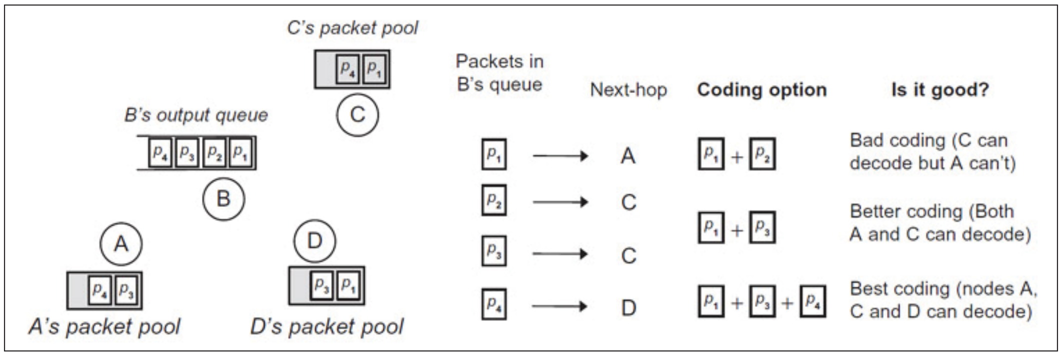
C's packet pool
B's output queue
A's packet pool
D's packet pool
Packets in B's queue
Next-hop
Coding option
Is it good?
Bad coding (C can decode but A can't)
Better coding (Both A and C can decode)
Best coding (nodes A, C and D can decode)

Figure 2.4 Opportunistic network coding
Adapted from Muriel Médard (2012)

Several works have studied ONC and have demonstrated that ONC has encoding complexity and scalability issues, and is not suitable scheme for the real-time scenario (Douik *et al.*, 2017).

Recently, after many analyses, a new scheme, known as IDNC, was proposed. IDNC is a subclass of ONC where each coded packet can be decoded instantly at the receivers such that no receiver is required to store any coded packets for future decoding opportunities (Douik *et al.*, 2017). This scheme relies on a cooperation between devices in the networks. IDNC significantly decreases the decoding complexity since each coded packet is decoded either instantly or discarded. Therefore, it is appropriate for devices with low battery and computational power (Douik *et al.*, 2017).

Fig. 2.5 shows an IDNC scheme example. It consists of a base station and five receivers. Here receivers 1, 3, and 5 did not receive some packets due to the lossy property of the wireless channel. There are two scenarios for the recovery of lost packets. In the first scenario, the base-station sends the required packets in three time slots. Thus, it sends packets 1, 2, and 4 in three time slots. Meanwhile, if we consider collaboration between devices, we need two time slots for data transmission. In the first time slot, receivers 2 and 4 send packets 2 and 1, respectively. Then in the second time slot, receiver 2 transmits 1⊕4, and receiver 4 communicates with packet 4.

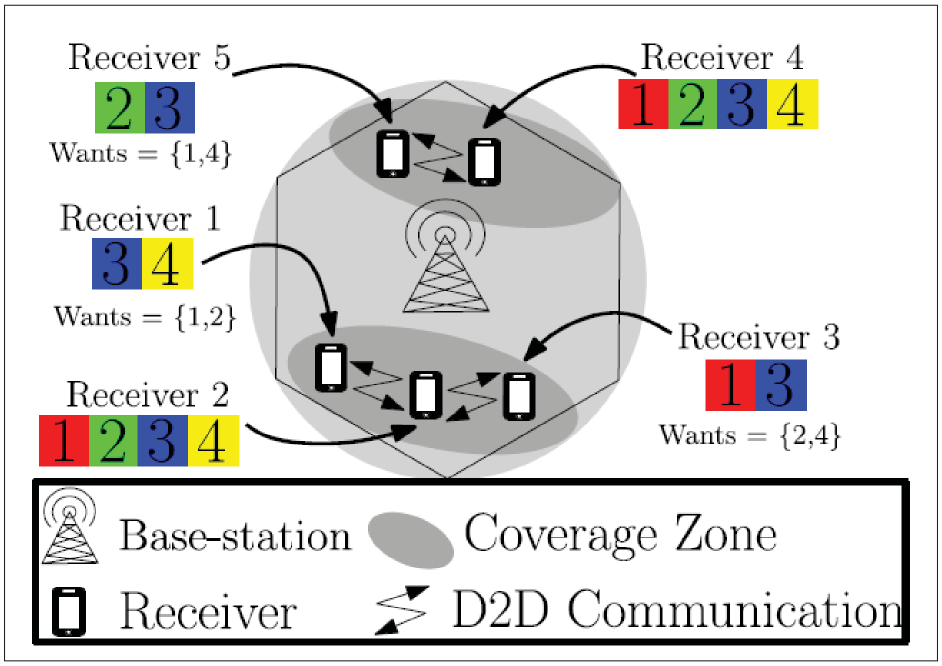

Figure 2.5 Instantly Decodable Network Coding
Adapted from Douik *et al.* (2017)

In addition to the ONC and IDNC schemes, generation-based RLNC presented in section 3.5 is another scheme used for lost packet recovery. This scheme, that we will detail in subsection 3.6, recovers lost packets by adding redundant coded packets in end of each block of code. A comparison is made between these schemes in Table 2.2. It is shown that the throughput is optimal for generation-based RLNC, while it is sub-optimal for ONC and IDNC. Moreover, generation-based RLNC suffers from large delays while ONC and IDNC go through moderate delays. The decoding complexity is low for IDNC compared to generation-based RLNC. Also, IDNC does not need any buffers. Conclusively, easy encoding and decoding, no buffer requirement, and instantaneous decoding are the most important advantages of IDNC (Douik *et al.*, 2017). However, dependency on the feedback is an important disadvantage of ONC and IDNC, since lost packet can be feedback packets in a real network. Furthermore, IDNC is a suitable choice for applications in which each source packet carries new information for the receiver irrespective of its order since IDNC cannot guarantee to deliver packets in-order, as explained by (Skevakis & Lambadaris, 2017; Aboutorab, Sadeghi & Sorour, 2014).

Table 2.2 Performance of Block-based RLNC, ONC, and IDNC
Adapted from Douik *et al.* (2017)

Criterion \Scheme	**Generation-Based RLNC**	**Opportunistic Network Coding**	**Instantly Decodable Network Coding**
Throughput	Optimal	Sub-optimal	Sub-optimal
Delay	Huge delay	Moderate depending on the scheme	Moderate depending on the scheme
Complexity	Large field size	Depends on the scheme	Binary field
Encoding	Mix using random independent coefficients	Mix using the diversity of lost and received packets	Mix using binary XOR
Decoding	Complexity cubical with the number of packets	Moderate depending on the scheme	Simple binary XOR
Progressive Decoding	Decoding is performed after getting the whole of frame	Depends on the scheme but usually better than RLNC	Instantaneous decoding
Overhead	Moderate depending on the scheme	Moderate depending on the scheme	Minimal
Buffer Size	As large as the frame size	Moderate depending on the scheme	No need for buffer
Feedback Load	Minimal feedback and can run even without feedback	More or less heavy depending on the scheme	Performance heavily depends on feedback
Broadcast Efficiency	Optimal	Sub-optimal	Sub-optimal
Multicast Efficiency	Inefficient	Depends on the scheme	Depends on the scheme

Finally, IDNC can be implemented in two distinct ways, namely centralized device-to-device communication and distributed device-to-device communication (Douik *et al.*, 2017). In the centralized method, a central controller in the network decides on the packets that should be communicated between devices and which devices should transmit packets (Douik *et al.*, 2017). This method is suitable for small scale networks and increases the signaling overhead in the network (Douik *et al.*, 2017). On the other hand, in distributed device-to-device communication, there is no central controller and devices collaborate to recover the lost packets (Douik, Sorour, Tembine, Alouini & Al-Naffouri, 2014). This method decreases the signaling overhead in the network (Douik *et al.*, 2014).

CHAPTER 3

KEY PRINCIPLES OF NETWORK CODING SCHEMES

3.1 Introduction

Network Coding (NC) is a new method for communicating between entities in the networks. Entities in the network coding method generate linear combinations of ingoing packets and send them as outgoing packets. This communication method increases network throughput and network reliability. In the beginning, a lot of studies were carried out on network coding theory, and then it was applied to different applications such as multicast and broadcast of packets, security, content distribution networks, and distributed storage systems. In the first part, we investigate NC from a historical point of view and then explain Random Linear Network Coding (RLNC) as one of the important methods of NC. Then, we study different network coding approaches and parameters of RLNC. Finally, we introduce the block-based RLNC and the sliding window network coding as two effective solutions to decrease communication delay.

3.2 Network Coding

In traditional point-to-point multicast scenarios communications, sources send their information bits and sinks receive a copy of them. In this scenario, the main problem is the calculation of the coding rate region. The coding rate region is the set of all possible data rates that can be sent and received by sources and sinks, respectively (Yeung, 2011). The Max-flow Min-cut theorem was proposed as a solution for resolving the coding rate region problem. This theorem provides an upper bound on the coding rate region but doesn't tell us how to achieve it. Thus, different techniques such as multicast tree design, attempted to achieve the maximum flow proposed by the Max-flow Min-cut theorem in a multicast scenario. However, designing a multicast tree requires complicated computations and does not always provide the maximum flow. Eventually, network coding was introduced as a promising method as it can achieve the coding rate region proposed by the Max-flow Min-cut theorem. In addition, it offers a tractable complexity compared to multicast trees. On the other hand, network coding has additional advantages as compared to traditional methods, including lower bandwidth and increased network robustness and reliability. As a result, it is currently employed in different applications such as security, data storage, channel coding, etc. In sections 3.3, we review and compare different network coding algorithms including Deterministic Network Coding and Random Linear Network Coding.

3.3 Random Linear Network Coding

In a network based on network coding, uncoded and coded packets are combined linearly by coefficients selected from a finite field in the source and intermediate nodes. These coefficients can be chosen in two different methods, including randomly and non-randomly. Depending on these methods, we have two network coding algorithms.

In centralized network coding, the coding coefficients are chosen carefully from a finite field with a specified size and are fixed for all the intermediate nodes. Centralized network coding is known as deterministic network coding (Kötter & Médard, 2003). When coding coefficients are fixed and stable throughout the network, small amounts of information are transmitted, which decreases network overhead. Deterministic network coding is suitable for small networks. However, it is not a proper solution for extensive networks with a large number of the intermediate nodes because the design of fixed coefficients becomes challenging in such cases. Thus, decentralized network coding was proposed for complex networks. In decentralized network coding, the coding coefficients are chosen randomly from a sufficiently large finite field. If the coefficients are chosen randomly and distributed among all intermediate nodes of the network, the robustness of the network in the presence of these changes or link failures increases (Ho, Médard, Kötter, Karger, Effros, Shi & Leong, 2006). This approach is known as RLNC (Muriel Médard, 2012; Sheng, Yang, Yu, Vasilakos, McCann & Leung, 2013).

RLNC has several advantages and disadvantages. Four noticeable advantages of RLNC are: reduced bandwidth requirements, increased throughput, increased robustness (reliability) of the network against deliberate malfeasances and link failures, and also reduced communication latency. While RLNC has such advantages, its weaknesses are not negligible. In fact, it is sensitive to errors because just one single packet can affect all the packets in the network when they are combined. In addition, the complexity of decoding in the sinks can be considered as an undeniable drawback of RLNC.

In this project, we will simulate schemes based on RLNC, where coefficients are chosen randomly from a sufficiently large finite field, to increase the reliability of IIoT networks and decrease communication latency and download time.

3.4 Network Coding Approaches

In this section, we explain how network coding can increase the throughput and reliability of wireless networks and introduce different network coding approaches.

Traditional computer networks were designed based on the store-and-forward method in which received packets are buffered in routers and forwarded without any change. However, in the network coding approach, routers combine packets before forwarding them. Combined packets can be related to a specific session or any other session. Depending on the considered scenario, network coding can increase the throughput or reliability in the network. Based on used scenario, network coding can be categorized into intra-session coding, inter-session coding, Analog Network Coding (ANC), and digital network coding (Muriel Médard, 2012; Al-Kofahi & Kamal, 2016).

(1) **Intra-Session Network Coding:** In this scenario, routers only combine packets that are sent to a destination in a specified session. Thus, routers are not required to decode the packets which are only decoded at the destination when it receives a sufficient number of coded packets. To clarify this approach, we have shown two examples presented in Fig. 3.1.a and 3.1.b. In Fig. 3.1.a, Amir sends his packets to Marwa. However, he is in a dead spot with 80% packet loss. Amir selects the best channel toward Marwa and has to send each packet 5 times because all wireless links are lossy. Then, packets are transferred to Marwa by a relay just once. While diversity can improve data transfer, combining it with

network coding can be an effective solution. Thus, Amir continues to send his packets. Each relay collects packets and combines them linearly and then sends a coded packet to Marwa. Marwa collects a set of coded packets and can decode them if she collects a sufficient number of coded packets.

