Reliable communication in multihop networks despite byzantine failures

Alexandre Maurer

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Alexandre MAURER

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Communication fiable dans les réseaux multi-sauts en présence de fautes Byzantines

Soutenue le 20 novembre 2014 devant le jury composé de :

M. Sébastien Tixeuil	Directeur de thèse
M. Michel Raynal	Rapporteur
M. Nicola Santoro	Rapporteur
M. Antonio Fernández Anta	Examinateur
M. Rachid Guerraoui	Examinateur
M. Pierre Sens	Examinateur

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Alexandre MAURER

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Reliable communication in multihop networks despite Byzantine failures

Soutenue le 20 novembre 2014 devant le jury composé de :

M. Sébastien Tixeuil	Directeur de thèse
M. Michel Raynal	Rapporteur
M. Nicola Santoro	Rapporteur
M. Antonio Fernández Anta	Examinateur
M. Rachid Guerraoui	Examinateur
M. Pierre Sens	Examinateur

Résumé

A mesure que les réseaux s'étendent, ils deviennent de plus en plus susceptibles de défaillir. En effet, leurs nœuds peuvent être sujets à des attaques, pannes, corruptions de mémoire... Afin d'englober tous les types de fautes possibles, nous considérons le modèle le plus général possible : le modèle Byzantin, où les nœuds fautifs ont un comportement arbitraire (et donc, potentiellement malveillant). De telles fautes sont extrêmement dangereuses : un seul nœud Byzantin, s'il n'est pas neutralisé, peut déstabiliser l'intégralité du réseau.

Nous considérons le problème d'échanger fiablement des informations dans un réseau multi-sauts malgré la présence de telles fautes Byzantines. Des solutions existent mais nécessitent un réseau dense, avec un grand nombre de voisins par nœud. Dans cette thèse, nous proposons des solutions pour les réseaux faiblement connectés, tels que la grille, où chaque nœud a au plus 4 voisins.

Dans une première partie, nous acceptons l'idée qu'une minorité de nœuds corrects échouent à communiquer fiablement. En contrepartie, nous proposons des solutions qui tolèrent un grand nombre de fautes Byzantines dans les réseaux faiblement connectés. Dans une seconde partie, nous proposons des algorithmes qui garantissent une communication fiable entre tous les nœuds corrects, pourvu que les nœuds Byzantins soient suffisamment distants. Enfin, nous généralisons des résultats existants à de nouveaux contextes : les réseaux dynamiques, et les réseaux de taille non-bornée.

Abstract

As modern networks grow larger and larger, they become more likely to fail. Indeed, their nodes can be subject to attacks, failures, memory corruptions... In order to encompass all possible types of failures, we consider the most general model of failure: the Byzantine model, where the failing nodes have an arbitrary (and thus, potentially malicious) behavior. Such failures are extremely dangerous, as one single Byzantine node, if not neutralized, can potentially lie to the entire network.

We consider the problem of reliably exchanging information in a multihop network despite such Byzantine failures. Solutions exist but require a dense network, where each node has a large number of neighbors. In this thesis, we propose solutions for sparse networks, such as the grid, where each node has at most 4 neighbors.

In a first part, we accept that some correct nodes fail to communicate reliably. In exchange, we propose quantitative solutions that tolerate a large number of Byzantine failures, and significantly outperform previous solutions in sparse networks. In a second part, we propose algorithms that ensure reliable communication between all correct nodes, provided that the Byzantine nodes are sufficiently distant from each other. At last, we generalize existing results to new contexts: dynamic networks, and networks with an unbounded diameter.

Contents

Chapter 1

Introduction

In this chapter, we introduce the context of our study, present our objectives and our approach, then explain the organization of this dissertation.

1.1 Context

In today's fast moving world, networks are omnipresent. The most famous example is the Internet, but there are plenty of other applications. For instance:

- Sensor networks collecting physical data such as temperature, pressure, humidity, or detecting forest fires (see Figure 1.1).
- Large computers grids for distributed calculation, such as physical simulations or data analysis (see Figure 1.2).
- Opportunistic networks of smartphones in crowded areas, such as subway stations or commercial centers (see Figure 1.3).
- Networks of mobile robots exploring dangerous or impracticable environments, such as radioactive or underwater zones (see Figure 1.4).

However, as networks grow larger and larger, they become more likely to fail. Indeed, the nodes of the network can be subject to failures, attacks, crashes, memory corruptions... This is a major issue, as complex systems are often "as weak as their weakest component".

FIGURE 1.1: The GreenOrbs sensor network for ecological surveillance in the forest, collecting data such as temperature, humidity, illumination, and carbon dioxide titer. Source: http://www.greenorbs.org/

FIGURE 1.2: A Google data center. Source: http://www.google.com/about/datacenters/

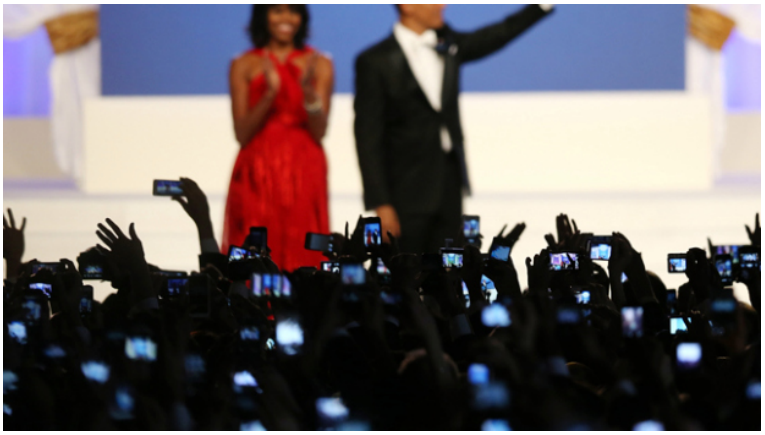

FIGURE 1.3: Public Inaugural Ball at the Walter E. Washington Convention Center, 2013. Almost everyone is equipped with a mobile computing device, which makes an excellent field for the deployment of wireless opportunistic networks. See, for instance: http://anr-crowd.lip6.fr/

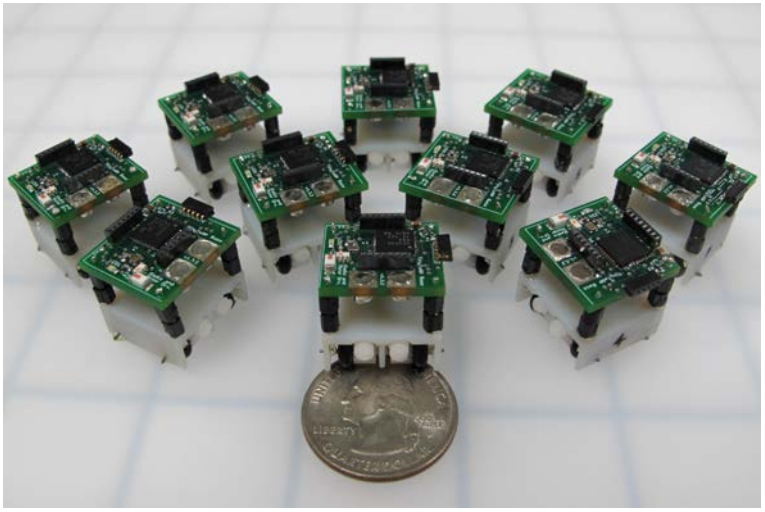

FIGURE 1.4: Cooperative swarming and exploration with TinyTeRP miniature robots. Source: http://robotics.umd.edu/2014_REU/projects.php

1.2 Fault tolerance

To overcome this difficulty, the concept of *fault tolerance* [1] has been introduced. The idea is to design networks such that, even if a small number of elements fail, this does not lead to the global failure of the system.

Many models of node failure have been studied so far, such as crash-stop failures [2], transient faults [3] or memory corruption [4]. In order to encompass *all* possible models of node failure, we have chosen to study the most general model: the *Byzantine* model [5], where the failing nodes have a totally arbitrary (and thus, potentially malicious) behavior. In other words, tolerating Byzantine failures implies to ensure that there exists no strategy, however unlikely it may be, enabling the Byzantine nodes to destabilize the network.

1.3 Our objective

In this dissertation, our goal is to enable the correct nodes to reliably broadcast information throughout the network. For instance:

- In a sensor network, the information can be a physical data such as luminosity or temperature.
- In a network of mobile robots, the information can be the position of the current robot or the identifiers of neighbor robots.

- In a computing grid, the information can be the result of a local calculation.

This is illustrated in Figure 1.5.

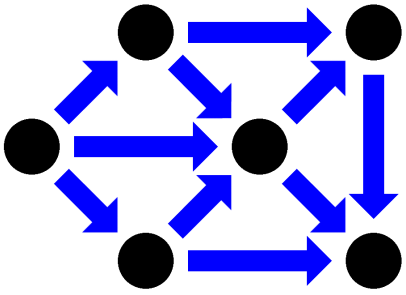

FIGURE 1.5: Example of information broadcast. The left node sends a "blue" message to its neighbors, which transmit it to their own neighbors, and so forth. Eventually, every node receives the "blue" message.

In this context, Byzantine failures are extremely dangerous. Indeed, one single Byzantine node, if not neutralized, can potentially lie to the entire network about the information broadcast by any correct node. This is illustrated in Figure 1.6.

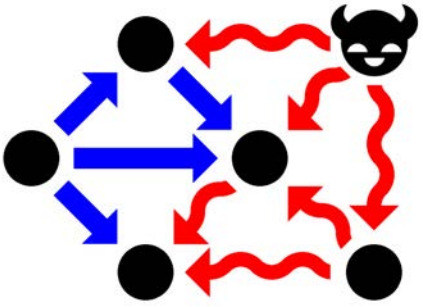

FIGURE 1.6: Example of information broadcast with a Byzantine node. Here, the upper-right node is Byzantine, and broadcasts a "red" message to make the network believe that "red" is the information sent by the left node (which is not the case).

Therefore, to solve this problem, we have to design *Byzantine-resilient* algorithm to broadcast information.

1.4 The field of distributed algorithmics

The classical approach to design algorithms is to implement and test them, in a real network or in a network simulator. However, is the case of Byzantine failures, this approach would require to make restrictive hypotheses on the behavior of Byzantine

nodes. Therefore, nothing guarantees that our experimentations encompass the worst possible situation and the worst possible strategy they could adopt.

We thus adopt a more theoretical approach: *distributed algorithmics* [6–8]. The idea is to prove mathematical properties on the proposed algorithm – for instance, proving that under a certain condition, the Byzantine nodes will never be able to lie to a correct node, whatever their behavior may be. We can thus provide very strong guarantees about the reliability of the network.

1.5 Organization of the thesis

This dissertation is organized as follows:

- In Chapter 2, we make a state of the art of previous works on this topic, and explain our contributions.
- In Chapter 3, we present our model for network analysis, and introduce some general definitions.
- In Part I, II and III, we present our contributions. A general overview of these contribution is given at the end of Chapter 2.
- In Chapter 10, we conclude and discuss about future perspectives.

Chapter 2

Related works and our contribution

In this chapter, we present the related works on the topic, then explain our contribution.

The chapter is organized as follows:

- In Section 2.1, we situate our focus in the domain of Byzantine-resilient algorithms.
- In Section 2.2, we present the works that are closely related to our focus.
- In Section 2.3, we show the limits of existing solutions.
- In Section 2.4, we give a general overview of our contributions.

2.1 Our focus

In this section, we situate our focus in the domain of Byzantine-resilient algorithms. This is graphically summarized in Figure 2.1.

2.1.1 The Byzantine generals problem

The concept of Byzantine failure was introduced by Leslie Lamport, Robert Shostak and Marshall Pease in their paper "The Byzantine Generals Problem" [5]. This problem is based on the following metaphor.

Several *generals* of the Byzantine army (see Figure 2.2) are surrounding an enemy city. They must decide on a common plan: "attack" (let us say 0) or "retreat" (let us say

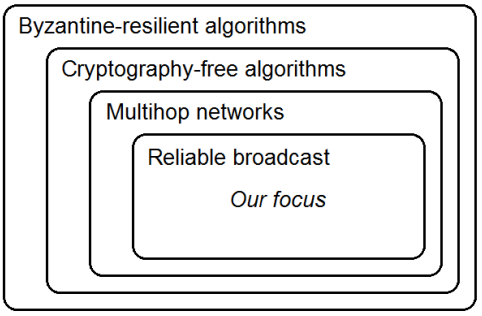

FIGURE 2.1: Graphical representation of our focus

1). As each general defends a different position, the generals can only communicate by sending *messengers*. However, there may be *traitors* among the generals. Thus, the problem is the following: we must find an algorithm enabling the loyal generals to always unanimously decide on a plan ("attack" or "retreat") or not decide at all.

FIGURE 2.2: A scene from the Byzantine-Bulgarian wars (5^{th} century)

In this metaphor, the generals represent the nodes of the network, and the messengers represent the communication channels. The loyal generals correspond to correct nodes, and the traitors correspond to Byzantine nodes. As we do not know the strategy of the traitors, we must therefore guarantee that there exists *no strategy* enabling the traitors to confuse the loyal generals. It was shown that, to solve this problem, it is necessary and sufficient to have $3k + 1$ generals, k being the maximal number of traitors.

The generality of this model has inspired a lot of subsequent works. Indeed, as we assume that the behavior of Byzantine nodes is totally arbitrary, they can represent any type of node failure. For instance, the Byzantine nodes can adopt a malicious strategy

(like broadcasting false messages), but also act randomly, do not act at all (like a crashed node) or even behave like a correct node. Therefore, if we tolerate Byzantine failures, we also tolerate any weaker (that is, more specific) model of node failure.

Note that, even if we consider that only the *nodes* can fail, it implictly encompasses the case where the *communication channels* can fail. Indeed, to represent a faulty channel, in it sufficient to consider that one of the two nodes connected to this channel is Byzantine.

2.1.2 Cryptographic and non-cryptographic approaches

There are two main approaches to deal with Byzantine failures.

- The first one is *cryptography* [9–13]. The idea is to use digital signatures to authenticate the sender across multiple hops. Thus, as the Byzantine nodes do not know some cryptographic secrets, they are not able to lie to correct nodes.
- The second one is non-cryptographic. Here, the idea is to make the assumption that there are only a minority of Byzantine failures. Therefore, if the correct nodes constitute themselves into a "voting system", they are likely to obtain the majority of the votes, and to mask the effect of Byzantine nodes. This is the case, for instance, of the algorithm solving the Byzantine generals problem [5] (see 2.1.1).

Cryptography has a lot of advantages: for instance, the Byzantine generals problem [5] can be solved with an unlimited number of Byzantine failures using cryptography [11]. Also, to ensure reliable communication in a multihop network, it is sufficient that a correct path exists between the two communicating nodes [13].

