GETRF Delivrable 3: **Network Coding**

DragonNet: Specification, Implementation, Experimentation and Performance Evaluation

Ichrak Amdouni, Antonia Masucci,
Hana Baccouch, Cedric Adjih
September 2014

This work has been supported by DGA/ASTRID/ANR-11-ASTR-0033

Contents

Chapter 1

Introduction and Overview of the Network Coding

1.1 Presentation of the Document

This document represents the synthesis of the work package 3, "Network Coding" of the GETRF project, and is the deliverable for this work package.

In the project, we have developed a solution, called DragonNet, for efficient communications with network coding. Following the context of the GETRF project, we have kept in mind tactical military applications, and focused on two use cases:

- Mobile ad-hoc networks (as used in tactical military networks with microwave or VHF radio communication equipment for instance).
- Wireless sensor networks (used for instance with the goal of monitoring intrusions in strategic areas).

Thus, it appears that these two cases are characterized with a common feature: they are *multi-hop* wireless networks. This fact has implications on the design of network coding solutions.

During the work package, we focused on two objectives:

- Specifying and developing a generic solution for network coding communications: the solution has the name of DragonNet. Our emphasis is on providing a modular and universal solution that can be adopted in the previously cited scenarios and, in reality, also in many other ones.
- Evaluating the performance of the solution, and of network coding in general, in the context of multi-hop wireless networks.

1.2 Synthesis of the Activities

Concerning specification-related activities, our main results that are summarized in this report are:

- Design of a complete modular solution: DragonNet. This solution is responsible of: coding, decoding, maintaining necessary information and the associated signaling (see Chapter 2). It is designed to be extensible.
- Detailed specification of DragonNet for multicast communications. This specification was proposed as an IRTF draft [4] and presented in the IRTF network coding research group NWCRG [1] (see Chapter 3), to generate discussion, and contribute to the research group.
- Detailed specification of DragonNet for wireless sensor networks. This specification derives from the previous one and mainly it updates the control message format (see Chapter 4).

Concerning implementation and performance evaluation, our main results are:

- Design of a building block for coding, decoding and signaling. This module is called CISEW. Mainly, it refines the module SEW in DragonNet and makes it more general. CISEW has been proposed as an IRTF draft [5] (see Chapter 5).
- Implementation and performance evaluation of DragonNet with:
  - Real sensor networks: IoT-LAB [2] (see Chapter 6).
  - ns3 simulator (see Chapter 7);
- Analytical analysis of network coding performance, in terms of capacity and energy, in multi-hop wireless networks. This work is published as an Inria research report [8] and in the conference [7] (see Chapter 8).
- Illustration of this theoretical result via the computation of the max-flow (see Chapter 9). This computation is done over the transmission graph that is obtained (1) first, by a pure TDMA scheduling of nodes, (2) Second, by scheduling nodes transmissions via a coloring algorithm called VCM (Vector based Coloring Method). This illustrates also the integration of two works under the GETRF project (integration of VCM with the theoretical result about the capacity and efficiency in grid networks).

[1] https://irtf.org/nwcrg
[2] https://www.iot-lab.info/

1.3 Related Work

1.3.1 Network Coding

In classical wired or wireless networks, coding is restricted to the sources and the end receivers, whereas the intermediate nodes are only in charge of routing and copying payloads. Network coding departs from this traditional end-to-end forwarding paradigm by enabling intermediate nodes to mix the received payloads. Hence, the intermediate nodes may recombine several input payloads into one or several output payloads.

The idea of *network coding* has been introduced by Ahlswede, Cai, Li and Yeung in [15]. Since then, research on network coding has attracted significant interest from the research community.

1.3.2 Fundamentals of Network Coding

In this section, we present terminology and fundamentals of network coding. We describe a general framework using the terminology adopted in this draft [6].

1.3.2.1 Linear Coding and Random Linear Coding

Network coding differs from classical routing by permitting coding at intermediate nodes. In network coding, there is **a source** that transmits information to the network. This information is called a **"flow"**. The flow represents a sequence of bytes at the source that needs to be broadcast. The source divides the flow in a sequence of **payloads**. The payloads are numbered, and can be identified by their payload index. The payloads of one flow may optionally be divided in several coding blocks (one by default). The source may have an arbitrary number of flows. We speak about **intra-flow** coding when each flow is coded independently. We speak about **inter-flow** coding when inter-flow coding is allowed.

One possible coding algorithm is **linear coding** that performs only linear transformations through **addition and multiplication** (see Li et al. [23] and Koetter et al. [16]). Precisely, linear coding assumes identically sized payloads. These payloads are vectors on a fixed **Galois field**. Let consider any source that multicasts k payloads $(p_j)_{j=1,\dots,k}$. At any time, any node v receives payload that is a linear combination of payloads p_j; that is:

i^{th} received coded payload at node v: $y_i^{(v)} = \sum_{j=1}^{j=k} g_{i,j} p_j$

The sequence of coefficients for a coded payload $y_i^{(v)}$ at the node v is $[g_{i,1}, g_{i,2}, ..., g_{i,k}]$ is called the "**coding vector**" of payload $y_i^{(v)}$. The matrix of coefficients $[g_{i,j}]_{i=1...n,j=1...k}$, where n is the number of payloads received by any node v is called the **coding matrix**. Consider the example depicted in Figure 1.1. The vector $P = [p_1, p_2, \ldots, p_n]$ represents the original payloads generated by the source. Node v has the vector $y_1^v, \ldots, y_n^v$ of coded payloads. This vector is in fact obtained by multiplying the matrix G by the vector P.

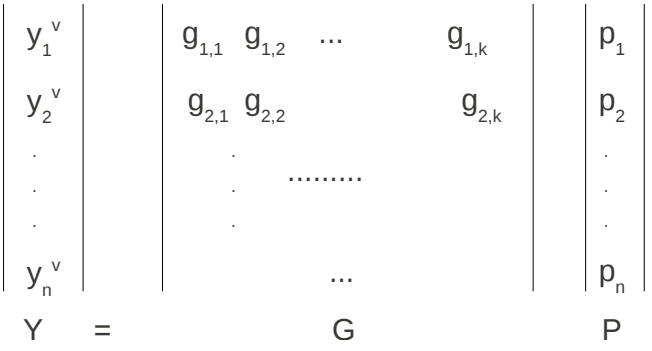


Figure 1.1: A set of coded payloads in a local buffer of a node v.

When a node generates a coded payload with linear coding, an issue is how to select coefficients. Whereas centralized deterministic methods exist, Ho and al. [24] presented a novel coding algorithm, which does not require any central coordination. This algorithm is called **random linear coding**: when a node transmits a payload, it computes a linear combination of all data it possesses with randomly selected coefficients and sends the result of the linear combination. In practice, a special header containing the coding vector of the transmitted payload may be added as proposed by Chou et al. [18].

Thinking in terms of coding vectors, at any point of time, it is possible to associate with one node v, **the vector space**, Π_v spawned by its coding vectors, and which is identified with its coding matrix. **The dimension of that vector space, denoted** D_v **,** $D_v \triangleq dim(\Pi_v)$**, is also the rank of the matrix**. By abuse of language, we call **rank of any node** v, that rank and dimension.

There are many network coding methods. We speak for instance about **block coding** when the original payload sequence is divided into blocks, called **coding blocks** (as known as **generations**), and coding is performed only over payloads within the same block.

There is also, the **sliding window coding** when the coding blocks are

selected based on a sliding window. For instance, for a window size = $[5, 9]$, the node should code payloads having indices between 5 and 19 (from p_5 to p_{19}). In this case, coding blocks of nodes may be partially overlapping, and, over time, moving to higher original payload sequence numbers. This method has the advantage to allow a real-time decoding; any node is not obliged to wait for the reception of a whole coding block to be able to decode payloads, as in the block coding method. The solution we propose for the network coding is based on a sliding window as we will see in the remaining of this report.

1.3.2.2 Decoding and Rank

The rank of a node is a direct metric for the amount of useful received payloads, and a received payload is called **innovative** when it increases the rank of the receiving node. Ultimately a node can decode all source payloads when its rank is equal to the total number of source payloads. In this case, we say that the decoding matrix **has full rank**. Decoding is done by **inverting the coding matrix** formed by the coding coefficients. As seen in the example of Figure 1.1, $Y = G \times P$. Hence, the original payloads $(p_i, i = 1 \dots n)$ can be determined by inverting the matrix G: $P = G^{-1} \times Y$. Matrix inversion can be performed by **Gauss Elimination**.

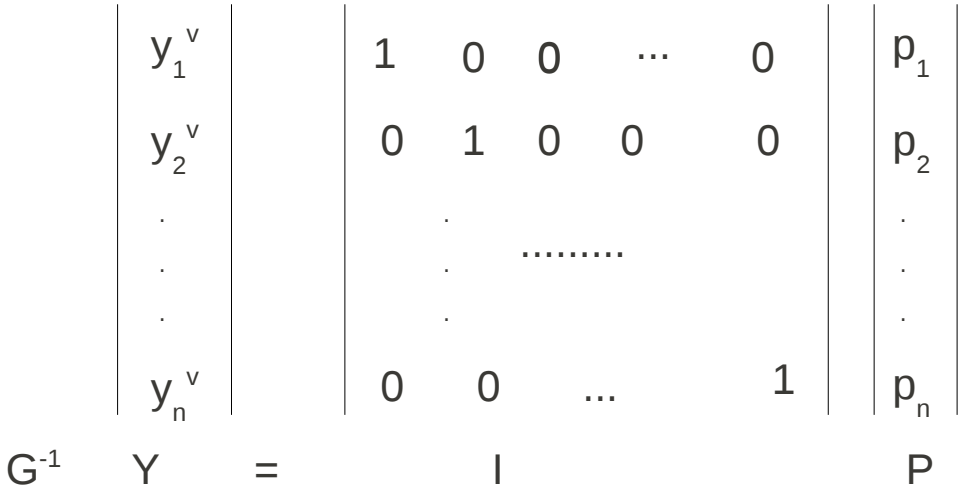


Figure 1.2: Decoding with Gauss Elimination.

1.3.3 Benefits of Network Coding

The efficiency provided by network coding in multicast and broadcast networks has been studied for instance, by Lun et al. [17], and Wu et al. [19]. In particular, they provide methods for determining optimal network coding parameters for a given network with specific model assumptions. The work of Fragouli et al. [20] gives insights for all-to-all broadcast and illustrates how gains could be obtained compared to classical routing.

It has been argued that the network coding is best suited to multi-hop wireless networks. Indeed, compared to wired networks, they have specific properties, see for instance [21], including:

- Wireless 'neighborcast': one wireless transmission by a node may reach several receivers. This property may be used to optimize broadcast.
- Time-variation: the visibility between two nodes may evolve with time, due to node mobility, physical changes in the propagation environment or other reasons.
- Unreliability of wireless communications: due to wireless channel conditions or properties, transmissions losses (packet erasures) potentially occur.

Intuitively, by combining the received packets, a coded packet sent by an intermediate node could benefit multiple receivers simultaneously, thus improving the bandwidth efficiency. In [15], it was shown that the multicast capacity (that is the maximum number of packets that can be sent from the source to a set of terminals per time unit) can be achieved by performing network coding at the intermediate nodes. A few years later, in [23], it was shown that for multicast networks, linear coding suffices to achieve the capacity limit, which is the max-flow from the source to each receiving node.

The authors of [25] show that random linear coding technique performs asymptotically as efficiently as any other network coding method in terms of capacity, for the case of single source multicast [25], and its performance is determined entirely by the average rates of nodes [26].

By reducing the number of transmissions required to transmit some amount of information, network coding achieves energy efficiency. In Chapter 8, we analyse the performance of network coding in torus networks. Results show that the capacity in a torus grid is equal to the number of neighbors of a node. Moreover, network coding in such networks is "near optimal" in terms of energy efficiency, in the sense that each transmission will provide innovative information (outside the vicinity of the source).

To benefit from the aforementioned advantages, we propose a generic framework for network coding called DragonNet. DragonNet derives from an existing solution called DRAGONCAST (various parts are described in [1–3]). The description of DRAGONCAST is proposed in the next section.

1.3.4 DRAGONCAST

DRAGONCAST is a protocol for broadcasting a set of packets from one source to the entire network with network coding. The base functioning is simple: the broadcast is initiated by transmissions from the source. Every node in the network retransmits coded payloads with a changing interval between transmissions. At the same time, every node collects received coded payloads and performs decoding as they are received. Finally, termination is automatically detected when all the nodes have successfully received all data.

DRAGONCAST provides a general framework with a modular architecture of three main components:

1. DRAGON "Dynamic Rate Adaptation from Gap with Other Nodes": is a rate adjustment method. Every node is retransmitting coded packets with a certain rate; this rate is adjusted dynamically. Essentially, the rate of the node increases if it detects some nodes that lack too many coded packets in the current neighborhood. This is called a "dimension gap" and the adaptation algorithm is a Dynamic Rate Adjustment from Gap with Other Nodes (DRAGON).

2. SEW "Sliding Encoding Window" is a real-time decoding method. The general idea of SEW is that it restricts the mixed original payloads within an encoded payload from a window of a fixed size. Consequently, nodes are not constrained to wait for the whole set of original coded payloads from the source, instead, real time decoding is possible.

3. A termination protocol for ensuring the termination of the broadcast: each node stops transmitting payloads once its known neighbors and itself have sufficient data to recover all source payloads.

As we will see in the remaining of this report, DragonNet specifies and completes these modules by the information base and the signaling required to perform network coding in a wireless network.

Part I

Description of DragonNet

Chapter 2

DragonNet: a Generic Solution for Network Coding

2.1 Introduction

In this chapter, we will present a general overview of DragonNet. We will also describe the architecture of DragonNet and its different building blocks.

2.2 Overview of DrangonNet

DragonNet is a generic framework for network coding in wireless networks. It is based on intra-flow coding where the source divides the flow in a sequence of payloads of equal size (padding may be used). The design keys of DragonNet are simplicity and universality; DragonNet does not use explicit or implicit knowledge about the topology (such as the direction or distance to the source, the loss rate of the links, ...). Hence, it is perfectly suited to the most dynamic wireless networks. The protocol is distributed and requires minimal coordination. DragonNet architecture is modular, it is based on 5 building blocks (LIB, SIG, Protocol, SEW and DRAGON). Each block is almost independent. This makes DragonNet generic and hence adaptable to many application scenarios.

DragonNet derives from an existing protocol called DRAGONCAST. Indeed, DragonNet shares the same principles and theoretical overview of DRAGONCAST. It enriches DRAGONCAST by the information base and signaling required to perform broadcast in wireless networks and in wireless sensor networks in particular. Furthermore, DragonNet provides a specification of its different blocks. The IRTF draft [4] is a detailed description of DragonNet and the material of this chapter derives from this draft. Notice

however, that in this draft, the same acronym of DRAGONCAST is kept. For clarity reasons, we replace it by DragonNet for the remaining of this report. Also, some acronyms are changed compared to this draft.

2.3 The Building Blocks of DragonNet

Figure 2.1 illustrates the different building blocks of DragonNet.

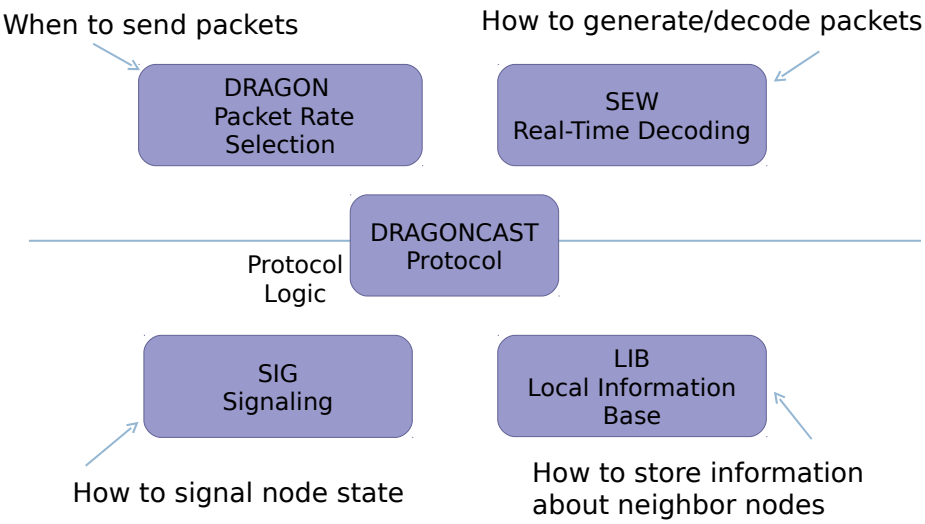


Figure 2.1: Organization of the different building blocks of DragonNet.

To make DragonNet the most universal possible, we separate the design of its building blocks into two categories: the protocol and the policy. The protocol category defines the general protocol aspects themselves and includes the information base LIB, the signalisation SIG and the Protocol itself. The policy category defines the higher protocol behavior and it includes the sliding encoding window SEW and the dynamic rate adaptation DRAGON.
In the following, we give a brief description of all these building blocks. A detailed specification of these modules is provided in Chapter 3.

2.3.1 LIB: Local Information Base

This module is responsible for maintaining all information required for the functioning of the protocol. This information base maintains information

about the flows, the decoding process, and the state of the neighbors.

2.3.2 SIG: SIGnalisation

This module provides the signaling for the control plane for DragonNet. The signaling consists mainly on the specification of a header for each coded payload (e.g piggybacking). It includes information relative to the state of the node, in addition to the packet encoding information. This allows each node to maintain information about the state of its neighbors.

2.3.3 Protocol

This module is the protocol itself with message generation and message processing.

2.3.4 SEW: Sliding Encoding Window

SEW is a real-time decoding method. This method relies on implicit cooperation between neighbor nodes, in order to allow recovery of some source payloads without requiring to decode all source payloads at once. Technically, as described in [1], it ensures the existence of a low triangle in the coding matrix during the online Gauss elimination process. The method SEW relies on two principles:

- SEW coding rule: generates only coded payloads that are linear combinations within a given window. The determination of this window is a policy for SEW.
- SEW decoding rule: when decoding, performs a Gaussian elimination in such a way that one coded payload is only used to eliminate the source payload with the highest possible index (i.e. the latest source payload).

2.3.5 DRAGON: Dynamic Rate Adaptation from Gap with Other Nodes

DRAGON is a dynamic payload rate adjustment policy. Every node transmits coded payloads with a specific payload rate. With DRAGON, this rate is adjusted dynamically. Essentially, the rate of the node increases if it detects that some nodes in the current neighborhood are "falling behind" in the decoding process. This is called a "dimension gap". DRAGON provides a heuristic to avoid this gap.

2.4 General Functioning

Figure 2.2 illustrates the general functioning of DragonNet.

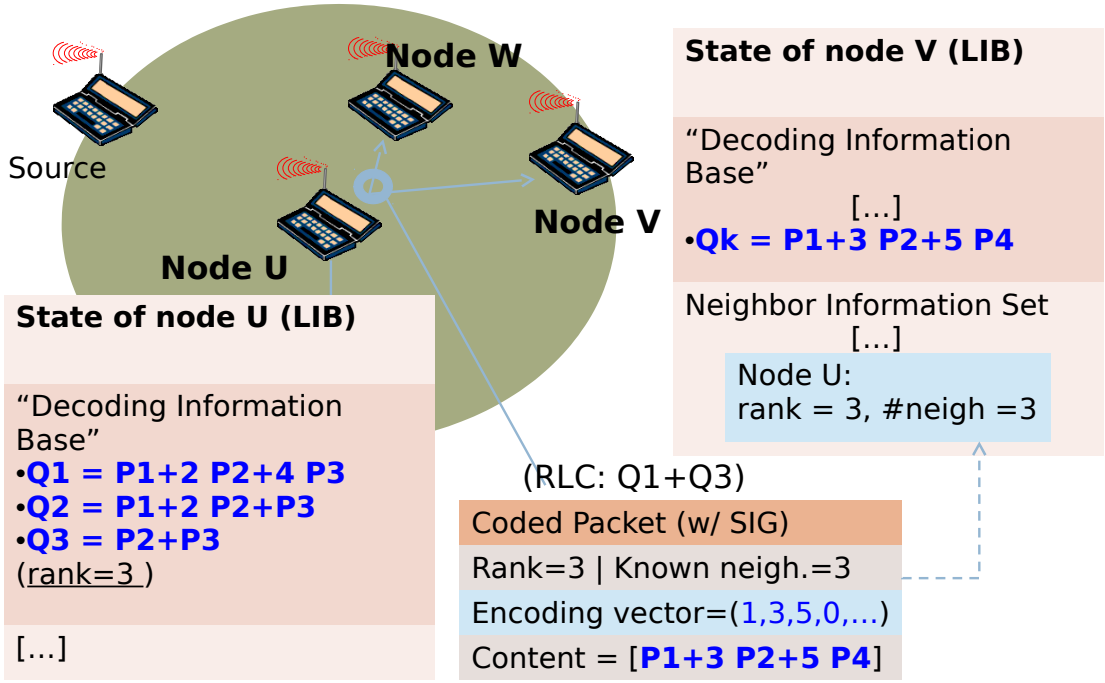


Figure 2.2: The general functioning of DragonNet.

The source initiates broadcasting by sending its original data payloads with a format specified in Section 3.2 from Chapter 3. These payloads have a predefined, constant size; padding can be used if necessary. Other nodes initiate transmission of encoded payloads upon receiving the first coded payload. As an example, we observe that the node u in Figure 2.2 has received payloads Q_1, Q_2 and Q_3 and has rank=3. These payloads are stored locally in the decoding information base as we will specify later. In addition to payloads, any node stores information relative to its neighbors. As an example, the node v has information about its neighbor u: (the rank and

the number of neighbors). Notice that received coded payloads are stored only when they are innovative. Then, nodes transmit payloads periodically. The transmission periodicity is decided by payloads rate selection algorithms. Precisely, when intermediate nodes receive a data payload that is a source payload or a coded payload, they start scheduling encoded data transmission. The scheduling interval is decided by the policy DRAGON. Payloads transmitted by intermediate nodes are coded payloads generated using random linear coding with a specified header. This header contains information needed for decoding. For instance, we see on Figure 2.2 hat node u piggybacks its message by adding its rank, the number of its neighbors, the encoding vector and the coded payload. Data transmission continues until nodes detect the termination condition, i.e. when themselves and all their neighbors have successfully decoded the data stream. However, a node may re-enter the transmission state. This happens when it receives a notification indicating that one neighboring node requires more coding vectors to recover some source payloads.

2.5 Conclusion

This chapter presented an overview about DragonNet, its building blocks and its functioning. Next chapter will specify the building blocks of DragonNet.

Part II

Specifications

Chapter 3

Specification of the Building Blocks of DragonNet

3.1 Introduction

The previous chapter gives an overview of DragonNet. This chapter details the specification of this architecture. Notice that this specification is available as a draft [4]. However, compared to the draft, the terminology used in this document is updated according to the network coding taxonomy draft [6].

3.2 SIG: The Signaling in DragonNet

DragonNet uses one single control message format based on a sequence of Type-Value including several protocol elements. The general message format is represented in Figure 3.1.

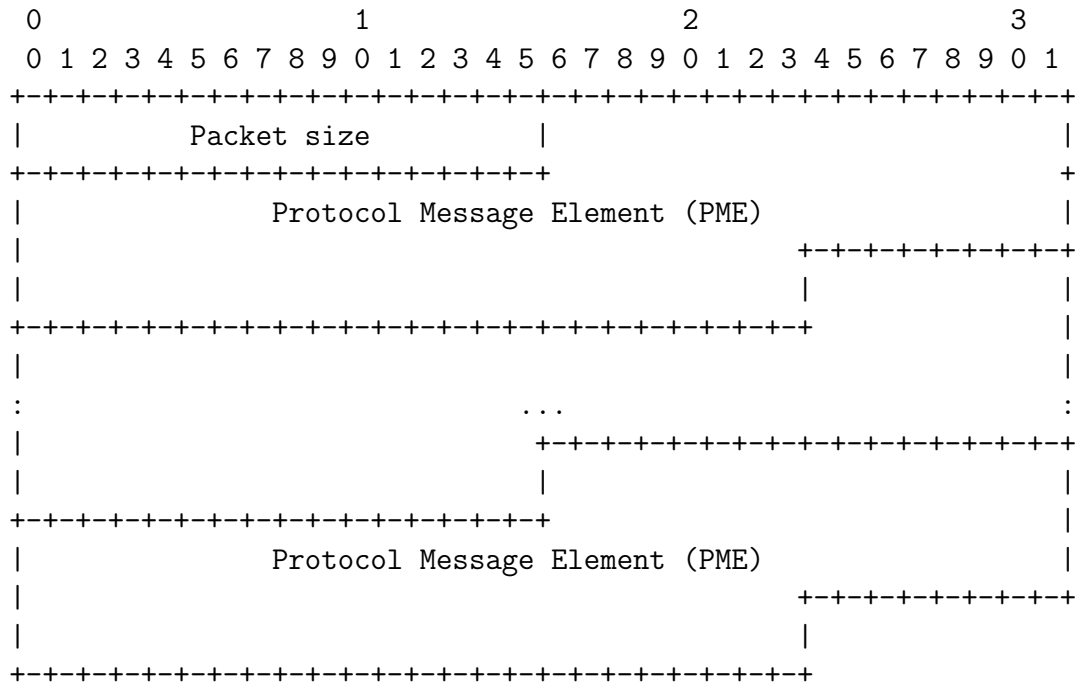


Figure 3.1: Format of DragonNet control message.

The following protocol elements are defined:

- The Flow Protocol Message Element (F-PME) depicted in Figure 3.2: it specifies information identifying the flow and the associated (constant) parameters.
- The State Protocol Message Element (S-PME) depicted in Figure 3.3: it specifies information relative to the state of the sender with respect to the decoding process.
- The Encoded Data Protocol Message Element (ED-PME) depicted in Figure 3.4: it specifies parameters of the encoding, along with the coding vector and includes the coded payload data.