In Fig. 3.1.b, the source sends two packets, $P1$ and $P2$, to router R and it broadcasts them to destinations $D1$ and $D2$. However, $D1$ only receives packet $P1$ and $D2$ only receives packet $P2$, hence, router R should send both packets again. However, it can combine packets and then send a coded packet rather than sending them separately. Each destination can decode the coded packet based on the packet it has already received. As shown in Fig. 3.1, intra-session network coding increases the reliability of the network. It also shows which intra-session network coding can be applied to different applications such as energy (Fragouli, Widmer & Le Boudec, 2008; Wu, Chou & Kung, 2005), secrecy (Cai & Yeung, 2002; Jaggi, Langberg, Katti, Ho, Katabi & Médard, 2007), content distribution (Gkantsidis & Rodriguez, 2005), and distributed storage (Jiang, 2006).

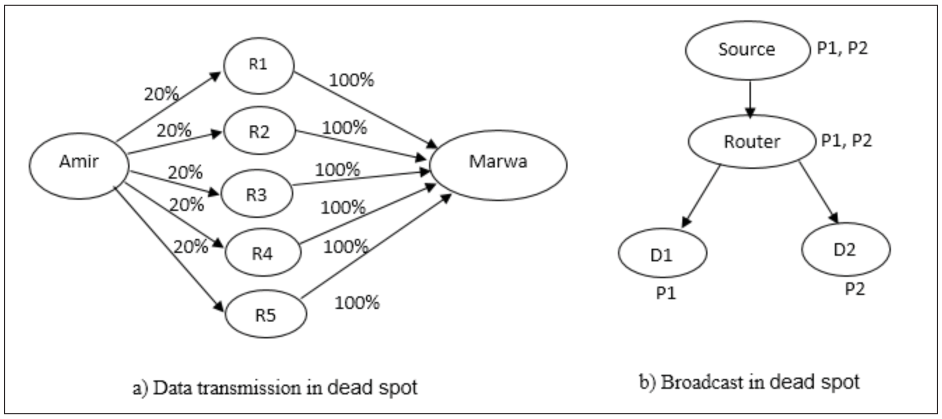

Figure 3.1 Intra-Session Network Coding
Adapted from Muriel Médard (2012)

(2) **Inter-Session Network Coding:** In this scenario, routers can combine the received packets from different communication sessions when they arrive at a common intermediate node in the network. Fig. 3.2 depicts a scenario in which Amir and Marwa are going to communicate with each other without using network coding technique. They communicate

with each other by a router because they do not stand in a common range of communication. As seen in Fig. 3.2, four-time slots are needed to transmit two packets between Amir and Marwa because each packet should be transmitted in a single time slot. However, we can decrease the number of packet transmissions by applying network coding technique, Fig. 3.3 shows this scenario. At first, Amir sends a packet in the first time slot and then Marwa sends her packet in the second time slot. The router receives and combines them and sends a coded packet to Amir and Marwa simultaneously in the third time slot. As a result, packets are transmitted in three-time slots instead of four. Finally, Amir and Marwa can decode the coded packet by existing packets present in their buffer. Thus, sending a single coded packet in a time slot rather than two packets in two-time slots increases network throughput.

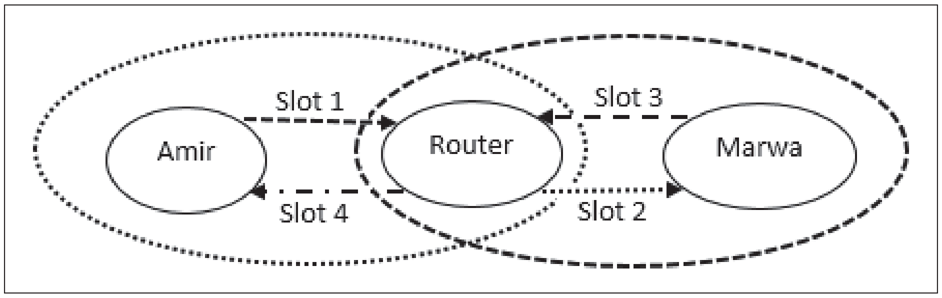

Figure 3.2 Data transmission without Network Coding
Adapted from Muriel Médard (2012)

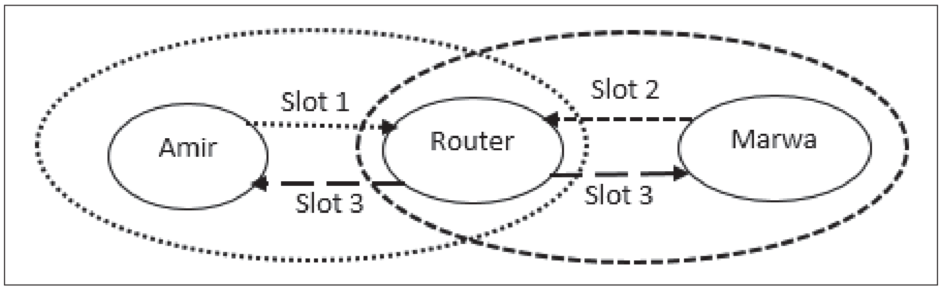

Figure 3.3 Inter-Session Network Coding
Adapted from Muriel Médard (2012)

(3) **Analog Network Coding (ANC):** In ANC, routers combine information bits directly instead of mixing the packets. Indeed, they mix analog signals instead of packets. For example, when two packets are sent simultaneously, they interfere with each other. However, the interfered signal can be considered as a linear combination of the two native signals (Muriel Médard, 2012). Fig. 3.4 demonstrates the ANC scenario. In this scenario, we can send packets in two-time slots instead of three since Amir and Marwa can send their packets simultaneously. Packets are combined in the router as an interfered signal and are sent to Amir and Marwa. Then, each user can decode its received signal to extract its own packet. Analog network coding is known as physical-layer network coding (Muriel Médard, 2012). In this scenario, symbol-level synchronization, carrier-frequency synchronization, and carrier-phase synchronization should be taken into consideration (Muriel Médard, 2012).

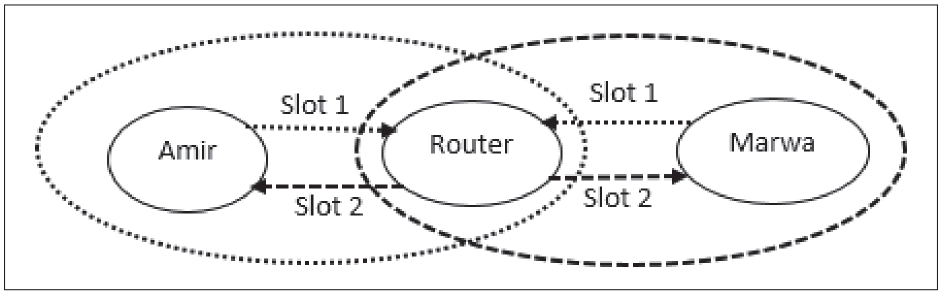

Figure 3.4 Analog Network Coding
Adapted from Muriel Médard (2012)

(4) **Digital Network Coding (DNC):** In DNC, unlike ANC in which information bits are mixed, routers mix received packets and send them to destinations. In DNC, the coding process is performed in the electronic domain when packets are received while in ANC, the coding process takes place in the electromagnetic domain at the physical layer (Al-Kofahi & Kamal, 2016). Moreover, in ANC, we do not need to receive the packets or their actual representations in bit format because this technique relies on the interference of packets. Fig. 3.5 shows a Digital Network Coding approach, and also communication between Amir and Marwa in three-time slots. At first, Amir sends a packet and Marawa sends her packets. The router mixes the received packets and sends a coded packet for

Amir and Marwa. Finally, they decode coded packet using their own packet. Inter-session network coding and intra-session network coding are a type of digital network coding since it combines packets instead of information bits.

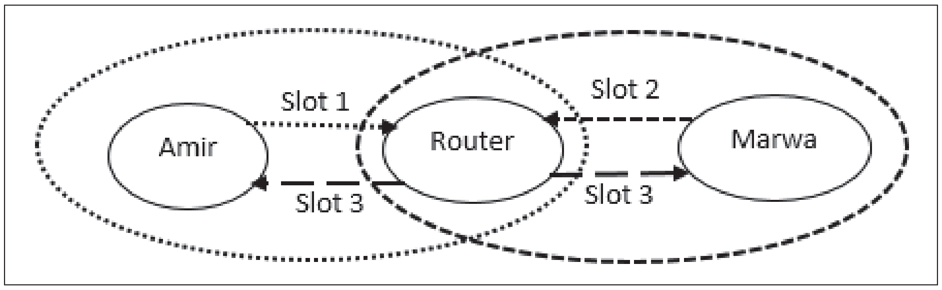

Figure 3.5 Digital Network Coding
Adapted from Muriel Médard (2012)

Each of these approaches has features that are suitable for different purposes. Table 3.1 explains some of these features. For example, Intra-session Network Coding increases reliability while Inter-session Network Coding increases throughput. Also, ANC improves the throughput, since routers amplify and deliver combined signals. It is noted that since inter-session network coding and intra-session network coding are a type of digital network coding, we did not explain digital network coding in the table 3.1 separately.

In this project, we will use intra-session network coding and digital network coding approaches to improve the reliability of IIoT networks in a one-hop multicast scenario.

3.5 Random Linear Network Coding Parameters

In RLNC system, the source divides a file of size B into $\lceil \frac{B}{m} \rceil$ pieces of size m, which are called symbols (original packets), then it creates generations. Each generation includes g symbols of size m. Thus, the total number of generations is $\lceil \frac{B}{g.m} \rceil$. Then, the source creates coded symbols, where each coded symbol consists of a linear combination of g symbols which are mixed by coefficients chosen from a sufficiently large finite field. Finally, the source transmits the coded packets that include coded symbols, along with their corresponding coding vectors (coefficients),

Table 3.1 Comparison between different approaches to network coding
Adapted from Muriel Médard (2012)

Features / Scenarios	**Intra-session Network Coding**	**Inter-session Network Coding**	**Analog Network Coding**
Coding method	Code packets from the same session	Code packets from different sessions	Code signals by transmitting them concurrently
Role of routers	Routers typically re-encode Decoding can be done at destination(s)	Routers typically decode and re-encode the packets	Routers need not decode, they amplify and forward mixed signals
Application	Typically used for multicast applications	Typically used for unicast applications	Used for both unicast and multicast
Used topologies and environments	Suitable for lossy links and unpredictable topologies	Suitable for low-loss static topologies	Suitable for high SNR channels
Benefit	Improves reliability	Improves throughput	Improves throughput

towards the sink (Heide, Pedersen, Fitzek & Medard, 2011). Henceforth, we employ the terms coded packet and coded symbol interchangeably throughout the proposed project. Fig. 3.6 shows the structure of a coded packet.

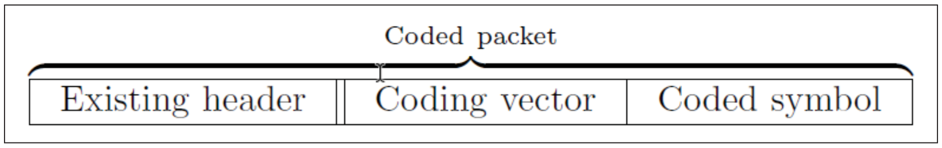

Figure 3.6 Structure of a coded packet
Adapted from Heide *et al.* (2011)

In RLNC system, recoding can also be performed in intermediate nodes before reaching the sink. The sink starts the decoding process if it collects a sufficient number of coded packets, which is equivalent to generation size g. Such a system has several challenges, including the decoding complexity, the overhead of the header due to the use of the coding vectors, and the overhead of the linear dependence of coded packets, i.e., transmission of redundant coded packets (Heide *et al.*, 2011). RLNCs can deal with these challenges by adjusting different parameters as follows.

Generation size : Generation sizes should be large enough to decrease the overhead of linear dependence (Heide *et al.*, 2011). This approach decreases the number of additional transmissions, which in turn decreases transmission energy (Heide *et al.*, 2011). However, choosing too large a generation size increases decoding delay, which is not suitable for real-time services such as voice or video streaming (Heide *et al.*, 2011). On the other hand, choosing a too large generation size increases decoding complexity and header overhead, which in turn increases the coding energy consumption and decreases coding throughput (Heide *et al.*, 2011). Besides, a large header overhead is not a proper choice for the networks transmitting small packets, such as IoT networks.

Signaling is another aspect of network coding. When a generation is decoded at the sink, the receiver sends an ACK to the source. Choosing a large generation size decreases the signaling overhead in the network since the receiver is required to send fewer ACKs to the source (Heide *et al.*, 2011).

Field sizes : A large field size decreases the overhead of linear dependence (Heide *et al.*, 2011). However, it increases decoding complexity and header overhead (Heide *et al.*, 2011), hence decreasing coding throughput (Shojania & Li, 2007).

Density : The proportion of non-zero scalars in a coding vector is called the density of the coded packet (Heide *et al.*, 2011). When the density of the coded packet increases, the overhead of linear dependence decreases, but the overhead of the header increases (Heide *et al.*, 2011). Besides, higher densities imply higher decoding complexity, and therefore lower coding throughput (Heide *et al.*, 2011).