However, we have chosen not to study cryptography in this dissertation. Our 3 initial reasons were the following:

- First, as stated previously, the idea of cryptography is to use the fact that the Byzantine nodes do not know some cryptographic secrets. However, if we want to consider Byzantine failures in the strong sense of the term, we must assume that they can adopt *any* strategy, however unlikely it may be. Therefore, we must assume that the Byzantine nodes are *omniscient*, and in particular, know any cryptographic secret.
- Second, even if we admit that the Byzantine nodes do not know some cryptographic secrets, manipulating asymmetric cryptography (with public and private keys) requires a lot of computing resources [14]. This may not always be available in low

energy networks, such as wireless sensor networks [15, 16] or autonomous robots networks [17]. Also, we could use symmetric cryptography, but then, the problem becomes to reliably exchange cryptographic keys, which is equivalent to reliably exchanging messages.

- Third, using cryptography implies to rely on a central authority that initially distributes cryptographic keys. Therefore, this initial infrastructure is an "Achilles heel" of the network: if it fails, the whole network fails. Here, we would like to have a totally *decentralized* network, where any element can fail independently without compromising the whole system.

In addition, the recent Heartbleed bug [18], discovered in the widely deployed *OpenSSL*, showed that cryptography is not unconditionally reliable. For these reasons, in the following of this dissertation, we focus on non-cryptographic approaches.

Besides, the *defense in depth* paradigm [19] advocates the use of multiple layers of security controls. Therefore, cryptographic and non-cryptographic strategies can be efficiently combined. For instance, if the cryptography-based security layer is compromised by a bug, a virus or intentional tampering, a cryptography-free communication layer can be used to safely broadcast a patch or to update cryptographic keys.

2.1.3 Fully connected networks and multihop networks

Most cryptography-free works about Byzantine failures consider a *fully connected* network: each process can communicate directly with any other process [5, 20–23]. Therefore, if there are n processes, each process must have $n-1$ communication channels. This may be a problem for large networks, as the number of communication channels per node may be physically limited. Thus, in the following, we consider Byzantine failures in *multihop* networks: only some pairs of processes (called *neighbors*) are linked by a communication channel. Therefore, two distant nodes must rely on intermediary nodes to communicate. This is illustrated in Figure 2.3.

Also, note that a fully connected network is often an abstraction of a multihop network where peer-to-peer communications are assumed to be reliable. Thus, conceiving algorithms for reliable communication in multihop network is sometimes a necessary step to execute algorithms designed for fully connected networks.

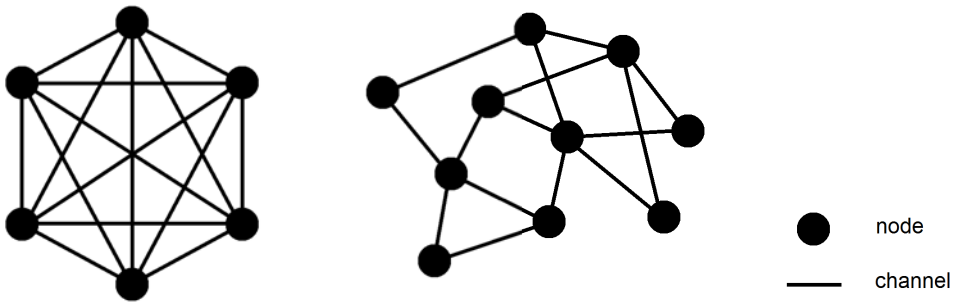

FIGURE 2.3: Fully connected network (left) and multihop network (right)

2.1.4 Space local and time local algorithms

In multihop networks, a notable class of algorithms tolerates Byzantine faults with either *space* or *time* locality.

- *Space local* algorithms [24–26] try to contain the fault as close to its source as possible. That is, they ensure that the nodes at a certain distance from Byzantine failures satisfy the desired property. This is only applicable to the problems where the information from remote nodes is unimportant, such as vertex coloring, link coloring or dining philosophers.

  It was shown that, if the problem involves nodes interacting at distance d, it is impossible to contain Byzantine failures with a radius smaller than d [25]. Thus, the local containment approach is not applicable to reliable communication, as it involves nodes at an arbitrary distance.

- *Time local* algorithms [27–31] try to limit the number of times where correct nodes are disturbed by Byzantine nodes (*disruption time*). Yet, time local algorithms presented so far can hold at most one Byzantine node, and are not able to mask the effect of Byzantine actions.

  Time locality can also be related to the paradigm of *self-stabilization* [3]. Self-stabilizing algorithms assume that the nodes of the network can have any faulty initial state. This can represent the fact that some nodes are Byzantine during a certain laps of time, then come back to a correct behavior.

In this thesis, we consider the problem of reliably exchanging information in the presence of Byzantine failures. None of these approaches is applicable to this problem.

2.2 Closely related works

In this section, we introduce the problem investigated in this thesis, and present the existing works on this topic.

2.2.1 The problem of reliable communication

As stated previously, we consider the following setting: a multihop communication network where some nodes may be Byzantine, and where cryptography is not allowed. We consider the problem of *reliable communication*. A more formal definition of this problem is given in Chapter 3. For the moment, let us just give an informal description.

Let p and q be two correct nodes of the network. The node p wants to broadcast a specific message m to the rest of the network. In a network where all nodes are correct, the following would happen: p sends m to its direct neighbors, that in turn send it to their neighbors – and so forth, until every correct node receives the message. Let us call this an *unsecured broadcast*.

In out setting however, some nodes may be Byzantine, and broadcast false messages. Thus, some correct nodes may believe that p sent a message $m' \neq m$, which is not true. Our goal is to design algorithms that ensure that the correct message (and only the correct message) is received. We say that an algorithm *ensures reliable communication between p and q* if q always eventually receives and accepts the message from p, and never accepts a false message pretending to be from p. This property must hold for *any* possible behavior of the Byzantine nodes.

Very few non-cryptographic algorithms exist for this problem. They can be split in two categories: the *local vote*, and the *vote on multiple paths*. Let us present those existing solutions.

2.2.2 Local vote

As seen in 2.2.1, in the case of simple broadcast, a node accepts and forwards a message as soon at it is received from *one* neighbor. Thus, one single Byzantine node is sufficient to initiate the broadcast of a false message. In [32], Koo proposed a simple solution to limit the power of Byzantine nodes: to accept and forward a message, a node must receive it, not from one, but from k distinct neighbors ($k > 1$). Thus, if less than k neighbors are Byzantine, they will never be able to cooperate to initiate the broadcast of a false message. This is illustrated in Figure 2.4. The only exception is for the

initiator of the message: as its neighbors know that it is the source, they accept its message directly.

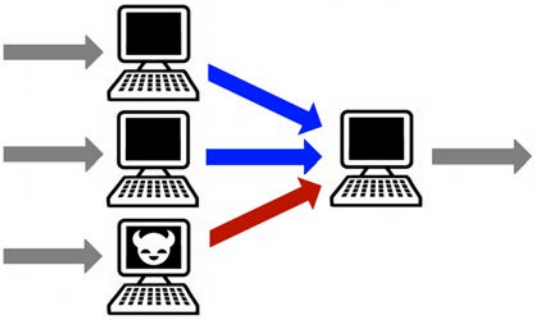

FIGURE 2.4: Certified Propagation Algorithm with $k = 2$

Although this algorithm is very simple, its analysis is not trivial. In [32], the setting is a radio network of nodes spatially organized on a grid. Each node is neighbor with nodes that are located within a certain radius (assumed to be greater than 1). Here, the criteria on Byzantine failures is the *fraction* of Byzantine neighbors per correct node. Both upper and lower bounds are given for this problem. When less than a $1/4\pi$ fraction of neighbors are Byzantine, the algorithm always ensures reliable communication between all pairs of correct nodes. On the other hand, if we have more than a $1/\pi$ fraction of Byzantine neighbors, there exists no algorithm ensuring reliable communication.

This result was later improved in [33], where the limit fraction of Byzantine neighbors is extended to $1/4$. In [34], a formula is given to determine whether or not this algorithm works for a given communication graph. At last, [35] showed the optimality of this result: if this algorithm does not ensure reliable communication on a given graph, then neither does any other algorithm (when the criteria is on the fraction of Byzantine neighbors).

2.2.3 Vote on multiple paths

Another strategy is to send the message through several disjoint paths between the sender and the receiver. The receiver collects the messages from different paths, then decides on the message to accept with a majority vote. Thus, if only a minority of paths are corrupted by a Byzantine node, the correct message is accepted. This is illustrated in Figure 2.5, with one corrupted path over three.

Here, we consider that up to k nodes may be Byzantine in the whole network. A first paper [36] showed that, to ensure reliable communication, it was necessary and sufficient that the network is $(2k+1)$-connected – that is, there exists at least $2k+1$ disjoint paths between each pair of nodes. Yet, this solution requires that each correct node is aware

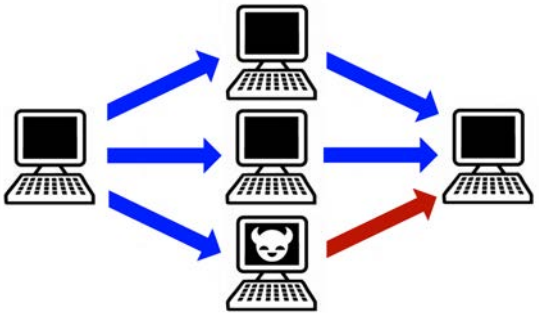

FIGURE 2.5: Example of vote on multiple paths

of the global topology of the network. This requirement was released in [37], where the topology of the network can be unknown.

The principle of the algorithm used to show the sufficient condition is the following: each node has a unique identifier, and the message registers the identifiers of the nodes that forward it. Thus, the receiver can check whether or not two paths are disjoint. The number of disjoint paths to collect before accepting the message determines the level of security of the algorithm: to tolerate k Byzantine failures with certainty, at least $2k+1$ disjoint paths must be collected.

The proof of the necessary condition relies on Menger's theorem [38], which ensures the equivalence between node cut and connectivity: the number of disjoint paths between two nodes p and q is also the minimal number of nodes that should be removed to disconnect p from q. Thus, if we only have $2k$ disjoint paths, the message received by q is entirely determined by $2k$ nodes. As k of them may be Byzantine, by symmetry, it is impossible to ensure that the correct message is accepted.

2.3 Limits of existing solutions

In this section, we show the limits of the aforementioned solutions.

2.3.1 Motivation

This thesis started with the following observation: both aforementioned solutions implicitly assume a dense network – that is, each node must have a large number of neighbors (for the local vote), and the network must have a high connectivity (for the vote on multiple paths). Thus, we raise the following question: what happens if the network is more sparsely connected?

To answer this question, we consider a simple and regular topology: the *grid* (see Figure 2.6), where each nodes has at most 4 neighbors. This topology appears in many applications, such as large-scale computation grids for industrial simulations [39]. A more formal definition is given in Chapter 3

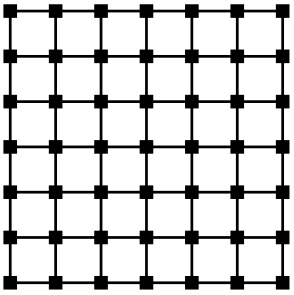

FIGURE 2.6: A 7×7 grid

Let us see what happens to both existing solutions on a grid. Note that the following remarks are also valid for less regular sparse networks.

2.3.2 Local vote

The Certified Propagation Algorithm (see 2.2.2) does not work on a grid. To understand this, let us consider the nodes of Figure 2.7. Let us consider the most tolerant setting of the CPA: to accept a message, a node must receive it from only two neighbors.

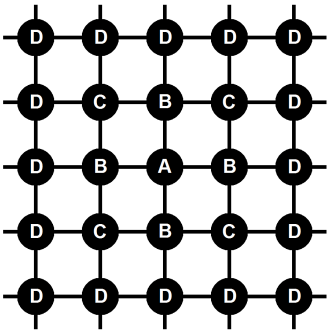

FIGURE 2.7: Limits of the CPA on a grid

Let us assume that the node A is the source. As the nodes of type B are neighbor with the node A, they directly accept its message. Then, as each node of type C has two neighbors of type B, they also accept and forward the message.

But then, no node of type D has more than one neighbor of type B and C. Thus, no node of type D accepts the message, and the broadcast stops here.

2.3.3 Vote on multiple paths

The vote on multiple paths (see 2.2.3) works on a grid, but tolerates at most one Byzantine failure. To understand this, let us suppose that 2 nodes are Byzantine, and let us consider any sender and receiver. As the behavior of Byzantine nodes is arbitrary, it is possible that they start broadcasting a false message before the sender is activated. Thus, the receiver will receive a false message from 2 disjoint paths. Then, as the receiver has at most 4 neighbors on a grid, it will receive the correct message from at most 2 disjoint paths. As both correct and false messages have 2 votes each, the receiver will not be able to decide on a message to accept. Thus, at most one Byzantine node can be tolerated here.

2.4 Our contribution

In this section, we give a general overview of our contributions.

As seen in 2.3, existing solutions can tolerate at most *one* Byzantine failure on a grid. Thus, our initial motivation is the following: to design algorithms that tolerate *many* Byzantine failures on a grid, and in other loosely connected networks.

This dissertation is organized in 3 parts:

- In Part I, we accept the idea that a *minority* of pairs of nodes may fail to communicate reliably, and propose algorithms that tolerate a large number of Byzantine failures with this concession.
- In Part II, we give algorithms that ensure perfect reliable communication provided that Byzantine failures are sufficiently distant.
- In Part III, we present extensions of existing solutions.

2.4.1 Part I: Quantitative Byzantine tolerance

In this part, we make a concession to counterbalance the low connectivity of the grid: we accept the idea that some pairs of nodes may fail to communicate reliably, provided that we ensure reliable communication between a *majority* of pairs of nodes.

In Chapter 4, we assume that the nodes know the topology of the network, and propose an algorithm based on *control zones* (arbitrary sets of nodes that filter false messages). We provide a methodology to determine whether two given nodes communicate reliably

or not. Then, with Monte-Carlo simulations, we show that this algorithm significantly outperforms previous solutions on a grid.

In Chapter 5, we now assume that the nodes do not know their position in the network, and propose an algorithm based on bounded disjoint paths. We then make a general comparison of available solutions.

2.4.2 Part II: Qualitative Byzantine tolerance

In this part, we propose algorithms that ensure reliable communication between *all* pairs of correct nodes, provided that the Byzantine failures are sufficiently distant.

In Chapter 6, we show than any 3-connected network admits a particular cycle decomposition. For instance, a grid can be decomposed in elementary square cycles, and a planar graph in elementary polygons. We give an algorithm that ensures perfect reliable communication when the distance between Byzantine failures is more than twice the diameter of the largest cycle. Then, we make this algorithm *self-stabilizing*: the nodes retrieve correct outputs even if the initial state of their variables is corrupt and totally arbitrary.

In Chapter 7, we consider a 8-connected lattice network. On this network, the Certified Propagation Algorithm [32] works provided that there is no *critical pair* - that is, a pair of Byzantine nodes that are too close from each other. We give an algorithm that can tolerate several critical pairs, provided that these pairs are sufficiently spaced.