They are used as the basis for DragonNet messages. Control information is sent in-band, prepended to encoded payloads. In the normal flow of the protocol, the majority of transmitted messages are "Data Payloads". In this version of the specification, the message MUST respect exactly one of the following formats:

- A regular data payload (with coded content): it MUST include the three following elements, in this order exactly: F-PME, S-PME, ED-PME.

- A termination control payload (e.g. without coded content): it MUST first include the two following elements, in this order exactly: F-PME, S-PME.

3.2.1 Format of the Flow Protocol Message Element F-PME

The Flow PME includes the following fields:

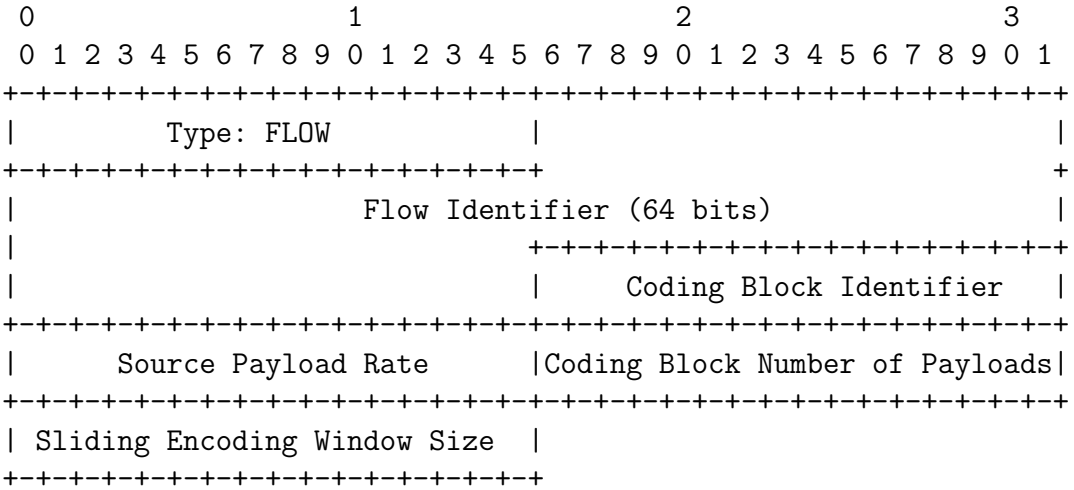


Figure 3.2: Format of the Flow PME (F-PME).

- Flow Identifier (Flow-ID): an identifier of size 8 bytes for the flow. Its semantics is opaque to DragonNet.
- Coding Block Identifier: DragonNet includes support for splitting a flow (with a given Flow-ID), which allows flows with more than 65535 payloads, and also allows to optionally operate with coding blocks.
- Source Payload Rate: expressed as the average inter-departure of the coded payloads in milliseconds (e.g. "10 payloads per second" yields the value 100).
- Coding Block Number of Payloads: total number of payloads in the coding block with the given coding block identifier.
- Sliding Encoding Window Size: the size of the encoding window, when generating coded payloads.

3.2.2 Format of the State Protocol Message Element S-PME

The State PME specifies state information of transmitter with respect to the coding block identified by a preceding F-PME. As depicted in Figure 3.3, the S-PME includes:

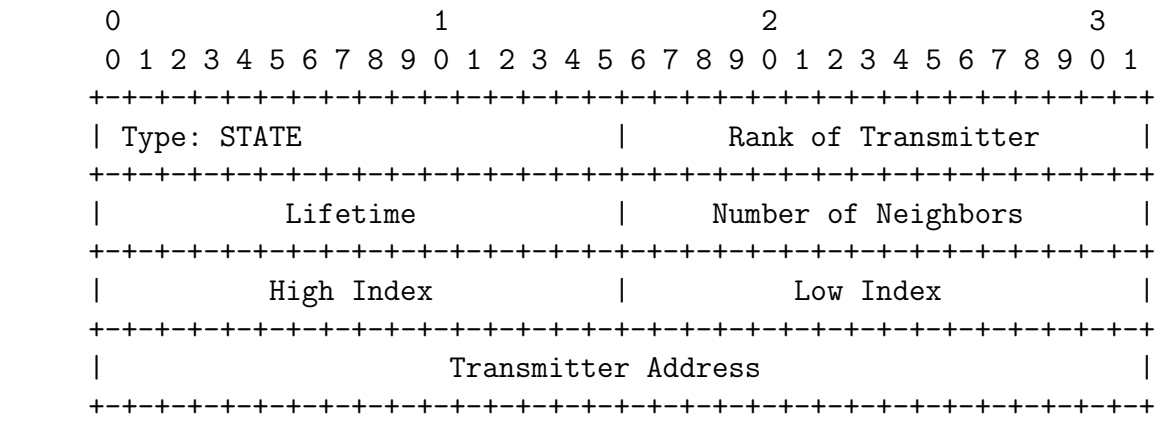


Figure 3.3: Format of the State PME

- Rank of Transmitter: denotes the current amount of innovative data of the transmitter.
- Lifetime: denotes the duration during which the information in this message (that is, the rank and the fact that the transmitter is a neighbor of a node receiving this message) is considered valid (after this it will expire).
- Number of neighbors: denotes the number of neighbors heard, that are not yet expired.
- High Index: specifies the highest index of the undecoded linear combination in the decoding table.
- Low Index: specifies the lowest index of the undecoded linear combination in the decoding table. Hence, all source payloads with lower indices have been decoded.
- Transmitter Address: the IP address of the transmitter of the message.

3.2.3 Format of the Encoded Data Protocol Message Element ED-PME

The Encoded Data PME holds actual coded payloads. It includes:

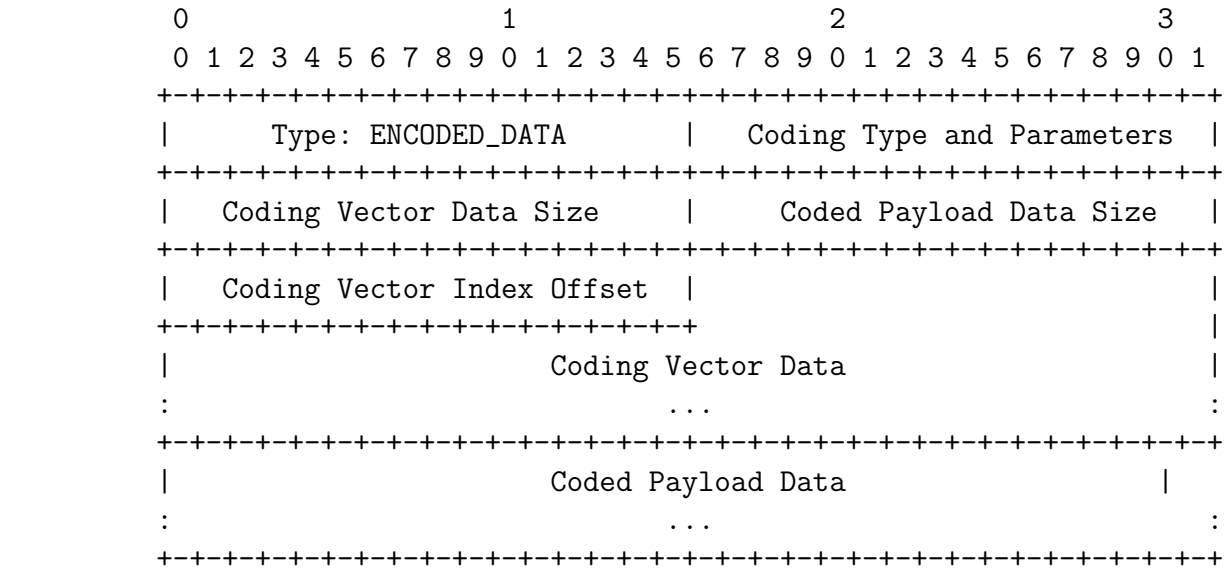


Figure 3.4: Format of the Encoded Data PME

- Encoding Type and Parameters: in this version of the specification, this fields contains one of the constants ENCODING_GF_2, ENCODING_GF_4, ENCODING_GF_16 or ENCODING_GF_256. This represents the fact that the fields GF(2), GF(4), GF(16) or GF(256) respectively are used as a basis for coding. As a result, the coefficients are respectively coded on 1, 2, 4 or 8 bits. Coding Vector Data Size: the size of the information representing the coding vector, in bytes. As indicated above, one byte will hold an integral number of vector coefficients.
- Coded Payload Data Size: the size of data payloads.

We define the following constants:

1. ENCODING_GF_2 = 0
2. ENCODING_GF_4 = 1
3. ENCODING_GF_16 = 2
4. ENCODING_GF_256 = 3

3.3 LIB: Local Information Base

Any node maintains a Local Information Base that records information about its decoding process and the state of its neighbors. The protocol state is maintained per flow: a flow is uniquely identified by a flow identifier. In addition, DragonNet supports the concept of partitioning a flow into coding blocks. In this version of the specification, each coding block is considered as an independent "flow".

The different information bases of LIB are structured hierarchically as follows:

- *Flow Information Set.* Each flow is independently associated with a Flow Information Tuple, which contains one or several coding blocks. The state of each coding block is maintained in a:
  - *Coding Block Information Set.* Each coding block contains information about the neighbors with respect to the propagation and the decoding of this coding block. This neighborhood information is stored in a:
    - *Neighbor Information Set.* Such information may also be provided by another protocol, such as OLSR [11] or NHDP [12].

    In addition, for decoding purposes, the flow information set includes the:
  - *Decoding Information Base.*

Now, we detail these different information bases.

3.3.1 Flow Information Base

Each node maintains a Flow Information Set which contains collected information about current flows. Specifically, the Flow Information Set consists of *Flow Information Tuples*. Each *Flow Information Tuple* contains the following information:

- F_flow_identifier: the identifier of the flow.
- F_source_rate: the payload rate of the source.
- F_coding_block_set: the Coding block Information Set associated to the flow.

3.3.2 Coding Block Information Base

Each node maintains, for each of its Flow Information Tuple, a *Coding Block Information Set*, which contains a *Coding Block Tuple* for each coding block. A *Coding Block Tuple* contains information relative to a coding block:

- G_coding_Block_identifier: an integer that identifies the coding block (coding blocks are numbered from 0).
- G_coding_block_size: number of coded payloads in the coding block.
- G_encoding_window_size: the size of the sliding encoding window (for SEW).
- G_decoding: the Decoding Information Base associated to this coding block.
- G_neighbor_set: the Neighbor Information Set associated to this coding block.

3.3.3 Neighbor Information Base

For each coding block, a node maintains a Neighbor Information Set which contains its known neighbors (with an expiration time), and information related to their state. Specifically, the *Neighbor Information Set* consists of *Neighbor Tuples*, each of which contains information about a single neighbor, for a given flow and for a given coding block, as follows:

- N_neighbor_address: address of the (heard) neighbor.
- N_neighbor_rank: the rank of the neighbor.
- N_high_index: high index of the neighbor.
- N_low_index: low index of the neighbor.
- N_validity_time: the validity time of the tuple (after which it expires).

3.3.4 Decoding Information Base

For each coding block, a node maintains a *Decoding Information Base* with the following content:

- D_coded_payload_set: a set of coded payloads. For each, the node maintains:

- Coding vector
- Coded payload content

During the decoding process, the decoding module (SEW) performs real-time decoding by performing Gaussian elimination on the list of coded payloads.

3.4 DragonNet Protocol Functioning

3.4.1 Source Payload Generation

A node that acts as a data source for a flow, also runs an instance of the DragonNet protocol for that flow (e.g. has a Flow Tuple with all associated information). In addition, it adds periodically source payloads in the associated Flow Information Base respecting the source rate.

3.4.2 Payload Processing

Whenever a node receives a coded payload:

- It updates its Flow Information Base related to the associated flow. This includes the rank and the expiration time.
- It notifies SEW for real-time decoding. Then, SEW will forward any decoded payload to the application.
- It notifies DRAGON which may update the transmission payload rate of the flow.

3.4.3 Payload Generation

For every "active" flow in its Flow Information Base, a node will generate coded payloads, with an interval between payloads determined by DRAGON. Based on its Local Information Base, every node is able to determine if it needs to send payloads, as described in [3]. If for a given flow and coding payload, in the associated neighbor set, no neighbor is known to require coded payloads, the payload is generated without an encoded payload (without Encoded Data PME).

3.5 DRAGON: Packet Rate Selection

In this section, we describe one payload rate selection algorithm, proposed for DragonNet.

3.5.1 DRAGON Rationale

The heuristic DRAGON has proposed and analyzed in [1] is inspired by Fragouli et al. [9]. We briefly summarize it in this section. The starting point of our heuristic DRAGON is the following observation. Indeed, for real-time decoding, the rank of nodes inside the network should be close to the index of the last source payload, and that in any case, they should at least evolve in parallel.

Thus, one would expect the rank of any node to grow at the same pace as the source transmission, as in the example of optimal rate selections for static networks. Decreasing the rates of intermediate nodes by a too large factor, would not permit the proper propagation of source payloads in real time. On the contrary, increasing excessively their rates, would not increase the rate of the decoded payloads (naturally bounded by the source rate) while it would decrease energy-efficiency (by increasing the amount of redundant transmissions).

The idea of the proposed rate selection is to find a balance between these two inefficient states. As we have seen, ideally the rank of a node would be comparable to the lastly sent source payload. Since we wish to have a simple decentralized algorithm, instead of comparing with the source, we compare indirectly the rank of a node with the rank of all its perceived neighbors.

The key idea is to perform a control so that the rank of neighbor nodes would tend to be equalized: if a node detects that one neighbor had a rank which is too low compared to its own, it would tend to increase its rate. Conversely, if all its neighbors have greater ranks than itself, the node does not need to send payloads in fact.

3.5.2 DRAGON (Dynamic Rate Adaptation from Gap with Other Nodes)

DRAGON is based on the following definitions. Precisely, let:

- D(v,t) denotes the rank of any node v at time t.
- N(v,t) denotes the number of neighbors of the node v at time t.

- g(v,t) denotes the maximum rank gap of v compared to its neighbors, normalized by the number of these neighbors. Then g(v,t) is evaluated as:

$$g(v,t) = Max_{for\ all\ u\ neighbor\ of\ v} \frac{D(v,t) - D(u,t)}{N(u,t)}$$

- We determine C(v,t): the payload rate of any node v at time t as follows:
  - if g(v,t) > 0 then: C(v,t) = A g(v,t) where A is some constant.
  - Otherwise, the node stops sending encoded payloads until g(v,t) becomes larger than 0.

When computing the payload rate selection, the node uses information about its neighbors stored in the Neighbor Information Base. Indeed, any node needs the rank of each of its neighbors as well as their total number. This information is deduced from the last received payloads. Although these payloads might not necessarily reflect the exact values at the computation time, they provide an estimate.

3.6 Real-time Decoding: Sliding Encoding Window (SEW)

In this section, we describe the method of DragonNet for real-time decoding, which allows recovery of some source payloads without requiring to decode all source payloads at once. It is related to the method from Sundararajan et al. [10] described for TCP.

The real-time decoding method, Sliding Encoding Window (SEW) relies on implicit cooperation between neighbor nodes in order to ensure the possibility of decoding. Technically, as described in [1], it ensures the existence of a low triangle in the coding matrix during the online Gauss elimination process.

The method SEW relies on two principles:

- SEW coding rule: generates only coded payloads that are linear combinations of consecutive source payloads within the first L source payloads, where L is a quantity that increases with time.
- SEW decoding rule: decoding is performed via Gaussian elimination in such a way that one coded payload is only used to eliminate the

source payloads with the highest possible index (i.e. the latest source payload).

In the following, we give the insights behind these rules.

3.6.1 Coding Rule of SEW

We introduce the following definitions.

Definition 1 (highest (resp. lowest) index of a coded payload) *The highest (resp. lowest) index of a coded payload, is the maximum (resp. minimum) index of its encoded payloads.*

Example:
For the payload $Q = P_3 + P_5 + P_7 + P_8$, the highest index is 8 and the lowest index is 3.

Because all coded payloads have their own highest index and lowest index, we can also compute the maximum of the highest indices of all undecoded payloads at any node, as well as the minimum of the lowest indices. Hence, we define:

Definition 2 (The high (resp. low) index of any node) *The high (resp. low) index of any node is the maximum (minimum) index of all its undecoded payloads.*

Notice that a node will generally decode the source payloads from 1 up to its low index.

To ensure real-time decoding, SEW uses knowledge about the state of neighbors of one node, namely their high and low index. Any node restricts the generated payloads to a subset of payloads of the source such that its perceived neighbors are able to decode nearly all of them, up to a margin K. Notice that once all these neighbors may decode up to the first L-K payloads, it is unnecessary for the node to include payloads $P_1, \dots P_L$ in its generated combinations.

Hence, the general idea of SEW is that it restricts the mixed original payloads within an encoded payload from a window of a fixed size K. In other words, any node v encodes only source payloads inside a fixed Encoding Window as:
i-th generated payload $q(v,i) = a(v,i,k)P_k + ... + a(v,i,k+K)P_{k+K}$, where the $(P_j, j = k, \dots k+K)$ is the set of payloads generated by the source, The sequence of coefficients for q(v,i) is the following coding vector: [0, 0, ..., a(v,i,k), a(v,i,k+1), ..., a(v,i,k+K), ...,0,0].

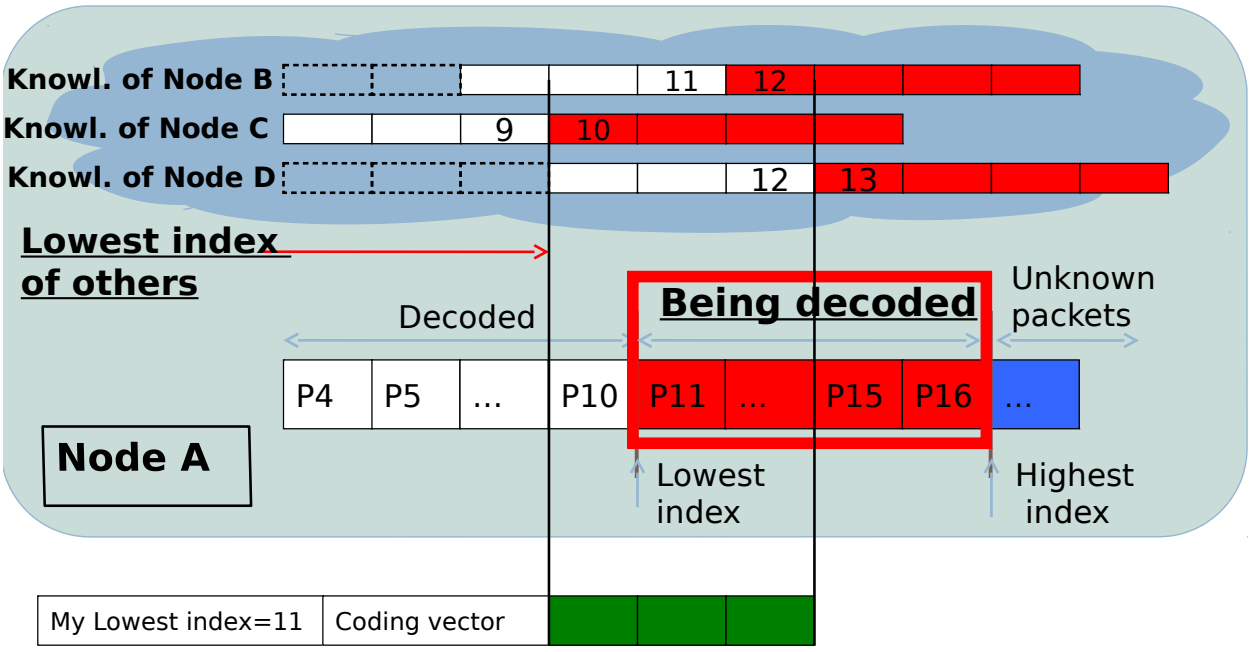


Figure 3.5: Illustration of coding rule of SEW

As an example, we consider the state of node A in Figure 3.5. Node A has decoded payloads until index 10. Its neighbors B, C and D have lowest indices equal to 12, 10 and 13. Hence, the lowest index at node A codes payloads starting from index 10 which is the minimum low index of its neighbors.

A node will repeat transmissions of new random combinations within the same window, until its neighbors progress in the decoding process.

3.6.2 Decoding Rule of SEW

The intent of the SEW decoding rule, is to guarantee proper functioning of the Gaussian elimination. An example of SEW decoding rule is the following: assume that node v has received payloads q_1 and q_2, for instance $q_1 = P_1 + P_9$ and $q_2 = P_1 + P_2 + P_3$. Then q_1 would be used to eliminate P_9 for newly

incoming payloads (the highest possible index is 9), and q_2 would be used to eliminate P_3 from further incoming payloads. On the contrary, if the SEW decoding rule was not applied and if q_1 were used to eliminate P_1, then it would be used to eliminate it in q_2, and would result into the computation of $q_2 - q_1 = P_2 + P_3 - P_9$; this quantity now requires elimination of P_9, a higher index than the initial one in q_2. In contrast, the SEW decoding rule guarantees the following invariant: during the Gaussian elimination process, the highest index of every currently undecoded payload will always stay identical or decrease.

Provided that the neighbor state is properly exchanged and known, the combination of the SEW coding rule and the SEW decoding rule, guarantees that ultimately every node will be able to decode the payloads in the window starting from its lowest index; that is, they guarantee early decoding.

Notice that improper knowledge of neighbor state might impact the performance of the method but not its correctness: if a node detects a previously unknown neighbor (for instance due to mobility), it will properly adjust its encoding window. Similarily, in DragonNet, obsolete neighbor information, for instance about disappeared neighbors, will ultimately expire.

3.7 Conclusion

This chapter provides a specification of the different building blocks of DragonNet. The design of DragonNet is modular and universal. The ideas presented here are sufficiently abstract to make their adaptation to many use cases in wireless networks possible. Examples of this adaptation are given in the next chapters.

Chapter 4

Specification of DragonNet for Wireless Sensor Networks

4.1 Introduction

This chapter describes the adaptation of the specification of DragonNet to wireless sensor networks (WSNs). The major modifications of the protocol are made on message format. Furthermore, the Local Information Base is updated. We also detail specifications of the decoding process.

4.2 DragonNet Message Format

New message format is introduced for DragonNet. The idea is to simply the header of this message. DragonNet is based on a unique message format. This message does not contain the Protocol Message Element as described in Section 3.2 from Chapter 3. However, it includes a sequence of values of several protocol elements. As depicted in Figure 4.1, this message includes:

- Node Id: The identifier of the message.
- Sequence Number: The sequence number of the message.
- Dragon Parameters: specifies decoding parameters.
- Flow Parameters: specifies informations about the flow.
- Coded Data : gives informations about encoded data.

We now detail these different parameters.

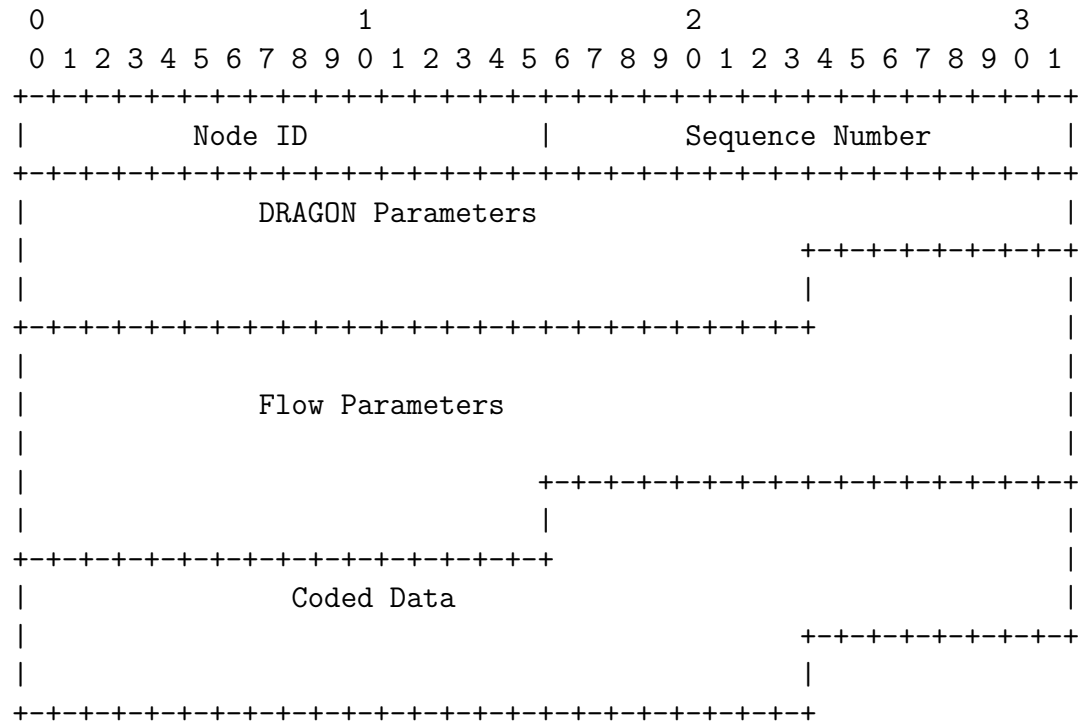


Figure 4.1: The format of DragonNet message as specified for WSNs.

4.2.1 Dragon Parameters

Dragon parameters (equivalent to the State Protocol Message Element defined previously in the old version of DragonNet) represents the informations relative to the decoding state of one neighbor. It includes :

- Rank: The rank of the neighbor.
- High Index: The highest index of the undecoded payloads of this neighbor.
- Low Index: The lowest index of the undecoded payloads of the neighbor.
- Neighbors number: The number of the neighbors.
- Last Time: The last time the current information was updated.