Overall, linear dependence and header overheads depend on the generation size, field size, and density. These parameters should be designed in such a way that linear dependence and header overheads are reduced. However, there are trade-offs between them and other aspects such as decoding complexity and coding throughput. Thus, adjusting these parameters is a big challenge in network coding.

In this project, we will simulate schemes with different generation sizes to observe the effects of generation sizes on decoding complexity and communication delay in IIoT networks. We assume that there is no linear dependency between coefficients in coded packets. Furthermore, we will combine source packets with non-zero coefficients to produce coded packets with high densities.

3.6 Block-based RLNC versus sliding window RLNC

RLNC provides redundant coded packets to recover lost packets in an erasure network. The conventional RLNC scheme is known as generation-based RLNC or block-based RLNC. In block-based RLNC, each coded packet consists of a linear combination of source packets which are mixed using coefficients taken from a sufficient large finite field. The number of source packets used in each coded packet determines the block size or generation size of the scheme. Fig. 3.7 (a) shows a block-based RLNC scheme (Fouli *et al.*, 2018). Non-systematic and systematic block-based RLNC schemes are derived from block-based RLNC. In a non-systematic block-based RLNC scheme shown in Fig. 3.7 (b), the source sends all packets as coded packets. In this scheme, the decoding process cannot be initiated unless the receivers collect a full set of coded packets. As a result, the communication delay and decoding complexity increases. However, if the source sends source packets at first and then sends coded packets, the receiver can start the decoding process without collecting a full set of coded packets. This new scheme is called systematic block-based RLNC. It decreases the decoding complexity and communication delay. The diminution of decoding complexity is based on the fact that systematic block-based RLNC is a type of sparse network coding scheme. Fig. 3.7 (c) shows a systematic block-based RLNC scheme (Fouli *et al.*, 2018).

One of the main issues in the block-based RLNC scheme is the selection of a proper generation size. The size of the generation is selected based on the required Quality of Service (QoS) of the application. For example, downloading a file needs a high throughput, while real-time services such as video streaming demand low communication delays (Zeng, Ng & Médard, 2012). Each application requires a specific generation size. Before detailing the generation size, we present two distinct definitions of delay in the network coding schemes as follows:

- **The average decoding delay:** the mean time required for the recovery of a generation (Chatzigeorgiou & Tassi, 2017).

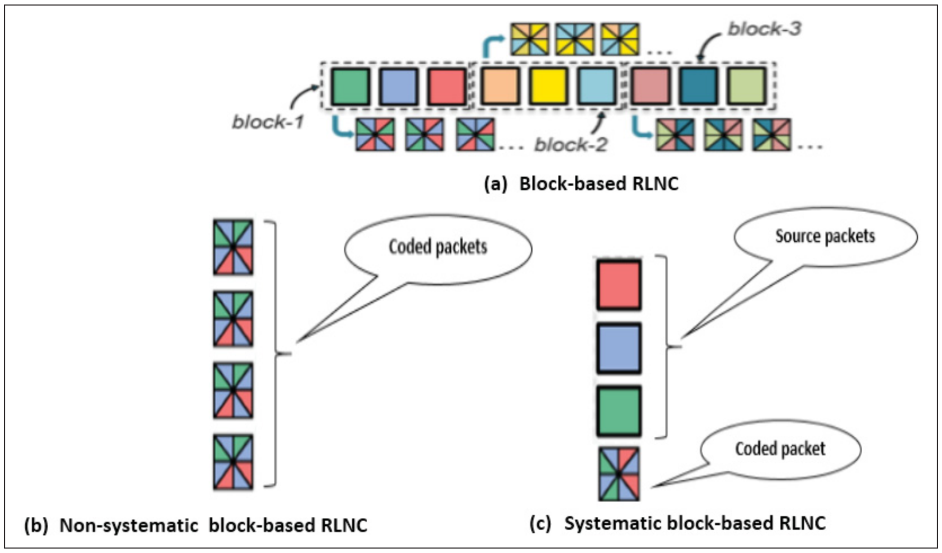

Figure 3.7 Block coding
Adapted from Fouli *et al.* (2018)

- **In-order delivery delay:** the total number of time slots that are elapsed, while sending a source packet by the transponder and delivering it in-order by the decoder to the upper layer at the receiver (Wunderlich *et al.*, 2017; Karzand *et al.*, 2017).

Traditionally, to achieve high throughputs in block-based RLNC, we increase the size of the generation. However, the decoding delay also increases since we need more time to complete a generation. On the other hand, to obtain a low decoding delay in a block-based RLNC, we decrease the size of generation. However, the throughput gain achieved by block-based RLNC vanishes. Overall, there is a compromise to be made between throughput and decoding delay, which is known as throughput delay trade-off (Zeng *et al.*, 2012). It is noted that there is no difference between systematic and non-systematic block-based RLNC in terms of the decoding delay in a fixed generation size (Chatzigeorgiou & Tassi, 2017). However, systematic and non-systematic block-based RLNC differ in terms of in-order delivery delay. Systematic block-based RLNC is preferred to non-systematic block-based RLNC. Since receivers receive source packets instead of coded packets in systematic block-based RLNC, they do not need to collect a sufficient number of coded packets for each generation and then start the decoding process. However, receivers in non-systematic block-based RLNC must collect a full set of coded packets for each generation and then starts the decoding process, which in turn increases in-order delivery delay. Furthermore, both systematic and non-systematic block-based RLNC can achieve lower in-order delivery delays for small generation sizes, since if a packet loss happens in the network, source packets spend less time in the receiver's buffer until the recovery process starts.

Overall, block-based RLNC has a rigid structure (Fouli *et al.*, 2018) that increases decoding complexity and communication latency (Fouli *et al.*, 2018). Thus, Sliding Window RLNC (SWRLNC) was proposed. Unlike block-based RLNC, sliding window RLNC does not include distinct blocks of source packets, but instead performs encoding on source packets within a sliding encoding window, such that each source packet can be seen in multiple generations (Fouli *et al.*, 2018). It has a flexible structure and decreases communication delay (Karzand *et al.*, 2017; Wunderlich *et al.*, 2017; Fouli *et al.*, 2018). Fig. 3.8 illustrates this encoding scheme (Fouli *et al.*, 2018).

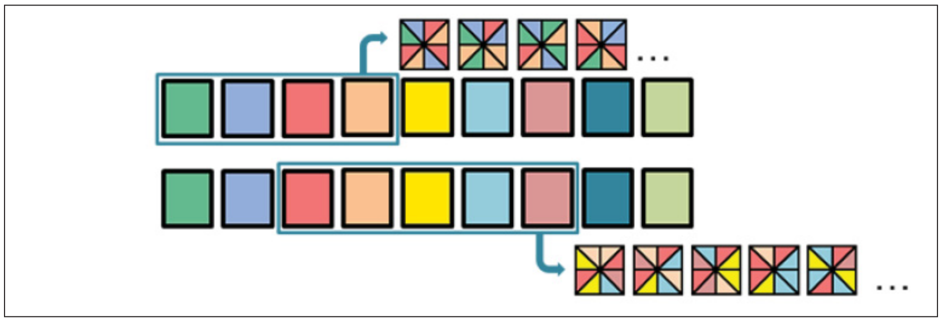

Figure 3.8 Sliding window coding
Adapted from Fouli *et al.* (2018)

Both the block-based RLNC and the sliding window RLNC provide redundant coded packets to recover lost packets in an erasure network. However, as mentioned earlier, the sliding window has a superior performance compared to block-based RLNC. Depending on network conditions, sliding window RLNC can be used with feedback or without feedback. For example, the work in (Zverev *et al.*, 2019) uses SWRLNC without feedback to recover lost packets. It adjusts the parameters of the scheme in (Wunderlich *et al.*, 2017) for a channel with a fixed loss rate to provide a zero packet loss rate (Zverev *et al.*, 2019). However, in practice, the channel situation is changing with time, and providing zero packet loss rate without feedback is impossible. In addition, (Zverev *et al.*, 2019) does not consider existing constraints in IIoT devices such as the small buffer size and low computational power. In (Karzand *et al.*, 2017), SWRLNC with feedback is used to reduce in-order delivery delay in a unicast scenario in satellite communications, where ARQ is not suitable. It is shown that SWRLNC with feedback surpasses block-based RLNC in terms of in-order delivery delay (Karzand *et al.*, 2017). In what follows, we elaborate on existing sliding window RLNC schemes, which can be classified according to their consideration of feedback.

SWRLNC Without Feedback :

The Caterpillar RLNC (CRLNC) scheme uses a finite sliding encoding window to provide redundant coded packets. In this scheme, first, source packets and then coded packets are sent (Wunderlich *et al.*, 2017). A coded packet consists of a linear combination of source packets within a finite sliding encoding window. The size of this sliding encoding window can change. For a sliding encoding window of size (W_e), we define the code rate R in equation (3.1) where N_S is the number of source packets sent before a coded packet is sent (Wunderlich *et al.*, 2017).

$$R = \mathbf{N}_s/(\mathbf{N}_s + 1) \tag{3.1}$$

Fig. 3.9 shows this scheme, where two source packets followed by a coded packet are transferred. Each coded packet is a linear combination of the previous four source packets within the finite sliding encoding window. In this example, the size of the encoding window is equals to four. It is noted that the first coded packet consists of two source packets since two source packets are available at the start point of the scheme. Here, the code rate is (2/3) since the number of source packets which are sent before a coded packet, N_s, is two. The receiver can recover lost packets if it collects a sufficient number of coded packets.

SWRLNC With Feedback :

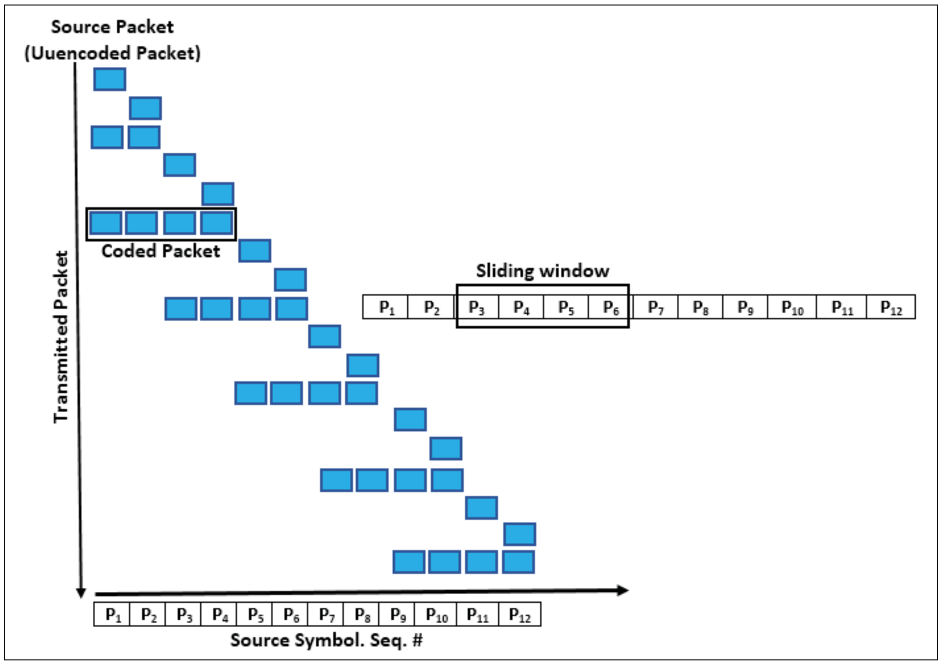

Figure 3.9 Caterpillar RLNC scheme
Adapted from Wunderlich *et al.* (2017)

In (Karzand *et al.*, 2017), a sliding window RLNC scheme that uses feedback to close the tail of the sliding encoding window and adjusts the coding rate is proposed. In this scheme, known as the low delay code scheme, coded packets are sent/distributed between source packets periodically. When packets are lost, the receiver starts collecting the coded packets. If the receiver receives a sufficient number of coded packets, it can recover the lost packets. Fig. 3.10 illustrates the structure of this code. Gray and white squares are used to represent coded packets and source packets, respectively. Meanwhile, crossed squares show lost packets. As demonstrated in the Fig. 3.10, the third coded packet includes more source packets since the source advances its sliding encoding window up to packet P_5, given the received feedback from the receiver. After the recovery of lost packets P_5 and P_6, the receiver sends periodic feedback, and the source advances its sliding encoding window up to packet P_{10}. Fig. 3.11 shows the value of the in-order delivery delay for each packet. This delay is equal to the waiting time of the source packets in the buffer before being delivered to the upper layer in-order.