2.4.3 Part III: Extensions

In this last part, we present extensions of existing solutions:

In Chapter 8, we extend the results of [36, 37] and give the necessary and sufficient condition to tolerate k arbitrarily placed Byzantine nodes in a *dynamic* network (where the topology can vary with time). We then study the satisfaction of this condition in several case studies.

In Chapter 9, we consider a grid with a uniform rate of Byzantine failures (each node has a probability λ to be Byzantine). In this setting, all aforementioned solutions fail if the size of the grid grows larger and larger. We propose a fractal algorithm that ensures a constant communication probability, however large the grid may be.

Chapter 3

Model and definitions

In this chapter, we provide general definitions and explain our model.

3.1 Definitions

3.1.1 Graph

A *graph* is a tuple (V, E) where:

- V is the set of *nodes*.
- $E \subseteq V \times V$ is the set of *edges*.

The nodes represent the processes of the network, and the edges represent the communication channels. Two nodes linked by an edge are said to be *neighbors*.

3.1.2 Paths

Definition 3.1 (Path)**.** A sequence of nodes $(u_1, \ldots, u_n)$ is a *path* connecting u_1 and u_n if, $\forall i \in \{1, \ldots, n-1\}$, u_i and u_{i+1} are neighbors.

Definition 3.2 (Disjoint paths)**.** Two paths $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_m)$ are *disjoint* if $\{u_1, \ldots, u_n\} \cap \{v_1, \ldots, v_m\} = \emptyset$. They are *internally disjoint* if $\{u_2, \ldots, u_{n-1}\} \cap \{v_2, \ldots, v_{m-1}\} = \emptyset$. A set of paths $\{P_1, \ldots, P_N\}$ is a set of disjoint paths if $\forall \{i, j\} \subseteq \{1, \ldots, N\}$, P_i and P_j are disjoint.

3.1.3 Metric

Definition 3.3 (Degree)**.** The *degree* of a node is the number of neighbors of this node. The degree of a network is the maximal degree of its nodes.

Definition 3.4 (Distance)**.** We say that two nodes p and q are at *distance* d if the shortest path $(u_1, \ldots, u_n)$ connecting p and q is such that $n = d$.

Definition 3.5 (Diameter)**.** The *diameter* of a network is the maximal distance between two nodes of this network.

3.1.4 Connectivity

Definition 3.6 (Connected set of nodes)**.** As set S of nodes is *connected* if, $\forall \{p, q\} \subseteq S$, there exists a path connecting p and q.

Definition 3.7 (Node cut)**.** As set of nodes X is a *node cut* isolating a node p from a node q if, after the removal of the nodes of X, there exists no path connecting p and q. X is a node cut isolating a set of nodes P from a set of nodes Q if, $\forall p \in P$ and $\forall q \in Q$, X is a node cut isolating p from q.

Definition 3.8 (Connectivity)**.** The *connectivity* of the network is C if, for any nodes p and q, there exists a set of C disjoint paths Ω such that, $\forall (u_1, \ldots, u_n) \in \Omega$, p and u_1 (resp. q and u_n) are neighbors.

3.1.5 Topologies

At last, let us define 4 network topologies that will be used in the following: the grid, the torus, the hexagonal grid ans the hexagonal torus.

Definition 3.9 (Grid and torus)**.** A $N \times M$ *grid* (resp. *torus*) is a network such that:

- Each node has a unique identifier (i, j), with $1 \leq i \leq N$ and $1 \leq j \leq M$.
- Two nodes (i_1, j_1) and (i_2, j_2) are neighbors if and only if one of these two conditions is satisfied:
  - $i_1 = i_2$ and $|j_1 - j_2| = 1$ (resp. 1 or M)
  - $j_1 = j_2$ and $|i_1 - i_2| = 1$ (resp. 1 or N)

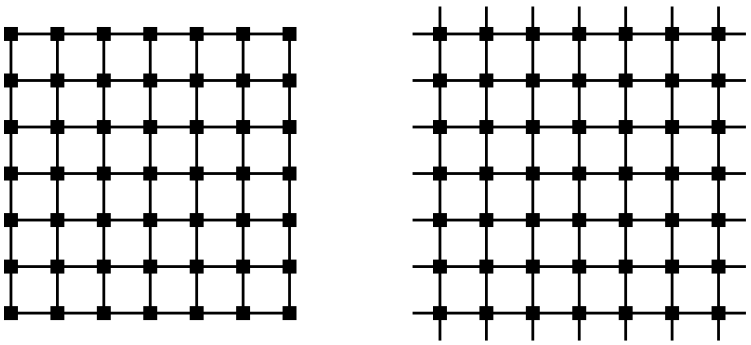

FIGURE 3.1: A 7×7 grid (left) and torus (right)

This is illustrated in Figure 3.1. The torus can be seen as a continuous version of the grid, where the upper (resp. right) side is connected to the lower (resp. left) side.

Now, let us define the hexagonal grid (resp. hexagonal torus), which is obtained by removing edges of a grid (resp. torus).

Definition 3.10 (Hexagonal grid and torus)**.** A $N \times M$ *hexagonal grid* (resp. *hexagonal torus*) is a $N \times M$ grid (resp. torus) with several edges removed: $\forall (i,j) \in \{1,\dots,N-1\} \times \{1,\dots,M\}$, if $i+j$ is odd, we remove the channel between (i,j) and $(i+1,j)$. We also remove the potential nodes with only one neighbor.

This transformation is illustrated in Figure 3.2.

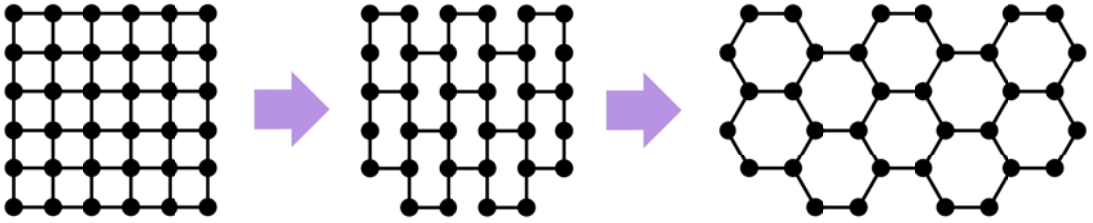

FIGURE 3.2: Turning a grid into a hexagonal grid.

3.2 Model

In this section, we present our hypotheses and the specification of the problem.

3.2.1 Hypotheses

We make the same hypotheses as the closely related previous works [32–37]:

- The nodes are independent processes, and two neighbor nodes can send *messages* to each other. We assume that any message sent is eventually received. However, we make no hypotheses on synchronicity: a message can be received at any time.
- Some nodes are *correct*, and follow a given algorithm. The other are assumed to be *Byzantine*, and have an unknown arbitrary behavior. Thus, we never make any hypotheses on the behavior of Byzantine nodes.
- Each node has a unique identifier. Besides, we assume that the communication channels are *authenticated*: when a node p receives a message from a neighbor q, p known the identity of q. Thus, a Byzantine node cannot forge its own identity.

3.2.2 Problem

Each node p holds a message $p.m_0$, and wants to broadcast this message throughout the network. We say that a node q *accepts* a message m from p when it considers that $m = p.m_0$ (that is, p is the author of the message m).

Definition 3.11 (Reliable communication)**.** For two correct nodes p and q, we have *reliable communication* from p to q when the two following conditions are satisfied:

- If q accepts m from p, then necessarily, $m = p.m_0$ (in other words, q never accepts a false message pretending to be from p).
- The node q always eventually accepts $p.m_0$ from p (the correct message is eventually accepted).

We also say that q is *reliable for* p. Is p is also reliable for q, we say that p and q *communicate reliably*.

Definition 3.12 (Reliable node set)**.** For a given correct node p, we say that a set of correct nodes S is *reliable for* p if $\forall q \in S$, q is reliable for p. A set of correct nodes S is simply *reliable* if $\forall \{p, q\} \subseteq S$, p and q communicate reliably.

In the following, we refer to the attribute X of a node p by $p.X$. We say that a node *multicasts* a message when it sends it to all its neighbors.

Part I

Quantitative Byzantine tolerance

Chapter 4

Control Zones

In this chapter, we propose a first algorithm to tolerate a large number of Byzantine failures in sparse networks.

For this purpose, we make the following concession: we accept that a small minority of correct nodes fail to communicate reliably. In exchange, we show that we can tolerate an important number of Byzantine failures while preserving reliable communication between a large majority of correct nodes.

We propose a protocol based on *control zones* and *authorizations*. Intuitively, control zones act as filters in the network: they limit the diffusion of Byzantine messages.

The chapter is organized as follows:

- In Section 4.1, we describe our protocol.
- In Section 4.2, we explain how to theoretically determine whether or not two given correct nodes communicate reliably.
- In Section 4.3, we use this theoretical methodology to statistically evaluate the performances of our protocol, and compare it with existing solutions.

The results of this chapter were published in the conferences AlgoTel [40], ICDCS [41] and in the journal TPDS [42].

4.1 Algorithm

In this section, we give an informal description of our protocol, define the notion of control zone and describe the algorithm.

4.1.1 Informal description

A correct node p tries to broadcast a tuple (p, m_0). The nodes receiving this tuple assume that m_0 is the message broadcast by p. Yet, a Byzantine node can broadcast a tuple (p, m_1), with $m_1 \neq m_0$, to make the network believe that p broadcast m_1.

To limit the action of Byzantine nodes, we define a set of *control zones*. A control zone is defined by:

- Its *core*, an arbitrary set of nodes.
- Its *boundary*, a node-cut isolating the core from the rest of the network.

An example of control zone is given in Fig. 4.1. The important point is that messages must pass through the *boundary* to access the *core*.

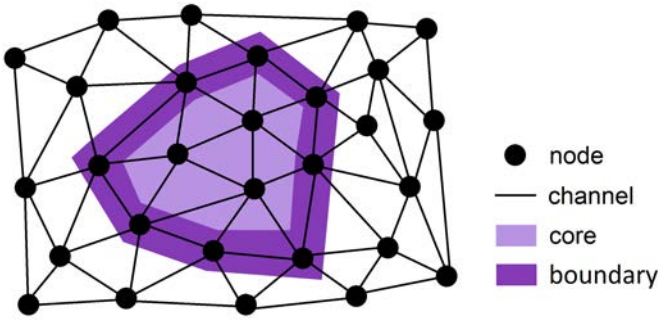

FIGURE 4.1: Example of control zone)

Here is the main idea of the protocol:

- When a message enters the *core* of a control zone, an *authorization* is broadcast on its *boundary*. This message, unlike standard messages, is not affected by other control zones.
- When the same message wants to exit the *core*, this *authorization* is required.

This mechanism does not disturb the broadcasting of correct messages.

Now, suppose that a Byzantine node is in the core of the control zone, and sends a false message (p, m_1), whereas p is *not* in the core of the control zone. Then, this message never gets the authorization to exit the core, as it never entered it. This is illustrated in Fig. 4.2.

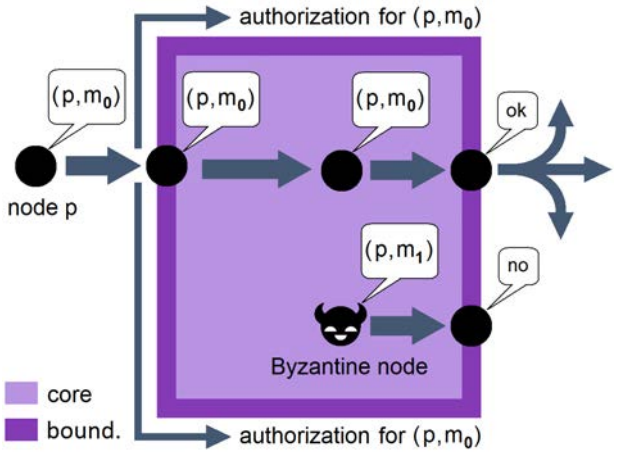

FIGURE 4.2: Principle of a control zone

Intuitively, this mechanism of control zones enables to limit the broadcast of Byzantine messages. The underlying idea is to define a lot of control zones on the network, intersecting each other, in order to minimize the broadcast of Byzantine messages. For instance, if a Byzantine node lies on the boundary of a first control zone, it can broadcast authorizations for its own false messages. However, if a second control zone with a correct boundary surrounds this Byzantine node, its messages will never cross the second control zone, even if they have the authorizations for the first control zone.

4.1.2 Control zones

Definition 4.1 (Control zone)**.** A *control zone* is a tuple $(Core, Boundary)$ of disjoint, connected node sets, such that $Boundary$ is a node-cut isolating $Core$ from the rest of the network.

We denote the core and the boundary of a control zone z by $core(z)$ and $boundary(z)$.

Before running the protocol, we choose an arbitrary set Ctr of control zones that will be used in the protocol. For each correct node p, let $p.myCtr$ be the set of control zones $z \in Ctr$ such that $p \in boundary(z)$. We assume that, for each zone $z \in myCtr$, p knows which nodes belong to $core(z)$ and $boundary(z)$.

4.1.3 Description of the protocol

In our protocol, two types of messages can be exchanged:

- *Standard messages*, of the form (s, m): a message claiming that the node s (*source*) broadcast m.
- *Authorization messages*, of the form (s, m, z): a message authorizing the *standard message* (s, m) to exit the control zone z.

Each node possesses three memory sets:

- $Wait$: set of messages received, but waiting for an authorization (initially empty). When $(s, m, q) \in Wait$, it means that p received the *standard message* (s, m) from a neighbor q.
- $Auth$: set of authorizations received (initially empty). When $(s, m, z) \in Auth$, it means that p has received the authorization for the *standard message* (s, m) on the control zone z.
- Acc: set of accepted messages (initially empty). When $(s, m) \in Acc$, it means that p has received (s, m) and all the corresponding authorizations, and has sent it to its neighbors.

At last, each correct node p obeys to the four following rules.

INIT - Initial broadcast. Executed initially.

- Send (p, m_0) to all neighbors.
- Add (p, m_0) to Acc.
- $\forall z \in myCtr$, send (p, m_0, z) to all neighbors.

ENTER - Message entering control zones. Executed when a standard message (s, m) is received from a neighbor q.

- If $(s, m) \in Acc$, ignore it.
- Else, add (s, m, q) to $Wait$.

DIFF - Diffusion of authorizations. Executed when an authorization message (s, m, z) is received from a neighbor q.

- If $(s, m, z) \in Auth$, or $q \notin boundary(z)$, ignore it.
- Else:
  - Add (s, m, z) to $Auth$.
  - Send (s, m, z) to all neighbors.

EXIT - Message exiting control zones. Executed when an element (s, m, q) of $Wait$ verifies the following condition: $\forall z \in myCtr$ such that $q \in core(z)$ and $s \notin core(z)$, we have $(s, m, z) \in Auth$.

- Add (s, m) to Acc.
- Send (s, m) to all neighbors.
- $\forall z \in myCtr$, send (s, m, z) to all neighbors.

4.2 Reliability properties

In this section, we give a theoretical methodology to determine the set of nodes that always communicate reliably, for a given placement of Byzantine nodes.

First, let us explain why this methodology is necessary to correctly evaluate the protocol.