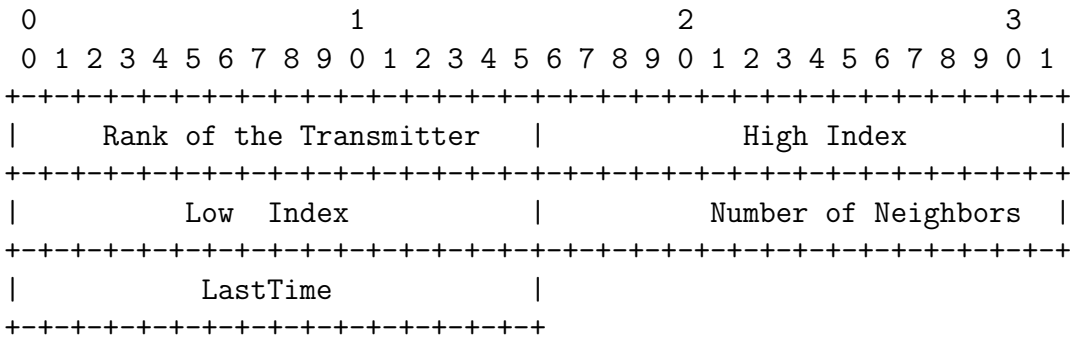


Figure 4.2: Dragon Related Parameters.

4.2.2 Flow Related Parameters

The Flow parameters (see Figure 4.3) specifies the parameters identifying the flow and the associated parameters. It includes:

- Massage Interval: inter message transmission interval.
- Sliding Encoding Window Size: The size of encoding window.
- Coding Block Size: the total number of payloads in the coding block.

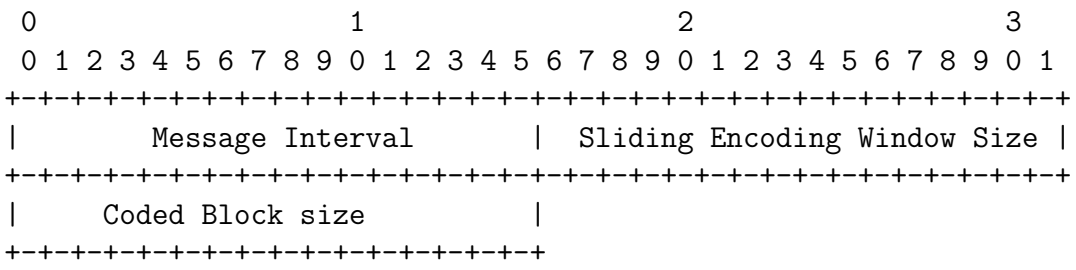


Figure 4.3: Flow parameters

4.2.3 Coded Data Related Information

The coded payload (see Figure 4.4) holds actual coded payload content and decoding informations.

- Coef Pos Min: The minimum index of the coded payload.

- Coef Pos Max: The maximum index of the coded payload.
- Data Size: The size of the coded data payloads.
- Coded Packet Data: The content of the coded payloads.

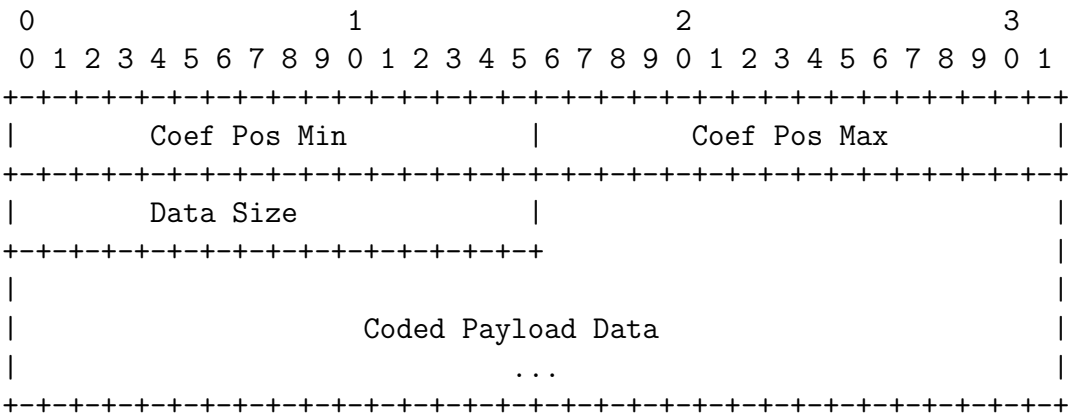


Figure 4.4: Coded Payload Related Parameters.

4.3 LIB: Local Information Base

A node maintains a Local Information Base containing information about his decoding process and its neighbors state.

Each node maintains a table containing its known neighbors and information related to their state:

- N_neighbor_rank : The rank of the neighbor.
- N_neighbor_high_index: The highest Index of the neighbor.
- N_neighbor_low_index: The lowest Index of the neighbor.
- Neighbors number : The number of the neighbors neighbors.
- N_last_time: The last time this information was updated.

When a node receives a packet data from its neighbor, it checks if the transmitter exists in the table. If the transmitter does not exist, the node creates a new entry in the table and saves the rank and the current time. If the transmitter exists already in the table, the receiver updates the current time and the rank relative to this neighbor.

4.4 Conclusion

This chapter presented a specification of the main modifications introduced on the DragonNet module to be adapted to wireless sensor networks. The main change was the use of a more compact (although less general) header, in order to accomodate for the shortest packet size of wireless sensor networks. Indeed, with 802.15.4, maximum packet size is 127, which would cause the previously proposed DragonNet header to induce a large overhead, on the order of 20 to 50 %.

Chapter 5

Specification of the Coding Interval-based Sliding Encoding Window (CISEW) for DragonNet

5.1 Introduction

This chapter focuses on CISEW; the sliding encoding window of DragonNet that refines and replaces SEW (CISEW is available as an individual IRTF draft [5]). CISEW offers a general framework for online network coding, with associated signaling. From this framework, several online coding policies can be implemented. Hence, we first start by proposing the general principles of CISEW. Then, we propose heuristics for the scenario of a real-time diffusion of content on nodes with limited resources.

5.2 Overview of CISEW

CISEW is a real time decoding method based on a sliding window. The method is proposed as a building block that could be integrated in a complete network coding protocol. This building block is responsible of:

- decoding: maintaining and decoding of the set of received coded payloads,
- recoding: selecting the content of generated coded payloads,

- signaling: allowing nodes to collect the information on the state of their neighbors.

Like SEW, CISEW aims at providing a real-time decoding via cooperation between neighbors. Every node, when it sends (re-)coded payloads also piggybacks information about its state, and specifically its decoding state (e.g. current matrix). CISEW enriches the signaling information used by SEW with additional information defined in this chapter.

CISEW processes received coded payloads with the same decoding rules as SEW. The difference comes from CISEW state advertisement, which is much richer, and is based on describing intervals of original payload indices, as indicated in Section 5.3.

As for SEW, this summarized state of the node is piggybacked on the coded payloads (along with encoding vector), and at each packet receipt, CISEW updates and maintains the information about its neighbors (for each node, it stores the interval indices).

Then, upon coded payload generation (re-coding), based on these intervals, the node determines the Encoding Window, that is the index interval of payloads it should encode. The intent of CISEW is to generate a coded payload that fits at best the requested intervals of these neighbors (with different possible policies for "at best").

Setting and processing the index intervals and determining the Encoding Window is a policy. Hence, CISEW design is general. Different policies are possible, and as a reference, we will specify some possible policies in Section 5.5.

Compared to SEW, the benefits of CISEW are:

- A finer adjustment of the coded payload to the decoding progress and the buffer state of neighboring nodes.
- An enhancement of the decoding possibilities upong receiving each coded payload.
- A management of the buffer overflow.
- A more extensive handling of buffer state cases, allowing for instance, for dropping part of the decoding buffer when a node "falls behind".
- CISEW design is general which makes it suitable to general applications.

5.3 Overview of CISEW Advertizement

CISEW main improvement starts with a much richer advertizement of the state of the node. The starting point is, the exact state of one node in its decoding buffer (that is, the matrix of Gaussian elimination). It is characterized as follows; each original payload, corresponding to an index, is one of:

1. original payloads that were decoded, but are no longer available (i.e. old, discarded, original payloads),
2. original payloads that were decoded, and are still available,
3. original payloads that are not yet decoded, but are present in coded payloads in the buffer (some of them are "seen", i.e. correspond to a pivot),
4. original payloads that are not yet decoded, and are not in any received coded payload.

CISEW invents the concept of Interval-based Coding, where the state of the node is concisely summarized by sets of intervals, with more precision than SEW (which only specifies the index of the first non-decoded payload).

Indeed, each node indicates (to its neighbors) which indices it needs, which ones it needs urgently, which ones it does not need, etc. To do so, CISEW determines and advertizes four types of index intervals:

1. Black: advertized interval of unwanted indices;
2. Grey: advertized interval of indices that the node is not interested in, but would not harm decoding;
3. Gold: advertized interval of indices that the node is interested in, in the near future.
4. White: advertized interval of indices that the node is interested in;

Thus, for instance, the "black interval" would typically correspond to old decoded original payloads that were already discarded: they can no longer be used by the node to decode newly received coded payloads, hence should be avoided.

Upon generation, the received, and stored, intervals of neighbors are taken into account. For instance, we can imagine that any node should not generate a payload with an index in the Black interval of all its neighbors, because this index is unwanted by all of them. However, it should try to mix payloads from the Gold interval of its neighbors.

5.4 CISEW Design and General Rules

In this section, we describe the design features of CISEW.

5.4.1 Signaling

CISEW uses the same type of local information base as defined in DrangonNet, specified in this document [4], and which maintains the state of the neighbors. However, some additional information are updated or added, reflecting the richer state advertizement of CISEW. This information concerns the Neighbor information set and the payload information set.

1. The Neighbor Information Set consists of Neighbor Tuples. For a given flow and for a given coding block, each neighbor tuple 'Neighbor Tuple' contains:

    - N_min_black_index: the minimum index of payloads in Black interval of this neighbor.
    - N_max_black_index: the maximum index of payloads in Black interval of this neighbor. It corresponds also to the start index of the Grey interval-1.
    - N_min_grey_index: the minimum index of payloads in Grey interval of this neighbor. It corresponds also to the end index of the Black interval+1.
    - N_max_grey_index: the maximum index of Grey payloads of this neighbor. It corresponds also to the start index of the Gold interval-1.
    - N_min_gold_index: the minimum index of payloads in Gold interval of this neighbor. It corresponds also to the end index of the Grey interval+1.
    - N_max_gold_index: the maximum index of Gold payloads of this neighbor. It corresponds also to the start index of the White interval-1.
    - N_min_white_index: the minimum index of payloads in White interval of this neighbor. It corresponds also to the end index of the Gold interval+1.
    - N_max_white_index: the maximum index of White payloads of this neighbor. It corresponds also to the end index of all payloads of this neighbor.

Notice that it is possible that any neighbor has an empty Black, Grey, Gold or White interval. In this case, the minimum and maximum indices associated to this interval are set to -1.

2. Payload Information Base: For each coding block, a node maintains a Payload Information Base with the following content:

   - D_payload_set: a set of coded and decoded payloads. For each, the node maintains: Coding vector Coded payload content
   - D_min_black_index: the minimum index of Black interval of this node.
   - D_max_black_index: the maximum index of Black interval of this node. It corresponds to the start index of its Grey interval-1.
   - D_min_grey_index: the minimum index of Grey interval of this node. It corresponds to the end index of its Black Interval+1.
   - D_max_grey_index: the maximum index in Grey interval of this node. It corresponds to the start index of its Gold interval-1.
   - D_min_gold_index: the minimum index of Gold interval of this node. It corresponds to the end index of its Grey interval+1.
   - D_max_gold_index: the maximum index of Gold interval of this node. It corresponds to the start index of its White interval-1.
   - D_min_white_index: the minimum index in White Interval of this node. It corresponds to the end index of its Gold interval+1.
   - D_max_white_index: the maximum index in White Interval of this node. It corresponds to the end index of all payloads in the payload set of this node.

   Notice that it is possible that any node has an empty Black, Grey, Gold or White interval. In this case, the minimum and maximum indices associated to this interval are set to -1.

5.4.2 CISEW Components

Figure 5.1 represents the organization of the different building blocks of CISEW.

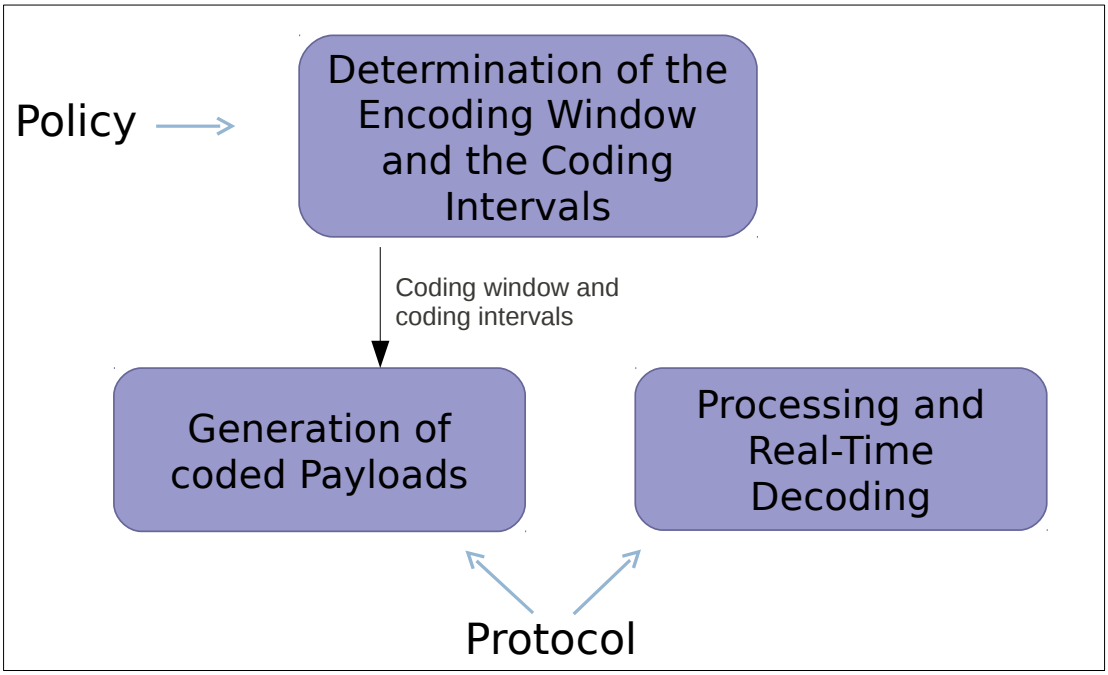


Figure 5.1: The components of CISEW

Now, we detail these components.

5.4.2.1 Determination of the Encoding Window and the Coding Intervals

This module is responsible for the determination of:

1. The encoding window: when any node attempts to generate a coded payload, this module starts by determining the index interval of the coded payload to be generated. The coding intervals (Black, Grey, Gold and White intervals) that each node should advertize.

This module is a policy for CISEW; hence CISEW rules can be applied to different policies. In Section 5.5.2 we propose sample heuristics for the implementation of this module. In practice, setting these intervals on a given node depends on:

1. The computing and the storage capacity of this node: as previously detailed, the intervals that the node advertizes indicate the indices it needs and the indices it does not need, etc. Hence, for instance, if the node advertizes a large Gold interval, it may receive a high number of undecoded indices that it should store and decode. Clearly, this is not suitable when the node has low computing and storage capacity.

2. The application requirements: as an example, we consider a real-time application and a code distribution application. A real-time application requires a real time decoding. Hence, nodes should evolve in parallel in their decoding. To ensure this, nodes should advertize coding intervals that are not too far apart. For instance, if a node notices that it is too late in the decoding process compared to all its neighbors (that is, its neighbors have already decoded many indices after its highest decoded index), it can increase its Gold interval even if it will never decode some payloads (and drop the older part of its decoding buffer). This prevents this node from delaying the global network.

3. In another type of application, such as code distribution (Over The Air reflashing of wireless sensor nodes for instance), all original payloads must be decoded. Hence, any node should still request the same indices of its undecoded payloads even if its neighbors are much more progressed compared to it.

5.4.2.2 Generation of Payloads

Each time the protocol DrangonNet requires a payload generation, it triggers CISEW that triggers itself this module.

If for a given flow and coding block, no neighbor is known to require coded payloads (all neighbors have already decoded all original payloads), the payload is empty. Otherwise, a coded payload is generated.

The CISEW procedure for generating a coded payload has the following rules:

The procedure has as an entry the encoding window interval determined by the previous component. All decoded or undecoded payloads from D_payload_set having indices in the encoding window interval are mixed.
Example: Consider a Decoding Information base containing the following payloads:

- q(1) = a(3) P(3) + a(5) P(5)
- q(2) = a(2) P(2) + a(6) P(6)

- q(3) = a(2) P(2) + a(3) P(3)

Assume that the encoding window is the interval [3, 6]. Hence, the node generates a random linear combination of payloads q(1) and q(2) but not q(3) since it includes the index 2.

5.4.2.3 Processing of Payloads

Whenever a node receives an encoded payload, CISEW:

1. Stores the coded payload received in this message. This step implies the management of the Coding Information Base, in particular the overflow of the payload set. Indeed, if the node receives a coded payload while its coded payload set, D_payload_set, is full, it should manage the situation. Many solutions are possible. For instance, this node may ignore the received payload, replace an old decoded payload by this payload, or replace another coded payload by the received one, etc. The choice of the best strategy is a policy for CISEW. We propose one in Section 5.5.2.

2. If the received payload is not ignored, CISEW:

   (a) Updates its Information Base related to the associated flow and neighbor.

   (b) Performs real-time decoding via a Gauss elimination applied to coded payloads stored in its D_payload_set.

5.5 Specification of Proposed Policies and Algorithms for CISEW

In this section, we propose specific policies, algorithms and heuristics for CISEW. We start by presenting one class of applications, deployment scenario, and associated assumptions, which inspired their design.

5.5.1 Applicability Statement

One typical class of application and scenario for our proposed policies and algorithm would be a real-time diffusion of content on nodes with limited resources. A prime example is the real time code update over a wireless sensor network. Regarding CISEW design, this context implies some considerations:

With limited computing capacity (such as with wireless sensors), CISEW should keep low decoding complexity. In particular, nodes should not decode in a very large encoding window. With limited memory (such as on sensor nodes), the decoding buffer will not hold the all decoded original payloads (and should drop them). We further assume that the DrangonNet protocol cannot use the secondary storage (e.g flash of sensors: this assumption holds because the flash access is time-consuming which is not preferable in a real time application). In a real-time application, the source generates coded payloads regularly and intermediate nodes decode these payloads. As discussed above, to guarantee that the global network evolves in parallel, every node should maintain a good decoding progress. For illustration, we give an example. Imagine a network with a late starting node. Assume that its neighbors have decoded the 10 first original payloads sent by the source and that these payloads are no longer stored by these nodes. The late node would request these payloads from its neighbors. However, in this case, it is not worthy for these neighbors to try to redecode these payloads. Because this may delay the whole network by prohibiting these neighbors to transmit random linear combinations of new payloads. Furthermore, it would be preferable for the late node to catch up other nodes rather than decoding old payloads. All these considerations are taken into account to define heuristics for CISEW as proposed in Section 5.5.2.

5.5.2 CISEW Heuristics

In this section, we will specify the CISEW algorithms.

5.5.2.1 Description of the Payload Set

Before specifying the procedures of CISEW, let us describe the payload set at any node denoted i.

Each node i has a payload set to store the decoded and undecoded payloads. However, it may be possible that this node withdraw some payloads. This happens for instance in case of memory overflow, or when the node judges that some payloads are no longer useful. Hence, we assume that each node i stores an admissibility variable denoted a(i) that is defined as follows:

• a(i) the minimum index of payloads that the node i stores and can process.

Consequently, at any instant, all index payloads smaller than a(i) are ignored. Payloads with indices $\geq$ a(i) can be decoded, undecoded or unheard.

The state of the payload set of any node i is described in Figure 5.2.

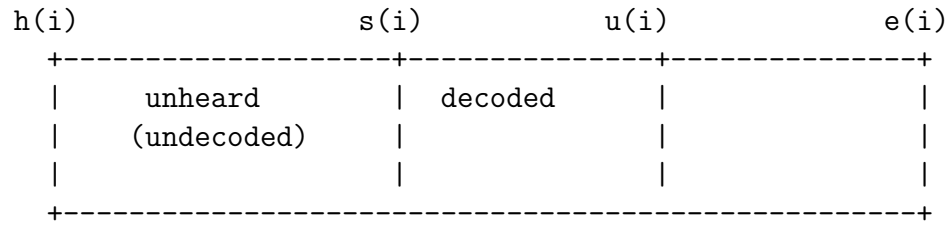


Figure 5.2: The buffer state of any node i.

We use the following notations,

1. s(i): ("coded payload set Start") minimum index of all payloads (decoded and undecoded) in the coded payload set of node i.

2. h(i): ("minimum unHeard") minimum index of unheard payloads having indices strictly smaller than the payloads in the coded payload set. Assume for instance that the source broadcasts payloads from index 1, and the node has as payloads: a(2)P(2) + a(3)P(3) and a(4)P(4) + a(5)P(5). Then h(i) is equal to 1, since the index 1 does not appear in any linear combination at node i. If there are not unheard payloads with indices smaller than the present payloads, then h(i) is set to -1.

3. u(i): ("minimum Undecoded") minimum index of undecoded payloads in the coded payload set.

4. e(i): ("coded payload set End") maximum index of payloads in the coded payload set at node i.

With these definitions, we have the following remarks:

1. If h(i) $\neq$ -1 then we have always h(i) $<$ s(i) by definition.

2. If h(i) $\neq$ -1 then the interval [h(i), s(i)] corresponds to unheard payloads (off course undecoded ones).

3. We have always s(i) $\leq$ u(i).

4. We can have s(i) = u(i) when the node i has not decoded any payload yet.

5. If s(i) $<$ u(i), then the interval [s(i), u(i)] corresponds to decoded payloads.

6. The first index in the interval [u(i), e(i)] corresponds to a decoded payload, the following ones are either decoded or undecoded.

To summarize, we can have one of the following schemas:

- **Case 1: h(i) = -1 and s(i) = u(i)**

This case corresponds to the initial state of any node. Initially, nodes receive undecoded payloads, store them but they are generally not able to decode them immediately. Also, for large Galois fields, the probability to mix payloads with low indices before payloads with higher indices is high, hence, h(i)=-1.

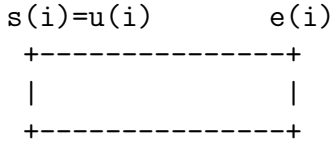


Figure 5.3: Case 1

- **Case 2: h(i) =-1**

This case corresponds to the scenario where the node i has started decoding some coded payloads. So, it is the normal case after a number of message exchanges.

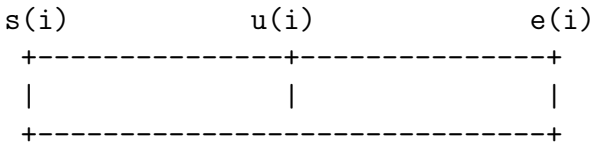


Figure 5.4: Case 2

- **Case 3: s(i) = u(i)**

The scenario that corresponds to this case is when the node i has not decoded any payload yet, and it did not heard some indices smaller than the indices it has stored. This scenario may correspond to the initial case where high indices are coded before low ones or for a late node that missed initial payloads transmitted in the network.

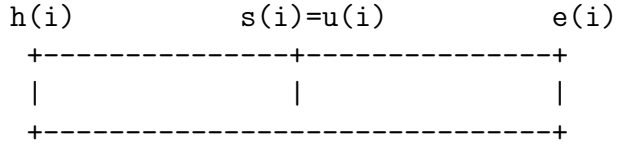


Figure 5.5: Case 3

- **Case4: h(i) < s(i) < u(i)**

This case may correspond to a late node that missed some payloads. However, this node succeeds to decode payloads because its neighbors are more progressed than it.

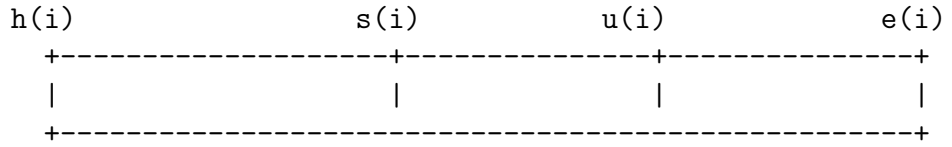


Figure 5.6: Case 4

5.5.2.2 Algorithm: Determination of the Coding Intervals

Consider any node i. The purpose of this algorithm is to determine the Coding Intervals that the node should advertize. The algorithm depends on the cases described in Section 5.5.2.1.

1. Case 1:
   (a) Black = [1,s(i)[.
   (b) Grey = empty.
   (c) Gold = [u(i), u(i)+MAX_GOLD].
   (d) White = [u(i)+MAX_GOLD, u(i)+MAX_GOLD+MAX_WHITE].
2. Case 2:
   (a) Black = [1,s(i)[. The node i has not the indices strictly smaller than s(i). Hence, all these indices are unwanted. Note that this interval starts from 1 because we assume that the source generates coded payloads starting from 1.
   (b) Grey = [s(i), u(i)]. This is because, all the payloads in [s(i), u(i)] are decoded.
   (c) Gold = [u(i), u(i)+MAX_GOLD].
   (d) White = [u(i)+MAX_GOLD, u(i) + MAX_GOLD+MAX_WHITE].
3. Case 3:
   (a) Black = empty.
   (b) Grey = empty.

(c) Gold = [u(i), u(i)+MAX_GOLD]. MAX_GOLD is the maximum size of this interval.

(d) White = [u(i)+MAX_GOLD, u(i)+MAX_GOLD+MAX_WHITE]. An arbitrary choice of the maximum size of the White interval is set to MAX_WHITE.

4. Case 4:

(a) Black = empty; this node needs all the payloads with indices starting from h(i). Hence, Black interval is empty.

(b) Grey = [s(i), u(i)]. This is because, all the payloads in [s(i), u(i)] are decoded. Consequently, they are not useful for the node, but does not harm the decoding. Indeed, when a node receives a linear combination with decoded payloads, it will replace these payloads by their stored value.