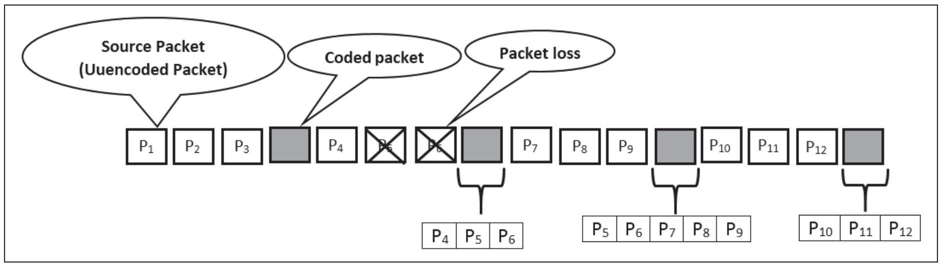

Figure 3.10 Low delay code structure
Adapted from Karzand *et al.* (2017)

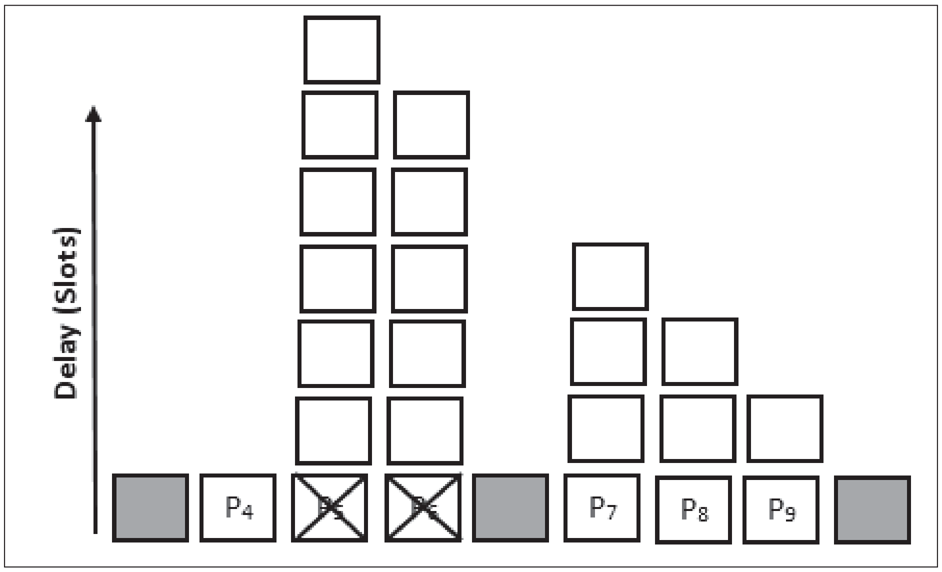

Figure 3.11 In-order delivery delay for each packet
Adapted from Karzand *et al.* (2017)

According to the results and achievements in previous works, sliding window RLNC and block-based RLNC have a high potential to be applied to IIoT networks and decrease in-order delivery delay. Sliding window RLNC has several advantages compared to block-based RLNC schemes as follows (Karzand *et al.*, 2017; Tournoux *et al.*, 2011; Fouli *et al.*, 2018):

- It has a flexible structure.
- It decreases the amount of feedback packets used in the network.
- It is robust to feedback loss in unicast scenarios since feedback only is used to close the tail of the sliding window.

- Parameters of the scheme can be adjusted readily according to changes in the channel situation.
- Communication delay can be controlled by changing the scheme's parameters.
- The decoding complexity can be adjusted to suit the computational power of devices.

In this project, we will put sliding window RLNC and block-based RLNC under light and investigate them from different points of view.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Introduction

Regarding the requirements of IIoT networks and their constraints, we investigate the performance of two main network coding methods that recover packet loss. Block-based RLNC and sliding-window RLNC with feedback have never been evaluated for a multicast scenario, so we do not know which one is the best. Following the problem statement, we implemented the two and conducted simulations to figure that out. We show the results and discuss which is the best one that satisfies IIoT requirements. To this end, in this project, we compare low delay code that uses sliding window RLNC method and systematic block-based RLNC in a one-hop multicast scenario. In this comparison, we consider the existing constraints of IIoT devices and the requirements of IIoT networks. Thus, we compare them in terms of packet in-order delivery delay, the number of feedback packets and coded packets used in the network, the required buffer size for receivers, and the decoding complexity.

In this chapter, first, we explain the system model used in simulation including, channel model, simulation method, and feedback system. Then, we present simulation results and initiate a discussion about them.

4.2 System Model of simulation

In this simulation, we consider an one-hop multicast scenario, which consists of an access point such as a WIFI network or any wireless network, and K devices fixed around an access point (without mobility). The access point uses only one frequency and the channels between the access point and users are lossy.

4.2.1 Channel models

In our simulation, we use two different channel models. In both channel models, the channels are considered independent from each other. In the first channel model, source packets are lost uniformly with an identical distribution (i.i.d.) for each device. In the second channel model, we use the Gilbert-Elliot model (Gilbert, 1960), since source packets are lost as burst errors in real scenarios (Zverev *et al.*, 2019).

Fig. 4.1 shows the Gilbert-Elliot channel model, which is based on a Markov chain with two states G (for good or gap) and B (for bad or burst). If the channel is in the good state, all of the packets (coded and unencoded) arrive at the receiver, while if the channel is in the bad state, all of them are lost. β and γ are the transition probabilities moving from the bad state to the good state and from the good state to the bad state, respectively. Having these parameters determined, the steady-state probability for the bad state π_B, and the expected sojourn time in the bad state $E[B]$ are calculated by equations (4.1) and (4.2), respectively (Wunderlich *et al.*, 2017).

$$\pi_B = \gamma/(\gamma + \beta) \tag{4.1}$$

$$E[B] = \mathbf{1}/\beta \tag{4.2}$$

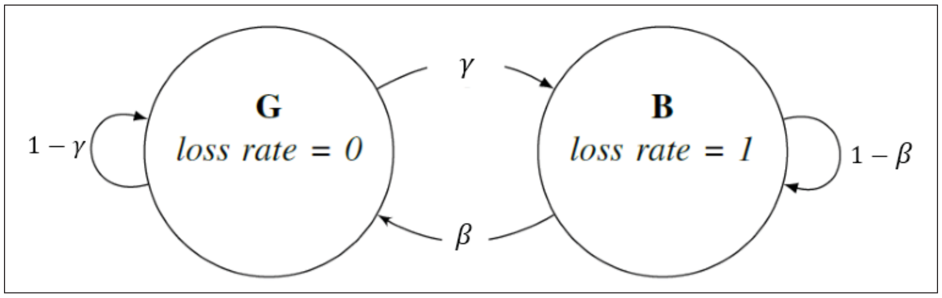

Figure 4.1 Gilbert-Elliot channel model
Adapted from Zverev *et al.* (2019)

4.2.2 Simulation method of sliding window RLNC and Block-based RLNC

In this project, we have simulated the low delay code, which uses sliding window RLNC, and the systematic block-based RLNC illustrated in subsection 3.6 for a one-hop multicast scenario. The low delay code takes advantage of sliding window RLNC, while the systematic RLNC has a block-based structure. To better understand both codes, we have compared them in terms of in-order delivery delay, the number of coded packets used in the network, the required buffer size for receivers, and the decoding complexity. In our simulations, we also use a sliding window in systematic block-based RLNC to encode source packets in each generation; however, there is no overlap between generations. Therefore, we consider the size of the generation and window as one. We have run simulations on the codes with different numbers of devices (5 and 50 devices) in the network because the number of devices is an important parameter in performance of the network.

We have simulated systematic block-based RLNC with three encoding window sizes small, medium, and large (8, 32, 128), because the size of encoding window has a direct impact on in-order delivery delay. We first taken into account the number of coded packets used in the network for each encoding window sizes. Then, we have distributed coded packets uniformly between source packets to recover lost packets in the low delay code. We have used this method since depending on the channel conditions, we require a different number of coded packets at end of generations in systematic block-based RLNC to achieve full reliability. We have considered two calculation methods to distribute coded packets between source packets in the low delay code. We have calculated an upper bound and a lower bound to distribute coded packets between source packets in the low delay code. For example, if we distribute 787 coded packets between 5000 source packets, we should put a coded packet after every 6.35 source packet. However, it is not allowed to transmit partial packets and, thus we have designed two scenarios. In the first scenario (lower bound), we put a coded packet after every 6th source packet, and in the second scenario (upper bound), we put a coded packet after every 7th source packet.

To calculate decoding complexity, we have counted the number of mathematical operations in the decoder to achieve an identity matrix and extract source packets. The decoder uses addition, multiplication, and division operations. We have considered that for two numbers with n digits, the complexity addition, multiplication, and division is $\Theta(n)$, $O(n^2)$, and $O(n^2)$, respectively.

Finally, it is noted that in a communication system based on network coding, packets are elements of a finite field, thus their sizes are fixed. However, in practice, the size of packets in IIoT networks are variables (Torres Compta, Fitzek & Lucani, 2015). There are different schemes to resolve this problem. For example, authors in (Torres Compta *et al.*, 2015) used the chain and fragmentation scheme to generate packets with a fixed size. They showed this scheme has a better performance in terms of header overhead compared to other schemes. Fig. 4.2 illustrates the chain and fragmentation scheme. In the first step, the scheme creates the chain of packets and adds zeros to the last packet P_g. In the second step, the scheme performs fragmentation to generate packets with equal sizes. Finally, in the third step, the scheme generates coded packets by the encoding vectors of EV.

In this simulation, we assume the access point uses this scheme to generate packets with a fixed size before sending them to devices. Using this scheme does not have any effect on in-order delivery delay, because if lost packets are recovered faster, source packets are also delivered to the application layer faster.

4.2.3 The feedback system

The feedback system is inspired by ALOHA (Abramson, 1970). ALOHA is a protocol which shares network channel between multiple users. ALOHA takes advantage of random access, while our feedback system uses both random access and systematic access based on the calculation of the last decoded packet in receivers. To use the feedback system, each device calculates the number of the last packet that it could decode. Then, each device activates a timer based on its computed number. The device with the smallest number sends feedback to the source quickly. We assume that other devices hear this feedback and stop their timers. This feedback includes

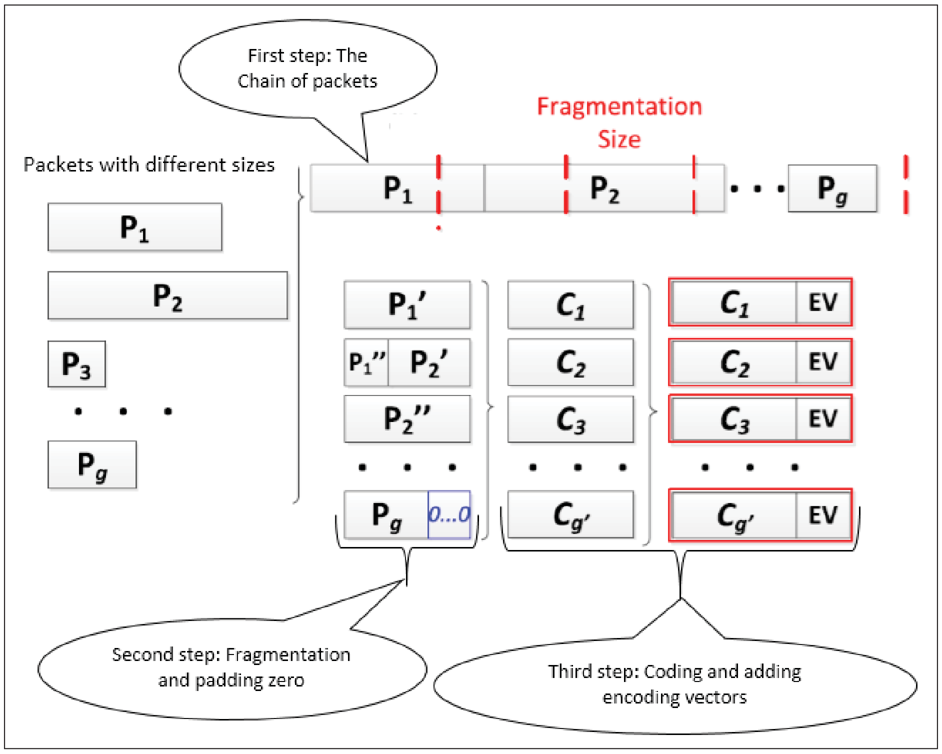
First step: The Chain of packets
Fragmentation Size
Packets with different sizes
P1
P2
P3
Pg
P1'
P1'' P2'
P2''
Pg 0...0
C1 C2 C3 Cg'
C1 EV
C2 EV
C3 EV
Cg' EV
Second step: Fragmentation and padding zero
Third step: Coding and adding encoding vectors

Figure 4.2 Example of the chain and fragmentation scheme
Adapted from Torres Compta *et al.* (2015)

the index of the last decoded packet of the device. The source receives feedback and moves the sliding encoding window according to feedback information. Then, the source generates coded packets by combining source packets existing in the encoding sliding window. It is noted that since several devices can have the same index simultaneously, devices also generate a random time to avoid collision in the network.

4.3 Simulation Setup

In the simulations, we use channel models described in subsection 4.2.1, where source packets are lost uniformly or consecutively (burst of errors). Hereafter, the first one is called uniform channel model, and the second one is called burst channel model. In both models, the fading channels between the access point and devices are independent. In the uniform channel model, Packet Erasure Rate (PER) of devices is changed from 10 % to 14 %. In the burst channel model, the transition probability from the good state to the bad state is varied from 0.01 to 0.09 and the transition probability from the bad state to the good state is changed from 0.25 to 0.50. Windows sizes used in simulations are 8, 32, and 128.