4.2.1 Motivation

To evaluate the performances of our protocol, a natural idea would be to directly simulate it. However, in the presence of Byzantine failures, things are not that simple. Indeed, simulating the protocol would imply to make restrictive assumptions on the order of activation of nodes, the order of reception of messages and the behavior of Byzantine nodes. This would considerably weaken the model, as nothing guarantees that we encompass the worst possible cases. In particular, as the behavior of faulty nodes is restricted (thus not totally arbitrary), these nodes cannot be called Byzantine anymore.

Therefore, instead of simulating the protocol, we provide a deterministic technique that, for a given set of Byzantine nodes, returns a set of correct nodes that *always* communicate reliably, independently of the order of execution and of the behavior of Byzantine

nodes. Thus, with this methodology, we can evaluate our protocol in Section 4.3 without adding restrictive assumptions.

Note that this methodology is *not* to be computed by correct nodes (which do not know the position of Byzantine nodes). This is an external view of the network.

The rest of the section is organized as follows:

- In 4.2.2, we provide some definitions.
- In 4.2.3, we explain how to determine the optimal *safe* node set, that is: the set of nodes that never accept any false message.
- In 4.2.4, we explain how to construct the optimal *communicating* node set, that is: the set of nodes that always accept the message of a given correct node p.
- In 4.2.5, we use the two aforementioned sets to determine the optimal *reliable* node set, that is: the set of nodes that always accept the message of p, and never accept any false message. Then, we show that *all* the nodes of this set communicate reliably.
- In 4.2.6, we evaluate the message complexity.

4.2.2 Definitions

We say that a correct node p *accepts* a message (s, m), when (s, m) is added to the set $p.Acc$. A message (s, m) is *correct* if s is correct and $m = s.m_0$. Else, it is *false*.

Definition 4.2 (Safe node set)**.** A node is *safe* if it never accepts a false message, in any possible execution. A set of nodes is safe if all its nodes are safe.

Definition 4.3 (Communicating node set)**.** Let p be a correct node. A set of nodes S is *communicating* for p if each node of S eventually accepts $(p, p.m_0)$, in any possible execution.

4.2.3 Determination of a safe node set

Let us give a methodology to determine the optimal *safe* node set (see Definition 4.2).

The condition for the existence of a safe node set is that each Byzantine node belongs to the core of a control zone with a correct boundary. Then, all the nodes that are not "enclosed" with a Byzantine node are safe.

In Theorem 4.4, we explain how to determine this set. In Theorem 4.5, we show that this set is optimal.

Theorem 4.4 (Determination of a safe node set)**.** *For each Byzantine node b, let $Z(b)$ be the set of control zones $z \in Ctr$ such that:*

- $b \in core(z)$
- *All the nodes of $boundary(z)$ are correct.*

Then, if for each Byzantine node b, $Z(b)$ is not empty, $S = V - \bigcup_{b \in Byz} \bigcap_{z \in Z(b)} core(z)$ is a safe node set.

Proof. The proof is by contradiction. Let us suppose the opposite: let (s, m_1) be a false message (that is, s is correct and $m_1 \neq s.m_0$), and let v be the first correct node such that:

1. $v \in S$
2. v accepts (s, m_1), that is: v is *not* safe.

Obviously, v did not accept (s, m_1) in INIT, as $m_1 \neq s.m_0$. So it was in EXIT. Thus, there exists $(s, m_1, q) \in v.Wait$ verifying the condition of EXIT. And the only way for (s, m_1, q) to have joined $v.Wait$, is that v received (s, m_1) from q in ENTER. Thus, two possibilities:

- Either q is a correct node, and accepted (s, m_1) in EXIT. As v is the first node to verify (1) and (2), it implies that $q \notin S$.
- Or q is a Byzantine node. Then, according to our hypotheses, $q \notin S$.

So, in both cases, we have $q \notin S$. Therefore, $q \in \bigcup_{b \in Byz} \bigcap_{z \in Z(b)} core(z)$. Let b be the Byzantine node such that $q \in \bigcap_{z \in Z(b)}$. As $q \in \bigcap_{z \in Z(b)}$ is neighbor with $v \notin \bigcap_{z \in Z(b)}$, according to Definition 4, there exists $z \in Z(b)$ such that $v \in boundary(z)$.

Then, by definition of $myCtr$ (see 2.3), $z \in v.myCtr$. As (s, m_1, q) verifies the condition of EXIT, $z \in v.myCtr$ implies that $(s, m_1, z) \in v.Auth$.

The only way for (s, m_1, z) to have joined $v.Auth$, is that v received (s, m_1, z) in DIFF from a neighbor in $boundary(z)$. Let u be the first node of $boundary(z)$ to send (s, m_1, z). According to our hypotheses, the nodes of $boundary(z)$ are correct, so u is correct. And

u did not send (s, m_1, z) in DIFF: otherwise, it would not be the first to do so. So it was in EXIT, implying that u accepted (s, m_1). So u verified (1) and (2) before v. This contradiction achieves the proof. □

Theorem 4.5 (Tightness of Theorem 4.4)**.** *The set S of Theorem 4.4 (possibly empty) contains all safe nodes.*

Proof. The proof is by contradiction. Let us suppose the opposite: there exists a safe node v such that $v \notin S$. Let s be a correct node, and let $m_1 \notin s.m_0$. Let us suppose that each Byzantine node b sends the following messages to its neighbors:

- (s, m_1)
- $\forall z \in Ctr$ such that $b \in boundary(z)$: (s, m_1, z)

As v is safe, v *a priori* never accepts (s, m_1).

First, let us suppose that there exists a Byzantine node b such that $Z(b)$ is empty, that is: there exists no control zone $z \in Ctr$ such that $b \in core(z)$ and all the nodes of $boundary(z)$ are correct. Let $(u_0, \ldots, u_n)$ be a path such that $u_0 = b$ and $u_n = v$, and let $k \geq 1$ be the first integer such that u_k never accepts (s, m_1). Therefore:

- Either u_{k-1} is Byzantine, and sends (s, m_1) to uk.
- Or u_{k-1} is correct, and eventually accepts (s, m_1). Therefore, u_{k-1} eventually sends (s, m_1) to u_k in EXIT.

So, in both cases, u_k eventually receives (s, m_1) from u_{k-1}, and adds (s, m_1, u_{k-1}) to $u_k.Wait$. Let $z \in u_k.myCtr$ be a control zone such that $u_{k-1} \in core(z)$ and $s \notin core(z)$.

- First, let us suppose that $b \notin core(z)$.
  - If there exists a Byzantine node b' such that $b' \in boundary(z)$, as $boundary(z)$ is connected, there exists a path P of nodes of $boundary(z)$ connecting b' to u_k. As b' sent (s, m_1, z) to its neighbors, by induction over P, u_k eventually adds (s, m_1, z) to $u_k.Auth$.
  - Otherwise, as $b \notin core(z)$, according to Definition 4, there exists $k' < k$ such that $u_{k'} \in boundary(z)$ and $u_{k'} \neq b$. As $boundary(z)$ is connected, there exists a path P of nodes of $boundary(z)$ connecting $u_{k'}$ to uk. As $u_{k'}$ eventually accepts (s, m_1), $u_{k'}$ eventually sends (s, m_1, z) to its neighbors. Thus, by induction over P, u_k eventually adds (s, m_1, z) to $u_k.Auth$ in DIFF.

So, in both cases, we eventually have (s, m_1, z) in $u_k.Auth$.

- Now, let us suppose that $b \in core(z)$. As $Z(b)$ is empty, there exists a Byzantine node b' in $boundary(z)$. As $boundary(z)$ is connected, there exists a path P of nodes of $boundary(z)$ connecting b' to u_k. As b' sent (s, m_1, z), by induction over P, all the nodes of P eventually add (s, m_1, z) to $Auth$ in DIFF, including u_k.

So, $\forall z \in u_k.myCtr$, we eventually have (s, m_1, z) in $u_k.Auth$. Therefore, u_k eventually accepts (s, m_1) in EXIT. This contradiction achieves the proof.

Now, let us suppose that, for each Byzantine node b, $Z(b)$ is not empty. Then, as $S = V - \bigcup_{b \in Byz} \bigcap_{z \in Z(b)} core(z)$, $v \in \bigcup_{b \in Byz} \bigcap_{z \in Z(b)} core(z)$. Let b be the Byzantine node such that $v \in \bigcap_{z \in Z(b)} core(z)$.

As $\forall z \in Ctr$, $core(z)$ is connected, $\bigcap_{z \in Z(b)} core(z)$ is also connected. Therefore, there exists a path $(u_0, \ldots, u_n)$ of nodes of $\bigcap_{z \in Z(b)} core(z)$ such that $u_0 = b$ and $u_n = v$. Let $k \geq 0$ be the greatest integer such that u_k is Byzantine. $\forall k' \in \{k+1, \ldots, n\}$, according to the definition of $\bigcap_{z \in Z(b)} core(z)$, there is no control zone $z \in u_{k'}.myCtr$ such that $u_{k'-1} \in core(z)$. Therefore, by induction over $P = (u_k, \ldots, u_n)$, all the nodes of P eventually accepts and send (s, m_1), including $u_n = v$. This contradiction achieves the proof.

□

4.2.4 Determination of a communicating node set

Let us give a methodology to determine the optimal *communicating* node set (see Definition 4.3) for a given correct node p.

In the following, we give a condition to construct a communicating node set for p "node by node". Let S be a communicating node set for p, and let v be a correct node. Then, the theorem tells us if $S \cup \{v\}$ is communicating for p, and so forth. To initiate the construction of S, we take $S = \{p\}$.

In Theorem 4.6, we explain how to determine this set. In Theorem 4.7, we show that this set is optimal.

Theorem 4.6 (Construction of a communicating node set)**.** *Let p be a correct node, and let S be a communicating node set for p. Let v be a correct node verifying the following condition:*

1. *v has a neighbor $u \in S$*
2. *$\forall z \in v.myCtr$ such that $u \in core(z)$ and $p \notin core(z)$, there exists a path of correct nodes on $boundary(z)$ connecting v and some node $w \in S$.*

Then, $S \cup \{v\}$ is also communicating for p.

Proof. As S is communicating for p, u eventually accepts $(p, p.m_0)$ in EXIT, and sends it to v. Then, according to ENTER, we eventually have $(p, p.m_0, u) \in v.Wait$. Now, let Z be the set of control zones $z \in v.myCtr$ such that $u \in core(z)$ and $p \notin core(z)$. Let us show that $\forall z \in Z$, we eventually have $(p, p.m_0, z) \in v.Auth$.

Let $z \in Z$. According to (2), there exists a path $(w_0, \ldots, w_n)$ of correct nodes of $boundary(z)$, such that $w_0 \in S$ and $w_n = v$. Let us prove the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1, \ldots, n\}$: we eventually have $(p, p.m_0, z) \in w_i.Auth$.

- First, let us show that $\mathcal{P}_1$ is true. As S is communicating for p, $w_0 \in S$ eventually accepts $(p, p.m_0)$. So, according to EXIT, w_0 sends $(p, p.m_0, z)$ to w_1. If we already have $(p, p.m_0, z) \in w_1.Auth$, $\mathcal{P}_1$ is true. Otherwise, as $w_0 \in boundary(z)$, according to DIFF, $(p, p.m_0, z)$ is added to $w_1.Auth$, and $\mathcal{P}_1$ is also true.
- Let us suppose that $\mathcal{P}_i$ is true, for $i < n$. As $(p, p.m_0, z) \in w_i.Auth$, according to DIFF, w_i sent $(p, p.m_0, z)$ to its neighbors. If we already have $(p, p.m_0, z) \in w_{i+1}.Auth$, $\mathcal{P}_{i+1}$ is true. Otherwise, as $w_{i+1} \in boundary(z)$, according to DIFF, $(p, p.m_0, z)$ is added to $w_{i+1}.Auth$, and $\mathcal{P}_{i+1}$ is also true.

Therefore, $\mathcal{P}_n$ is true, and we eventually have $(p, p.m_0, z) \in v.Auth$.

So we eventually have $(p, p.m_0, u) \in v.Wait$ and, $\forall z \in Z$, $(p, p.m_0, z) \in v.Auth$. Thus, according to EXIT, v eventually accepts $(p, p.m_0)$. Therefore, $S \cup \{v\}$ is communicating for p.

□

Theorem 4.7 (Tightness of Theorem 4.6)**.** *Let p be a correct node, and let S be a communicating node set for p constructed with Theorem 4.6. Let S be any communicating node set for p. Then, $S' \subseteq S$.*

Proof. The proof is by contradiction. Let us suppose the opposite: there exists at least one correct node of S' that does not belong to S.

Let there be an execution where the Byzantine nodes did not send any message, where all the nodes of S have accepted $(p, p.m_0)$, and no other node has accepted $(p, p.m_0)$ yet. Such an execution is possible, as the construction of S with Theorem 3 does not require that any node $x \notin S$ accepts $(p, p.m_0)$. Then, let v be the first node of $S' - S$ accepting $(p, p.m_0)$.

As $v \neq p$, according to EXIT, it implies that there exists a node u such that ...

1. $(p, p.m_0, u) \in v.Wait$
2. $\forall z \in v.myCtr$ such that $u \in core(z)$ and $p \notin core(z)$, $(p, p.m_0, z) \in v.Auth$

According to (1), v received $(p, p.m_0, u)$ from u in ENTER. As no Byzantine node sent any message, u is correct and sent $(p, p.m_0)$, implying that u accepted $(p, p.m_0)$. Thus, as v is the only node of $V - S$ that accepted $(p, p.m_0)$, $u \in S$.

According to (2), v received $(p, p.m_0, z)$. Let $(u_0, \ldots, u_n)$ be a sequence of nodes such that $u_0 = v$, u_i received $(p, p.m_0, z)$ from u_{i+1}, and u_n didn't receive $(p, p.m_0, z)$. As no Byzantine node sent any message, the nodes $\{u_1, \ldots, u_n\}$ are correct. Therefore, they sent $(p, p.m_0, z)$ in DIFF, implying that they belong to $boundary(z)$. Let $w = u_n$. As w is correct and didn't receive $(p, p.m_0, z)$, w sent $(p, p.m_0, z)$ in EXIT, implying that w accepted $(p, p.m_0)$. Thus, according to our hypothesis, $w \in S$. Thus, $(u_0, \ldots, u_n)$ is a path of correct nodes of $boundary(z)$ connecting v and $w \in S$.

Therefore, v has a neighbor $u \in S$ verifiyng the conditions of Theorem 3. So v should belong to S: contradiction. Thus, the result.

□

4.2.5 Determination of a reliable node set

Let us give a methodology to determine the maximum *reliable* node set (see Definition 3.12) for a given correct node p. For this purpose, we simply make the intersection of the two aforementioned *safe* and *communicating* node sets.

In Theorem 4.8, we explain how to determine this set. In Theorem 4.9, we show that this set is optimal. At last, in Theorem 4.10, we show that all the nodes of this set communicate reliably.