(c) Gold = [h(i), h(i)+MAX_GOLD]. First, the node i needs to receive linear combinations of payloads that it has not heard, that is in [h(i), s(i)]. Second, the undecoded payloads start at index u(i). Hence, the node needs to receive linear combinations starting at this index. Third, linear combinations in [s(i), u(i)] are not needed. However, for simplicity, we include them in the Gold interval.

(d) White = [u(i)+MAX_GOLD, u(i)+MAX_GOLD+MAX_WHITE]. White payloads are useful in the future. Hence, the White interval starts at the end of the Gold interval.

5.5.2.3 Algorithm: Determination of the Encoding Window Interval

When a sender node attempts to generate a coded payload, it needs to determine the encoding window for the payload it should generate. To do so, CISEW adopts the following general rules:

The selected encoding window should have a maximum size. Dealing with sensor networks, decoding has to keep a low complexity. Hence, the window size should not exceed a given threshold. Also, large encoding window increases the probability to include new payload indices in the coded payload generated. As a consequence, the least advanced node from decoding point of view (called latest node) would have new undecoded payloads instead of higher decoding opportunities. Let MAX _WINDOW_SIZE be the maximum size of the encoding window. Given an encoding window, any node may fail to generate a coded payload within this window. This happens when all the

payloads of this node include an index outside this window. In this case, the algorithm should adjust this window in order to avoid the generation of empty coded payload. The aim is to increase the number of neighbors whose rank is increased by receiving this payload. To determine the encoding window interval, any sender node proceeds as follows:

Any node determines the encoding window based on the advertized coding intervals of its neighbors. Note that the coding intervals of these neighbors may be overlapped or completely disjoint. For the use case described above, our strategy is to take into account the advertized state of the latest neighbor, that is the neighbor with the smallest value of h(i). If h(i) $\neq$ -1, or the smallest value of s(i) otherwise. We say that the encoding window is associated to this node.

Note that given the coding intervals associated to the latest node, the sender node may fail to generate a coded payload. This happens when all coded and undecoded payloads at this sender node contain at least one index outside the coding intervals of its latest neighbor. In this case, the sender node proceeds in two steps:

1. The sender node increases the encoding window (as we will see hereafter) and tries again to generate a coded payload within this modified window.

2. If the node fails to generate a coded payload at the end of the first step, it considers the second latest node and try to generate a coded payload associated to this node. If it fails again, it considers the third neighbor, etc. The node repeats this procedure while looping through the neighbors starting from the latest ones, until it generates a non empty payload.

The detailed algorithm is as follows:

1. Initialisation: S is empty

2. WHILE there are non visited neighbors and S is empty DO

   (a) Let N be the latest node among non visited neighbors.

   (b) W = union of Gold and White intervals of N without exceeding MAX_WINDOW_SIZE; that is W=[N.N_min_gold_index, minimum(N.N_min_gold_index+MAX_WINDOW_SIZE, N.N_max_white_index)]

   (c) S = set of coded payloads with indices in W

   (d) If S is not empty, then return W

(e) Otherwise, the coding intervals of N may be in one of the cases detailed in Section 5.5.2.2. Setting W depends on these cases as follows:

i. Case1: (Black—Gold—White) or Case 3: (Gold—White)
We cannot extend W in this case, so come back to the step 1 in the loop (select the next node N).

ii. Case2: (Black—Grey—Gold—White) or Case4: (Grey—Gold—White)
• Increase W to include the Grey interval of node N in addition to its Gold and White intervals, without exceeding MAX_WINDOW_SIZE; that is W=[x,y] where:
◦ x = N.N_min_grey_index
◦ y = minimum (x+MAX_WINDOW_SIZE, N.N_max_white_index)
• If S is still empty, then come back to the step 1 in the loop (select the next node N).

5.5.2.4 Algorithm: Processing

Let P be a received payload at any node i. To process this payload, i proceeds as follows:

1. If P includes indices that are lower than a(i), then P is ignored.

2. If the payload set is not full then i stores P in its payload set and performs Gauss elimination.

3. If the buffer is full, then i should decide whether it should keep P or ignore it. Hence, i starts by evaluating whether the payload P may help it to decode other payloads in the near future or not. If it is the case, then P is kept. Furthermore, the node i should empty memory space and discard another payload to be able to store the payload P. The procedure of the management of the payload set is detailed in Section 5.5.2.5. If upon applying this procedure, the payload P is kept, then P is stored and Gauss elimination is performed.

5.5.2.5 Algorithm: Management of the Payload Set Overflow

Before detailing the procedure for the management of the payload set overflow, we introduce the following notations relative to any payload.

1. min_index: the minimum index of this payload if it is undecoded. Otherwise, if this payload is decoded, then min_index is the index of this payload.

2. max_index: the maximum index of this payload if it is undecoded. Otherwise, if this payload is decoded, then max_index: the index of this payload.

This procedure is called when any node i receives a payload P while its payload set is full. Hence the node i proceeds as follows:

1. If P is in the Grey or Gold Interval of node i, then P is useful. Hence, i should discard the oldest decoded payload and store P.

2. If P is in Gold and White Interval of node i, then:

   (a) If P is "rather Gold" (that is: |i.N_max_gold_index - min_index| $\geq$ |max_index - i.N_min_white_index|) then P is kept and the oldest decoded payload is ignored.

   (b) If P is "rather White" (that is: |i.N_max_gold_index - min_index| $<$ |max_index - i.N_min_white_index|) then P is ignored.

5.5.2.6 Algorithm: Decoding and Gauss Elimination

CISEW keeps the same decoding rule of SEW. Indeed, when decoding, CISEW performs a Gaussian elimination, in such a way that one coded payload is only used to eliminate the source payload with the highest possible index (i.e. the latest source payload).

The intent of the CISEW decoding rule is to guarantee proper functioning of the Gaussian elimination. As an example, assume that node v has received payloads q(1) and q(2), for instance q(1) = P(1) + P(9) and q(2) = P(1) + P(2) + P(3). Then q(1) would be used to eliminate P(9) for newly incoming payloads (the highest possible index is 9), and q(2) would be used to eliminate P(3) from further incoming payloads. On the contrary, if the CISEW decoding rule was not applied and if q(1) were used to eliminate P(1), then it would be used to eliminate it in q(2), and would result into the computation of q(2) - q(1) = P(2) + P(3) - P(9); this quantity now requires elimination of P(9), an higher index than the initial one in q(2). In contrast, the CISEW decoding rule guarantees the following invariant: during the Gaussian elimination process, the highest index of every currently undecoded payload will always stay identical or decrease.

5.6 Conclusion

In this chapter, we presented a specification of the building block CISEW. CISEW allows a real time decoding by encoding payloads within a sliding

window. We also provided heuristics of CISEW for real time application over a limited resources hardware. One of the great improvements of CISEW over the previously designed module SEW is to be applicable (and well specified) in the case where one or several nodes are behind in the decoding process (possibly due to losses, due to late start, or some other cuase), and the network coding protocol decides that some parts would not be decoded.

Part III

Performance Evaluation

Chapter 6

Experiments of DragonNet on a real WSN testbed IoT-LAB

6.1 Introduction

In this chapter, we run experiments of DragonNet on a remote open testbed, IoT-LAB [13]. IoT-LAB is a very large scale testbed, remotely accessible, and includes a total number of 2728 nodes (in 6 sites); the nodes are mostly of type "wireless sensor nodes" with a wireless radio, and are well suited to perform wireless protocol experiments. A variant of this protocol was run on IoT-LAB: some results were presented in [14].

6.2 Setting of the Experiment

We run experiment on the Euratech testbed in the region of Lille using 20 nodes arranged in a line. Figure 6.1 shows a part of this testbed.

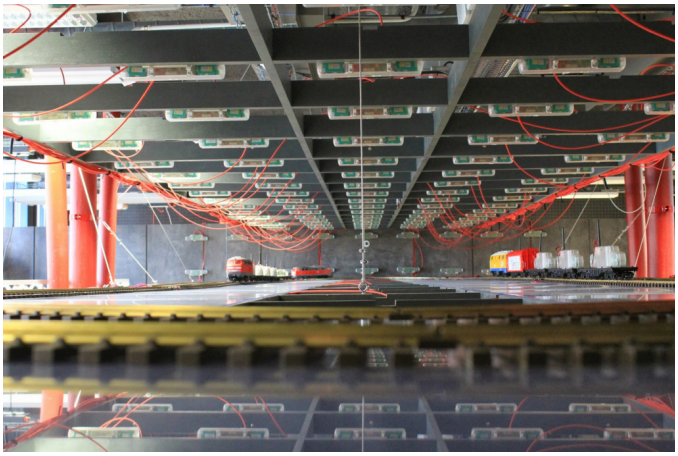


Figure 6.1: Euratech testbed in Inria Lille.

The nodes are equipped with MSP430F1611 micro-controller (16-bit, 48kB flash and 10kB RAM) and CC2420 radio.

We run experiment of the version of DragonNet adapted to wireless sensor networks with the simplified design of SEW. The rate adaptation is not activated in this version.

In the configuration tested, the cumulative distribution of the loss rate is illustrated in Figure 6.2: notice that the network is highly lossy. Less than 20% of the links have less than 18% loss rate, and in the same spirit, more than 40% have a loss rate greater than 40%.

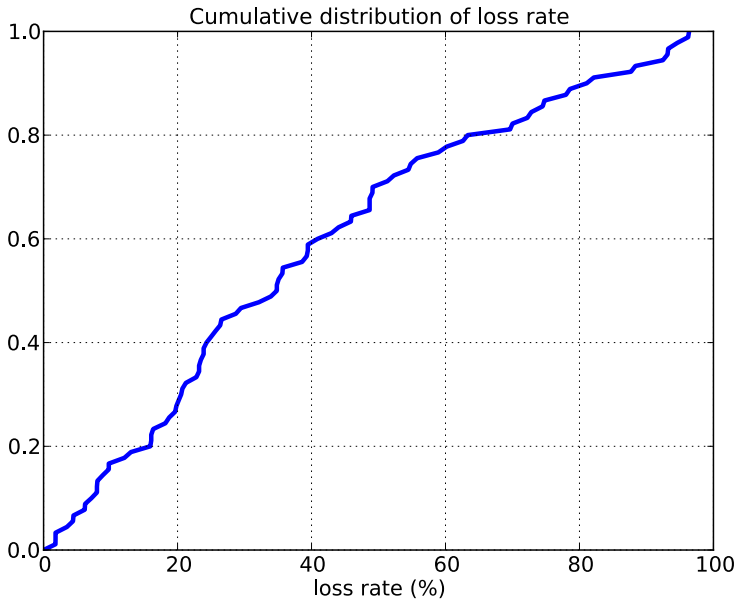


Figure 6.2: Cumulative distribution of the loss rate.

6.3 Results

We first study the evolution of the network in terms of rank and number of decoded packets. These results are depicted in Figures 6.3, 6.4 and 6.5. We set the window size to 15.

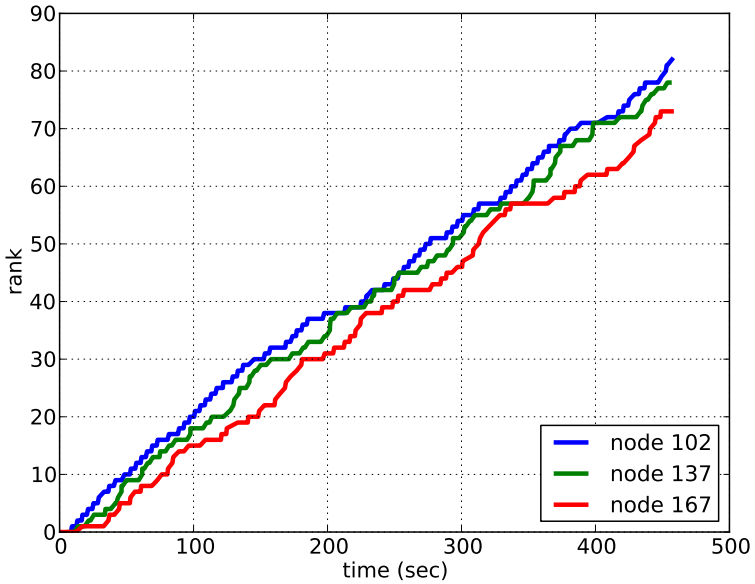


Figure 6.3: The rank evolution of nodes 102, 137 and 167.

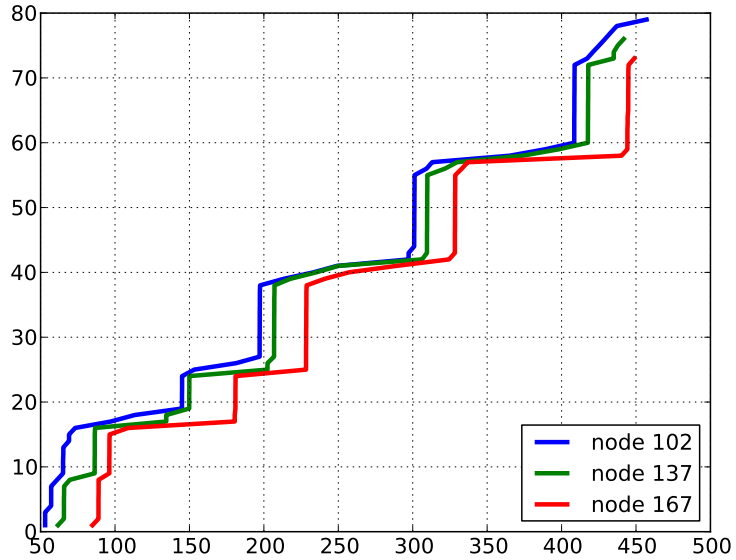


Figure 6.4: The number of decoded packets at nodes 102, 137 and 167.

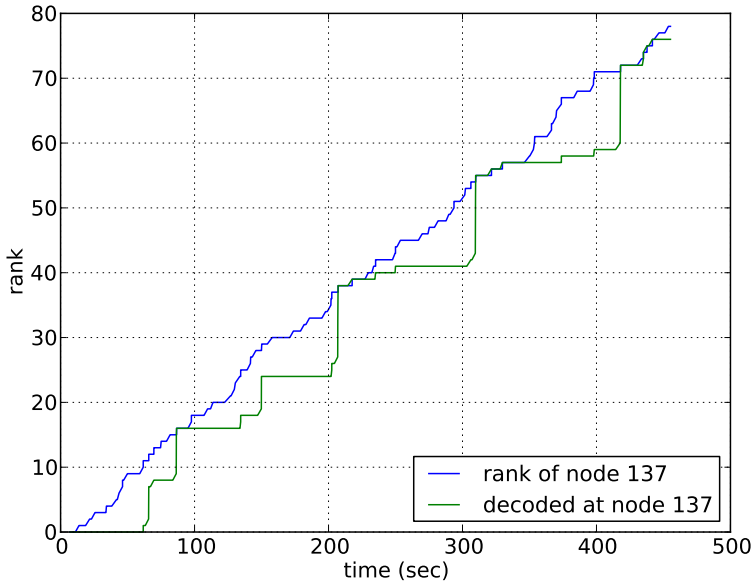


Figure 6.5: The evolution of the rank and the number of decoded packets at node 137.

From these figures, we notice the following remarks.

- The rank and the decoding process evolve progressively in time, which means that the DragonNet nodes are not blocked.
- The rank and the number of packets evolve very closely (see Figure 6.5). This is the advantage of the real time decoding of DragonNet. Whereas, with classical random linear coding, decoding might occur at the end.
- However, we still notice some plateau in the number of decoded packets plot. This is expected and corresponds to the instants where the neighbors of selected nodes advance their encoding window. Indeed, nodes code packets from their low_index to $low\_index + K$, where K is the window size. When this index in incremented, the receiver nodes cannot decode the received packets immediately because these packets have not been seen previously. This is confirmed by the result in Figure 6.6 that illustrates the evolution of the lowest index at nodes 172 and 162 which are neighboring nodes of node 167.

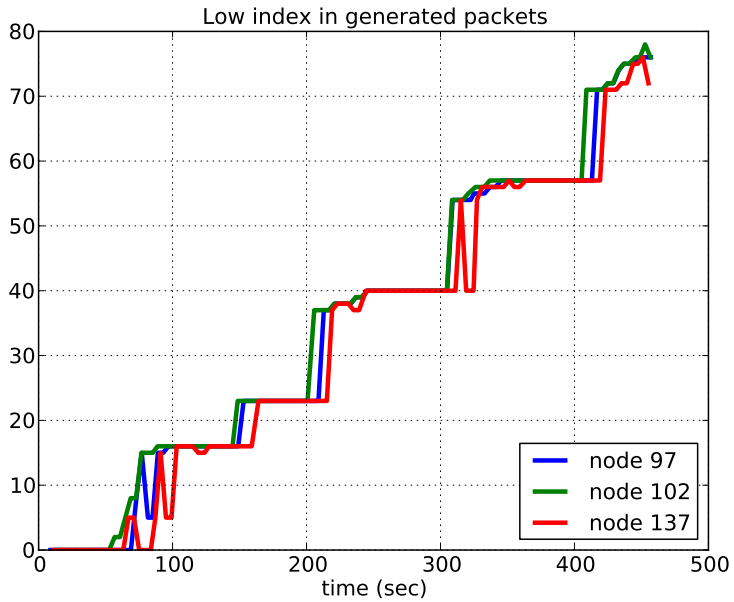


Figure 6.6: Low index of generated packets at nodes 172 and 162.

- Notice that, globally, the progress of the rank at the selected nodes is not very different. Same remark applies for the number of decoded packets. This means that the network nodes evolve in parallel. Of course, this result is positive when the nodes progress is good. This is the case here. The window mechanism of SEW allows nodes to have near progress speed which is beneficial mainly for real time applications.

  However, we notice that the progress of the rank of node 167 is the slowest one. This is because this node is the farthest node from the source compared to other nodes (see Figure 6.7 where the source is node 97).

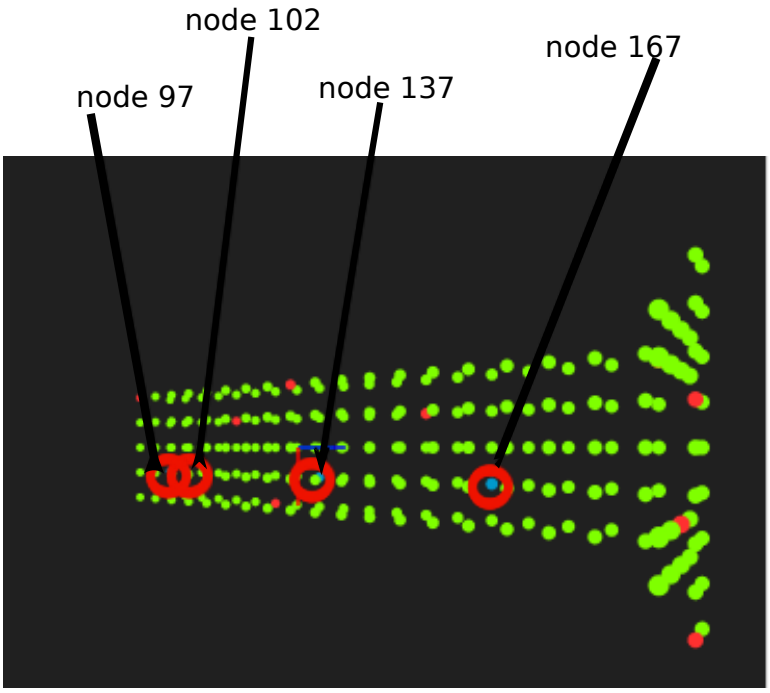
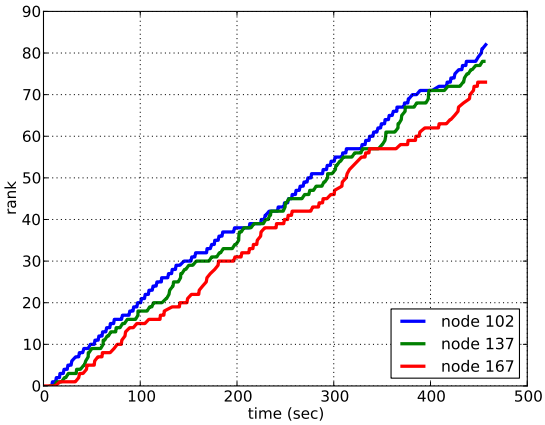


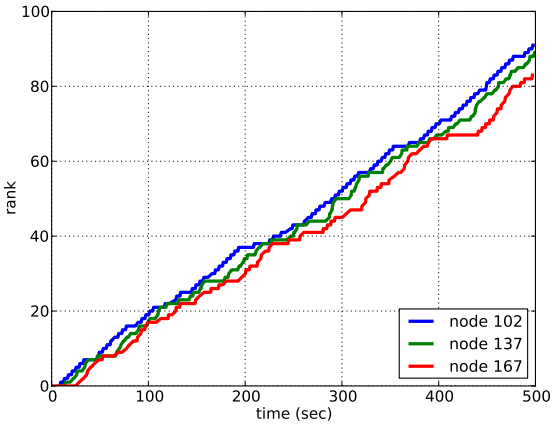
Figure 6.7: .

Indeed, the probability to receive innovative packets is higher at nodes close to the source. Hence, these nodes would decode earlier.

Now, we set the window size to 25 and run the same experiment. We want to determine the impact of the window size.

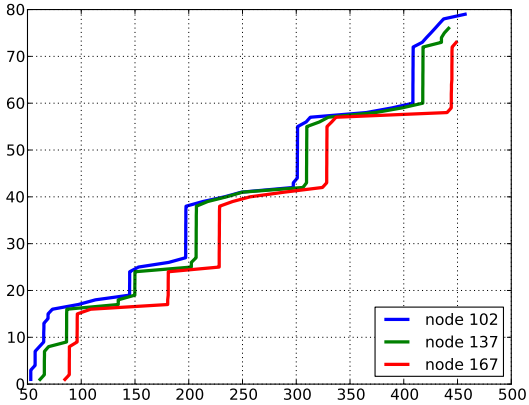


(a) window size = 15

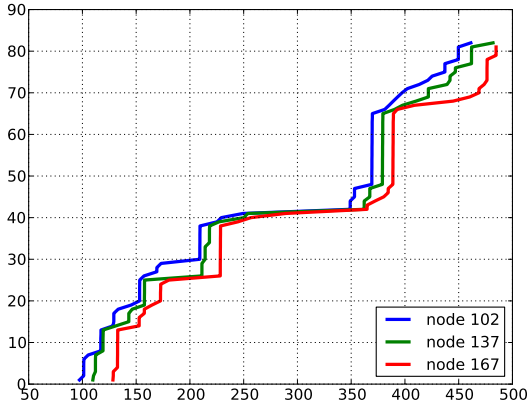


(b) window size = 25

Figure 6.8: The rank of nodes 102, 137 and 167 for two different window sizes.



(a) window size = 15



(b) window size = 25

Figure 6.9: The number of decoded packets at nodes 102, 137 and 167 for two different window sizes.

We observe that the rank evolution is almost not impacted by the window size. The number of decoded packets is slightly impacted; when the window size increases, the decoding is relayed. For instance, node 102 decodes 40 packets at instant 200 seconds if the window size is 15, and decodes the same number of packets at 350 seconds if the window size is 25. This is because, when the window size increases, the number of undecoded packets at any

node increases (new column in the coding matrix), and hence more time is needed for the node to decode them.

6.4 Conclusion

This chapter provides proof of concept that network coding and namely DragonNet works well on real wireless networks. Results highlighted the principles of SEW that allowed nodes to perform real time decoding, even on wireless sensor nodes with extremely limited resources. Of great interest is also the fact that the network coding operates well even with highly lossy and unreliable links found in our testbed.

Chapter 7

Performance of DragonNet under ns3 Simulator

7.1 Introduction

To evaluate the performance of DragonNet for wireless sensor networks, we have integrated it in the ns3 simulator. This implementation corresponds to the specification detailed in Chapter 4. In this chapter, we describe this integration and present some simulation results.

7.2 Description of the Framework

The ns-3 network simulator is a recently released next generation simulator intended to replace the popular ns-2 simulator. It introduces many features over its predecessor and aims to become the leading network simulator. The key objects in the ns3 simulator are Nodes, Packets and Channels as depicted in Figure 7.1. A node represents a network element, to which Applications, Protocol stack, and NetDevices can be added. Each NetDevice is attached to a channel, over which it sends and receives packets. A Node may be connected to more than one Channel through multiple NetDevices. A node may use multiple protocol components, which handle packets received by the NetDevice. Each node can also handle a list of Applications. The binding interfaces between the components of the node are designed to be similar to those in real systems. The interface between Application and Protocols is based on a class called Socket. This class is used by the application to send and receive traffic to the protocol stacks that are attached to a node. A Channel can represent a wired network cable or a wireless transmission medium. Each Channel object may be connected to a list of NetDevice.