4.3.1 Simulation Results

To evaluate the results, we investigate different key performance indicators (KPIs) in the networks, including in-order delivery delay, the number of coded packets, buffer size, average decoding complexity, maximum decoding complexity, the number of feedback packets, average in-order delivery delay, and maximum in-order delivery delay.

Figs. 4.3, 4.4, and 4.5 show a comparison between low delay code and systematic block-based RLNC for 5 devices, of window sizes 8, 32, and 128, and uniform channel model. The Y-axis in Figs. 4.3, 4.4, and 4.5 states cumulative average in-order delivery delay. It shows the percentage of source packets that are delivered to the application layer in slot X^{th}. As it is shown in Fig. 4.3.a, where the window size is 8, systematic block-based RLNC and the low delay code U_B (upper bound) decode 80% of packets before the fifth slot with the same average in-order delivery delay. It is noted that the low delay code U_B achieves this performance with less number of coded packets. However, systematic block-based RLNC overtakes low delay code U_B after the fifth slot. On the other hand, for the window sizes 32 and 128 in Figs. 4.3.b and 4.3.c, the low delay code U_B has a better performance before 26th and 111th slots, respectively. Because coded packets are distributed between source packets in the low delay code, while coded packets are located at the end of each generation in the systematic block-based RLNC. Thus, the lost packets recovery process in the systematic block-based RLNC starts at the end of transmission of each generation, while the lost packets recovery process in the low delay code can start if receivers receive enough coded packets at any time. Fig. 4.3.d shows the number of coded packets used in the network.

Fig. 4.4.a illustrates the buffer size in the receivers. As it is shown in Fig. 4.4.a , the low delay code requires a larger buffer size than that of systematic block-based RLNC since there is no

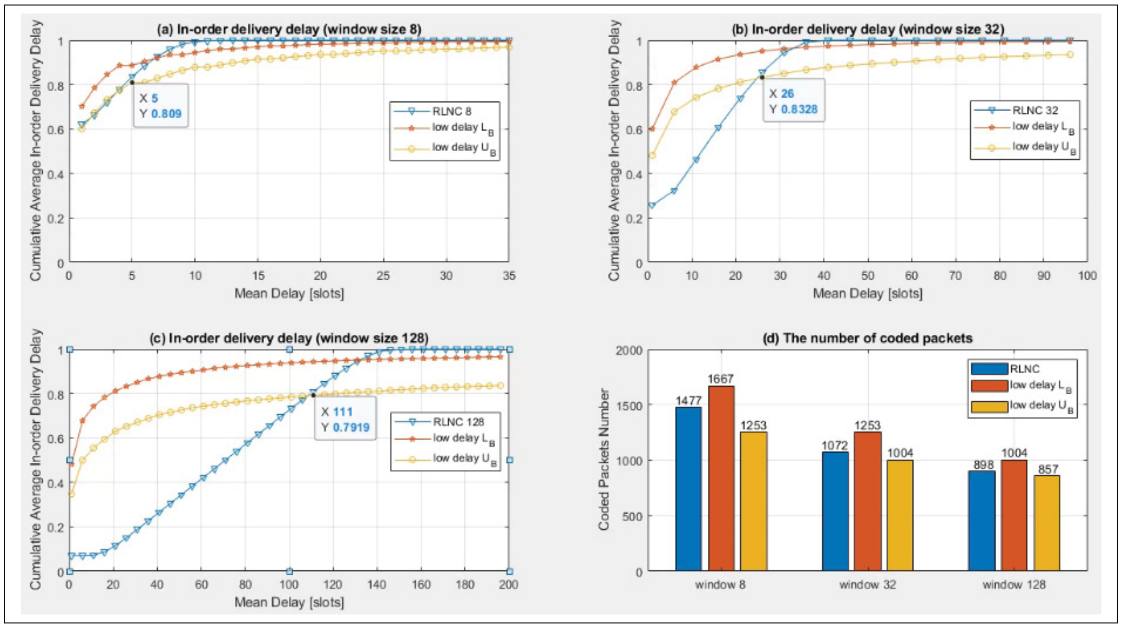

Figure 4.3 Comparison between low delay code and systematic block-based RLNC (5 devices, window sizes 8,32, and 128, uniform channel model)

mechanism to control the size of buffer in the low delay code. Thus, sometimes more packets should wait in the buffer for starting lost packets recovery. However, systematic block-based RLNC requires a smaller buffer size since systematic RLNC has a block-based structure, and each block of code includes a specific number of source packets. This is also true for average decoding complexity and maximum decoding complexity in Figs. 4.4.b and 4.4.c. Since the size of sliding window in the low delay code is dynamic, we can see an increase in decoding complexity. Fig. 4.4.d depicts the number of feedback packets used in the network. The low delay code uses less feedback packets for the window size 8. However, for window sizes 32 and 128, the systematic block-based RLNC uses less feedback packets compared to the low delay code. It is noted that since the low delay code has a flexible structure, we can decrease the number of feedback packets used in the low delay code for window sizes 32 and 128.

Fig. 4.5 shows an average in-order delivery delay and the maximum in-order delivery delay for the window sizes 8,32, and 128, uniform channel model, and 5 devices. As it is shown in Fig. 4.5.a, average in-order delivery delay in the low delay code L_B (lower bound) is less than that

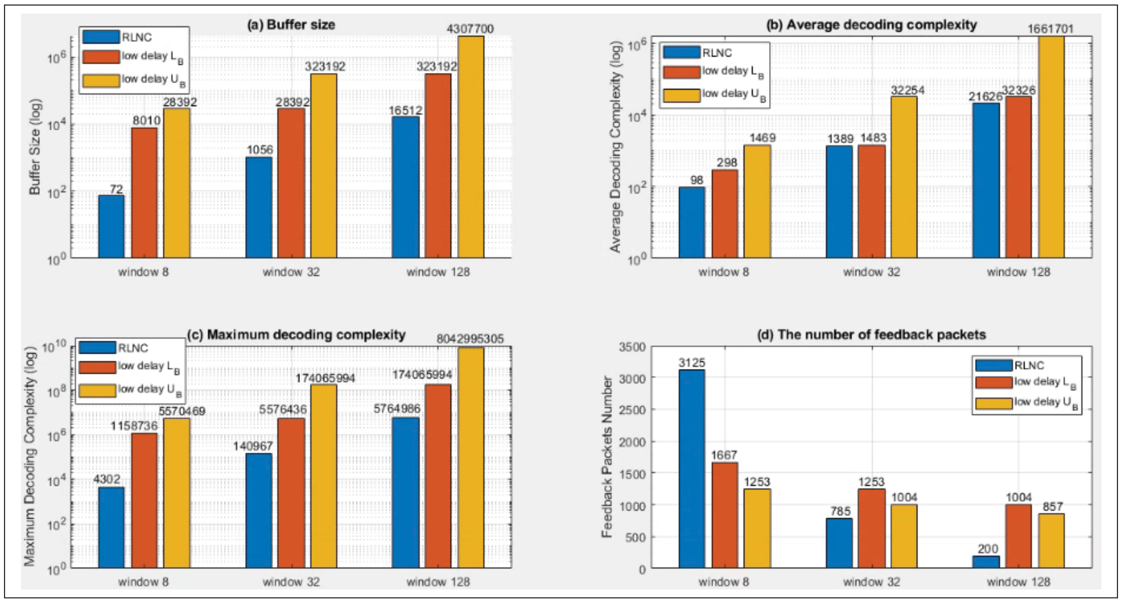

Figure 4.4 Comparison between low delay code and systematic block-based RLNC (5 devices, window sizes 8,32, and 128, uniform channel model)

of the systematic block-based RLNC, while average in-order delivery delay in the low delay code U_B is larger than that of the systematic block-based RLNC, since the low delay code L_B uses more coded packets compared to the low delay code U_B. As it is depicted in Fig. 4.5.b, the low delay code has a larger maximum in-order delivery delay compared to the systematic block-based RLNC since the low delay code does not have any mechanism to control in-order delivery delay.

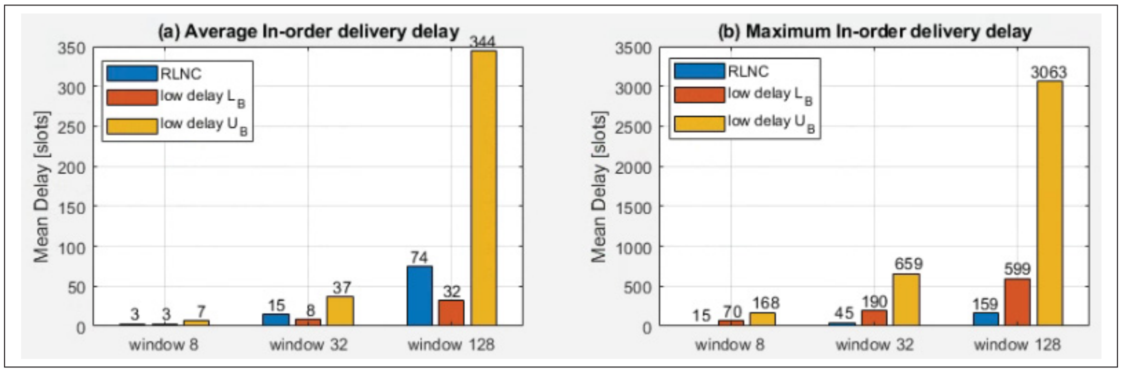

Figure 4.5 Comparison between low delay code and systematic block-based RLNC (5 devices, window sizes 8,32, and 128, uniform channel model)

Figs. 4.6, 4.7, and 4.8 depict the comparison between low delay code and systematic block-based RLNC for burst channel model. For the window size 8 in Fig. 4.6.a, the systematic block-based RLNC has a better performance than the low delay code U_B . However, with the increase in the window size, the low delay code U_B decreases in-order delivery delay more than that of the systematic block-based RLNC.

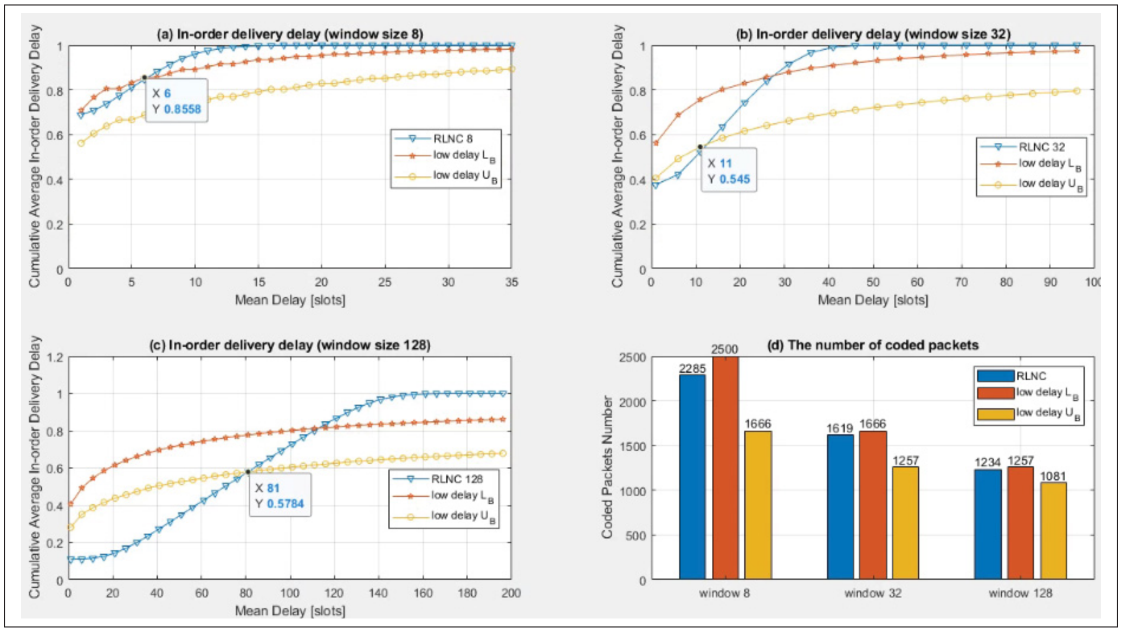

Figure 4.6 Comparison between low delay code and systematic block-based RLNC (5 devices, window sizes 8,32, and 128, burst channel model)

The trends in Figs. 4.7.a, 4.7.b, 4.7.c, and 4.7.d that use burst channel model are similar to the trends of Figs. 4.4.a, 4.4.b, 4.4.c, and 4.4.d that use uniform channel model. Similarly, with the increase of the window size, buffer size, average decoding complexity, and maximum decoding complexity increase, since the size of generation increases. Moreover, the number of feedback packets for the low delay code is less than that of the systematic block-based RLNC for window size 8. However, with an increase of window size, the number of feedback packets decreases for the systematic block-based RLNC as compared with the low dealy code, since the size of generation increases. Similarly, the average in-order delay delivery delay and the maximum in-order delay delivery increase in Fig. 4.8 with the increase of window size.