Theorem 4.8 (Determination of a reliable node set)**.** *Let p be a correct node. Let S_A be a safe node set (determined with Theorem 4.4). Let S_B be a communicating node set for p (determined with Theorem 4.6). Then, $S = S_A \cap S_B$ is a reliable node set.*

Proof. Let $u \in S$. As $u \in S_A$, u never accepts a false message. As $u \in S_B$, u eventually accepts $(p, p.m_0)$. Then, according to Definition 3.12, S is reliable. □

Theorem 4.9 (Tightness of Theorem 4.8)**.** *Let p be a correct node, and let S be a reliable node set for p constructed with Theorem 4.8. Let S be any reliable node set for p. Then, $S' \subseteq S$.*

Proof. The proof is by contradiction. Let us suppose the opposite: there exists $v \in S'$ such that $v \notin S$. Therefore, two (non-exclusive) possibilities:

- Either $v \notin S_A$. Therefore, as v is safe, S_A is not the optimal safe node set, which contradicts Theorem 2. Thus, the result.
- Or $v \notin S_B$. Therefore, as v eventually accepts $(p, p.m_0)$, S_B is not the optimal communicating node set for p, which contradicts Theorem 4. Thus, the result.

□

Theorem 4.10 (Set of nodes communicating reliably)**.** *Let p be a correct node, and let S be the reliable node set for p constructed with Theorem 4.8. Then, all the nodes of S communicate reliably.*

Proof. As $S = S_A \cap S_B$, let $(p_0, \ldots, p_n)$ be the history of construction of S_B with Theorem 3, with $p_0 = p$. $\forall i \in \{0, \ldots, n\}$, Let $S_i = \{p_0, \ldots, p_i\}$. Let us prove the following property $\mathcal{P}_i$ by induction: $\forall (s, q) \in S_i^2$, q eventually accepts $(s, s.m_0)$.

$\mathcal{P}_0$ is true, as $S_0 = \{p\}$. Now, let us suppose that $\mathcal{P}_i$ is true, $\forall i \in \{1, \ldots, n\}$. To show that $\mathcal{P}_{i+1}$ is true, we have to show that ...

1. $\forall s \in S_i$, p_{i+1} eventually accepts $(s, s.m_0)$.
2. $\forall q \in S_i$, q eventually accepts $(p_{i+1}, p_{i+1}.m_0)$.

The proof of (1) is the same as the proof of Theorem 3, if we replace $(p, p.m_0)$ by $(s, s.m_0)$. Now, let us show that (2) is true. As p_{i+1} verifies the conditions of Theorem 3, let $v = p_{i+1}$, and let u and w be the corresponding nodes of Theorem 3. Let $q \in S_i$, and let Z be the set of control zones $z \in v.myCtr$ such that $u \in core(z)$.

First, let us establish two preliminary results:

- (a) Initially, v sends $(v, v.m_0)$ to u. According to ENTER, $(v, v.m_0, v)$ is added to $u.Wait$. As there is no control zone $z \in Ctr$ such that $v \in core(z)$ and $v \notin core(z)$, the condition of EXIT is directly satisfied. Thus, u eventually accepts $(v, v.m_0)$.
- (b) Let $z \in Z$, and let $x \in S_i \cap boundary(z)$.
  - As $z \in v.myCtr$, v sent $(v, v.m_0, z)$ in INIT.
  - According to (2), there exists a path of correct nodes of $boundary(z)$ connecting v and w.

  So, for the same reason as in Theorem 3, w eventually receives the authorization $(v, v.m_0, z)$. As S_i is a set of correct nodes, $w \in S_i$ and $boundary(z)$ is connected, there also exists a correct path on $boundary(z)$ between w and y. Then, similarly, x eventually receives $(v, v.m_0, z)$.

Now, let there be a configuration in which we have reached the states described in (a) and (b). Then, in such a configuration, $(v, v.m_0)$ becomes indistinguishable from $(u, u.m_0)$. Indeed, the only part of the protocol that could distinguish these messages is the condition of EXIT, for the nodes of $boundary(z)$ with $z \in Z$. But, as we have reached the state described in (b), all these nodes have received the authorizations $(v, v.m_0, z)$, $z \in Z$. So EXIT behaves the same way in both cases. Thus, as q eventually accepts $(u, u.m_0)$, q eventually accepts $(v, v.m_0)$.

Therefore, $\mathcal{P}_{i+1}$ is true. Thus, by induction, $\mathcal{P}_n$ is true: $\forall (s, q) \in S_B^2$, q accepts $(s, s.m_0)$, and, symmetrically, s accepts $(q, q.m_0)$. Therefore, as S_A is safe, according to Definition 9, all the nodes of $S = S_A \cap S_B$ communicate reliably.

□

4.2.6 Message complexity

In addition, let us evaluate the message complexity of our protocol, when the whole network is a reliable node set.

We define the following parameters:

- n, the number of nodes of the network.
- d, the *degree* of the network, that is: the maximal number of neighbors of a node.
- N_{Ctr}, the number of control zones in Ctr.
- N_{Bound}, the maximal number of nodes in the boundary of a control zone.

Let u be any node. Let us evaluate the number of messages related to u:

- All nodes once accept and send $(u, u.m_0)$ to their neighbors, which makes at most dn messages.
- All nodes on the boundary of a control zone z once send $(u, u.m_0, z)$ to their neighbors, which makes at most $dN_{Bound}N_{Ctr}$ messages.

Thus, at most $dn(n + N_{Bound}N_{Ctr})$ messages are sent in the network. Therefore, if we assume that N_{Bound} is bounded and that N_{Ctr} is $O(n)$, the message complexity is $O(n^2)$, the same as an unsecured broadcast protocol.

4.3 Experimental evaluation

In this section, we provide an experimental evaluation of our protocol. We explain our motivations, then describe the setting and the methodology. Next, we comment on the results, and make a comparison with existing solutions.

4.3.1 Motivations

A classical approach to deal with Byzantine failures is to consider the "worst case" placement of a given number of Byzantine nodes. However, with this approach, tolerating more Byzantine failures implies increasing the network connectivity. As we consider sparsely connected networks here, we are more interested in quantitative fault tolerance. Therefore, we assume a random distribution of Byzantine failures. Our metric is the *communication probability*, that is: the probability that two nodes communicate reliably.

In practice, there are many situations where the Byzantine failures occur randomly. First of all, the Byzantine failures can simply model the probability that each node has to misbehave (memory overflow, bit flips, etc ...). Also, in the case of an external attack, the Byzatine adversary does not always choose the position of Byzantine nodes. This is the case, for instance, in a peer-to-peer overlay, where each node joining the network receives a random identifier, and therefore a random position in the virtual topology. Besides, many virus propagation mechanisms use random epidemic schemes to spread across networks.

4.3.2 Setting

We use 100×100 square and hexagonal grids for our evaluation (see Definition 3.9 and 3.10).

In this evaluation, we use *concentric* control zones. A concentric control zone of width n is a tuple $(Core_n, Boundary_n)$ defined as follows:

- $Core_1$ is a single node.
- $Core_2$ is an elementary cycle (here, a square of 4 nodes, or a hexagon of 6 nodes).
- $Boundary_n$ is the smallest cycle that isolates $Core_n$ from the rest of the network.
- $Core_{n+2}$ is the union of $Core_n$ and $Boundary_n$.

An example of such control zones is given in Figure 4.3 (for a square grid) and Figure 4.4 (for a hexagonal grid).

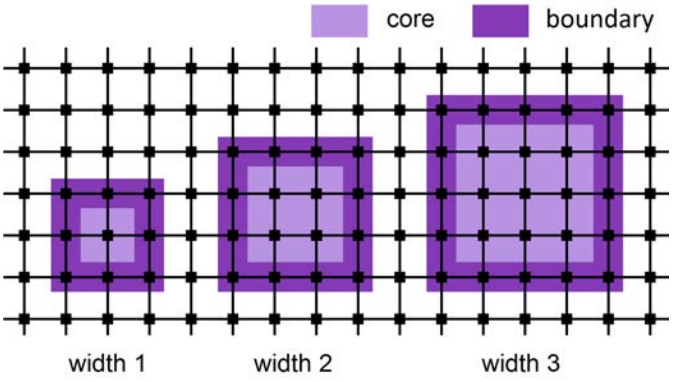

FIGURE 4.3: Example of square control zones

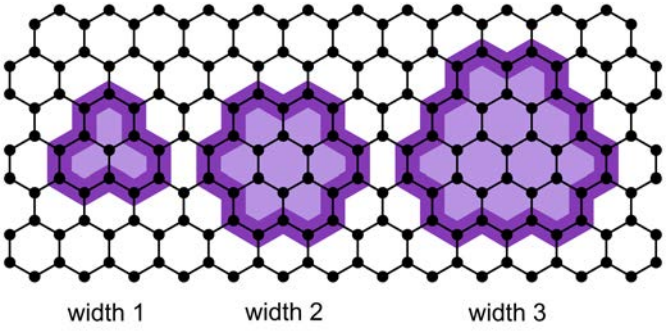

FIGURE 4.4: Example of hexagonal control zones

We say that we run the protocol at *order* W if we use all concentric control zones of width $1, 2, \ldots, W$. This parameter enables us to modulate the number of control zones used by the protocol.

Note that, if the nodes know their position in the grid (their (i, j) identifier) and the order W, they can easily determine to which control zones they belong. Therefore, there is no need to describe each control zone node by node.

4.3.3 Methodology

For a given number n_B of randomly distributed Byzantine failures, we would like to evaluate the probability $P(n_B)$ that two correct nodes communicate reliably. For this purpose, we use a Monte-Carlo method:

- We generate a large number of random placements of n_B Byzantine nodes.
- For each placement, we randomly choose two correct nodes, and check if they communicate reliably. If they do, the simulation is a success.
- On a large number of simulations, the fraction of successes approaches $P(n_B)$.

To check is two nodes communicate reliably, we proceed as follows:

- We randomly choose two correct nodes p and q.
- We use Theorem 4.8 to construct the optimal reliable node set S for p.
- If q belongs to S, according to Theorem 4.10, p and q communicate reliably.

In Figure 4.5, we give a toy example illustrating the construction of a reliable node set on a square grid, for an order $W = 3$ (see 4.3.2). Let us comment on this figure step by step.

1. First, we determine the optimal *safe* node set, using Theorem 4.4. As each Byzantine node belongs to a control zone with a correct boundary, the safe node set is not empty. The blank zones correspond to the intersections of the cores containing Byzantine nodes. Here, having control zones of width 3 is an advantage: if we only had $W = 2$, for instance, the upper-left group of Byzantine nodes would be impossible to neutralize.

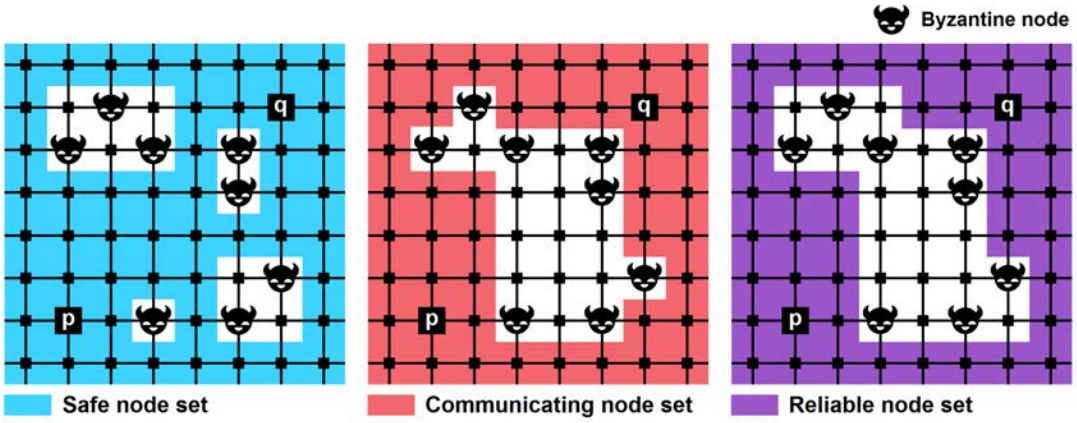

FIGURE 4.5: Example of construction of a safe, communicating and reliable node set

2. Then, we construct the optimal *communicating* node set for p, using Theorem 4.6. We notice that some correct nodes, surrounded by too many Byzantine nodes, cannot be added to to this set. Here, having control zones of width 3 is a drawback: the presence of these zones make the conditions of Theorem 4.6 harder to satisfy, which limits the size of the communicating node set.

3. Finally, we determine the optimal *reliable* node set for p, using Theorem 4.8. According to this theorem, we simply take the intersection of the two aforementioned sets.

4.3.4 Results

We present the simulation results in Figure 4.6 and 4.7. Let us comment on these results.

1. The first plot represents the *safety* probability – that is, the probability of existence of a safe node set (see Theorem 4.4). This probability *increases* with the order. Indeed, when we increase the number of control zones, we increase the probability to satisfy the conditions of Theorem 4.4.

2. The second plot represents the mean size of the reliable node set, when it exists – that is, the fraction of correct nodes covered by the reliable node set. This fraction *decreases* with the order. Indeed, when we increase the number of control zones, we make the conditions of Theorem 4.6 harder to satisfy. Thus, the construction of the communicating node set is made more difficult.