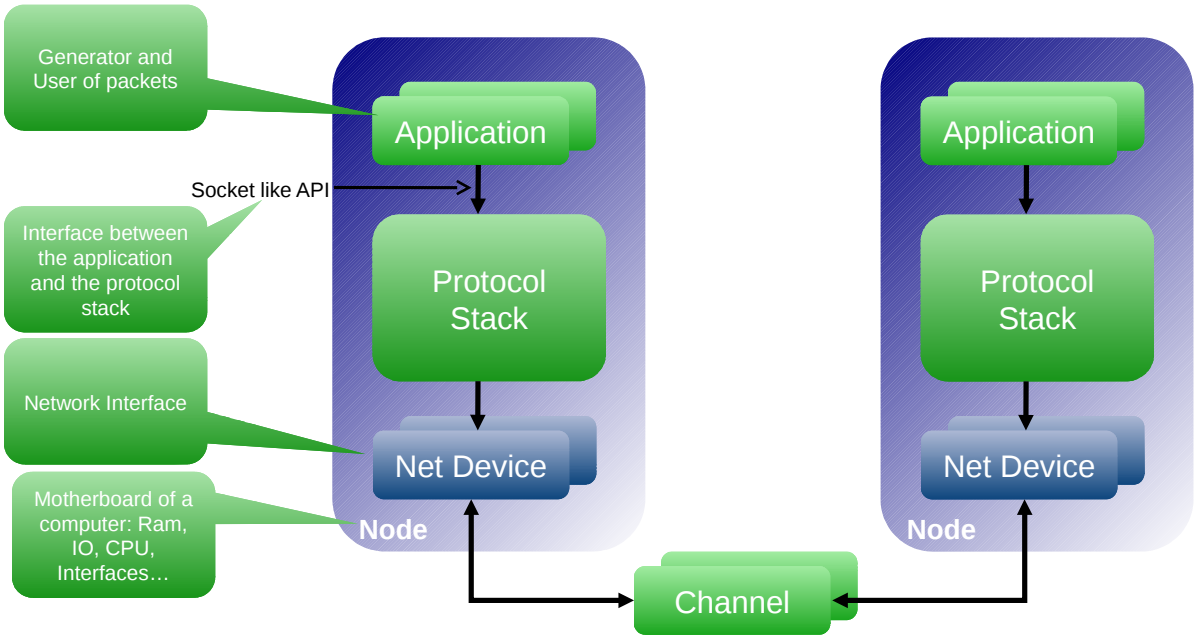


Figure 7.1: Architecture of ns3 Simulator.

We implemented DragonNet module as a C library. To integrate it in ns3 simulator, we have developed a ns3 DragonNet module respecting the architecture of ns3; it is a network module. It is parametrized and interoperable with all ns3 modules.

As illustrated in Figure 7.2, The DragonNet module is implemented as a multicast routing protocol. It implements the abstract class Ipv4RoutingProtocol which is a minimal interface consistiong mainly of two methods: RouteOutput and RouteInput. DragonNetCoding class imports methods from DragonNet C library. It handles coding and decoding of packets.

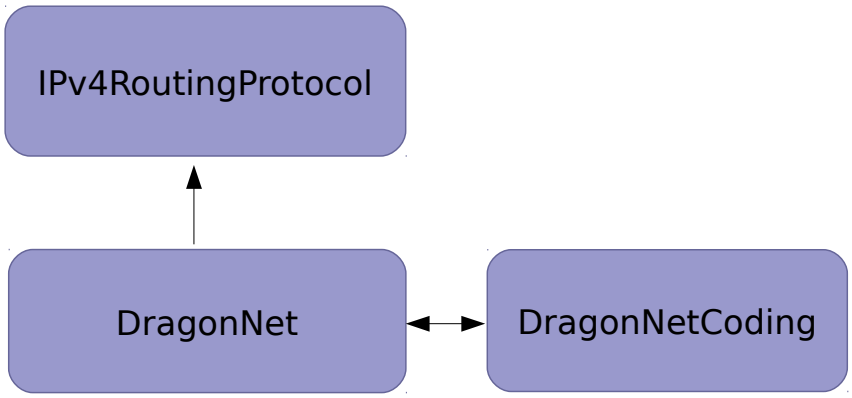


Figure 7.2: DragonNet ns-3 module.

7.3 Simulation Results

To evaluate the performance of DragonNet protocol as designed for wireless sensor networks. The rate adaptation policy is not activated in the simulated version. We run different simulation scripts. For each scenario tested, we evaluate particularly two metrics: the rank and the rate of innovative packets. We also consider mobile networks.

7.3.1 Grid scenario

In this scenario, we focus on a multi-hop network. We consider a grid of 11×11 nodes. The source transmits payloads with constant rate equal to 4 payloads per Second. The simulation runs for 1000 simulated seconds. We first compute the rank of some nodes selected randomly (see Figure 7.3).

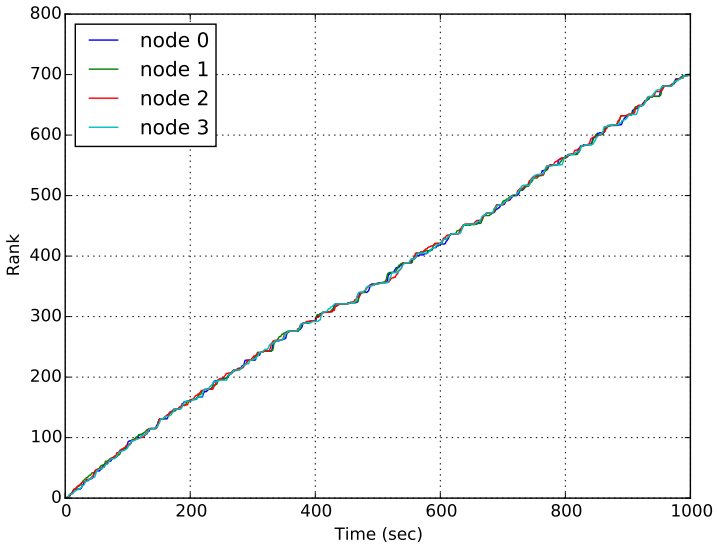
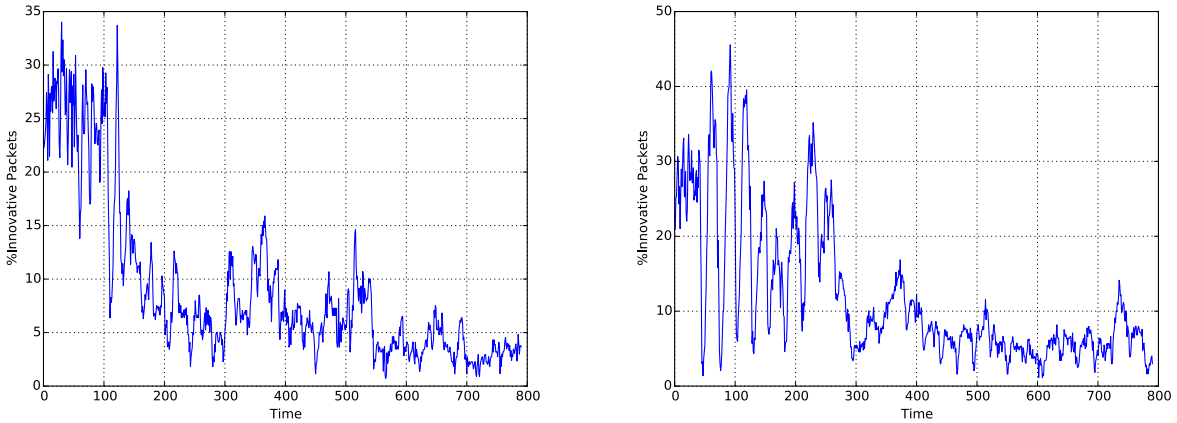


Figure 7.3: Variation of the rank in a grid network.

Results show that the rank at the selected nodes evolve almost in parallel and increases continuously. This plot is proof of concept also for DragonNet.

We also plot the rate of innovative packets received by nodes in the network at each time unit (see Figure 7.4) for two different window sizes.



(a) window size = 16

(b) window size = 25

Figure 7.4: Rate of the innovative packets in the network.

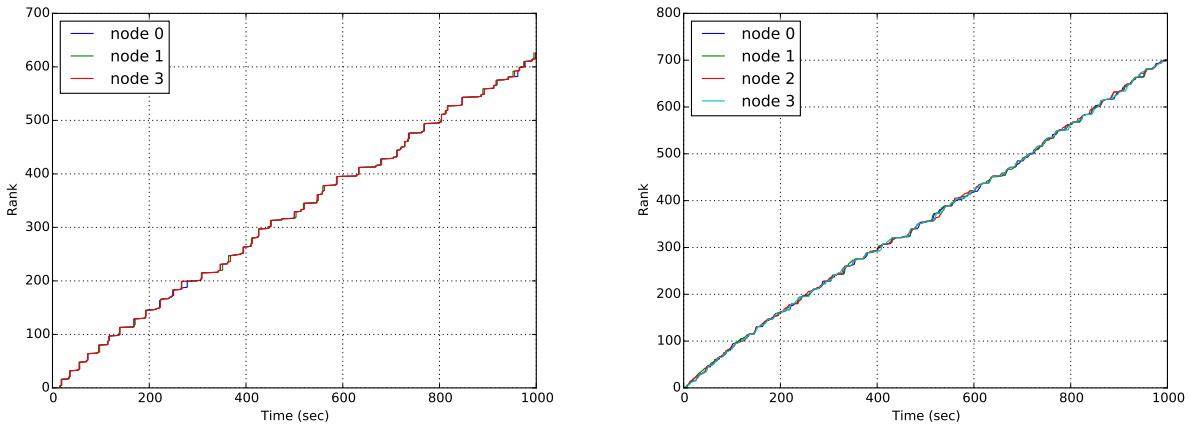
Results show that when the window size increases, the rate of innovative

packets increases also. This is expected since the probability to have coded combinations that are linearly independent increases when the encoding window increases.

7.3.2 Mobility scenario

In this scenario we use the same 11×11 nodes grid. The initial distance between the nodes is fixed to 140m. Nodes move according to RandomWaypointMobilityModel with a speed of $3000m/s$ for high speed mobility scenario $150m/s$ for low speed mobility scenario with no pause time within a $1540 \times 1540m$ grid region. The simulation runs for 1000 simulated seconds. Source rate is fixed to 4 packets per Second.

Figure 7.5 illustrates the evolution of the rank of the nodes. Results show that the rank increases more rapidly in low mobility network. This can be explained by the higher loss rate in high mobile scenario.

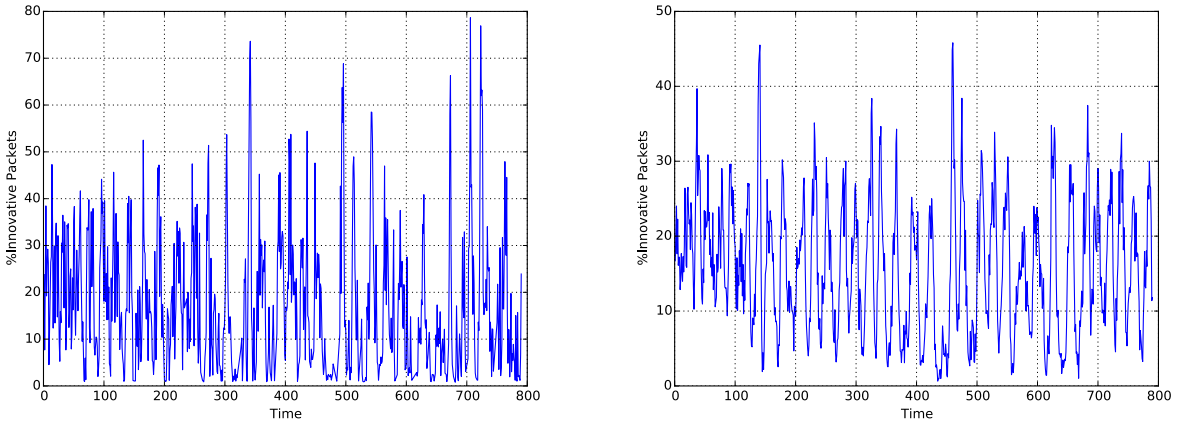


(a) High speed

(b) Low speed

Figure 7.5: Evolution of the rank in a mobile grid.

Figure 7.6 illustrates the rate of innovative payloads for high mobility scenario and low mobility scenario.



(a) High speed (b) Low speed

Figure 7.6: Rate of innovative packets in a mobile grid.

Results show that the rate of innovative packets is slightly sensible to the speed of nodes. This means also that packet losses have no impact on this value. This is because we are plotting the rate of innovative packets.

7.3.3 Military Scenario

Our goal here is to emulate a scenario of a military application. We consider a surveillance application where soldiers monitor a strategic area and anyone among them can alert the group in case of intrusion detection. In general, soldiers are in the border of the region. Hence, we consider a set of nodes arranged in a rectangle of size 600×500, nodes occupy the border of this region. We select one source node which generates alarms and broadcasts it in the whole network. The rank and the rate of innovative packets are illustrated in Figures 7.7 and 7.8.

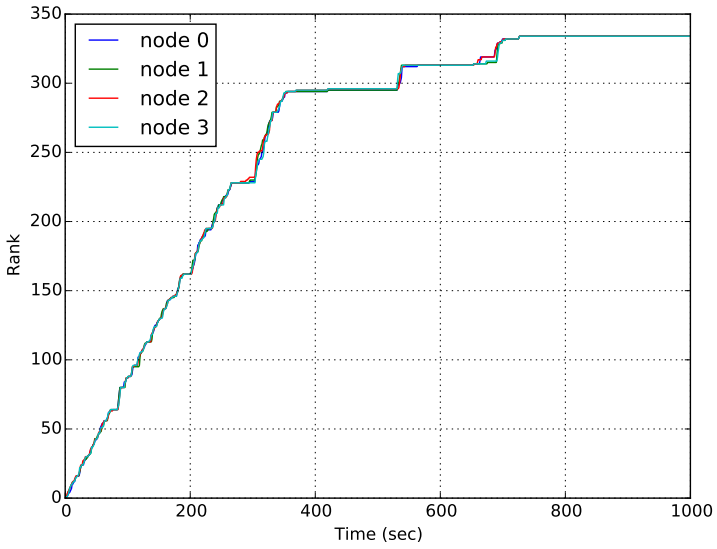


Figure 7.7: Evolution of the rank in a military scenario.

Like other scenarios, the rank at the selected nodes is almost the same. However, we observe some plateau in this plot. This is due to the fact that as arranged in the border of the region, the average number of neighbors per node decreases compared to the other scenarios. Hence, in average the node receives less packets. This is also confirmed by the rate of innovative packets in the network (Figure 7.8) which decreases compared to other scenarios.

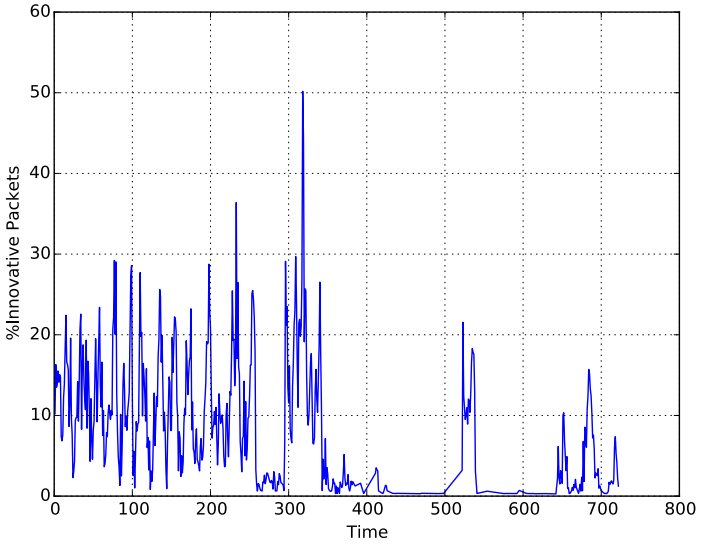


Figure 7.8: Rate of innovative packets in a military scenario.

7.4 Conclusion

Simulations provided for a proof of concept of the correctness of DragonNet implementation. Moreoever, the scenarios selected included larger deployments, typical area surveillance scenario, and mobility: it was show that network coding continues to perform extremely well in all cases, even in presence of mobility. The rank of nodes increases continuously illustrating proper evolution of the decoding in all nodes. The simulation results also make the case for future improvements on rate adaptation in DragonNet.

Chapter 8

Efficiency of Broadcast with Network Coding in Wireless Networks

8.1 Introduction

The goal of this chapter is to evaluate the performance of broadcast with network coding in wireless multi-hop networks regarding the capacity and the energy metrics. Wireless communication is modeled as a hyper-graph where the same transmission from one node achieves many of its neighbors and we analyse the case where the nodes are arranged on a torus grid. We evaluate the broadcast capacity of wireless network coding when all nodes have the same transmission rate, with the exception of the source. We prove also that in this case the network coding is "near optimal" in terms of energy efficiency. In this chapter, we detail thee results as well as the methodology to obtain them.

8.2 Problem and Background

8.2.1 Problem

In multi-hop wireless networks, a natural application of network coding is to reduce the number of transmissions required to transmit some amount of information to the destinations. This kind of application allows to achieve energy efficiency for networks where the cost of wireless communication is a critical design factor. We focus on one specific form of communication: broadcasting information from some sources to all the nodes in a wireless

multi-hop network. Such communication is commonly used in wireless networks, for instance, for management, information dissemination, multimedia content distribution, or as a simplified form of multicast.

The idea of energy efficient broadcast communication can be expressed as follows:

> • With some given broadcast sources, minimize the total number of (re)transmissions used to allow all nodes in the network to get the information.

This work seeks and provides some answers for the following question:

> • **Problem Statement**: How efficient is broadcast with network coding?

8.2.2 Related Work

The efficiency provided by network coding in multicast and broadcast networks has been studied for instance, by Lun et al. [17], and Wu et al. [19]. In particular, they provide methods for determining optimal network coding parameters for a given network with specific model assumptions. The work of Fragouli et al. [20] gives insights for all-to-all broadcast and illustrate how gains could be obtained compared to routing.

The work in [22] focused on homogeneous planar random networks, where the density and the area of the network would increase towards infinity. The authors have shown that, wireless network coding was asymptotically "optimal" for a strong definition of optimality, *transmission-level optimality*, where nearly every received packet would be innovative. This relied on the computation of the broadcast rate where nodes are organized on a grid, and on a specific *rate selection*, where most nodes would have the same rate. However, because the grid was not a torus, nodes near the edge of the network would have a smaller neighborhood, and would be in the bottleneck for the computation of the maximum broadcast rate. For this reason, the rate selection was modified to handle them specially.

Our methodology to answer this problem is explained in the Section 8.2.3.

8.2.3 Methodology and Summary of Results

We consider a torus grid where the transmission rate (coded payloads per second) is identical for all nodes except for the source. We investigate the maximum broadcast capacity which is the maximum rate (payload per second) at which the source can inject payloads, while ensuring that the receiver nodes can decode (with probability tending to 1). Our results are:

- We find this capacity in the considered topology to be equal to the number of neighbors of a node. In order to do this we translate the min-cut problem on a hypergraph in an equivalent problem of additive combinatorics.

- Moreover, network coding in such networks is "near optimal" in terms of energy efficiency, in the sense that each transmission will provide innovative information (outside the vicinity of the source).

- We analyse also the performance of network coding in torus networks with uniform loss rate and non-uniform loss rate: we find that in these two cases the maximum broadcast capacity is the expected number of neighbors that receive information.

8.3 Model

We study the problem of broadcasting information from one source to all nodes in the network. We consider multi-hop wireless networks with a certain number of nodes, without mobility. We also assume an ideal wireless model. More precisely, wireless transmissions are without losses, collisions or interferences. We assume that each node of the network is operating well below its maximum transmission capacity. Additionally, the network is a packet network with fixed packet size. We consider the torus grid as network topology.

A fundamental concept is represented by the idea of "neighbors". We give here a definition.

Definition 3 *We say that two nodes in the network are neighbors if their distance is less than a fixed radius that we denote by r (integer).*

We will use the following general notation:

- $\mathcal{V}$: set of nodes in the network;

- C_v: the retransmission rate of packets of a node v;

Some of the notation is more specifically targeted to a network of nodes organized on a lattice. Assume that $\mathcal{V}$ is included in a larger set $\hat{\mathcal{V}}$ (for a lattice, $\mathcal{V} \subset \hat{\mathcal{V}} = \mathbb{Z}^n$). We use the following notations for concepts related to neighborhood:

- $\mathcal{N}(X)$: open set of neighbors of $X \in \mathcal{V}$, $\mathcal{N}(X) \subset \hat{\mathcal{V}}$;

- $\mathcal{N}[X]$: closed set of neighbors of $X \in \mathcal{V}$, that is nodes and their neighbors $\mathcal{N}[X] \triangleq \mathcal{N}(X) \cup X$;
- R: the set of neighbors of the origin node;

8.4 Network Coding: Maximum Broadcast Rate

The maximum broadcast rate for the source represents the rate limit for the source which ensures that every destination in the network may decode. It is given by the minimum cut from the source to each particular destination in the network, where connectivity is described as an hypergraph [27].

An *hypergraph* is a graph where edges are replaced by hyper-arcs which are generalizations of arcs that may have more than one end node.

Let us consider the source s, and one of the multicast destinations $t \in \mathcal{V}$.

Definition 4 *An s-t* cut *is a partition of the set of nodes $\mathcal{V}$ in two sets S, T such that $s \in S$ and $t \in T$.*

Let $Q(s,t)$ be the set of such *s-t cuts*: $(S,T) \in Q(s,t)$.

We denote ΔS the set of nodes of S that are neighbors of at least one node of T:

$$\Delta_S \triangleq \{v \in S | \mathcal{N}(v) \cap T \neq \emptyset\}. \tag{8.1}$$

The *capacity of the cut* $(S,T) \in Q(s,t)$, denoted by $C(S)$, is defined as the maximum rate between the nodes in S and the nodes in T:

$$C(S) \triangleq \sum_{v \in \Delta_S} C_v. \tag{8.2}$$

In other terms, the idea is to cut the network into two parts, and check the total quantity of information transmitted from nodes in the part including the source, to nodes of the other part. The *min-cut* between s and t, that we denote by $C_{\min}(s,t)$, is the cut of $Q(s,t)$ with the minimum capacity:

$$C_{\min}(s,t) \triangleq \min_{(S,T) \in Q(s,t)} C(S). \tag{8.3}$$

When we consider the multicast case, there are several destinations t for the same source s, the min-cut is the minimum of the $s-t$ min-cut for all destinations t.

In the case of broadcast to all nodes, the min-cut is the minimum for all nodes different from s; we denote the broadcast capacity $C_{\min}(s)$,

$$C_{\min}(s) \triangleq \min_{t \in \mathcal{V} \setminus \{s\}} C_{\min}(s,t). \tag{8.4}$$

8.5 Broadcast Rate in Torus Grid

In order to compute the global min-cut $C_{\min}(s)$ in the considered topology, we consider a destination node t in the network. We link the capacity of the cut between the nodes of S and the nodes of T with the number of nodes of S who are neighbors of nodes of T, that can be written by a Minkowski sum. Moreover, we use tools from group theory in order to verify that there is no problem of neighborhood for the nodes that are at the border.

8.5.1 Minowski Sum and Neighborhood

The Minkowski addition is a classical way to express the neighborhood of an area, see [28] and the figure within.

Definition 5 *Given two subsets A and B of a group, the Minkowski sum $A \oplus B$ is defined as the set of all sums generated by all pairs of points in A and B, respectively:*

$$A \oplus B \triangleq \{a + b \ : \ a \in A, b \in B\}. \tag{8.5}$$

In the torus grid, the closed set of neighbors of one node t, that we denote $\mathcal{N}[t]$, can then be redefined in terms of the Minkowski sum as

$$\mathcal{N}[t] = \{t\} \oplus R. \tag{8.6}$$

This is indeed a translation of R, the set of neighbors of the node at $(0, 0)$ to the node t.

This extends to the neighborhood of subsets:

$$\mathcal{N}(A) = A \oplus R. \tag{8.7}$$

To see why, the expression (8.5) can be rewritten as:

$$A \oplus R = \bigcup_{a \in A} \{a\} \oplus R,$$

which corresponds to the union of the closed neighborhood of each node in A.

We consider the torus grid G, and we write the 2-dimensional torus grid as:

$$G = \frac{\mathbb{Z}}{n_X \mathbb{Z}} \times \frac{\mathbb{Z}}{n_Y \mathbb{Z}}, \tag{8.8}$$

where n_X and n_Y are the width and height of the grid and $\frac{\mathbb{Z}}{n\mathbb{Z}}$ is the set of integers modulo n. Since each $\frac{\mathbb{Z}}{n\mathbb{Z}}$ is a group, the Minkowski sum of subsets

of G is well-defined. The set R, neighborhood of the node $(0,0)$, can also be more precisely defined. Let $r > 0$ be the radio range, we define:

$$R \triangleq \{(x \bmod n_X, y \bmod n_Y) : (x,y) \in \mathbb{Z}^2 \text{ and } x^2 + y^2 \leq r^2\}. \quad (8.9)$$

We observe that R is symmetric with respect to the origin, that is $(x,y) \in R \Rightarrow (-x,-y) \in R$. In particular, the number of elements in R less 1 represents the number of neighbors of a node : $M = |R| - 1$. In Figure 8.1, we show an example of R with range $r = 3$ and the Minkowski sum with a set of nodes A.

We now introduce the essential notion of "large neighborhood" related to the fact that we have no problem of neighborhood for the nodes that are on the border of the grid.

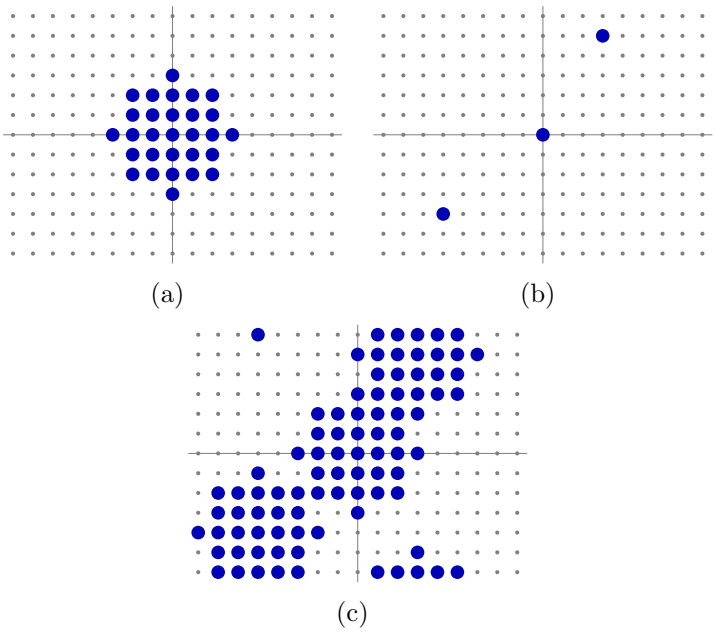


Figure 8.1: (a) R for $r = 3$; (b) Set of nodes A; (c) Minkowski sum $R \oplus A$.