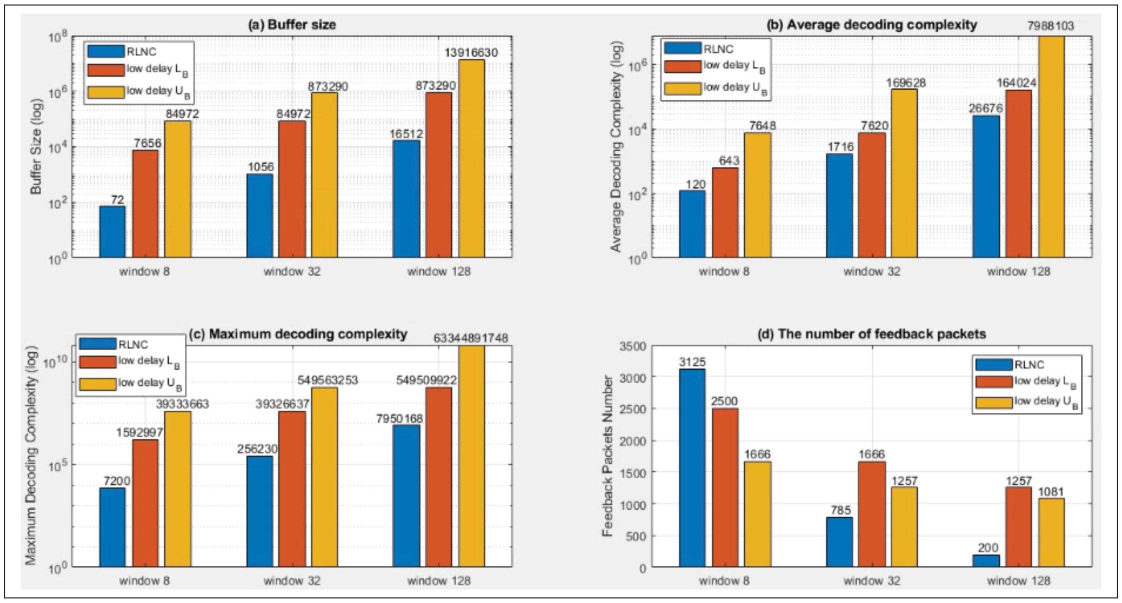

Figure 4.7 Comparison between low delay code and systematic block-based RLNC (5 devices, window sizes 8,32, and 128, burst channel model)

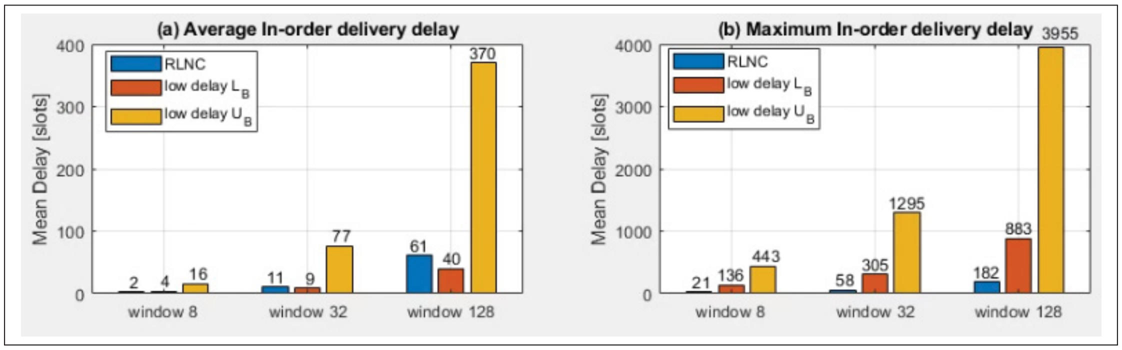

Figure 4.8 Comparison between low delay code and systematic block-based RLNC (5 devices, window sizes 8,32, and 128, burst channel model)

In Figs. 4.9, 4.10, and 4.11, where there are 50 devices and an uniform channel model is used, the low delay code has a better performance for all the window sizes. Almost more than 90% of packets in Figs. 4.9.a, 4.9.b, and 4.9.c are decoded with a lower in-order delivery delay in the low delay code U_B (upper bound) than that of systematic block-based RLNC. Fig. 4.9.d depicts the number of coded packets used in the network and explains the reason of the decrement in in-order delivery delay for 50 devices. When there are 50 devices, we need more coded packets to recover lost packets compared to the same network with 5 devices. As a result, using more coded packets reduces in-order delivery delay dramatically.

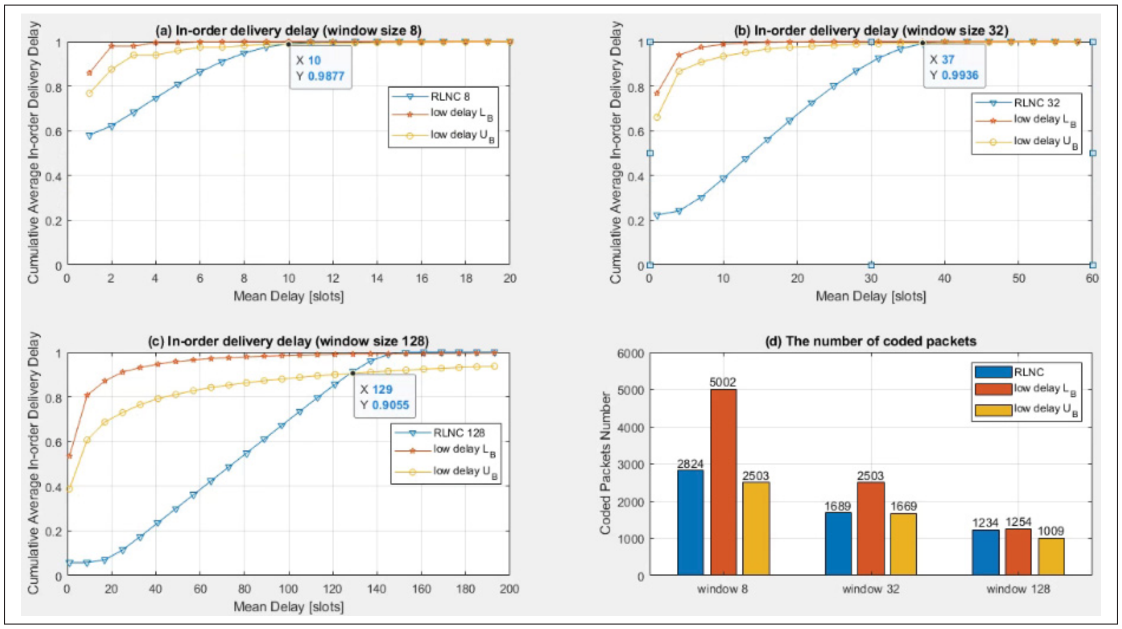

Figure 4.9 Comparison between low delay code and systematic block-based RLNC (50 devices, window sizes 8,32, and 128, uniform channel model)

As shown in Figs. 4.10.a, 4.10.b, and 4.10.c, the low delay code needs a larger buffer size and has a larger average and maximum decoding complexity than those of the systematic block-based RLNC. The analysis of this behavior is similar to Figs. 4.4.a, 4.4.b, and 4.4.c. The systematic block-based RLNC has a block-based structure to control buffer size and decoding complexity, while the low delay code does not have any mechanism to limit them. Fig. 4.10.d illustrates the number of feedback packets used in the network. The low delay code uses less feedback packets compared to systematic block-based RLNC for all window sizes.

Fig. 4.11 shows the average and maximum in-order delivery delay for 50 devices and uniform channel model. Unlike Fig. 4.5.a, the low delay code has a lower average delivery delay since more coded packets are distributed between source packets and recover lost packets. It is noted that with decrease in average in-order delivery delay, network throughput also reduces. Network throughput is a function of the number of devices in the network. The network throughput

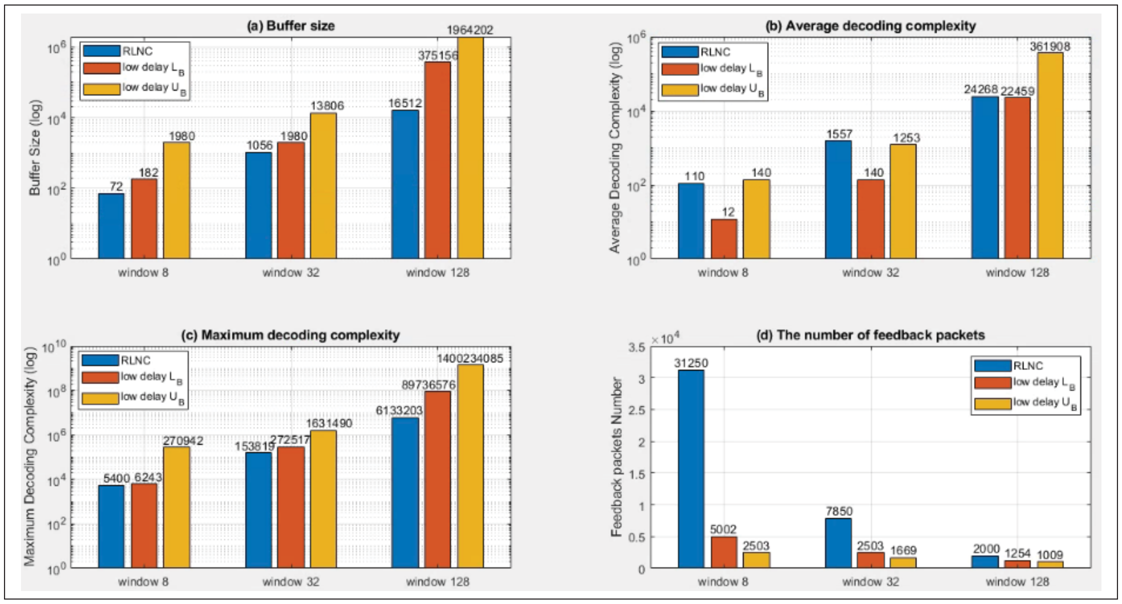

Figure 4.10 Comparison between low delay code and systematic block-based RLNC (50 devices, window sizes 8,32, and 128, uniform channel model)

decreases with the increase in the number of devices. Overall, the number of devices in the network and device's channel status are two important factors that affect the scalability of the network. Maximum in-order delivery delay in Fig. 4.11.b has a similar behavior to Fig. 4.5.b. The low delay code has a larger maximum in-order delivery delay as compared to systematic block-based RLNC.

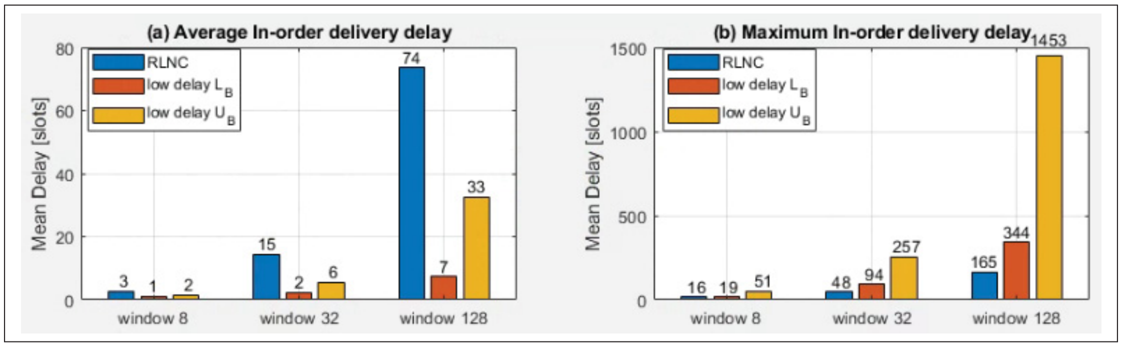

Figure 4.11 Comparison between low delay code and systematic block-based RLNC (50 devices, window sizes 8,32, and 128, uniform channel model)

Figs. 4.12, 4.13, and, 4.14 show the status of the network for the Gilbert-Elliot channel model. The analysis of Figs. 4.12 is similar to previous corresponding figures. In Fig. 4.14.a, the low delay code decreases the average in-order delivery delay compared to the systematic block-based RLNC for the window sizes 8 and 32. However, the systematic block-based RLNC achieves a better average in-order delivery delay for the window size 128. Since the Gilbert-Elliot channel model generates burst of errors, devices recover lost packets with more delay for the window size 128. Similar to Figs, 4.5.b, 4.8.b, and 4.11.b, the low delay code has a larger maximum in-order delivery delay in Fig. 4.14.b. This is because the low delay code does not have any mechanism to control in-order delivery delay.