3. At last, the third plot represents the communication probability $P(n_B)$. In the previous plots, we showed that increasing the order of the protocol had a *positive*

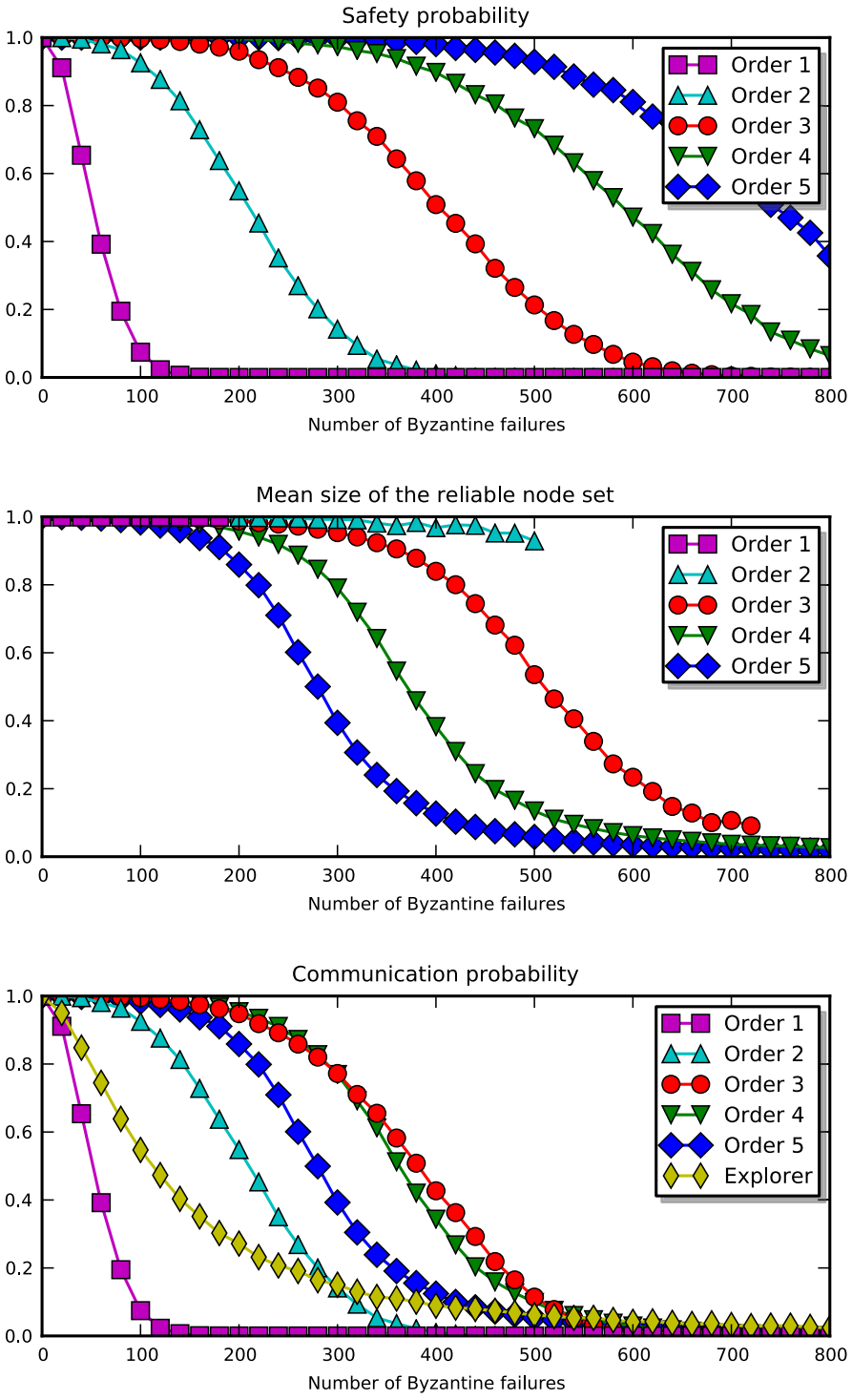

FIGURE 4.6: Simulation results on a 100×100 grid.

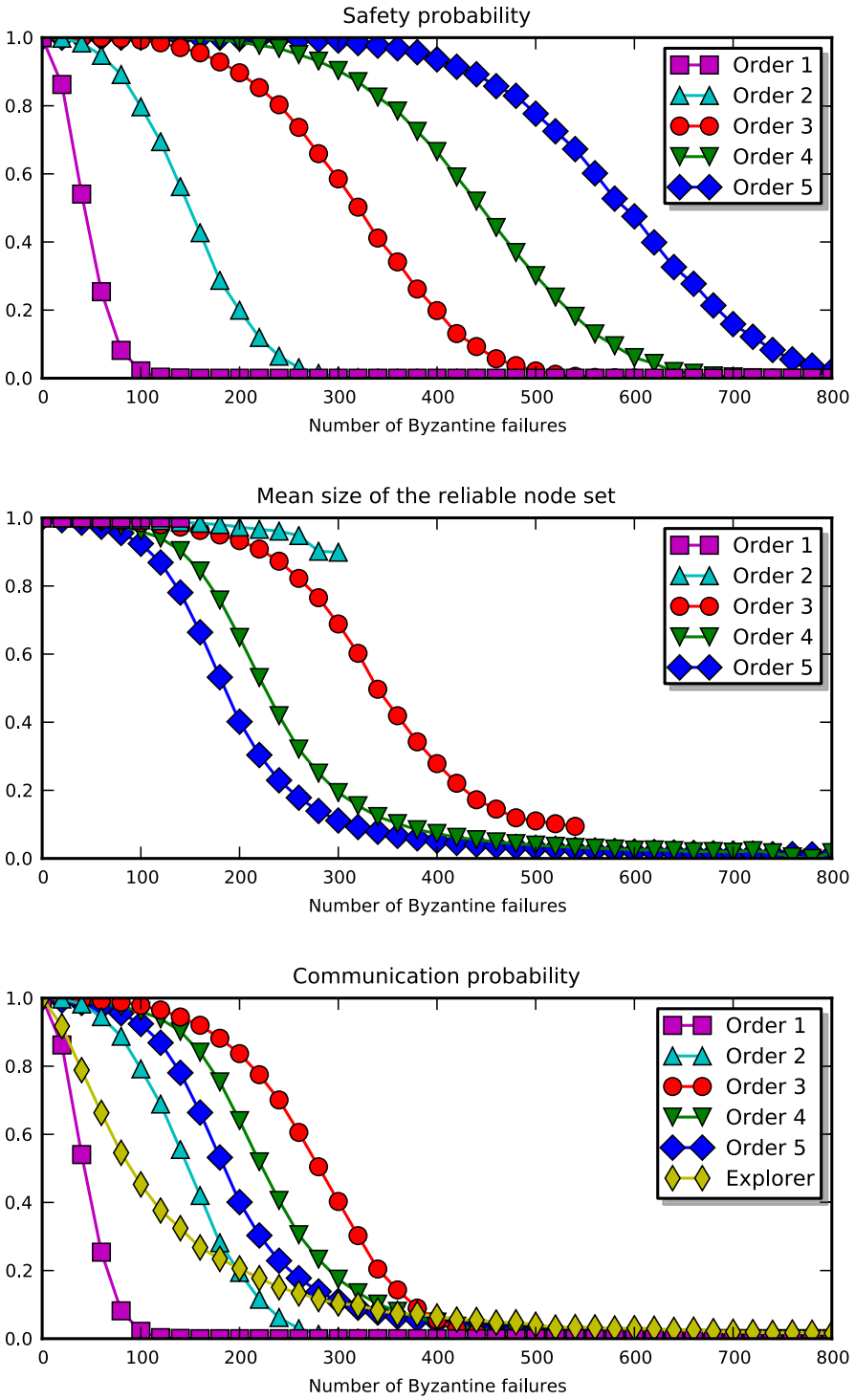

FIGURE 4.7: Simulation results on a 100×100 hexagonal grid.

influence on the existence of a reliable node set, but a *negative* influence on its size. Therefore, a compromise between these two tendancies appears for order 3, for which the communication probability is optimal.

This last point suggests the existence of an optimal number of control zones, for a given topology.

4.3.5 Comparison with existing solutions

The classical Certified Propagation Algorithm [32] does not work in square and hexagonal grid: there are not enough channels to confirm and retransmit a message directly. Among the existing solutions, the only protocol that actually works on square or hexagonal grids is *Explorer* [37]. This solution consists in generating node-disjoint paths between each pair of nodes. Therefore, if, for instance, 1 path among 3 is corrupted by a Byzantine node, this is unimportant, as we still have a strict majority of correct paths.

The protocol given in [37] tolerates at most 1 Byzantine failure in square and hexagonal grids, whatever its position is. However, this does not use the fact that the nodes knows their position in the network. Therefore, in order to have a fair comparison, we used a modified version of *Explorer*, where the messages follow optimal predetermined paths between each pair of nodes. Thus, we now have good probabilities to tolerate *more* than 1 Byzantine nodes. The performances of the modified *Explorer* are represented in the last plot of Figure 4.6 and 4.7.

We thus observe that our protocol outperforms the modified *Explorer* by a significant margin. For example, if a communication probability of 0.99 is required, *Explorer* can tolerate at most 5 (resp. 3) Byzantine failures in a square (resp. hexagonal) grid, while our protocol can tolerate at least 120 (resp. 70) Byzantine failures.

In terms of message complexity, according to the formula given in 4.2.6, our protocol requires 49 times more messages than a simple unsecured broadcast (at order 3 on a square grid). This additional cost remains proportional to the number of nodes, whereas *Explorer* requires a number of messages that grows exponentially with the number of nodes.

4.4 Conclusion

In this chapter, we proposed an algorithm based on predetermined control zones. We gave a theoretical methodology to determine whether or not two nodes communicate

reliably, for a given placement of Byzantine nodes. Using this methodology, we evaluated the performances of our protocol on grids, with a tunable number of control zones. For a communication probability of 0.99, our protocol outperforms previous solutions with a factor 20.

As pointed out in 4.3.4, there seems to exist an optimal number of control zones: too much or two few control zones lead to bad performances. Rigorously determining the optimal set of control zones for a given network topology remains a challenging open problem. Also, we used regular lattices for our evaluation, but an interesting problem would be to define control zones dynamically (for instance, in a network of mobile robots).

Chapter 5

Fixed disjoint paths

In this chapter, we propose an alternative to Chapter 4 in the case where the nodes do not know their position in the network. As the methodology is similar, we recommend to read Chapter 4 first.

In Chapter 4, a certain level of topology knowledge is required to compute control zones. However, this hypothesis is difficult or impossible to satisfy in many types of networks, such as self-organized wireless sensor networks or peer-to-peer overlays.

Here, we assume that the nodes do not know their position, and propose an algorithm using a fixed number of disjoint paths to accept and forward messages. The chapter is organized as follows:

- In Section 5.1, we describe the algorithm.
- In Section 5.2, we explain how to determine a reliable node set.
- In Section 5.3, we evaluate and compare the performances of our protocol with simulations.

A first version of these results was published in the DISC conference [43]. The final version was published in the journal JPDC [44].

5.1 Algorithm

5.1.1 Informal description

Each correct node s wants to broadcast a message $s.m_0$ to the rest of the network.

First, $s.m_0$ is directly accepted and retransmitted by the neighbors of s. Then, to accept a message, the other correct nodes must receive confirmations from several distinct nodes, through a fixed number of disjoint paths. For instance, in Figure 5.1, the right node accepts a message if and only if it is received through 3 disjoint paths of at most $H_1 = 3$ (resp $H_2 = 4$ and $H_3 = 2$) hops. The same requirement stands for every correct node. Once the message is accepted, the node retransmits it for more distant nodes, and the same principle is repeated over and over.

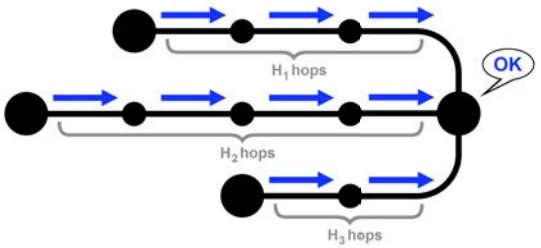

FIGURE 5.1: Principle of the protocol: correct case.

This specific setting of the protocol can be described by the tuple $(H_1, H_2, H_3) = (3, 4, 2)$. More generally, a setting of the protocol is described by a tuple $(H_1, \ldots, H_n)$, each H_i being a positive integer. The integer n (not to confuse with the number of nodes) and the values H_i are assigned arbitrarily: we do not know *a priori* their impact on the global performances, which is studied further in Section 5.3.

The underlying idea is as follows: if the Byzantine nodes are sufficiently spaced, they cannot cooperate to make a correct node accept a false message. Indeed, with setting $(3, 4, 2)$, a correct node can accept the first false message *only* if there exists 3 distinct Byzantine nodes distant of at most 3 (resp. 4 and 2) hops. This critical case is illustrated in Figure 5.2.

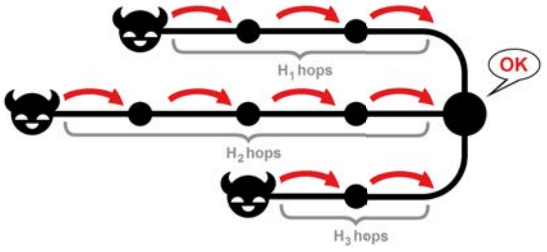

FIGURE 5.2: Principle of the protocol: critical case.

However, if, for instance, the third Byzantine node is located at *more* than 2 hops (*e.g.* 3 hops), the false message is never accepted. This is illustrated in Figure 5.3. This intuitive idea is demonstrated further in Theorem 5.1.

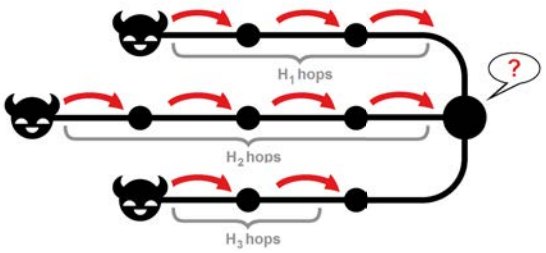

FIGURE 5.3: Principle of the protocol: safe case.

5.1.2 Description of the algorithm

The setting of the protocol is described by a n-tuple of integers $(H_1, \ldots, H_n)$, with $0 \leq H_1 \leq \cdots \leq H_n$, known by all correct nodes. These values, should be considered as an inherent part of the protocol: they are hard-coded with the rest of the algorithm, and are not to be learned by correct nodes. The problem of the choice of the parameters $(H_1, \ldots, H_n)$ is discussed further in Section 5.3. Note that previous solutions [32–34, 36, 37] also have fixed parameters. Let $H = max_{i \in \{1,\ldots,n\}} H_i$.

The correct nodes can send and receive tuples of the form (s, m, Ω), where m is the message broadcast by s (or pretending to be it) and Ω is a set containing the identifiers of nodes already visited by the message. This set is used to certify that the paths are actually disjoint. The Byzantine nodes can, of course, forge and forward any message of the form (s, m, Ω).

Each correct node p holds a dynamic set $p.Rec$, where the tuples (s, m, Ω) received are recorded. Each correct node p initially multicasts $(p, p.m_0, \emptyset)$, then executes the following algorithm:

- When a tuple (s, m, Ω) is received from a neighbor q:
  - If $q = s$:
    - Accept m from s and multicast $(s, m, \emptyset)$.
  - If $q \notin \Omega$ and $card(\Omega) < H$:
    - Add $(s, m, \Omega \cup \{q\})$ to $p.Rec$.
    - Multicast $(s, m, \Omega \cup \{q\})$.
- When there exists s, m and $(\Omega_1, \ldots, \Omega_n)$ such that:
  1. $\forall i \in \{1, \ldots, n\}$, we both have $(s, m, \Omega_i) \in p.Rec$ and $card(\Omega_i) \leq H_i$
  2. $\forall \{i, j\} \subseteq \{1, \ldots, n\}$, $\Omega_i \cap \Omega_j = \emptyset$

  Then, accept m from s and multicast $(s, m, \emptyset)$.

5.2 Reliability properties

In this section, we give a methodology to characterize the pairs of nodes that always communicate reliably, for a given placement of Byzantine nodes. The motivation are the same as in Chapter 4.

In 5.2.1, we give the condition for *safety*, that is: no correct node can accept a false message. In 5.2.2, we give a methodology to characterize which pairs of correct nodes communicate reliably. In 5.2.3, we show the tightness of our conditions. In 5.2.4, we show a linear message complexity.

5.2.1 Safety

We give a condition on the placement of Byzantine nodes to ensure that no correct node ever accepts a false message. The following theorem is the demonstration of the intuitive idea exposed in 5.1.1, and partially shows the correctness of our algorithm. However, this condition is not sufficient to ensure that the correct nodes actually accept the good messages: this aspect is studied further in 5.2.2.