Definition 6 *If for all subsets $A \subset G$ at least one of this conditions is verified:*

$$A \oplus R = G \quad (8.10)$$

$$\text{or} \quad |A \oplus R| \geq |A| + |R| - 1 \quad (8.11)$$

we say that R verifies the large neighborhood condition.

Lemma 1 *In the case R verifies the large neighborhood condition, let (S,T) be a partition of G and $\Delta_S = \mathcal{N}[T] \setminus T$, where $\mathcal{N}[T] \triangleq \mathcal{N}(T) \cup T$ is the "closed neighborhood" of nodes of T. If there exists $\sigma \in S$ such that $\sigma \notin \Delta_S$ then*

$$|\Delta_S| \geq |R| - 1.$$

Proof: Since Δ_S represents the set of nodes of S which are neighbor of at least a node of T and it can be rewritten as $\Delta_S = \mathcal{N}[T] \setminus T$, we have

$$\begin{aligned} |\Delta_S| &= |\mathcal{N}[T] \setminus T| \\ &\overset{(a)}{=} |\mathcal{N}[T]| - |T| \\ &\overset{(b)}{=} |T \oplus R| - |T| \end{aligned}$$

where (a) is coming from $T \subset \mathcal{N}[T]$ and (b) from $\mathcal{N}[T] = T \oplus R$. Now the hypothesis (large neighborhood condition) is that one of the conditions (8.10) or (8.11) is true. The condition (8.10) would imply that $N[T] = G$. This means that σ is neighbor of T, and since σ in never in T we have that $\sigma \in \Delta_S$. But we know that Δ_S does not include σ then this is a contradiction: (8.10) can never be verified. As a consequence (8.11) must be true: $|T \oplus R| \geq |T| + |R| - 1$, which implies

$$\begin{aligned} |\Delta_S| &\geq |T| + |R| - 1 - |T| \\ &\geq |R| - 1. \end{aligned} \tag{8.12}$$

■

8.5.2 Maximum Broadcast Rate

We focus on our main problem, computing the maximum broadcast rate of the source s.

Theorem 1 *We consider a network G which is represented by a torus grid with a neighborhood defined by the set R, and with the following rate selection:*

- *rate $C_v = 1$ for all nodes $v \neq s$,*
- *rate $C_s = M = |R| - 1$ for the source s.*

if R verifies the large neighborhood condition then the maximum broadcast rate of the source is $|R| - 1$.

Proof: Consider a fixed source s. In the previous section, we said that the maximum broadcast rate of the source is the min-cut $C_{\min}(s)$. We will assume that the source transmits at the maximum broadcast rate, that is $C_s = C_{\min}(s)$. Let us now consider any cut $(S,T) \in Q(s,t)$. The capacity of this cut is

$$C(S) \triangleq \sum_{v \in \Delta_S} C_v \quad \text{with} \quad \Delta_S \triangleq \{v \in S : \mathcal{N}(v) \cap T \neq \emptyset\}. \tag{8.13}$$

• Case (i): If $s \in \Delta_S$, then T includes at least one node which is neighbor of the source. Thus $C(S) \geq C_s$, and this cut never constraints the maximum broadcast rate since $C_s = |R| - 1$ and therefore $C(S) \geq |R| - 1$.

• Case (ii): Otherwise, Δ_S includes only nodes different from the source, hence with transmission rate 1. Therefore,

$$C(S) = \sum_{v \in \Delta_S} C_v = |\Delta_S|. \tag{8.14}$$

Therefore, applying Lemma 1 we have

$$C(S) \geq |R| - 1. \tag{8.15}$$

We observe that in the case of the particular cut $(S,T) = (\mathcal{V} \setminus \{t\}, t)$ we have $\Delta_S = \{u \in S | \mathcal{N}(u) \cap \{t\} \neq \emptyset\} = R \setminus \{t\}$, then

$$C(\mathcal{V} \setminus \{t\}) = |R| - 1.$$

■

8.5.3 The logic for Energy-Efficiency

In this section, we see why the previous results imply energy-efficiency in the network.

We have proved in the considered networks that the maximum broadcast rate of the source is equal to the number of its neighbors. If we consider a node which is not neighbor of the source, it will receive on average M coded packets per unit time. Our result implies that, on average, it receives in particular M "innovative" coded packets per unit time, where innovative are the packets that provide new informations. This means that on average each transmission will be innovative for each receiver. In other words, the transmission in these networks is efficace in terms of energy since we could not do better.

We underline that this is not true in general (see experiments in [29] for instance) but it is strictly linked to the network topology and its homogeneity. We have extended here the results presented in [22], where a modification of rate selection is needed since the network is not a torus and nodes near the border of the network would have a smaller neighborhood. Without this modification the network would be in the bottleneck for the computation of the maximum broadcast rate.

8.5.4 Inequalities for Sumsets

Our goal is to prove in our case sufficient conditions that appear in Theorem 1, and thus in Definition 6.

In the case of a torus, these relations are a difficult problem and closely linked to the number theory and additive combinatorics. To prove the conditions (8.10) and (8.11) we use the following result due to Kneser [30].

Proposition 1 (Kneser's Theorem) *Let G be a finite abelian group, A and B nonempty finite subsets:*

$$|A \oplus B| \geq |A \oplus H| + |B \oplus H| - |H| \tag{8.16}$$

where $H \triangleq \{h \in G : x + h \in A \oplus B,\ \forall x \in (A \oplus B)\}$ is a subgroup of G and it is called stabilizer.

In our case, if n_x and n_y are prime (equal or not), we prove the desired properties.

Theorem 2 *Let n_x and n_y be prime, A a nonempty finite subset of G, and R defined in (8.9). Then*

$$\begin{aligned} &|A \oplus R| \geq |A| + |R| - 1 \\ \text{or} \quad &A \oplus R = G. \end{aligned}$$

Proof: We consider Kneser's relation (8.16) with $B = R$:

$$|A \oplus R| \geq |A \oplus H| + |R \oplus H| - |H|. \tag{8.17}$$

H is a subgroup of G and we know that the subgroups of G are: $\{0\}$, G, $\{(x, 0) : x \in \frac{\mathbb{Z}}{n_Y\mathbb{Z}}\}$ and $\{(0, y) : x \in \frac{\mathbb{Z}}{n_X\mathbb{Z}}\}$.
• Case $H = \{\mathbf{0}\}$: We have

$$\begin{aligned} |A \oplus R| &\geq |A \oplus \{\mathbf{0}\}| + |R \oplus \{\mathbf{0}\}| - |\{\mathbf{0}\}| \\ &\geq |A| + |R| - 1. \end{aligned} \tag{8.18}$$

• Case $H = \{(x, 0) : x \in \frac{\mathbb{Z}}{n_Y \mathbb{Z}}\}$: We observe that

$$|A \oplus H| \geq |A|. \tag{8.19}$$

This is true since $\mathbf{0} \in H$ implies that $A \oplus \{\mathbf{0}\} \subset A \oplus H$ which gives $A \subset A \oplus H$.

We focus, now, our attention on $|R \oplus H|$. Since we are on a torus, the Minkowski sum of an horizontal line H and R is equal to a rectangle (see Figure 8.2), where the height is the diameter of R equal to $2r + 1$ and the width is the lenght of the line H. We consider the upper edge of the rectangle

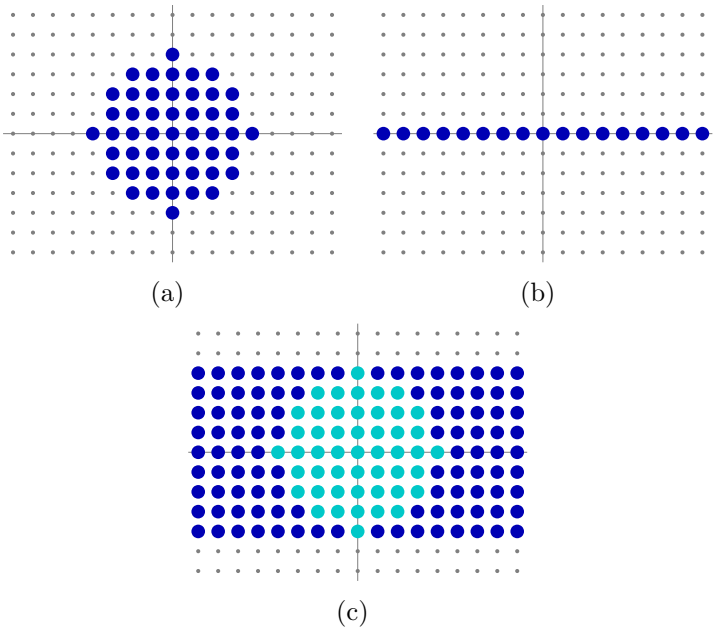


Figure 8.2: (a) R for $r = 4$; (b) $H = \{(x, 0) : x \in \frac{\mathbb{Z}}{n_Y \mathbb{Z}}\}$; (c) Minkowski sum of R and H.

which is given by the horizontal line denoted by

$$L^{'} = \{(x, y) \in \mathbb{Z}^2 : y = r\}$$

which has the same lenght of the line H : $|L^{'}| = |H|$. By definition of Minkowski sum and of R, this line passes by the unique point with coordinates $(0, r)$. We observe that

$$\begin{aligned} &(L^{'} \setminus \{(0, r)\}) \cap R = \emptyset \quad \text{and} \\ &(L^{'} \setminus \{(0, r)\}) \cup R \subseteq R \oplus H. \end{aligned} \tag{8.20}$$

This means that

$$\begin{aligned} |R \oplus H| &\geq |L^{'} \setminus \{(0,r)\} \cup R| \\ &= |L^{'}| - 1 + |R| - |L^{'} \setminus \{(0,r)\} \cap R| \\ &= |H| + |R| - 1. \end{aligned} \tag{8.21}$$

Then, if we consider equations (8.26), (8.19) and (8.21), we obtain

$$\begin{aligned} |A \oplus R| &\geq |A \oplus H| + |R \oplus H| - |H| \geq \\ &\geq |A| + |R| + |H| - 1 - |H| \\ &= |A| + |R| - 1. \end{aligned}$$

• Case $H = \{(0,y) : x \in \frac{\mathbb{Z}}{n_x\mathbb{Z}}\}$: It is similar to the case $H = \{(x,0) : x \in \frac{\mathbb{Z}}{n_Y\mathbb{Z}}\}$.
• Case $H = G$: If the stabilizer is G by definition we have that

$$(A \oplus R) \oplus G = A \oplus R.$$

This implies that

$$|A \oplus R| = |(A \oplus R) \oplus G| \geq |G|$$

since $G \subset (A \oplus R) \oplus G$. Therefore, $A \oplus B = G$. ■

We observe that Theorem 2 allows us to prove Theorem 1 in the case the neighborhood R is a discretized circle given in (8.9). What happens if R is a general subset of integer points? In the following we give some sufficient conditions such that the inequalities (8.10) and (8.11) hold for a general neighborhood.

Definition 7 *A set $B \subset G$ is connected if for all u and v in B there exists a path from u to v in B such that any two consecutive points in the path differ by at most one in each coordinate.*

Theorem 3 *Let n_x and n_y be prime, A a nonempty finite subset of G, and B a subset of G. Let n_1^z and n_2^z be positive integers such that $n_z = n_1^z + n_2^z + d_{max}$ where $z \in \{x,y\}$ and $d_{max} = max\{|z_u - z_v|, \forall u,v \in B\}$. Then*

$$|A \oplus B| \geq |A| + |B| - 1 \tag{8.22}$$

$$\text{or} \quad A \oplus B = G \tag{8.23}$$

is true in the following cases:

1. *In the case B is connected, (8.22) or (8.23) is true if*

$$d_{max} \leq (n_1 + n_2)(h-1) + 1 \tag{8.24}$$

where $h = max\{|z_u^c - z_v^c|, \forall u, v \in B\}$ with $z^c = y$ (respectively x) if $z = x$ (respectively y).

2. *In the case B is disconnected, (8.22) or (8.23) is true if*

$$d_{max} \leq (n_1 + n_2)(\hat{h} - 1) + 1 \tag{8.25}$$

where $\hat{h}$ is the number of rows (columns) of B with at least one element.

Proof: We consider Kneser's relation (8.16)

$$|A \oplus B| \geq |A \oplus H| + |B \oplus H| - |H|. \tag{8.26}$$

H is a subgroup of G and we know that the subgroups of G are: $\{0\}$, G, $\{(x,0) : x \in \frac{\mathbb{Z}}{n_Y\mathbb{Z}}\}$ and $\{(0,y) : y \in \frac{\mathbb{Z}}{n_X\mathbb{Z}}\}$.
The cases $H = \{0\}$ and $H = G$ are similar to the ones discussed in the previous theorem. In particular,
• Case $H = \{\mathbf{0}\}$: We have

$$|A \oplus B| \geq |A| + |B| - 1.$$

• Case $H = G$: We have

$$|A \oplus B| = |(A \oplus B) \oplus G| \geq |G|.$$

• Case $H = \{(x,0) : x \in \frac{\mathbb{Z}}{n_Y\mathbb{Z}}\}$: We suppose B connected and we denote $n_1^x = n_1$, $n_2^x = n_2$ and we have $d_{max} = max\{|x_u - x_v|, \forall u, v \in B\}$. The Minkowski sum of B and H is equal to a rectangle where the height is h and the width is the lenght of the line H which is n_x, see Figure 8.3.

Since $n_x = n_1 + n_2 + d_{max}$, the rectangle can be written as disjoint union of three smaller rectangles S_1, S_2 ans S_3, where S_1 has $n_1 h$ elements, S_2 has $d_{max}h$ elements and it includes B and S_3 with $n_2 h$ elements. So,

$$B \oplus H = S_1 \cup S_2 \cup S_3. \tag{8.27}$$

Therefore, we have

$$\begin{aligned} |B \oplus H| &= |S_1 \cup S_2 \cup S_3| \\ &= |S_1| + |S_2| + |S_3| \\ &\geq n_1 h + |B| + n_2 h \\ &= |B| + n_1 + n_2 + n_1(h-1) + n_2(h-1) \\ &\overset{(a)}{\geq} |B| + n_1 + n_2 + d_{max} - 1 \\ &= |B| + |H| - 1, \end{aligned} \tag{8.28}$$

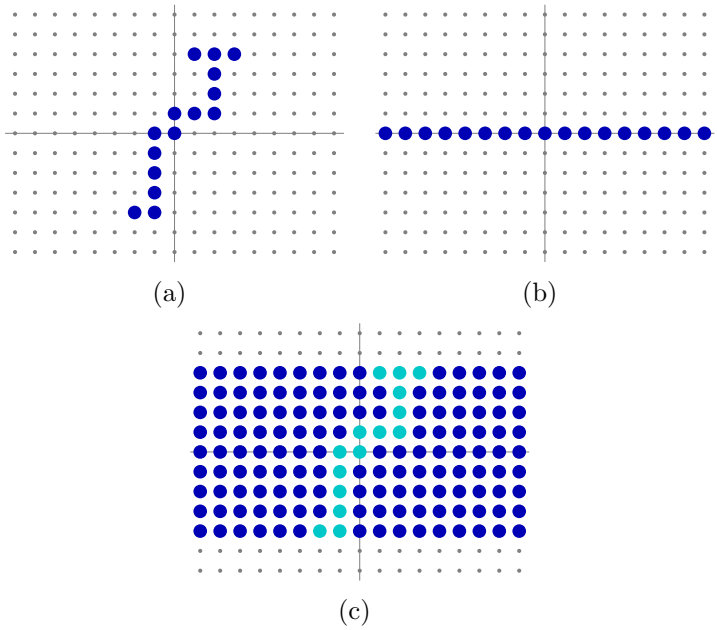


Figure 8.3: (a) B connected; (b) $H = \{(x, 0) : x \in \frac{\mathbb{Z}}{n_Y\mathbb{Z}}\}$; (c) Minkowski sum of B and H.

where (a) comes from applying hypothesis $d_{max} \leq (n_1 + n_2)(h - 1) + 1$.

We consider now B a disconnected subset of G. We have by hypothesis that $\hat{h}$ is the number of rows of B with at least one element. In this case, the Minkowski sum of B and H is equivalent to a rectagle with height $\hat{h}$ and width n_x, see Figure 8.4.

Then, as in the previous case, we can write the rectangle as disjoint union of three rectangles and, in partcular, we have

$$\begin{aligned} |B \oplus H| &= |S_1 \cup S_2 \cup S_3| \\ &= |S_1| + |S_2| + |S_3| \\ &\geq n_1\hat{h} + |B| + n_2\hat{h} \\ &= |B| + n_1 + n_2 + n_1(\hat{h} - 1) + n_2(\hat{h} - 1) \\ &\overset{(b)}{\geq} |B| + n_1 + n_2 + d_{max} - 1 \\ &= |B| + |H| - 1, \end{aligned} \tag{8.29}$$

where (b) comes from applying hypothesis (8.25).
• Case $H = \{(0, y) : x \in \frac{\mathbb{Z}}{n_x\mathbb{Z}}\}$: It is similar to the case $H = \{(x, 0) : x \in \frac{\mathbb{Z}}{n_Y\mathbb{Z}}\}$. ■

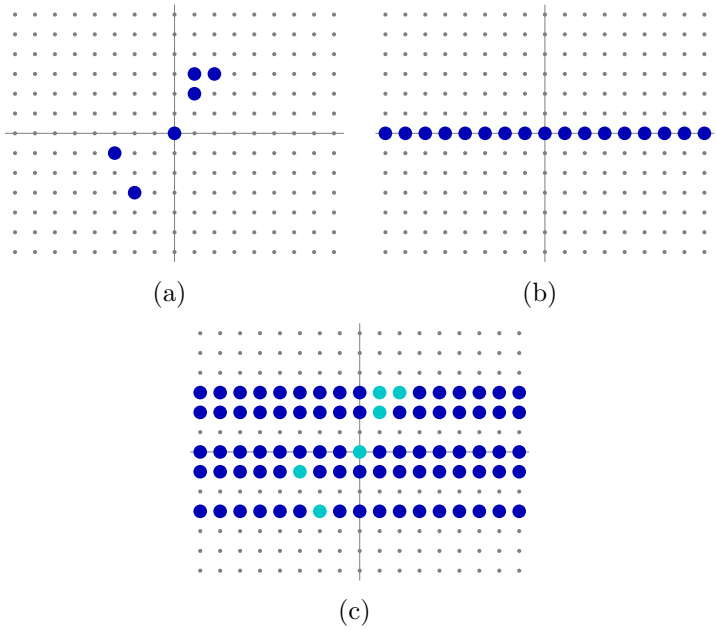


Figure 8.4: (a) B disconnected; (b) $H = \{(x, 0) : x \in \frac{\mathbb{Z}}{n_Y \mathbb{Z}}\}$; (c) Minkowski sum of B and H.

We observe that hypothesis (8.24) and (8.25) allow us to exclude the case where the neighborhood is too large, which represents a situation not interesting from the point of view of network coding. Moreover, these conditions imply that h and $\hat{h}$ are > 1, which means that there is at least one node different from the source in the network.

8.6 Performance of NC in Lossy Wireless Networks

In order to study the performance of network coding in lossy wireless networks, we consider the following model, [31]:

- For a node u, we consider that there exists an hyperarc to any set of nodes K, with $K \subset \mathcal{V} \setminus \{u\}$.
- The transmission rate for hyperarc (K, u) is given by $C_u \pi_u(K)$ where C_u is the transmission rate of the node u and $\pi_u(K)$ is the probability that at least one node in K receives from u.

Then, the capacity of the cut is the quantity:

$$\begin{aligned} C(S) &\triangleq \sum_{u\in\Delta_S}\sum_{K\not\subset S} C_u \pi_u(K) \\ &= \sum_{u\in\Delta_S} C_u \sum_{K\not\subset S} \pi_u(K) \end{aligned} \tag{8.30}$$

We observe that $K \not\subset S$ means that we are considering subsets $K|K \subset \mathcal{V}\backslash\{u\}$ and $K \cap T \neq \emptyset$.

We consider the event

$$\xi = \{\text{at least one node in } T \text{ receives from } u\} \tag{8.31}$$

and we consider the exhaustive set $\{U_1, \ldots, U_n\}$ made of all subsets of $\mathcal{V}\backslash\{u\}$ such that $U_i \cap T \neq \emptyset$ for all $i = 1, \ldots, n$. Then we can write the event ξ as follows

$$\begin{aligned} \xi &\{\text{at least one node in } U_1 \text{ receives from } u\} \text{or } \{\text{at least one node in } U_2 \text{ receives from } u\} \ \ldots \\ &\ldots \text{or } \{\text{at least one node in } U_n \text{ receives from } u\} \\ &= \{K = U_1\} \text{ or } \{K = U_2\} \text{ or } \ldots \text{or } \{K = U_n\}. \end{aligned} \tag{8.32}$$

The probability of the event ξ can then be expressed as follows:

$$\begin{aligned} Pr\{\xi\} &= \sum_{U_i\not\subset S} Pr\{K = U_i\} \\ &= \sum_{U_i\not\subset S} Pr\{\text{at least one node in } U_i \text{ receives from } u\} \\ &= \sum_{U_i\not\subset S} \pi_u(U_i). \end{aligned} \tag{8.33}$$

This means that we can write the capacity of the cut

$$\begin{aligned} C(S) &= \sum_{u\in\Delta_S} C_u \sum_{K\not\subset S} \pi_u(K) \\ &= \sum_{u\in\Delta_S} C_u Pr\{\xi\} \\ &= \sum_{u\in\Delta_S} C_u Pr\{\text{at least one node in } T \text{ receives from } u\}. \end{aligned} \tag{8.34}$$

8.6.1 Uniform loss rate

We consider uniform loss rate in the network.

Theorem 4 *We consider a network G which is represented by a torus grid with a neighborhood defined by the set R in (8.9) and we denote by q the probability that a node receives a packet from its neighbor. We consider the following rate selection:*

- *rate $C_v = 1$ for all nodes $v \neq s$,*
- *rate $C_s = C_{max,s}$ (defined in Theorem 5 later) for the source s.*

If R verifies the large neighborhood condition then the maximum broadcast rate of the source is $(|R| - 1)q$.

Before proving Theorem 4, we establish intermediary results: Lemma 2 and Theorem 5.

Lemma 2 *Under conditions of Theorem 4 and the case we are near the source ($s \in \Delta_S$), we have*

$$C(S) \geq C_s(1 - (1-q)^{\ell}) + (|R| - \ell - 1)q. \tag{8.35}$$

Proof: We assume now that we are near the source: $s \in \Delta_S$. Moreover, we consider Δ_{TS} the set of neighbors of s which are not in Δ_S and whose cardinality is denoted by ℓ and $\ell \geq 1$ because $s \in \Delta_S$. We can have two cases: all the neighbors of the source are included in $\Delta_S \cup T$ or some of the neighbors of the source are not included in $\Delta_S \cup T$.
First, we analyze the case $\mathcal{N}(s) \subset \Delta_S \cup T$, see Figure 8.5.

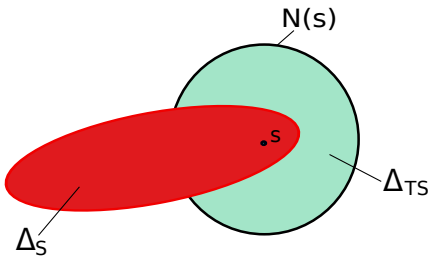


Figure 8.5: $\mathcal{N}(s) \subset \Delta_S \cup T$

We observe that

$$\mathcal{N}(s) = (\mathcal{N}(s) \backslash \Delta_S) \cup (\mathcal{N}(s) \cap \Delta_S) = \Delta_{TS} \cup (\mathcal{N}(s) \cap \Delta_S) \text{ and } \Delta_{TS} \cap (\mathcal{N}(s) \cap \Delta_S) = \emptyset$$

and since $|\mathcal{N}(s)| = |R|$ we have

$$|R| = |\Delta_{TS}| + |\mathcal{N}(s) \cap \Delta_S|.$$

Therefore,

$$|\Delta_S| \geq |\mathcal{N}(s) \cap \Delta_S| = |R| - |\Delta_{TS}| = |R| - \ell. \tag{8.36}$$

Then from (8.34)

$$\begin{aligned} C(S) &\geq C_s(1-(1-q)^\ell) + (|\Delta_S| - 1)q \\ &= C_s(1-(1-q)^\ell) + (|R| - \ell - 1)q \end{aligned} \tag{8.37}$$

where $(1-(1-q)^\ell)$ represents the probability that at least one node in Δ_{TS} receives from s.

In the case $\mathcal{N}(s) \not\subset \Delta_S \cup T$, see Figure 8.6: the fact that $\mathcal{N}(s) \not\subset \Delta_S \cup T$ corresponds exactly to conditions of Lemma 1, then applying Lemma 1 we have $|\Delta_S| \geq |R| - 1$. Therefore, we have

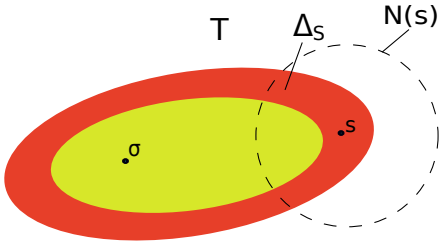


Figure 8.6: $\mathcal{N}(s) \not\subset \Delta_S \cup T$

$$\begin{aligned} C(S) &\geq C_s(1-(1-q)^\ell) + (|\Delta_S| - 1)q \\ &\geq C_s(1-(1-q)^\ell) + (|R| - 2)q \\ &\geq C_s(1-(1-q)^\ell) + (|R| - \ell - 1)q \end{aligned} \tag{8.38}$$

since $\ell \geq 1$.