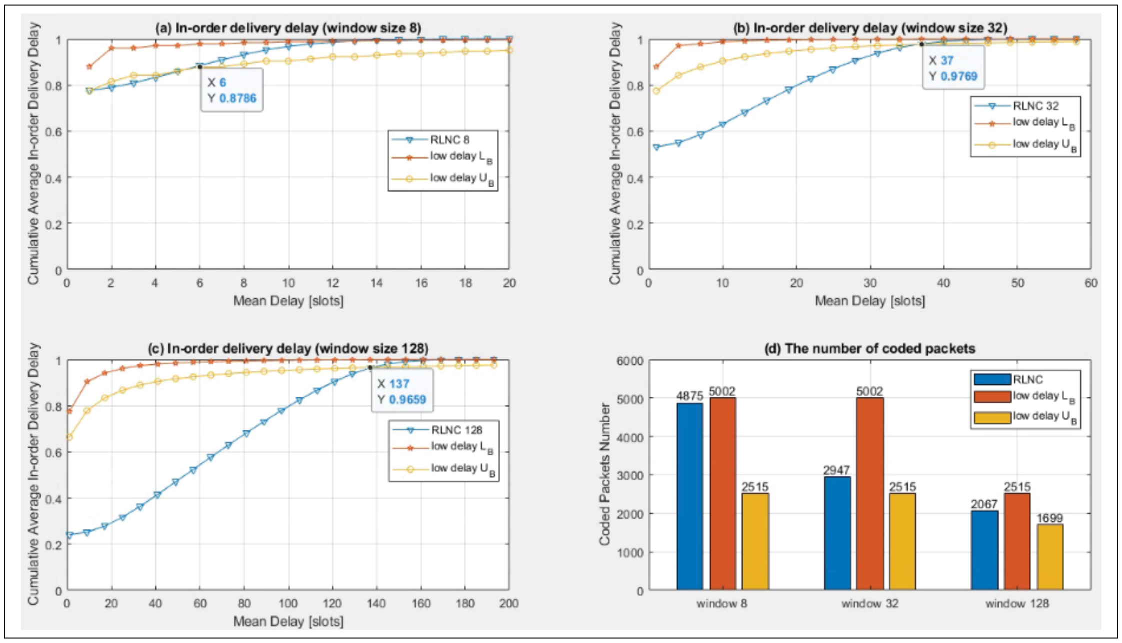

Figure 4.12 Comparison between low delay code and systematic block-based RLNC (50 devices, window sizes 8,32, and 128, burst channel model)

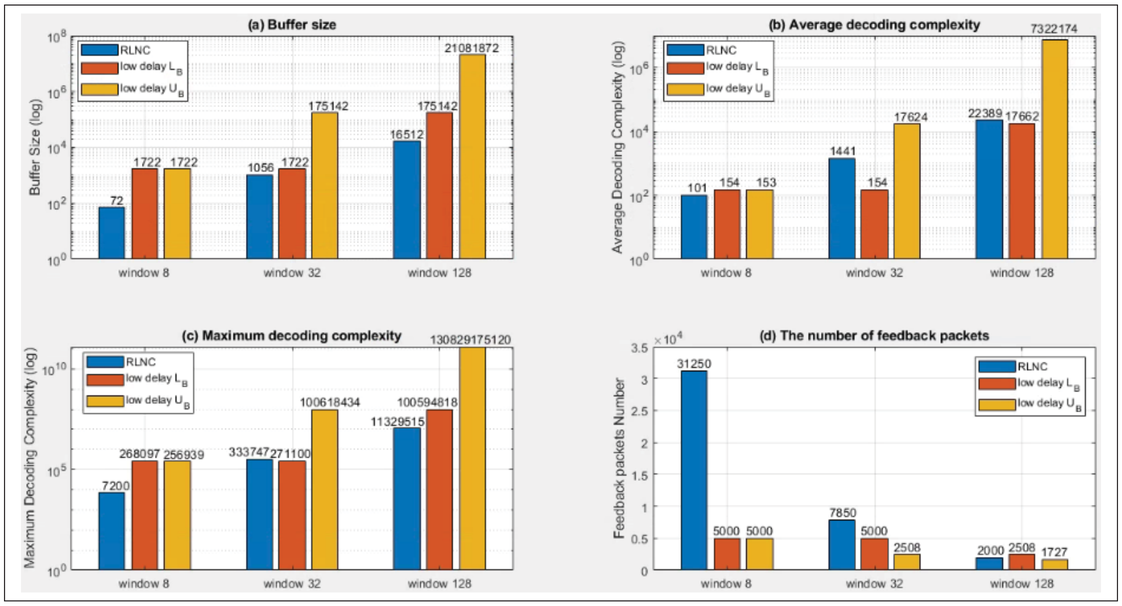

Figure 4.13 Comparison between low delay code and systematic block-based RLNC (50 devices, window sizes 8,32, and 128, burst channel model)

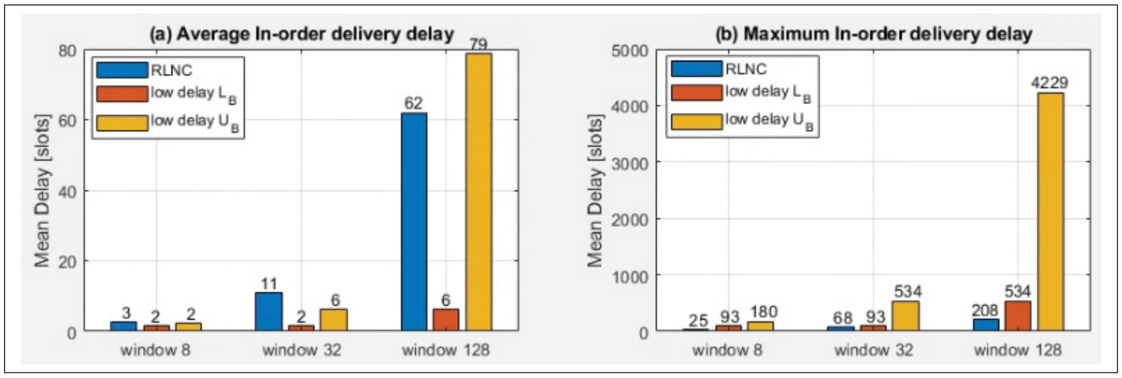

Figure 4.14 Comparison between low delay code and systematic block-based RLNC (50 devices, window sizes 8, 32, and 128, burst channel model)

CONCLUSION AND RECOMMENDATIONS

In this thesis, we have investigated the performance of the systematic block-based and sliding window RLNC in terms of in-order delivery delay, number of coded packets used in the network, required buffer size for receivers, decoding complexity, and number of feedback used in the network. The results have shown us that the low-delay code (based on sliding window RLNC network coding) has a high potential to be applied to IIoT networks. It can decrease in-order delivery delay. However, the low-delay code has an average decoding complexity greater than that of systematic block-based RLNC, since the block-based structure of the systematic RLNC limits decoding complexity automatically, while the low-delay code does not have any mechanism to limit decoding complexity. This argument is also true for the buffer size of devices. Systematic block-based RLNC needs a smaller buffer than the system using the low-delay code. To overcome the problems of the decoding complexity and buffer size in the low-delay code system, we need to design new mechanisms. These mechanisms can increase the speed of packet loss recovery, and reduce decoding complexity and buffer size in receivers.

BIBLIOGRAPHY

Aboutorab, N., Sadeghi, P. & Sorour, S. (2014). Enabling a Tradeoff between Completion Time and Decoding Delay in Instantly Decodable Network Coded Systems. *IEEE Trans. on Commun.*, 62(4), 1296-1309. doi: 10.1109/TCOMM.2014.021614.130172.

Abramson, N. (1970). The ALOHA system: Another alternative for computer communications. *Proceedings of the November 17-19, 1970, fall joint computer conference*, pp. 281–285.

Abudaqa, A., Mahmoud, A., Abu-Amara, M. & Sheltami, T. (2020). Survey of Network Coding Based P2P File Sharing in Large Scale Networks. *Appl. Sci.*, 10, 2206. doi: 10.3390/app10072206.

Ahlswede, R., Cai, N. & Yeung, R. (1998). Network Information Flow. *IEEE Trans. Inf. Theory*, 46. doi: 10.1109/ISIT.1998.708784.

Ahmed, S. & Kanhere, S. (2006a, 01). VANETCODE: network coding to enhance cooperative downloading in vehicular ad-hoc networks. pp. 527-532. doi: 10.1145/1143549.1143654.

Ahmed, S. & Kanhere, S. (2006b, 01). VANETCODE: network coding to enhance cooperative downloading in vehicular ad-hoc networks. pp. 527-532. doi: 10.1145/1143549.1143654.

Al-Kofahi, O. & Kamal, A. (2016). *Resilient Wireless Sensor Networks The Case of Network Coding*. doi: 10.1007/978-3-319-23965-1.

Alic, K. & Svigelj, A. (2018). Opportunistic Network Coding. In *Network Coding and Subspace Designs* (pp. 319–338). Cham: Springer International Publishing. doi: 10.1007/978-3-319-70293-3_12.

Cabrera Guerrero, J., Lucani, D. & Fitzek, F. (2016, 09). On network coded distributed storage: How to repair in a fog of unreliable peers. pp. 188-193. doi: 10.1109/ISWCS.2016.7600898.

Cai, N. & Yeung, R. W. (2002). Secure network coding. *Proc. IEEE Int. Symp. on Inf. Theory,*, pp. 323.

Chao, C., Chou, C. & Wei, H. (2010). Pseudo Random Network Coding Design for IEEE 802.16m Enhanced Multicast and Broadcast Service. *In Proc. Of IEEE 71st Vehicular Technology Conference*, pp. 1-5.

Chatzigeorgiou, I. & Tassi, A. (2017). Decoding Delay Performance of Random Linear Network Coding for Broadcast. *IEEE Trans. on Veh. Technol.*, 66(8), 7050-7060.

Diestel, R. (2005). Graph Theory: Electronic Edition 2000.

Dimakis, A. G., Ramchandran, K., Wu, Y. & Suh, C. (2011). A Survey on Network Codes for Distributed Storage. *Proc. of the IEEE*, 99(3), 476-489.

Dimakis, A., Godfrey, P., Wu, Y., Wainwright, M. & Ramchandran, K. (2010). Network Coding for Distributed Storage Systems. *Inf. Theory, IEEE Trans. on*, 56, 4539 - 4551. doi: 10.1109/TIT.2010.2054295.

Douik, A., Sorour, S., Tembine, H., Alouini, M. & Al-Naffouri, T. Y. (2014). A game theoretic approach to minimize the completion time of network coded cooperative data exchange. *In Proc. Of IEEE Global Communications Conference*, pp. 1583-1589.

Douik, A., Sorour, S., Al-Naffouri, T. Y. & Alouini, M. (2017). Instantly Decodable Network Coding: From Centralized to Device-to-Device Communications. *IEEE Commun. Surv. Tut.*, 19(2), 1201-1224.

Feizi, S., Lucani, D. E. & Médard, M. (2012). Tunable sparse network coding. doi: 10.3929/ETHZ-A-007052510.

Feizi, S., Lucani, D., Sørensen, C., Makhdoumi, A. & Médard, M. (2014, 06). Tunable sparse network coding for multicast networks. pp. 1-6. doi: 10.1109/NETCOD.2014.6892129.

Fouli, K., (MIT), M. M. & Shroff, K. (2018). Coding the Network: Next Generation Coding for Flexible Network Operation. *Code On Netw. Coding LLC Published*.

Fragouli, C., Lun, D., Medard, M. & Pakzad, P. (2007). On Feedback for Network Coding. *In Proc. Of 41st Annual Conference on Information Sciences and Systems*, pp. 248-252.

Fragouli, C., Widmer, J. & Le Boudec, J.-Y. (2008). Efficient Broadcasting Using Network Coding. *IEEE/ACM Trans. on Netw.*, 16. doi: 10.1145/1373990.1374006.

Fu, A., Sadeghi, P. & Médard, M. (2012). Dynamic Rate Adaptation for Improved Throughput and Delay in Wireless Network Coded Broadcast. *Netw., IEEE/ACM Trans. on*, 22. doi: 10.1109/TNET.2013.2292613.

Gabriel, F., Wunderlich, S., Pandi, S., Fitzek, F. H. P. & Reisslein, M. (2018). Caterpillar RLNC With Feedback (CRLNC-FB): Reducing Delay in Selective Repeat ARQ Through Coding. *IEEE Access*, 6, 44787-44802.

Gilbert, E. N. (1960). Capacity of a burst-noise channel. *Bell system technical journal*, 39(5), 1253–1265.

Gkantsidis, C. & Rodriguez, P. R. (2005). Network coding for large scale content distribution. *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, 4, 2235-2245 vol. 4.

Gkantsidis, C. & Rodriguez, P. (2005, 04). Network Coding for Large Scale Content Distribution. 4, 2235 - 2245 vol. 4. doi: 10.1109/INFCOM.2005.1498511.

Hassan, q. f. (2018). *Internet of Things A to Z: Technologies and Applications*.

Heide, J. & Lucani, D. (2015). Composite extension finite fields for low overhead Network Coding: Telescopic codes. *In Proc. Of IEEE International Conference on Communications (ICC)*, pp. 4505-4510.

Heide, J., Pedersen, M. V., Fitzek, F. H. P. & Medard, M. (2011). On Code Parameters and Coding Vector Representation for Practical RLNC. *In Proc. Of IEEE International Conference on Communications (ICC)*, pp. 1-5.