Theorem 5.1 (Safety)**.** *For a given correct node u, let $Critical(u)$ be the following proposition: there exist at least n distinct Byzantine nodes $(b_1, \dots, b_n)$ and n internally disjoint paths $(\mathcal{X}_1, \dots, \mathcal{X}_n)$ such that, $\forall i \in \{1, \dots, n\}$, $\mathcal{X}_i$ is a path of at most H_i hops connecting u and b_i.*

If, for every correct node u, $Critical(u)$ is false, then no correct node ever accepts a false message.

Proof. The proof is by contradiction. Let us suppose the opposite: for every correct node u, $Critical(u)$ is false, yet at least one correct node accepts a false message. Let s be a correct node, and let v be the first correct node to accept a message $m \neq s.m_0$ from s. In the following, we show that $Critical(v)$ is necessarily true, contradicting the previous statement. This contradiction proves the result.

As v is correct and accepts m from s, according to the protocol, there exists $(\Omega_1, \dots, \Omega_n)$ such that, $\forall i \in \{1, \dots, n\}$, $(s, m, \Omega_i) \in v.Rec$ and $card(\Omega_i) \leq H_i$.

Consider now a given index $i \in \{1, \dots, n\}$, and let $q_0 = v$. Let $\mathcal{P}_k^i$ be the following proposition: there exists a path $(q_1, \dots, q_k)$, with $\{q_1, \dots, q_k\} \subseteq \Omega_i$, such that q_{k-1} received $(s, m, \Omega_i - \{q_1, \dots, q_k\})$ from $q_k \in \Omega_i - \{q_1, \dots, q_{k-1}\}$. In our notations, $\{q_1, \dots, q_{k-1}\} = \emptyset$ for $k = 1$.

- First, we show that $\mathcal{P}_1^i$ is true. According to the protocol, the statement $(s, m, \Omega_i) \in v.Rec$ implies that v received $(s, m, \Omega_i - \{q_1\})$ from a node $q_1 \in \Omega_i$. It is actually possible, as $card(\Omega_i - \{q_1\}) \leq H_i - 1 < H$. So $\mathcal{P}_1^i$ is true.

- Now, let us suppose that $\mathcal{P}_k^i$ is true, for $k < card(\Omega_i)$. So q_k sent $(s, m, \Omega_i - \{q_1, \ldots, q_k\})$ to q_{k-1}. Let us suppose that q_k is correct. Then, according to the protocol, it implies that q_k received $(s, m, \Omega_i - \{q_1, \ldots, q_{k+1}\})$ from a node $q_{k+1} \in \Omega_i - \{q_1, \ldots, q_k\}$. It is actually possible, as $card(\Omega_i - \{q_1, \ldots, q_{k+1}\}) \leq H_i - k - 1 < H$. So either $\mathcal{P}_{k+1}^i$ is true or q_k is Byzantine.

Therefore, by induction:

- Either there exists an index $k \in \{1, \ldots, card(\Omega_i) - 1\}$ and a path $(q_1, \ldots, q_k)$ such that q_k is Byzantine. Let $b_i = q_k$, and let $\mathcal{X}_i$ be the path $(v, q_1, \ldots, q_k)$.

- Or $\mathcal{P}_{card(\Omega_i)}^i$ is true, and $q_{card(\Omega_i)}$ sent $(s, m, \Omega_i - \{q_1, \ldots, q_{card(\Omega_i)}\}) = (s, m, \emptyset)$ to $q_{card(\Omega_i - 1)}$. The node $q_{card(\Omega_i)}$ cannot be s, as $m \neq s.m_0$. Also, it cannot be a correct node, as v is the first correct node to accept m from s. So $q_{card(\Omega_i)}$ is necessarily Byzantine. Let $b_i = q_{card(\Omega_i)}$, and let $\mathcal{X}_i$ be the path $(v, q_1, \ldots, q_{card(\Omega_i)})$.

In both cases, the path $\mathcal{X}_i$ connects v and b_i with at most $card(\Omega_i) \leq H_i$ hops.

For $\{i, j\} \subseteq \{1, \ldots, n\}$, the path $\mathcal{X}_i$ and $\mathcal{X}_j$ are internally disjoint if $(\Omega_i - \{b_i\}) \cap (\Omega_j - \{b_j\}) = \emptyset$. As v is correct and accepts m from s, according to the protocol: $\forall \{i, j\} \subseteq \{1, \ldots, n\}, \Omega_i \cap \Omega_j = \emptyset$. Therefore, the paths $(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ are disjoint, and the nodes $(b_1, \ldots, b_n)$ are distinct. Thus, $Critical(v)$ is true. This contradiction achieves the proof.

□

5.2.2 Reliability

Here, we suppose that the condition of Theorem 5.1 is satisfied: no false message can be accepted by a correct node. We now consider a given correct node s, and give a methodology to characterize a set of nodes reliable for s.

Similarly to Chapter 4, the following theorem enables to construct a set of nodes reliable for s step by step: for a given set $\mathcal{R}$ of nodes reliable for s, and a given correct node v, Theorem 5.2 tells us if v is also reliable for s.

Theorem 5.2 (Reliability)**.** *Let s be a correct node, and let us suppose that the condition of Theorem 5.1 is satisfied (for every correct node u, $Critical(u)$ is false). Let $\mathcal{R}$ be a set of nodes reliable for s, and let $v \notin \mathcal{R}$ be a correct node. If there exist at least n distinct nodes $\{r_1, \ldots, r_n\} \subseteq \mathcal{R}$ and n disjoint correct paths $(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ such that, $\forall i \in \{1, \ldots, n\}$, $\mathcal{X}_i$ is a path of at most H_i hops connecting v and r_i, then v is also reliable for s.*

Proof. According to Theorem 5.1, no correct node can accept a false message. So, if a correct node accepts a message, this is necessarily a correct message.

We consider a given index $i \in \{1, \ldots, n\}$. Let $\mathcal{X}_i = (q_0, \ldots, q_M)$, with $q_0 = r_i$ and $q_M = v$. By definition, we have $M \leq H_i \leq H$. Let us prove the following property $\mathcal{P}_k^i$ by induction, $\forall k \in \{1, \ldots, M\}$: the node q_k eventually receives $(s, s.m_0, \{q_0, \ldots, q_{k-2}\})$ from q_{k-1}. In our notations, $\{q_0, \ldots, q_{k-2}\} = \emptyset$ for $k = 1$.

- First, we show that $\mathcal{P}_1^i$ is true. As $\mathcal{R}$ is set of nodes reliable for s, the node $r_i \in \mathcal{R}$ eventually accepts $s.m_0$ from s. According to the protocol, it implies that r_i also multicasts $(s, s.m_0, \emptyset)$. So q_1 eventually receives $(s, s.m_0, \emptyset)$ from $q_0 = r_i$, and $\mathcal{P}_1^i$ is true.
- Now, let us suppose that $\mathcal{P}_k^i$ is true for $k \leq M$. As $q_{k-1} \notin \{q_0, \ldots, q_{k-2}\}$, and $card(\{q_0, \ldots, q_{k-2}\}) < M \leq H$, q_k eventually multicasts $\{q_0, \ldots, q_{k-1}\}$. So q_{k+1} eventually receives $\{q_0, \ldots, q_{k-1}\}$ from q_k, and $\mathcal{P}_{k+1}^i$ is true.

So $\mathcal{P}_M^i$ is true and the node $q_M = v$ eventually receives $(s, s.m_0, \{q_0, \ldots, q_{M-2}\})$ from q_{M-1}. As $q_{M-1} \notin \{q_0, \ldots, q_{M-2}\}$ and $card(\{q_0, \ldots, q_{M-2}\}) < M \leq H$, v eventually adds $(s, s.m_0, \{q_0, \ldots, q_{M-1}\})$ to the set $v.Rec$. Let $\Omega_i = \{q_0, \ldots, q_{M-1}\}$.

So, $\forall i \in \{1, \ldots, n\}$, we have $(s, s.m_0, \Omega_i) \in v.Rec$ and $card(\Omega_i) < H_i$. Besides, as the paths $(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ are disjoint, $\forall \{i, j\} \subseteq \{1, \ldots, n\}$, we have $(\Omega_i - \{r_i\}) \cap (\Omega_j - \{r_j\}) = \emptyset$. Thus, as the nodes $(r_1, \ldots, r_n)$ are distinct, we have $\Omega_i \cap \Omega_j = \emptyset$. Therefore, according to the protocol, v eventually accepts $s.m_0$. Thus, v is reliable for s.

□

5.2.3 Bounds tightness

We now show that the condition for safety (Theorem 5.1) is tight, and that the methodology to characterize the reliable nodes (Theorem 5.2) is optimal in a safe network.

Theorem 5.3 (Bounds tightness for Theorem 5.1)**.** *If the condition of Theorem 5.1 is not satisfied, it is impossible to guarantee that the network is safe.*

Proof. Let us suppose the opposite: the condition of Theorem 5.1 is not satisfied, yet the network is safe, that is: no correct node can accept a false message. As the condition of Theorem 5.1 is not satisfied, there exists at least one correct node u such that $Critical(u)$ is true. In other words, there exist at least n distinct Byzantine nodes $(b_1, \ldots, b_n)$ and n disjoint paths $(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ such that, $\forall i \in \{1, \ldots, n\}$, $\mathcal{X}_i$ is a path of at most H_i hops connecting u and b_i.

Thus, it is possible that the Byzantine nodes $(b_1, \ldots, b_n)$ unanimously multicast $(s, m', \emptyset)$, with $m' \neq m$. If so, with a reasoning similar to the proof of Theorem 5.2, we show that u eventually accepts m' from s. Therefore, the network cannot be safe. This contradiction completes the proof.

□

Theorem 5.4 (Bounds tightness for Theorem 5.2)**.** *Suppose that the network is safe, and let s be a correct node. Then, the set constructed with Theorem 5.2 contains* all *the nodes reliable for s.*

Proof. Let us suppose the opposite: the set $\mathcal{R}$ constructed with Theorem 5.2 does not contain all the nodes reliable for z. Let $\mathcal{R}'$ be the set of nodes reliable for s.

Let there be an execution where all the nodes of $\mathcal{R}$ accept $s.m_0$ from s, but no other correct node accepted $s.m_0$ from s so far. Such an execution is possible, as the construction of $\mathcal{R}$ with Theorem 5.2 does not require that any node $u \notin \mathcal{R}$ accepts $s.m_0$ from s.

Let v be the first node of $\mathcal{R}' - \mathcal{R}$ to accept $s.m_0$ from s in the following of the execution. Then, by a reasoning similar to the proof of Theorem 5.1, we show that there must exist at least n distinct nodes $(u_1, \ldots, u_n)$ that have previously accepted $s.m_0$ from s, and n disjoint correct paths $(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ such that, $\forall i \in \{1, \ldots, n\}$, $\mathcal{X}_i$ is a path of at most H_i hops connecting v and r_i.

As the only correct nodes that have previously accepted $s.m_0$ from s are the nodes of $\mathcal{R}$, we have $\{u_1, \ldots, u_n\} \subseteq \mathcal{R}$, and the condition of Theorem 5.2 is satisfied for $\mathcal{R}$ and v. So v could actually be added to $\mathcal{R}$, and $\mathcal{R}$ is not the largest set of nodes reliable for s that can be constructed with Theorem 5.2. This contradiction completes the proof. □

5.2.4 Message complexity

We now evaluate the message complexity of our protocol – that is, the number of tuples (s, m, Ω) sent by the correct nodes. We only consider the case where all nodes are correct, as the Byzantine nodes can send as many messages as they want.

Let $|V|$ be the number of nodes, and let Δ be the maximal degree of the network – that is, the maximal number of neighbors for a single node. Let s be a correct node. According to our protocol, when a node v accepts $s.m_0$ from s, it sends $(s, s.m_0, \emptyset)$ to each neighbor, which makes at most Δ messages. Then, each neighbor of v multicasts $(s, s.m_0, \{v\})$, which makes at most $\Delta + \Delta^2$ messages. This process is repeated H times, which makes at most $\Delta + \Delta^2 + \cdots + \Delta^H = O(\Delta^H)$ messages. So, $O(\Delta^H |V|)$ messages related to s are sent by the protocol, which makes $O(\Delta^{2H}|V|^2)$ messages in total.

Therefore, if we consider that the degree Δ and protocol parameter H are bounded (for instance, in a torus network, where $\Delta = 4$), the message complexity is $O(|V|^2)$: the same as an unsecured broadcast protocol.

5.3 Evaluation of the protocol

In this section, we perform simulations to compare the performance of different settings of our protocol. We also provide a quantitative comparison with previous solutions, and show the improvement.

5.3.1 Settings

Let us choose a reasonable set of settings $(H_1, \ldots, H_n)$ for our simulations, and explain this choice.

First, we exclude the case $n = 1$, which corresponds to an unsecured broadcast (see 5.1.1). Besides, any setting with $n > 4$ would not work: n disjoint paths are required, and a node has only 4 neighbors. Therefore, we restrain ourselves to $n \in \{2, 3, 4\}$.

Then, we introduce the notion of *minimal setting*.

Definition 5.5 (Minimal setting)**.** For a given network, and for $n \geq 1$, a setting $(H_1, \ldots, H_n)$ is:

Smaller than a setting $(H'_1, \ldots, H'_n)$ if, $\forall i \in \{1, \ldots, n\}$, $H_i \leq H'_i$, and there exists $j \in \{1, \ldots, n\}$ such that $H_j < H'_j$.

Covering if, when there are no Byzantine nodes, all nodes are reliable.

Minimal if no smaller setting is covering.

We therefore choose the minimal settings for a torus network, for $n \in \{2, 3, 4\}$:

- Setting A: $(1, 2)$
- Setting B: $(1, 2, 5)$
- Setting C: $(1, 3, 3)$
- Setting D: $(1, 2, 5, 5)$

This choice is motivated by the two following reasons:

1. Any *smaller* setting is not covering, and therefore does not enable reliable broadcast.
2. Any *greater* setting requires longer paths. This would only increase the probability to have Byzantine nodes on the paths, and reduce the safety probability without any compensation.

Of course, we do not claim that this preliminary choice is optimal. Rigorously determining the optimal settings remains a challenging open problem.

5.3.2 Results

Let λ be the *Byzantine rate*, that is: the probability for a node to be Byzantine.

The simulation results are presented in Figures 5.4 and 5.5. Figure 5.4 corresponds to a 10×10 torus, and Figure 5.5 corresponds to a 50×50 torus. For each case, we represented the probability that the network is safe (no correct node can accept a false message), then the communication probability, varying λ. Note that the scale is different for Figure 5.4 and Figure 5.5.

First, let us comment on the probability that the network is safe, according to Theorem 5.1.

- This probability *increases* with the complexity of the setting (2 paths for setting A, 3 paths for settings B and C, 4 paths for setting D). Indeed, as we use more paths, it becomes less likely to have a critical placement of Byzantine nodes (see Figure 5.2).