This means that in any case we have

$$C(S) \geq C_s(1-(1-q)^\ell) + (|R| - \ell - 1)q. \tag{8.39}$$

■

We observe in the case near the source: $C(S) \geq C_s q$ with $q \geq 0$, by setting C_s to a sufficintly high value, we can get $C(S)$ arbitrarly high. Hence, the capacity of this cut is not limiting provided that C_s is set appropriety. In the next theorem, we give an appropriate value for C_s, $C_{max,s}$. It verifies: $\forall \ell \geq 1, \quad C_s \geq \frac{\ell q}{1-(1-q)^\ell}$.

Theorem 5 *Under conditions of Lemma 2, let*

$$C_{max,s} = \max_{\ell \in \{1,\ldots,|R|-1\}} \frac{\ell q}{1-(1-q)^{\ell}}$$

then

$$C_{max,s} = \frac{(|R|-1)q}{1-(1-q)^{|R|-1}}.$$

Proof: We first prove that $f(\ell) = \frac{\ell q}{1-(1-q)^{\ell}}$ is an increasing function in the variable ℓ: we compute

$$\begin{aligned} \frac{d}{d\ell} f(\ell) &= \frac{q}{1-(1-q)^{\ell}} + \frac{q\ell(1-q)^{\ell}\log(1-q)}{(1-(1-q)^{\ell})^2} \\ &= \frac{q}{(1-(1-q)^{\ell})^2}\left[1-(1-q)^{\ell}) + \ell(1-q)^{\ell})\log(1-q)\right]. \quad (8.40) \end{aligned}$$

We focus our attention on $1-(1-q)^{\ell}+\ell(1-q)^{\ell}\log(1-q)$ that we denote by $g(\ell)$: we compute

$$\frac{d}{d\ell} g(\ell) = \ell(1-q)^{\ell}(\log(1-q))^2 \geq 0. \quad (8.41)$$

This implies that $g(\ell)$ is increasing for $\ell \geq 0$ and we observe that

$$g(1) = q + \log(1-q)(1-q) \geq 0 \quad \forall q \in [0,1[\,, \quad (8.42)$$

thus $g(\ell) \geq 1$ for $\ell \geq 1$. This means that $\frac{d}{d\ell} f(\ell) \geq 0$ and then $\frac{\ell q}{1-(1-q)^{\ell}}$ is an increasing function in the variable ℓ.

Since $\frac{\ell q}{1-(1-q)^{\ell}}$ is an increasing function in the variable ℓ, the maximum is achieved for $\ell = |R|-1$. We have

$$\begin{aligned} C_{max,s} &= \max_{\ell \in \{1,\ldots,|R|-1\}} \frac{\ell q}{1-(1-q)^{\ell}} \\ &= \max_{\ell \in \{1,\ldots,|R|-1\}} f(\ell) \\ &= f(|R|-1) \\ &= \frac{(|R|-1)q}{1-(1-q)^{|R|-1}} \geq 1 \quad (8.43) \end{aligned}$$

since $f(\ell)$ is increasing and $f(1) = 1$.

■

Proof of Theorem 4: In the case we are near the source, $s \in \Delta_S$, from Lemma 2 and Theorem 5 we have for $\ell \in \{1, \ldots, |R|-1\}$

$$\begin{aligned} C(S) &\geq C_s(1-(1-q)^\ell) + (|R|-1-\ell)q \\ &\overset{(a)}{\geq} \frac{\ell q}{1-(1-q)^\ell}(1-(1-q)^\ell) + (|R|-1-\ell)q \\ &= \ell q + (|R|-1-\ell)q \\ &= (|R|-1)q \end{aligned} \tag{8.44}$$

where (a) comes from the fact that $C_s = C_{max,s} = \max_{\ell \in \{1,\ldots,|R|-1\}} \frac{\ell q}{1-(1-q)^\ell}$.

We analyze the case of nodes far from the source : $s \notin \Delta_S$.

We observe that given $u \in \Delta_s$ there exists a node $v_u \in T$ such that v_u is neighbor of u. This implies that

$$Pr\{\text{at least one node in } T \text{ receives from } u\} \geq Pr\{v_u \text{ receives from } u\} = q \tag{8.45}$$

Therefore, the capacity of the cut

$$C(S) \geq \sum_{u \in \Delta_S} C_u q = |\Delta_S| q. \tag{8.46}$$

Using Lemma 1, we have

$$|\Delta_S| \geq |R| - 1, \tag{8.47}$$

therefore,

$$C(S) \geq |\Delta_S| q \geq (|R|-1)q. \tag{8.48}$$

We observe that the capacity of the particular cut $(\mathcal{V} \setminus \{t\}, t)$ is given by

$$C(\mathcal{V} \setminus \{t\}) = (|R|-1)q \tag{8.49}$$

since in this case $\Delta_S = \Delta_{\mathcal{V}\setminus\{t\}} = R \setminus \{t\}$ and $|R|-1$ is the number of neighbors of t.

■

We observe that the maximum broadcast rate $(|R|-1)q$ corresponds to the expected number of neighbors that receive information.

8.6.2 Non-Uniform loss rate

First, we define $R_1, \ldots, R_k$ the neighborhoods of node u such that $R_1 \supset R_2 \supset \cdots \supset R_k$ and R_i does not contains 0 for $i = 1, \ldots, k$ with $k \geq 2$. We denote

q_i $(i = 1, \ldots, k)$ the probability that a node in R_i $(i = 1, \ldots, k)$ receives a packet from u and we have $q_{i+1} \geq q_i$. As before we consider a cut (S, T) and we associate to each R_i a subset $\Delta_i = (T \oplus (R_i \cup \{0\})) \setminus T$ such that $\Delta_1 \supset \Delta_2 \supset \cdots \supset \Delta_k$ and, denoting r_i and d_i the cardinalities of R_i and Δ_i respectively. In this case,

$$C(S) \stackrel{def}{=} \sum_{i=1}^{k} \sum_{u \in \Delta_i \setminus \Delta_{i+1}} C_u Pr\{\text{at least one node in } T \text{ receives from } u\} \text{ with } \Delta_{k+1} = \emptyset$$

$$= \sum_{i=1}^{k-1} q_i(|\Delta_i| - |\Delta_{i+1}|) + q_k|\Delta_k| \tag{8.50}$$

$$= \sum_{i=1}^{k-1} q_i(d_i - d_{i+1}) + q_k d_k. \tag{8.51}$$

Lemma 3 *In the case $R_i \cup \{0\}$ verifies the lage neighborhood condition, if there exists $\sigma \in S$ such that $\sigma \notin \Delta_i$ for $i = 1, \ldots, k$ then*

$$d_i \geq r_i \quad \forall\ 1 \leq i \leq k.$$

Proof: We have $\Delta_i = (T \oplus (R_i \cup \{0\})) \setminus T$ where the nodes in $T \oplus R_i \cup \{0\}$ are neighbors of T. In the hypothesis, $R_i \cup \{0\}$ verifies the condition of large neighborhood, then one of the conditions (8.10) or (8.11) is true. The condition (8.10), implies that $T \oplus (R_i \cup \{0\}) = G$. This means that σ is neighbor of T, but we know that it does not belong to T, then $\sigma \in \Delta_i$. But in the hypothesis $s \notin \Delta_i$, then T can never verify (8.10). As a consequence (8.11) must be true: $|T \oplus (R_i \cup \{0\})| \geq |T| + |(R_i \cup \{0\})| - 1$. We observe that

$$|T \oplus (R_i \cup \{0\})| \geq |T| + |R_i \cup \{0\}| - 1 = |T| + |R_i| + 1 - 1 = |T| + |R_i|.$$

Therefore,

$$|\Delta_i| = |T \oplus (R_i \cup \{0\})) \setminus T| \geq |T| + |R_i| - |T| = |R_i|,$$

which means $d_i \geq r_i$. ■

Theorem 6 *We consider a network G which is represented by a torus grid with neighborhoods defined by R_i $(i = 1, \ldots, k)$ and we denote by q_i the probability that a node receives a packet from its neighbor in R_i. We consider the following rate selection:*

- *rate $C_v = 1$ for all nodes $v \neq s$,*

- *rate* $C_s = C_{max,s} = \max_{\ell_1 \in \{1,\ldots,r_1-1\},\ldots,\ell_k \in \{1,\ldots,r_k-1\}} \frac{\sum_{i=1}^{k} \ell_i q_i}{1-\Pi_{i=1}^{k}(1-q_i)^{\ell_i}}$ *for the source* s.

If every $R_i \cup \{0\}$ $(i = 1, \ldots, k)$ *verifies the large neighborhood condition then the maximum broadcast rate of the source is given by* $\sum_{i=1}^{k-1} q_i(r_i - r_{i+1}) + q_k r_k$.

Lemma 4 *Under conditions of Theorem 6 and the case we are near the source* $(s \in \Delta_i)$*, we have*

$$C(S) \geq C_s(1-(1-q_1)^{\ell_1} \ldots (1-q_k)^{\ell_k}) + (r_1 - r_2 - \ell_1)q_1 + (r_2 - r_3 - \ell_2)q_2 + \ldots (r_k - \ell_k)q_k. \tag{8.52}$$

Proof: We consider Δ_{TS_i} the set of neighbors of s which are not in Δ_i and whose cardinality is denoted by ℓ_i and $\ell_i \geq 1$ because $s \in \Delta_i$. We can have two cases: all the neighbors of the source are included in $\Delta_i \cup T$ or some of the neighbors of the source are not included in $\Delta_i \cup T$.
First, we analyze the case $\mathcal{N}_i(s) \subset \Delta_i \cup T$. In Figure 8.7, we can see the case $k = 2$.

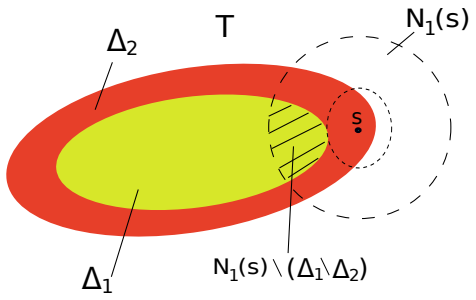


Figure 8.7: $\mathcal{N}_1(s) \subset \Delta_1 \cup T$

We observe that

$$\mathcal{N}_i(s) = (\mathcal{N}_i(s) \backslash (\Delta_i \backslash \Delta_{i+1})) \cup (\mathcal{N}_i(s) \cap (\Delta_i \backslash \Delta_{i+1})) = \Delta_{TS_i} \cup (\mathcal{N}_i(s) \cap (\Delta_i \backslash \Delta_{i+1}))$$

and

$$\Delta_{TS_i} \cap (\mathcal{N}_i(s) \cap (\Delta_i \setminus \Delta_{i+1})) = \emptyset$$

and since $|\mathcal{N}_i(s)| = |(R_i \setminus R_{i+1}) \cup \{0\}|$ we have

$$|(R_i \setminus R_{i+1}) \cup \{0\}| = |\Delta_{TS_i}| + |\mathcal{N}_i(s) \setminus (\Delta_i \setminus \Delta_{i+1})|.$$

Therefore,

$$|\Delta_i \backslash \Delta_{i+1}| \geq |\mathcal{N}_i(s) \backslash (\Delta_i \backslash \Delta_{i+1})| = |(R_i \backslash R_{i+1}) \cup \{0\}| - |\Delta_{TS_i}| = r_i - r_{i+1} + 1 - \ell_i. \tag{8.53}$$

Then

$$\begin{aligned} C(S) &\geq C_s(1-(1-q_1)^{\ell_1}\dots(1-q_k)^{\ell_k}) + (|\Delta_1 \setminus \Delta_2| - 1)q_1 + \dots (|\Delta_k| - 1)q_k \\ &= C_s(1-(1-q_1)^{\ell_1}\dots(1-q_k)^{\ell_k}) + (r_1 - r_2 + 1 - \ell_1 - 1) + \dots + (r_k - \ell_k) \\ &= C_s(1-(1-q_1)^{\ell_1}\dots(1-q_k)^{\ell_k}) + (r_1 - r_2 - \ell_1) + \dots + (r_k - \ell_k) \end{aligned} \quad (8.54)$$

where $(1-(1-q_1)^{\ell_1}\dots(1-q_k)^{\ell_k})$ represents the probability that at least one node in Δ_{TS_i} receives from s.

In the case $\mathcal{N}_i(s) \not\subset \Delta_i \cup T$, see Figure 8.8: it represents the conditions of Lemma 3, then applying Lemma 3 we have $|\Delta_i| \geq |R_i|$. Therefore, we have

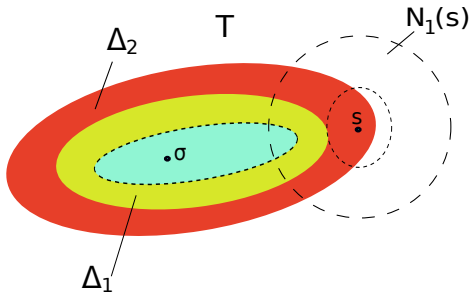


Figure 8.8: $\mathcal{N}_1(s) \not\subset \Delta_1 \cup T$

$$\begin{aligned} C(S) &\geq C_s(1-(1-q_1)^{\ell_1}\dots(1-q_k)^{\ell_k}) + (|\Delta_1 \setminus \Delta_2| - 1)q_1 + \dots (|\Delta_k| - 1)q_k \\ &\geq C_s(1-(1-q_1)^{\ell_1}\dots(1-q_k)^{\ell_k}) + (r_1 - r_2 - \ell_1) + \dots + (r_k - \ell_k) \end{aligned} \quad (8.55)$$

since $\ell_i \geq 1$.

This means that in any case we have

$$C(S) \geq C_s(1-(1-q_1)^{\ell_1}\dots(1-q_k)^{\ell_k}) + (r_1 - r_2 - \ell_1) + \dots + (r_k - \ell_k) \quad (8.56)$$

■

Lemma 5 *Defining*

$$C_j(S) \stackrel{def}{=} \sum_{i=1}^{j-1} q_i(d_i - d_{i+1}) + q_j d_j \quad \textit{and} \quad C_j^b(S) \stackrel{def}{=} \sum_{i=1}^{j-1} q_i(r_i - r_{i+1}) + q_j r_j \quad (8.57)$$

we have

$$C_{j+1}(S) = C_j(S) + d_{j+1}(q_{j+1} - q_j) \quad \textit{and} \quad C_{j+1}^b(S) = C_j^b(S) + r_{j+1}(q_{j+1} - q_j) \quad (8.58)$$

for all $j \geq 1$.

Proof: We have

$$\begin{aligned} C_{j+1}(S) &= \sum_{i=1}^{j} q_i(d_i - d_{i+1}) + q_{j+1}d_{j+1} \\ &= \sum_{i=1}^{j-1} q_i(d_i - d_{i+1}) + q_j(d_j - d_{j+1}) + q_{j+1}d_{j+1} \\ &= C_j(S) - q_j d_{j+1} + q_{j+1}d_{j+1} \\ &= C_j(S) + d_{j+1}(q_{j+1} - q_j). \end{aligned}$$

Similarly, we get

$$C^b_{j+1}(S) = C^b_j(S) + r_{j+1}(q_{j+1} - q_j) \tag{8.59}$$

■

We give in the following the proof of Theorem 6.

Proof of Theorem 6: In the case we are near the source, $s \in \Delta_i$, from Lemma 4 we have

$$\begin{aligned} C(S) &\geq C_s(1 - (1-q_1)^{\ell_1} \dots (1-q_k)^{\ell_k}) + (r_1 - r_2 - \ell_1)q_1 + \dots + (r_k - \ell_k)q_k \\ &\overset{(a)}{\geq} \frac{\sum_{i=1}^k \ell_i q_i}{1 - \Pi_{i=1}^k (1-q_i)^{\ell_i}} (1 - (1-q_1)^{\ell_1} \dots (1-q_k)^{\ell_k}) + (r_1 - r_2 - \ell_1)q_1 + \dots + (r_k - \ell_k)q_k \\ &= \frac{\sum_{i=1}^k \ell_i q_i}{1 - \Pi_{i=1}^k (1-q_i)^{\ell_i}} (1 - (1-q_1)^{\ell_1} \dots (1-q_k)^{\ell_k}) + \sum_{i=1}^{k-1}(r_i - r_{i+1})q_i - \sum_{i=1}^{k} \ell_i q_i + q_k r_k \\ &\geq \sum_{i=1}^{k-1}(r_i - r_{i+1})q_i + r_k q_k \end{aligned} \tag{8.60}$$

where (a) comes from the fact that $C_s = C_{max,s} = \max_{\ell_1 \in \{1,\dots,r_1-1\},\dots,\ell_k \in \{1,\dots,r_k-1\}} \frac{\sum_{i=1}^k \ell_i q_i}{1-\Pi_{i=1}^k (1-q_i)^{\ell_i}}$.

We assume to be far from the source: $s \notin \Delta_i$ for all $i = 1, \dots, k$.

We have proved the case $k = 1$ in uniform case. We consider $k = 2$ and we have:

$$\begin{aligned} C(S) &= q_1(|\Delta_1| - |\Delta_2|) + q_2|\Delta_2| \\ &= q_1(d_1 - d_2) + q_2 d_2 \\ &= q_2 r_2 + q_2(d_2 - r_2) + q_1(d_1 - d_2) \\ &\geq q_2 r_2 + q_1(d_2 - r_2) + q_1(r_1 - d_2) \\ &= q_2 r_2 + q_1(r_1 - r_2) \end{aligned} \tag{8.61}$$

where the inequality comes from applying $q_{i+1} \geq q_i$ and $d_i \geq r_i$ for all $i \in \{1, \ldots, k\}$.

Assuming that

$$C_j(S) \geq C_j^b(S) \tag{8.62}$$

and using Lemma 5 we have

$$\begin{aligned} C_{j+1}(S) - C_{j+1}^b(S) &= C_j(S) + d_{j+1}(q_{j+1} - q_j) - C_j^b(S) - r_{j+1}(q_{j+1} - q_j) \\ &= (d_{j+1} - r_{j+1})(q_{j+1} - q_j) + C_j(S) - C_j^b(S) \\ &\geq (d_{j+1} - r_{j+1})(q_{j+1} \quad \text{beacause of assumption} \\ &\geq 0 \end{aligned} \tag{8.63}$$

We observe that in the particular case of the cut $(\mathcal{V} \setminus \{t\}, t)$, we have that $\Delta_i = (\{t\} \oplus (R_i \cup \{0\})) \setminus \{t\} = R_i$ for $i = 1, \ldots, k$ and

$$\begin{aligned} C(\mathcal{V} \setminus \{t\}) &= \sum_{i=1}^{k} \sum_{u \in \Delta_i} C_u Pr\{\text{at least one node in } T \text{ receives from } u\} \\ &= \sum_{i=1}^{k-1} q_i(r_i - r_{i+1}) + q_k r_k = C_k^b(S) \end{aligned} \tag{8.64}$$

since T corresponds to the destination t. ■

Remark 1 *In the case $\Delta_1 = ... = \Delta_{m-1} = \emptyset$, we denote $\Delta_1^* = \Delta_m, \ldots, \Delta_{k-m+1}^* = \Delta_k$ and $q_1^* = q_m, \ldots, q_{k-m+1}^* = q_k$ We have*

$$\begin{aligned} C_j^*(S) &\stackrel{def}{=} \sum_{i=1}^{j-1} q_i^*(d_i^* - d_{i+1}^*) + q_j^* d_j^* \\ &= \sum_{i=1}^{j-1} q_{i+m-1}(d_{i+m-1} - d_{i+m}) + q_{j+m-1} d_{j+m-1} \end{aligned} \tag{8.65}$$

We set $\ell = i + m - 1$, then $i = \ell - m + 1$, and

$$C_j^*(S) = \sum_{\ell=m}^{j+m-2} q_\ell(d_\ell - d_{\ell+1}) + q_{j+m-1} d_{j+m-1}. \tag{8.66}$$

In particular,

$$C_{k-m+1}^*(S) = \sum_{\ell=m}^{k-1} q_\ell(d_\ell - d_{\ell+1}) + q_k d_k. \tag{8.67}$$

and we have

$$C^*_{k-m+1}(S) \geq C^{b*}_{k-m+1}(S). \tag{8.68}$$

Hence

$$\sum_{\ell=m}^{k-1} q_\ell(d_\ell - d_{\ell+1}) + q_k d_k \geq \sum_{\ell=m}^{k-1} q_\ell(r_\ell - r_{\ell+1}) + q_k r_k. \tag{8.69}$$

8.7 Conclusions

In this chapter, we studied network coding applied to the case of information broadcast in wireless networks. We have provided the maximum broadcast capacity of the source, for networks modeled by hyper-arcs such as torus grid, which is equal to the number of neighbors. In particular, each node receives, in average, M innovative packets where M is the number of neighbors. Network coding in such networks is efficient in terms of energy efficiency, in the sense that each transmission will provide innovative information. In order to prove this we have translated the min-cut problem on a hypergraph in an equivalent problem of additive combinatorics and we use tools from group theory. Moreover, we have considered torus networks with uniform loss rate and non-uniform loss rate, we have analyzed the maximum broadcast capacity of the source also in these cases and we have found that it is the expected number of neighbors that receive information.

Chapter 9

Performance Computation over TDMA Wireless Networks with VCM Scheduled Transmissions

9.1 Introduction

In Chapter 8, we proved that network coding guarantees that a source node has broadcast capacity equal to the number of neighbors of this source, when all nodes are retransmitting with equal periodicity and for a specific topology. To prove this result, we have considered a torus grid. However, this assumption is naturally not verified in real networks. Moreover the results are asymptotic with time grows towards infinity. Hence, we try in this chapter to illustrate this result "experimentally" (i.e. with simulated transmissions) for a practical case. We consider nodes disposed on a finite grid. Then, we use an existing scheduling algorithm based on TDMA (with a coloring called VCM) to schedule nodes. We then evaluate the performance of the network, i.e. the amount of information that a source can transmit to one selected destination, assuming network coding: it is computed as a max-flow over the resulting *transmission graph*.

9.2 Methodology

9.2.1 Max-flow Computation

Many algorithms have been elaborated for the maximum flow computation, including the Ford-Fulkerson algorithm [1]. To compute the max-flow, we

[1] `http://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm`

apply this algorithm not over the topology, but over the *graph resulting from the transmissions* of nodes, called the transmission graph, know as event-driven graph in [33], and using the approach of that reference.

In our scenario, indeed, nodes transmit each in turn (TDMA scheme) assuming no packet losses; all neighbors receive the emitted packet. In the transmission graph, a node is represented by a set of vertices, each of them representing the node at a given time (see [33]). Edges are created as follows: when a transmission occur, new edges from the vertex of the transmitter (at the associated time) to the vertices, at the given time, of the destinations of this transmission are added to the transmission graph: because wireless transmissions are actually implying an hypergraph, we apply the transformation of a hyperarc to a set of edges as proposed by Y. Wu (in his thesis). Finally, Ford-fulkerson algorithm is applied to this new graph.

9.2.2 VCM: The Vector-based Coloring Method

Graph coloring is one technique to perform node scheduling in wireless sensor networks. Indeed, in spatial TDMA (STDMA), nodes are colored such that no two interfering nodes share the same color. Then, each color is associated to one time slot. Nodes having the same color access the medium during the slot corresponding to their color. We have developed the VCM method [32] for grid coloring. VCM provides a regular coloring, that is a coloring obtained via the repetition of a color pattern. The advantage of this solution is that despite it is distributed, it does not require a coordination or message exchange between nodes to perform coloring. In this chapter, we used VCM to schedule nodes transmissions, as a good exemple of what would be applied in practice.

9.3 Results

We consider a grid with a transmission range= 2.5× the grid step. Figure 9.3 illustrates the neighborhood of the central node. Notice that this node has 20 neighbors.

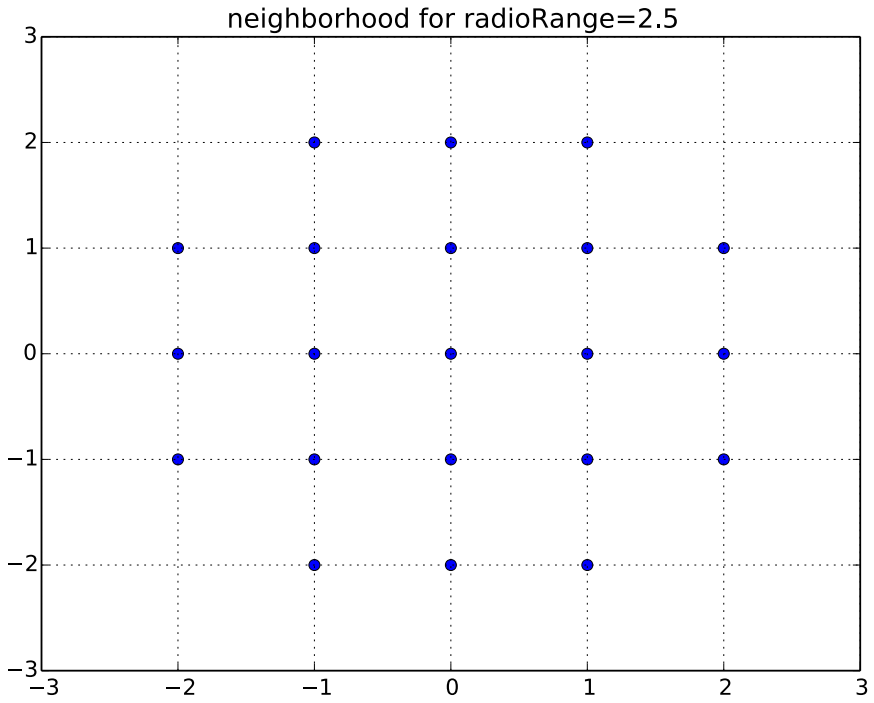


Figure 9.1: Neighborhood of the central node for radio range = 2.5.

Each node transmits a packet at each unit time, except for the source which has an higher rate: it transmits a number of packets equal to the number of neighbors. We assume there is no packet loss. We consider a source node (blue node in Figure 9.2 and a destination node. Then, we compute the max-flow over the transmission graph using Ford-Fulkerson algorithm.