Heide, J., Pedersen, M. V., Fitzek, F. H. P. & Medard, M. (2014). A Perpetual Code for Network Coding. *In Proc. Of IEEE 79th Vehicular Technology Conference (VTC Spring)*, pp. 1-6.

Ho, T., Kötter, R., Médard, M., Karger, D. & Effros, M. (2003). The Benefits of Coding over Routing in a Randomized Setting. doi: 10.1109/ISIT.2003.1228459.

Ho, T., Médard, M., Kötter, R., Karger, D., Effros, M., Shi, J. & Leong, B. (2006). A Random Linear Network Coding Approach to Multicast. *IEEE Trans. on Inf. Theory*, 52. doi: 10.1109/TIT.2006.881746.

Hundeboll, M., Ledet-Pedersen, J., Sluyterman, G., Madsen, T. K. & Fitzek, F. H. P. (2014). Peer-Assisted Content Distribution with Random Linear Network Coding. *In Proc. Of IEEE 79th Vehicular Technology Conference (VTC Spring)*, pp. 1-6.

Jaggi, S., Langberg, M., Katti, S., Ho, T., Katabi, D. & Médard, M. (2007, 06). Resilient Network Coding in the Presence of Byzantine Adversaries. 54, 616 - 624. doi: 10.1109/INFCOM.2007.78.

Jiang, A. (2006, 08). Network Coding for Joint Storage and Transmission with Minimum Cost. pp. 1359 - 1363. doi: 10.1109/ISIT.2006.262068.

Jiang, D., Xu, Z., Li, W. & Chen, Z. (2015). Network Coding-Based Energy-Efficient Multicast Routing Algorithm for Multi-hop Wireless Networks. *J. of Syst. and Softw.*, 104. doi: 10.1016/j.jss.2015.03.006.

Karzand, M., Leith, D. J., Cloud, J. & Médard, M. (2017). Design of FEC for Low Delay in 5G. *IEEE J. on Sel. Areas in Commun.*, 35(8), 1783-1793.

Korhonen, J. & Frossard, P. (2009). Flexible Forward Error Correction Codes with Application to Partial Media Data Recovery. *Signal Process. : Image Communication*, 24, 229-242. doi: 10.1016/j.image.2008.12.005.

Kötter, R. & Médard, M. (2003). An algebraic approach to network coding. *Netw., IEEE/ACM Trans. on*, 11, 782 - 795. doi: 10.1109/TNET.2003.818197.

Lacan, J., Roca, V., Peltotalo, J. & Peltotalo, S. (2009). Reed-solomon forward error correction (FEC) schemes, RFC 5510.

Lamaazi, H., Benamar, N., Jara, A. J., Ladid, L. & El Ouadghiri, D. (2014, 08). Challenges of the Internet of Things: IPv6 and Network Manageme. doi: 10.1109/IMIS.2014.43.

Li, B., Bi, S., Zhang, R., Jiang, Y. & Li, Q. (2016, 05). Random Network Coding Based on Adaptive Sliding Window in Wireless Multicast Networks. pp. 1-5. doi: 10.1109/VTC-Spring.2016.7504440.

Li-Yong Ren, Hai-Ying He, Zhao Di & Ming Lei. (2009). Content distribution system based on segmented network coding. *In Proc. Of International Conference on Apperceiving Computing and Intelligence Analysis*, pp. 258-261.

Liu, Z., Wu, C., Li, B. & Zhao, S. (2010). UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding. *In Proc. Of Proceedings IEEE INFOCOM*, pp. 1-9.

Lopes, J. (2014, April, 1). Raptor codes. Consulted at https://www.slideshare.net/zemasa/fountain-codes.

Luby, M. (2002). LT codes. *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 271-280.

Lucani, D., Pedersen, M., Heide, J. & Fitzek, F. (2014). Fulcrum Network Codes: A Code for Fluid Allocation of Complexity.

Manna, S., Bhunia, S. & Mukherjee, N. (2014, 05). Vehicular pollution monitoring using IoT. pp. 1-5. doi: 10.1109/ICRAIE.2014.6909157.

Maymounkov, P., Harvey, N. & Lun, D. (2008). Methods for Efficient Network Coding.

Muriel Médard, A. S. (2012). Network Coding: fundamentals and applications. *Academic Press*, 99, 366 - 371. doi: ISBN 9780123809186.

Ning, H. & Wang, Z. (2011). Future Internet of Things Architecture: Like Mankind Neural System or Social Organization Framework? *IEEE Commun. Lette.*, 15, 461-463. doi: 10.1109/LCOMM.2011.022411.110120.

Ortolf, C., Schindelhauer, C. & Vater, A. (2009, 01). Paircoding: Improving File Sharing Using Sparse Network Codes. pp. 49-57. doi: 10.1109/ICIW.2009.16.

Pakzad, P., Fragouli, C. & Shokrollahi, A. (2005). Coding schemes for line networks. *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 1853-1857.

Peled, O., Sabag, O. & Permuter, H. H. (2019). Feedback Capacity and Coding for the $(0, k)$ -RLL Input-Constrained BEC. *IEEE Trans. on Inf. Theory*, 65(7), 4097-4114.

Peralta, G., Cid-Fuentes, R., Bilbao, J. & Crespo, P. (2018). Network Coding-Based Next-Generation IoT for Industry 4.0. In Matin, M. A. (Ed.), *Network Coding*. Oxford University Press. doi: 10.5772/intechopen.78338.

Peralta, G., Garrido, P., Bilbao, J., Aguero, R. & Crespo, P. (2019). Fog to cloud and network coded based architecture: Minimizing data download time for smart mobility.

Roca, V., Neumann, C. & Furodet, D. (2008). Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes.

Roca, V., Cunche, M., Lacan, J., Bouabdallah, A. & Matsuzono, K. (2013). Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME.

Roca, V., Teibi, B., Burdinat, C., Tran, T. & Thienot, C. (2016). *Block or Convolutional AL-FEC Codes? A Performance Comparison for Robust Low-Latency Communications*. working paper or preprint. Consulted at https://hal.inria.fr/hal-01395937.

Sahai, A. (2008). Why Do Block Length and Delay Behave Differently if Feedback Is Present? *Inf. Theory, IEEE Trans. on*, 54, 1860 - 1886. doi: 10.1109/TIT.2008.920339.

Sethi, P. & Sarangi, S. (2017). Internet of Things: Architectures, Protocols, and Applications. *J. of Elect. and Comput. Eng.*, 2017, 1-25. doi: 10.1155/2017/9324035.

Sheng, Z., Yang, S., Yu, Y., Vasilakos, A., McCann, J. & Leung, K. (2013). A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities. *Wirel. Commun. IEEE*, 20, 91-98. doi: 10.1109/MWC.2013.6704479.

Shojania, H. & Li, B. (2007). Parallelized Progressive Network Coding With Hardware Acceleration. *In Proc. Of Fifteenth IEEE International Workshop on Quality of Service*, pp. 47-55. doi: 10.1109/IWQOS.2007.376547.

Shokrollahi, A. (2006). Raptor codes. *IEEE Trans. on Inf. Theory*, 52(6), 2551-2567.

Shokrollahi, A. (2006). Raptor Codes. *IEEE Trans. on Inf. Theory*, 52, 2551 - 2567. doi: 10.1109/TIT.2006.874390.

Sipos, M., Fitzek, F. H. P., Lucani, D. E. & Pedersen, M. V. (2014). Dynamic allocation and efficient distribution of data among multiple clouds using network coding. *In Proc. Of IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pp. 90-95.

Sipos, M., Fitzek, F., Lucani, D. & Pedersen, M. (2014, 10). Dynamic allocation and efficient distribution of data among multiple clouds using network coding. pp. 90-95. doi: 10.1109/CloudNet.2014.6968974.

Skevakis, E. & Lambadaris, I. (2017). Delay optimal scheduling for network coding broadcast. *In Proc. Of IEEE International Conference on Communications (ICC)*, pp. 1-6.

Stolpmann, D., Petersen, C., Eichhorn, V. & Timm-Giel, A. (2018, 08). Extending On-the-fly Network Coding by Interleaving for Avionic Satellite Links. pp. 1-5. doi: 10.1109/VTC-Fall.2018.8690876.

Sundararajan, J. K., Shah, D., Médard, M. & Sadeghi, P. (2017). Feedback-Based Online Network Coding. *IEEE Trans. on Inf. Theory*, 63(10), 6628-6649.

Sundararajan, J., Shah, D. & Médard, M. (2008, 08). ARQ for network coding. pp. 1651 - 1655. doi: 10.1109/ISIT.2008.4595268.

Tao, S., Huang, J., Yang, Z., Cheng, W. & Liu, W. (2008). An Improved Network Coding-Based Cooperative Content Distribution Scheme. *ICC Workshops - In Proc. Of IEEE International Conference on Communications Workshops*, pp. 360-364.

Torres Compta, P., Fitzek, F. H. & Lucani, D. E. (2015). Network coding is the 5G key enabling technology: effects and strategies to manage heterogeneous packet lengths. *Transactions on Emerging Telecommunications Technologies*, 26(1), 46–55.

Tournoux, P.-U., Lochin, E., Lacan, J., Bouabdallah, A. & Roca, V. (2011). On-the-Fly Erasure Coding for Real-Time Video Applications. *Multimedia, IEEE Trans. on*, 13, 797 - 812. doi: 10.1109/TMM.2011.2126564.

Vasseur, J.-P. & Dunkels, A. (2010). The IP for Smart Object Alliance. In *Interconnecting Smart Objects with IP* (pp. 289-294). Morgan Kaufmann. doi: 10.1016/B978-0-12-375165-2.00018-1.

Vasudevan, D., Subramanian, V. & Leith, D. (2010, 07). On ARQ for Packet Erasure Channels with Bernoulli Arrivals. pp. 1793 - 1797. doi: 10.1109/ISIT.2010.5513297.

Vater, A. (2011). *Efficient coding schemes for file sharing networks*. (Ph.D. thesis).

Vater, A., Schindelhauer, C. & Ortolf, C. (2010, 01). Tree network coding for peer-to-peer networks. pp. 114-123. doi: 10.1145/1810479.1810503.

Wang, M. & Li, B. (2008). Network Coding in Live Peer-to-Peer Streaming. *Multimedia, IEEE Trans. on*, 9, 1554 - 1567. doi: 10.1109/TMM.2007.907460.

Wang, N. & Ansari, N. (2011). Downloader-Initiated Random Linear Network Coding for Peer-to-Peer File Sharing. *Syst. J., IEEE*, 5, 61 - 69. doi: 10.1109/JSYST.2010.2063377.

Wu, H., Chen, M. & Guan, X. (2012). A Network Coding Based Routing Protocol for Underwater Sensor Networks. *Sensors (Basel, Switzerland)*, 12, 4559-77. doi: 10.3390/s120404559.

Wu, Y., Chou, P. & Kung, S.-Y. (2005). Minimum-Energy Multicast in Mobile Ad Hoc Networks Using Network Coding. *Commun., IEEE Trans. on*, 53, 1906 - 1918. doi: 10.1109/T-COMM.2005.857148.

Wunderlich, S., Gabriel, F., Pandi, S., Fitzek, F. H. P. & Reisslein, M. (2017). Caterpillar RLNC (CRLNC): A Practical Finite Sliding Window RLNC Approach. *IEEE Access*, 5, 20183-20197.

Xiaocong, Q. & Jidong, Z. (2010, 12). Study on the structure of “Internet of Things(IOT)” business operation support platform. pp. 1068 - 1071. doi: 10.1109/ICCT.2010.5688537.

Xu, J., Wang, X., Zhao, J. & Lim, A. (2012). I-Swifter: Improving chunked network coding for peer-to-peer content distribution. *Peer-to-Peer Netw. and Appl.*, 5, 30-39. doi: 10.1007/s12083-011-0105-7.

Yazdani, N. & Lucani, D. E. (2019). Revolving Codes: High Performance and Low Overhead Network Coding. *In Proc. Of IEEE 2nd Wireless Africa Conference (WAC)*, pp. 1-5.

Yeung, R. (2011). Network Coding: A Historical Perspective. *Proc. of the IEEE*, 99, 366 - 371. doi: 10.1109/JPROC.2010.2094591.

Yeung, R. W., y Li, S. & Cai, N. (2006). Network Coding Theory (Foundations and Trends(R) in Communications and Information Theory).

Zeng, W., Ng, C. T. K. & Médard, M. (2012). Joint coding and scheduling optimization in wireless systems with varying delay sensitivities. *In Proc. Of 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pp. 416-424.

Zverev, M., Garrido, P., Agüero, R. & Bilbao, J. (2019). Systematic Network Coding with Overlap for IoT Scenarios. *In Proc. Of International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1-6.

Čolaković, A. & Hadzialic, M. (2018). Internet of Things (IoT): A Review of Enabling Technologies, Challenges, and Open Research Issues. *Comput. Netw.*, 144. doi: 10.1016/j.comnet.2018.07.017.