Results show that the max-flow computed is 20 (packets) per unit time, which is in fact the number of neighbors of the source node (and the number of packets it generates by unit time). This confirms the theoretical result that we have shown in Chapter 8, which implies (asymptotic) optimality of network coding operation in this scenario: indeed (except from neighborhhod of the source), asymptotically, every transmission is innovative for every receiver, a strong result.

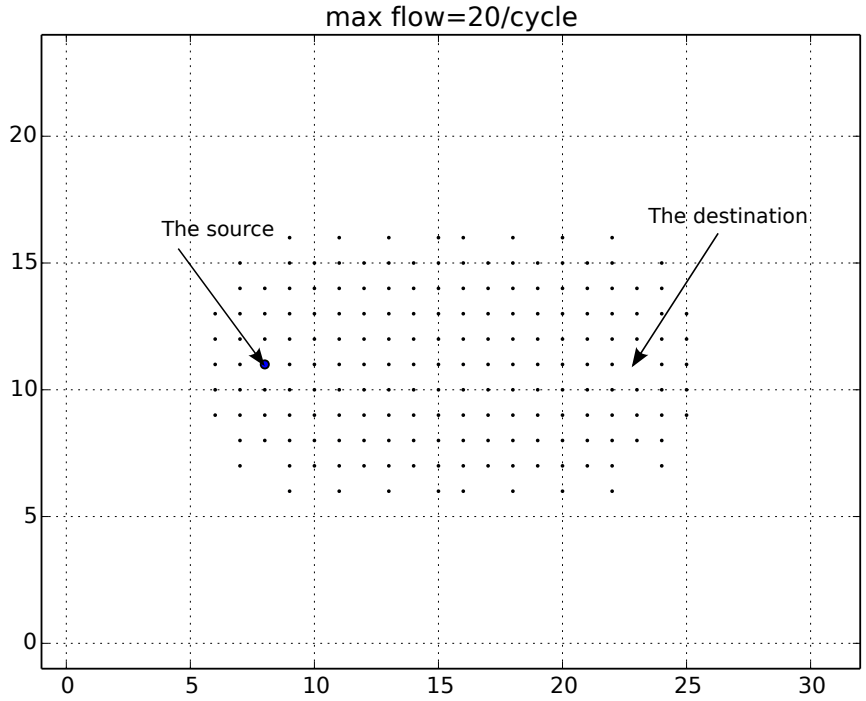


Figure 9.2: Max-flow computation for a finite grid.

Nodes depicted in Figure 9.2 belong to the different 20 paths obtained. A modified Ford-Fulkerson was used with a path selection heuristic prefering paths closer to the source and destination.

The result is an asymptotic result, i.e. what is the number of packets that could be transmitted per unit time from the source to the destination when the network running time converges towards infinity.

Now, we consider a grid colored with VCM, and actual transmissions. Figure 9.3 illustrates the coloring obtained.

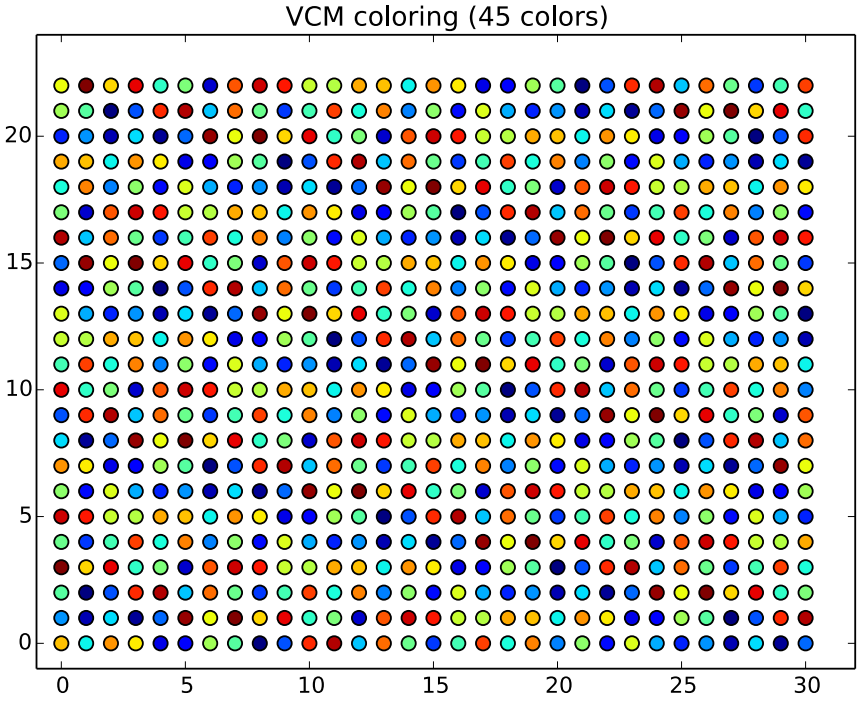


Figure 9.3: The coloring obtained with VCM.

As shown in Figure 9.4, we look at how much (decoded) information the source can send to the destination, when the number of cycles increase. We can see that after a starting phase, the number of packets that can be decoded (the amount of information that can be send, the max-flow on the transmission graph), increases linearly. The phenomenon of existence of starting phase, followed by a steady state phase with constant increase was predicted in [34]. When the number of cycles is ≥ 3, the linear increase is exactly $20/cycle$, as predicted asymptotically by our previous results.

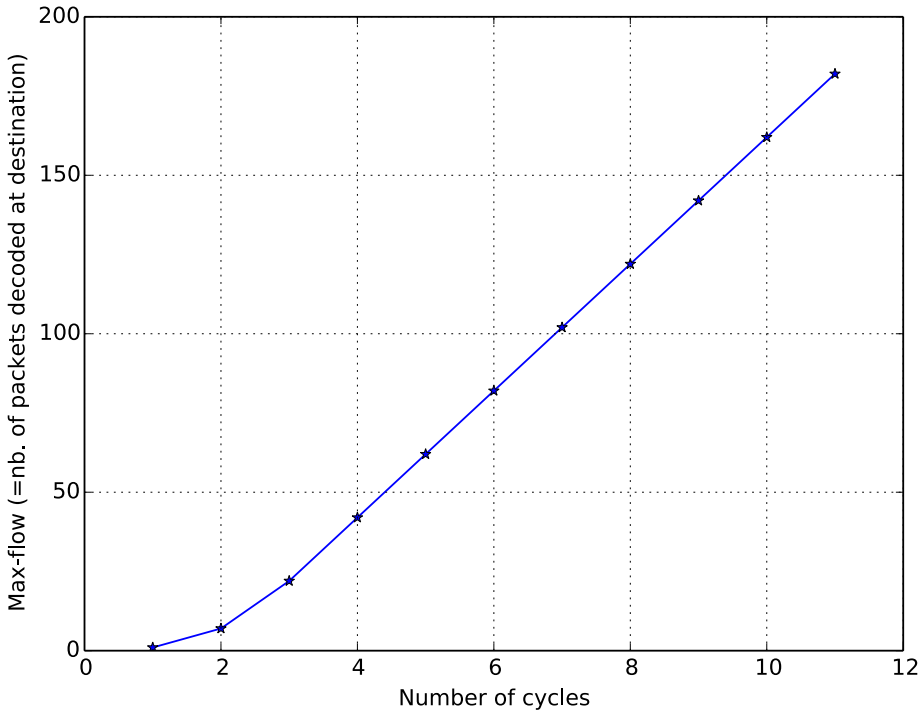


Figure 9.4: With a colored grid: amount of information decodable at the destination, depending on the number of cycles.

Figure 9.5, Figure 9.6 and Figure 9.7 illustrate the evolution of the paths found between the source and destination for successively a number of cycles equal to 1, 2, 3 (starting phase), 4, 6 and 11. For instance, for a cycle of length 1, only one path could be found, whereas for two cycles, 7 paths can be found. After that, the paths found at each cycle (by Ford-Fulkerson + heuristic) tend to repeat themselves, and indeed the max-flow increases linearly.

As verified in Figure 9.7: we compute the max-flow over the transmission graph of this colored network for 11 cycles. The result found is 182. This result is comparable to the one depicted in Figure 9.2 ($182 \approx 20 \times 11$), and due to the fact that the linear increase evidenced in Figure 9.4 is exactly as predicted, 20 per cycle.

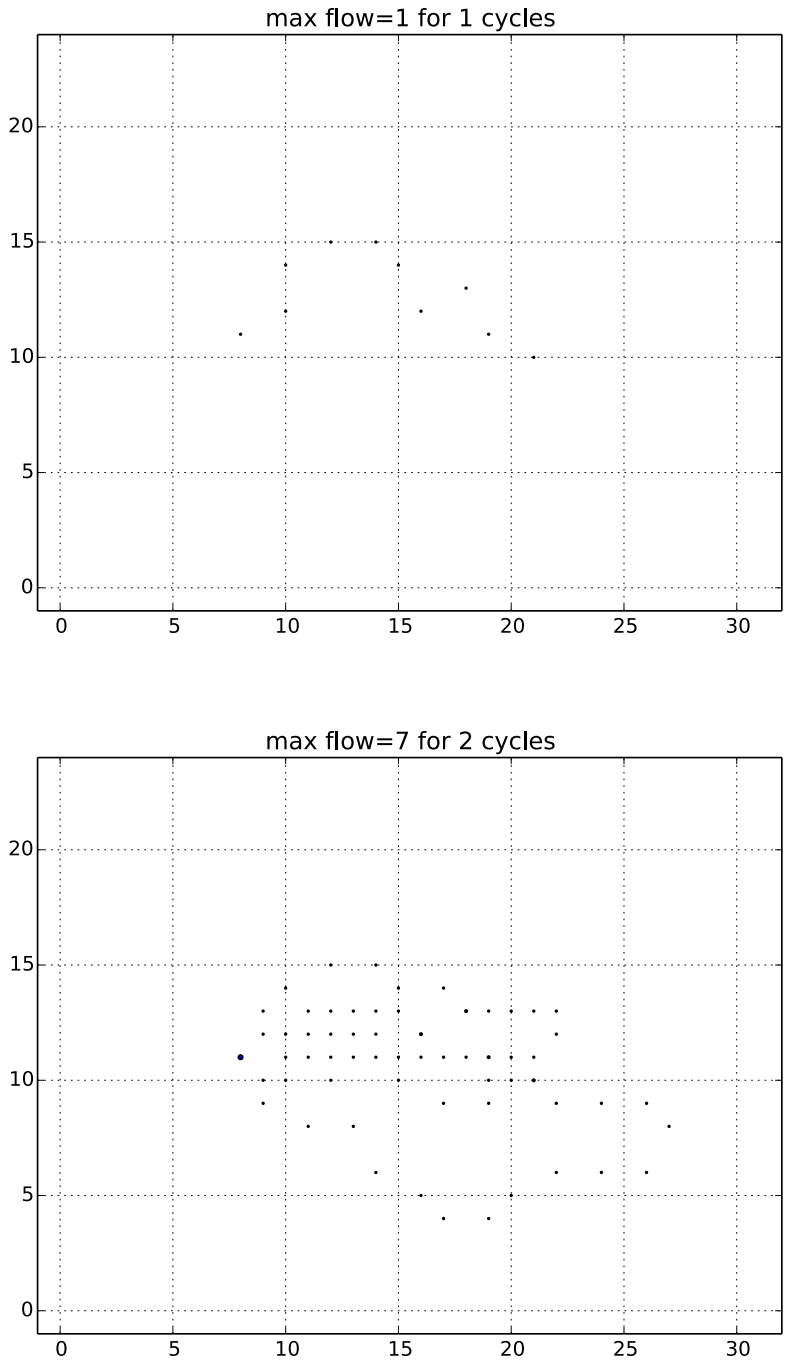


Figure 9.5: Max-flow computation of the colored grid .

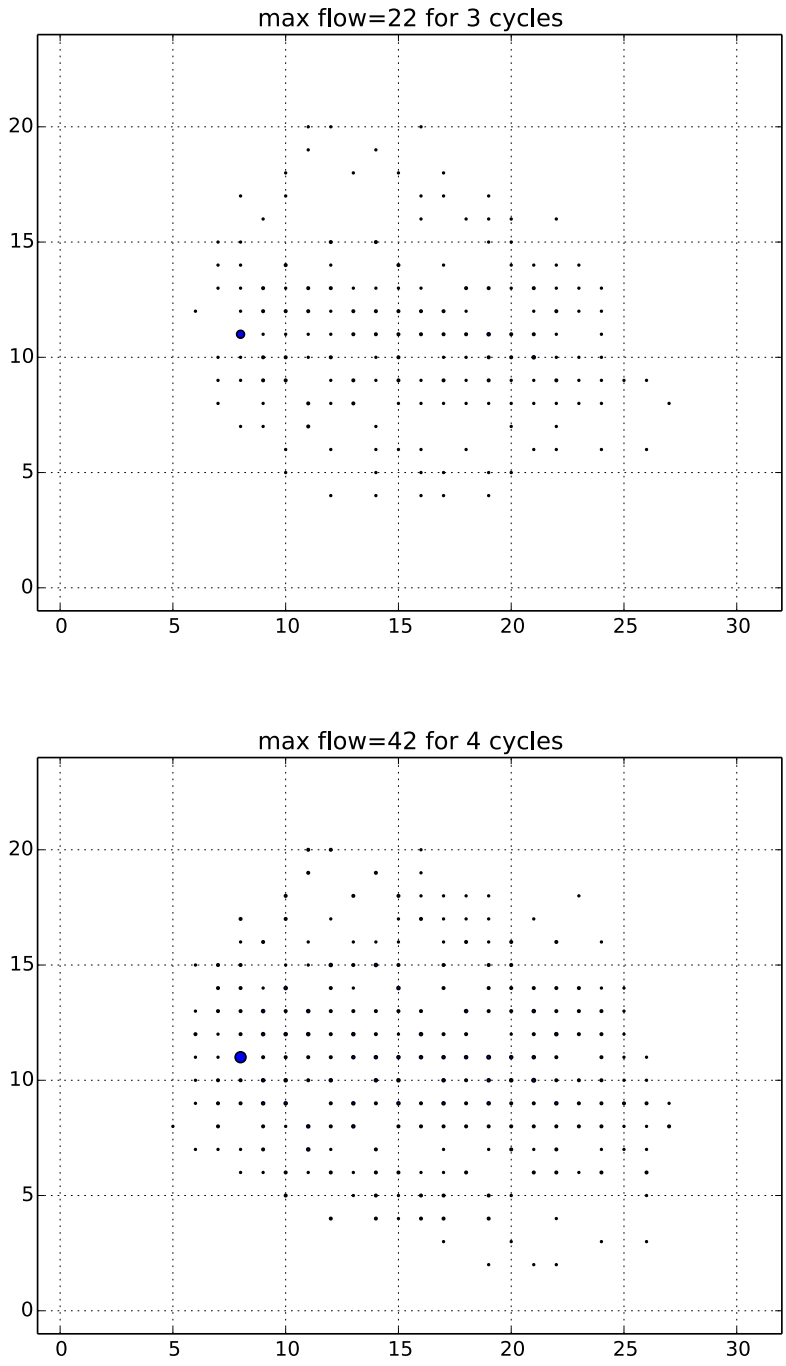


Figure 9.6: Max-flow computation of the colored grid .

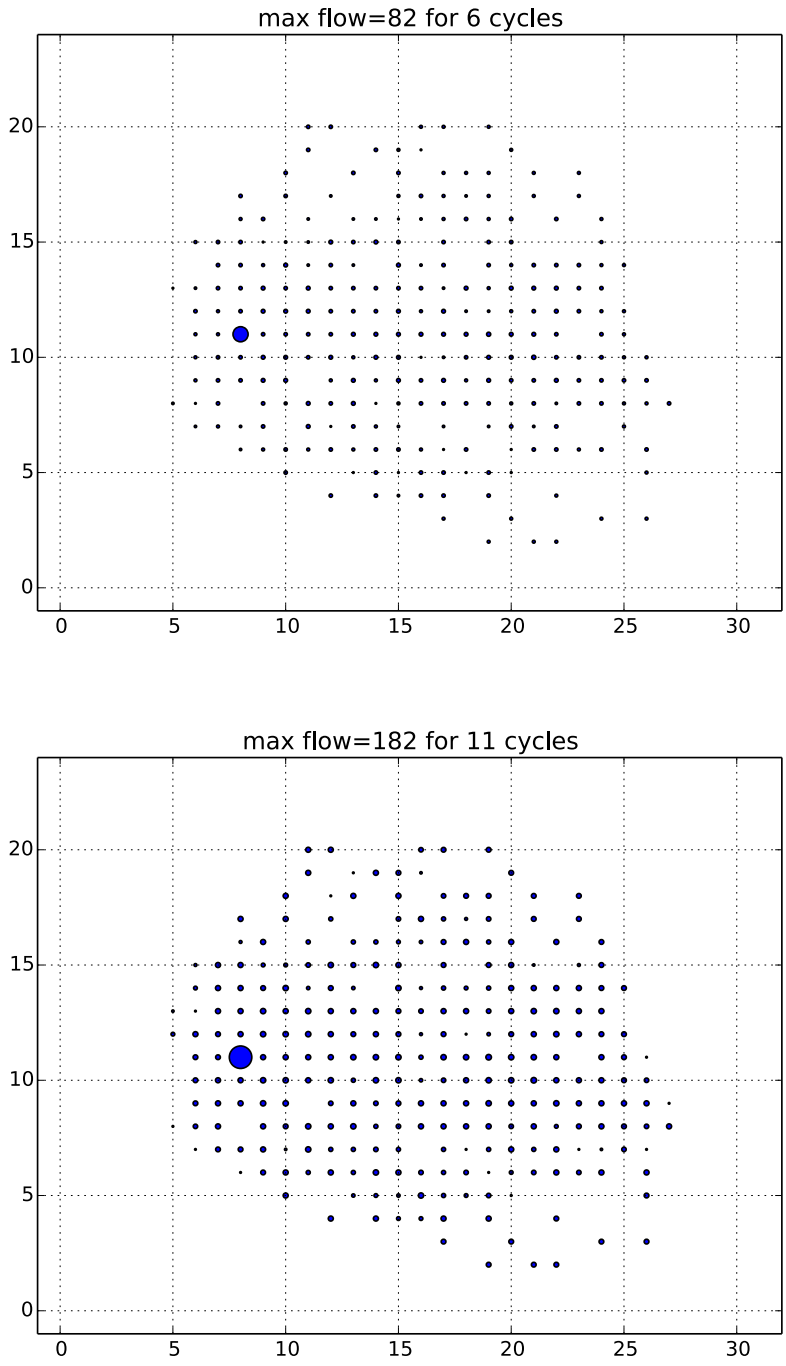


Figure 9.7: Max-flow computation of the colored grid .

9.4 Conclusion

This chapter provided an illustration of the theoretical result that we have presented previously concerning the broadcast capacity of network coding. It was shown in an non-asymptotic setting: interestingly in the case of a discrete grid, the asympotic result corresponds *exactly* to the linear increase after a starting phase. It illustrates also the integration of two of our results in the GETRF project: VCM coloring with network coding.

Bibliography

[1] Cho S-Y., Adjih C., *W*ireless Broadcast with Network Coding: Dynamic Rate Selection, MedHocNet 2008, June 2008.

[2] Cho S-Y., Adjih C., *W*ireless Broadcast with Network Coding: DRAGONCAST, Inria Research Report RR-6569, July 2008. `http://hal.inria.fr/docs/00/29/28/67/PDF/RR-6569.pdf`

[3] Cho S-Y., *E*fficient Information Dissemination in Wireless Multi-Hop Networks, Ecole Polytechnique PhD thesis, September 2008. `http://hal.archives-ouvertes.fr/view_by_stamp.php?&halsid=0a2rq3pq31n2ocg7rosveb4s30&label=LIX&langue=fr&action_todo=view&id=pastel-00004228&version=1&view=extended_view`

[4] Adjih C., Cho S-Y., Baccelli E., *Broadcast With Network Coding: DRAGONCAST*, draft-adjih-dragoncast-00 (work in progress), July 2013. `http://tools.ietf.org/html/draft-adjih-dragoncast-00`

[5] Amdouni I., Adjih C., *Coding Interval-based Sliding Encoding Window*, draft-amdouni-nwcrg-cisew-00 (work in progress), July 2014. `http://tools.ietf.org/html/draft-amdouni-nwcrg-cisew-00`

[6] Firoiu V., Adamson B., Roca V., Adjih C., Bilbao J., Fitzek F., Masucci A., M. Montpetit, *N*etwork Coding Taxonomy, draft-firoiu-nwcrg-network-coding-taxonomy-01 (work in progress), March 2014. `http://tools.ietf.org/html/draft-firoiu-nwcrg-network-coding-taxonomy-01`

[11] Clausen T., Jacquet P., *Optimized Link State Routing Protocol (OLSR)*, RFC 3626, October 2003. `https://www.ietf.org/rfc/rfc3626.txt`

[12] Clausen T., Dearlove C., Dean J., *Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)*, RFC 6130, April 2011. `http://tools.ietf.org/html/rfc6130`

[7] Masucci A., Adjih C., *Capacité de Diffusion avec Codage Réseau dans les Grilles Torique*, ALGOTEL 2014, 16ćemes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, pp. 1-4, June 2014.

[8] Masucci A-M, Adjih C., *Efficiency of Broadcast with Network Coding in Wireless Networks*, Inria research report, February 2014.

[9] Fragouli C., Widmer J., J. Le Boudec, *A Network Coding Approach to Energy Efficient Broadcasting*, INFOCOM 2006, April 2006.

[10] Sundararajan J., Shah D., Medard M., Mitzenmacher, M., J. Barros, *Network Coding Meets TCP*, INFOCOM 2009, April 2009.

[11] Clausen T., Jacquet P., *Optimized Link State Routing Protocol (OLSR)*, RFC 3626, October 2003.

[12] Clausen T., Dearlove C., Dean J., *Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)*, RFC 6130, April 2011.

[13] `https://www.iot-lab.info/`

[14] Adjih C., Amdouni I., Baccouch H., Masucci A., *Experiments with Broadcast with Network Coding*, Presentation at the Network Coding Research Group, IRTF, IETF/IRTF 89, London, March 2014. `http://www.ietf.org/proceedings/89/slides/slides-89-nwcrg-1.pdf`

[15] Ahlswede R., Ning Cai, Li S.-Y.R., Yeung R.W., *Network information Flow*, Information Theory, IEEE Transactions on, 46(4):1204-1216, July 2000.

[16] Koetter R., Medard M., *An algebraic approach to network coding*, IEEE/ACM Trans. Networking, vol. 11, no. 5, pp. 782-95, Oct. 2003.

[17] Lun D.S., Ratnakar N., Medard M., Koetter R., Karger D.R., Ho T., Ahmed E., Zhao F., *Minimum-cost multicast over coded packet networks*, IEEE Transactions on Information Theory, 52(6):2608-2623, June 2006.

[18] Chou P. A., Wu Y., Jain K., *Practical Network Coding*, 43^{th} Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, Oct. 2003.

[19] Wu Y., Chou P. A., Sun-Yuan Kung, *Minimum-energy multicast in mobile ad hoc networks using network coding*, IEEE Transactions on Communications, 53(11):1906-1918, 2005.

[20] Fragouli C., Widmer J., Le Boudec J-Y., *Efficient Broadcasting Using Network Coding*, IEEE/ACM Transactions on Networking, 16(2):450-463, 2008.

[21] Baccelli E., Perkins C., *Multi-hop Ad Hoc Wireless Communication*, draft-baccelli- manet-multihop-communication-02 (work in progress), July 2013.

[22] Adjih C., Cho S. Y., Jacquet P., *Near optimal broadcast with network coding in large sensor networks*, First International Workshop on Information Theory for Sensor Netwoks (WITS 2007), Santa Fe, USA, 2007.

[23] Li S-Y R., Yeung R. W., Cai N., *Linear network coding*, IEEE Transactions on Information Theory, vol. 49, no. 2, pp. 371-381, Feb. 2003.

[24] Ho T., Koetter R., Médard M., Karger D.R., Effros M., *The benefits of coding over routing in a randomized setting*, In Proceedings of 2003 IEEE International Symposium on Information Theory, 2003.

[25] Lun D. S., Médard M., Koetter R., Effros M., *Further results on coding for reliable communication over packet networks*, CoRR, abs/cs/0508047, 2005.

[26] Lun D. S., Médard M., Koetter R., Effros M., *On coding for reliable communication over packet networks*, Physical Communication, 1(1):3-20, 2008.

[27] Deb S., Effros M., Ho T., Karger D. R., Koetter R., Lun D. S., Médard M., Ratnakar N., *Network coding for wireless applications: A brief tutorial*, International Workshop on Wireless Ad-hoc Networks (IWWAN), 2005.

[28] Lee I. K., Kim M. S., Elber G., Wien t., *The Minkowski sum of 2d curved objects*, 1998.

[29] Cho S. Y., Adjih C., Jacquet P.. *Heuristics for network coding in wireless networks* CoRR, abs/0706.4175, 2007.

[30] Tao T., Vu V.H., *Additive Combinatorics*, Cambridge Studies in Advanced Mathematics, Cambridge University Press, 2009.

[31] Lun D.S., Medard M., Koetter R., Effros M., *On coding for reliable communication over packet networks*, Physical Communication, 1(1):3-20, March 2008.

[32] Adjih C., Amdouni I., Minet P., *VCM: the vector-based coloring method for grid wireless ad hoc and sensor networks*, MSWIM 2012, October 2012.

[33] Neglia G., Zhang X., Kurose J., Towsley D., Wang H., *On Optimal Packet Routing in Deterministic DTNs*, IEEE Vehicular Technology Conference, (VTC'77), Dresden, Germany, June 2013.

[34] M. Jafari Siavoshani, C. Fragouli, and S. Diggavi, *"Subspace Properties of Network Coding and their Applications"*, IEEE Transactions on Information Theory, vol. 58, no. 5, pp. 2599-2619, May 2012.