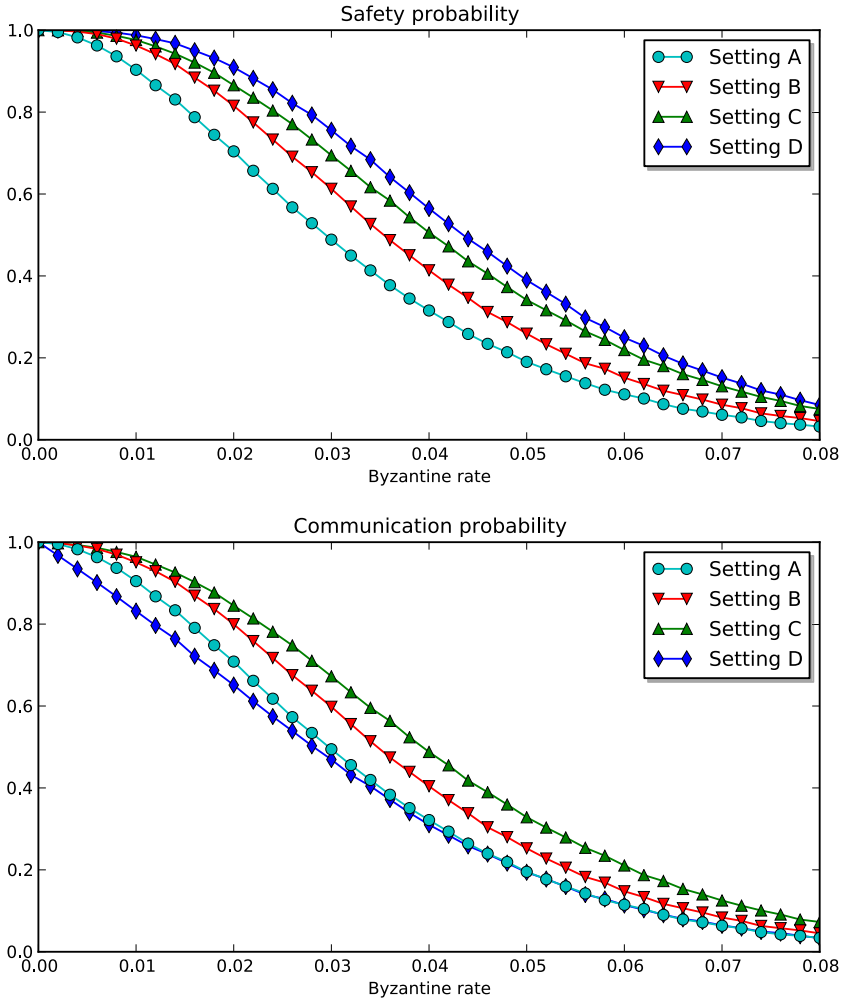

FIGURE 5.4: Simulation results on a 10×10 torus.

- This probability *decreases* with the size of the network. Indeed, for a given Byzantine rate, the frequency of critical placements increases with the size of the network. Besides, the disparities between the different settings also increase with the size: the results on Figure 5.5 are more dispersed than on Figure 5.4.

Now, let us comment on the communication probability.

- There seems to exist an optimal setting of the protocol. Indeed, increasing the number of paths makes the safety conditions of Theorem 5.1 easier to satisfy, but the reliability conditions of Theorem 5.2 harder to satisfy. Therefore, there is a compromise to find. This is illustrated in Figure 5.4, where setting D (the most complex setting) offer the best safety probability, but also the worst communication probability.

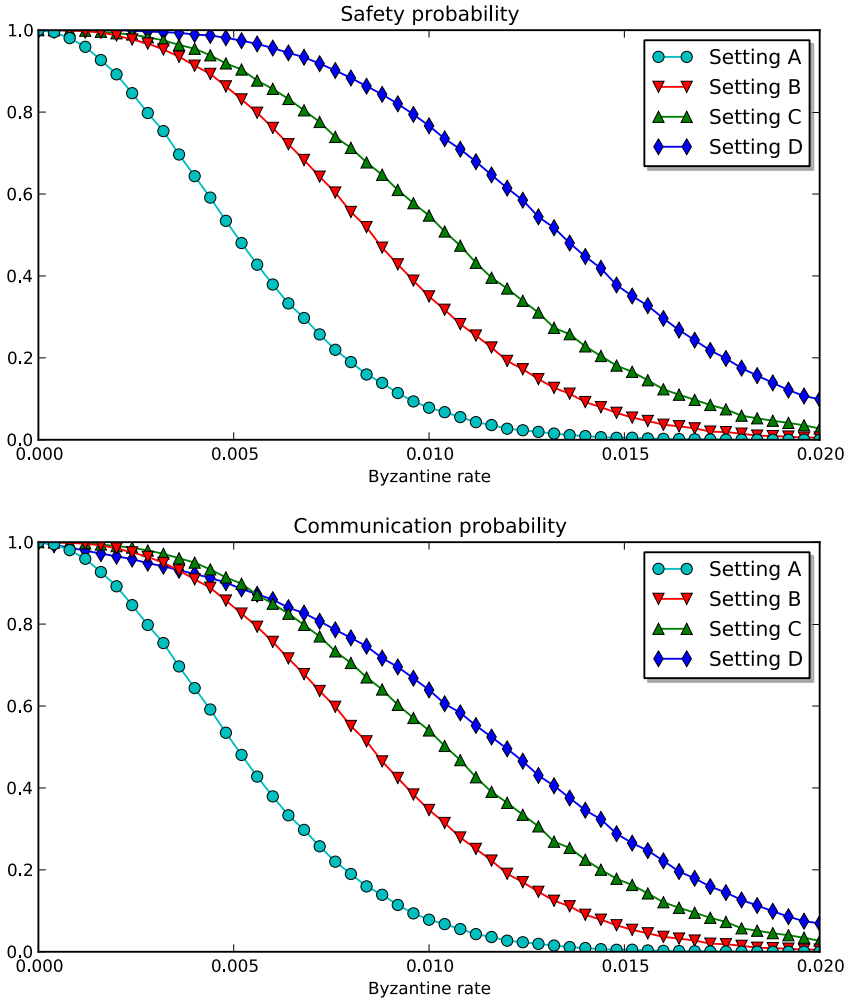

FIGURE 5.5: Simulation results on a 50×50 torus.

- Besides, the size of the network impacts this optimum. Indeed, on in Figure 5.5, setting D now offers the best communication probability for the Byzantine rates $\lambda > 0.005$. However, this is not the case for smaller Byzantine rates.

Therefore, setting C offers the best performances in both networks. We use this setting for the comparison with previous solutions.

5.3.3 Comparison with previous solutions

Finally, we provide a quantitative comparison with previous solutions.

According to our hypotheses, the nodes are not aware of their position on the network. Therefore, our protocol must be compared with other protocols that still work despite this constraint and the low degree of the network. This is only the case of protocol [37].

Let us suppose that we want to guarantee a communication probability $P(\lambda)$ of at least 0.99 on a 50×50 torus. Then, we can tolerate a Byzantine rate λ of:
– 4×10^{-6} with an unsecured broadcast
– 5×10^{-5} with protocol [37]
– 2×10^{-3} with our protocol (improvement of factor 40)

These performances are to be compared with the protocol of Chapter 4 (Control Zones), which only works when the nodes know their position in the network. With this protocol, we can tolerate a Byzantine rate of 8×10^{-3}. As we can see, assuming that the nodes do not know their position still has a cost in term of performances, and filling this gap remain a challenging open problem.

All these solutions are represented in Figure 5.6, on a logarithmic scale.

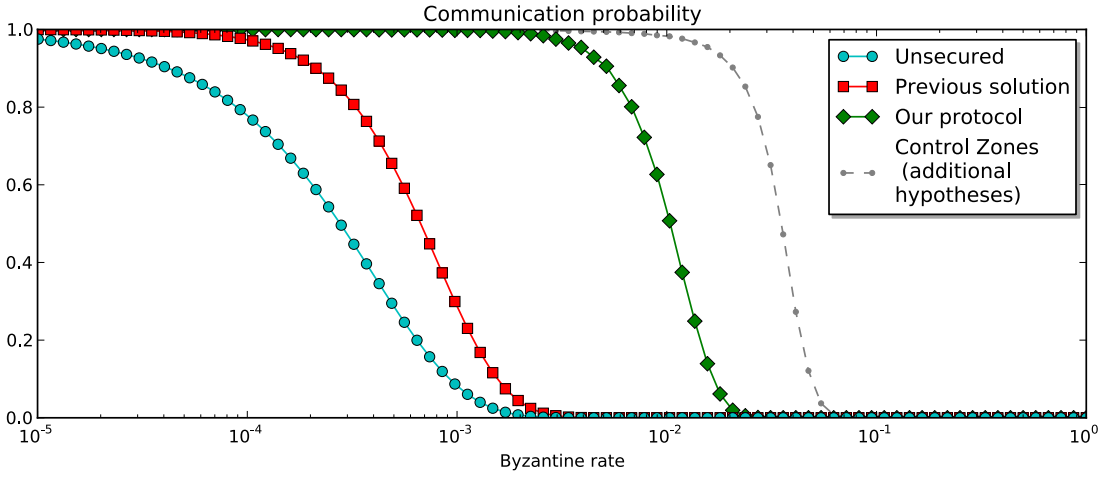

FIGURE 5.6: Comparison of existing protocols on a logarithmic scale (50×50 torus).

5.3.4 Application to other topologies: the hexagonal torus

In this section, we discuss the possibility to apply our scheme to other topologies. As an example, we consider the case of the the hexagonal torus. In this topology, each node has 3 neighbors instead of 4.

The minimal settings on this topology (following the methodology used in previous sections) are the following:

- Setting E: $(1, 3)$
- Setting F: $(2, 2)$

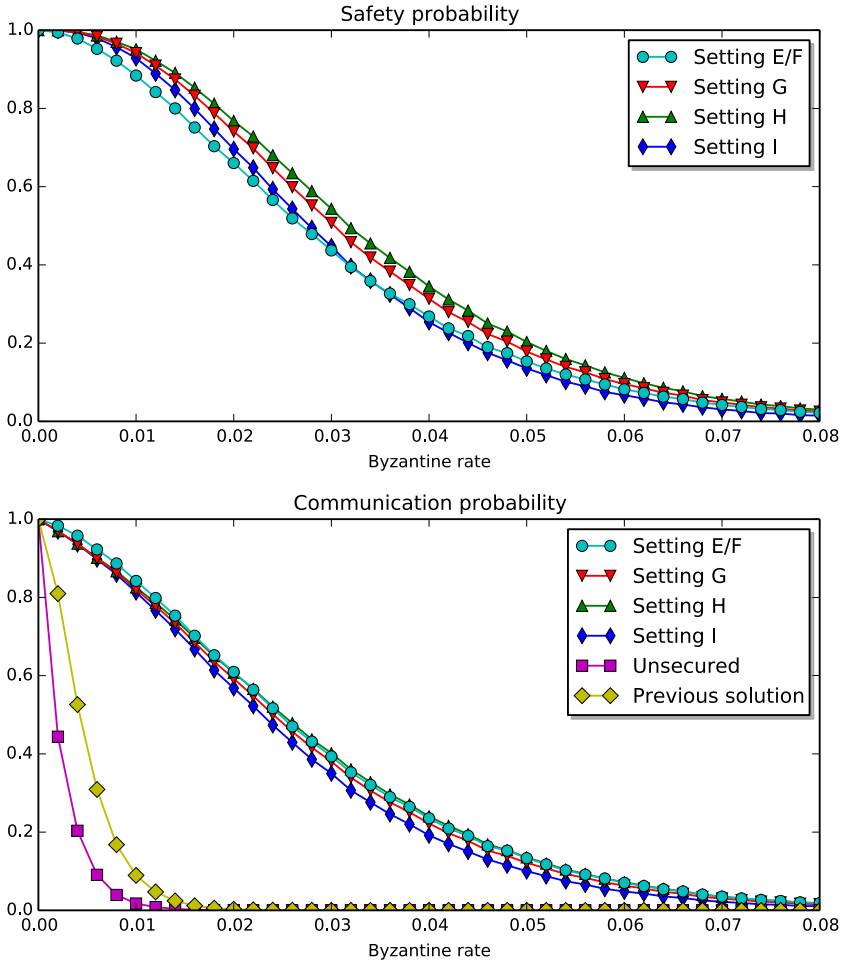

FIGURE 5.7: Simulations results on a 10×10 hexagonal torus

- Setting G: $(1, 3, 7)$
- Setting H: $(2, 2, 10)$
- Setting I: $(2, 6, 6)$

We performed simulations on a 10×10 hexagonal torus. The results are represented on Figure 5.7. Setting E and F give exactly the same results, thus we represented it on the same plot.

We notice that the setting has very few influence here. As previously, the settings with 3 parameters ensure a better safety than those with only 2 parameters. Yet, the settings with 2 parameters ensure a better communication probability. Indeed, when the setting has 3 parameters, the nodes with Byzantine neighbors cannot accept the correct message, as they only have 2 correct neighbors.

We also plotted the performances of both unsecured broadcast and solution [37]. To achieve a communication probability of 0.99, our protocol can tolerate a 1.2×10^{-3} Byzantine rate on a 10×10 hexagonal torus (versus 5×10^{-3} on a 10×10 torus). Thus, the removal of channels seems to have a negative impact on the performances. However, we still tolerate a Byzantine rate 10 times higher than solution [37].

5.4 Conclusion

In this chapter, we proposed a paremeterizable approach for Byzantine-resilient broadcast in sparsely connected networks, with a minimal number of hypotheses. We showed that, by properly tuning the parameters of the protocol, we can optimize the fault-tolerance in the presence of randomly distributed Byzantine failures, and outperform previous solutions in the same setting. However, the absence of position knowledge still has a strong impact on Byzantine tolerance.

To go further, it would be interesting to experiment this approach on less regular networks, such as sensor networks, robot networks or peer-to-peer overlays, where the lack of global positioning is a common assumption. Also, a motivating open challenge is to obtain theoretical probabilistic guarantees with global network parameters (diameter, node degree, connectivity...) in order to automatically compute the optimal parameter settings.

Part II

Qualitative Byzantine tolerance

Chapter 6

Cycle decomposition

In this chapter, we propose algorithms that ensure perfect reliable communication: all pairs of correct nodes communicate reliably.

Existing algorithms use conditions on the *number* [36, 37] or the *density* [32–35] of Byzantine failures. This density was represented by the maximal number of Byzantine failures per correct node. In sparse networks however, this criteria is too strong, as we have seen in Chapter 2. Therefore, we consider a more relaxed density criteria: the minimal *distance* between two Byzantine failures.

Here is the content of the chapter:

- We first show than any 3-connected network admits a particular cycle decomposition. For instance, a grid can be decomposed in elementary square cycles, and a planar graph in elementary polygons.
- Then, we give an algorithm that ensures perfect reliable communication when the distance between two Byzantine failures is more than twice the diameter of the largest cycle.
- At last, we give a self-stabilizing version of the previous algorithm: the nodes retrieve correct outputs even if the initial state of their variables is corrupt and totally arbitrary. Thus, by combining tolerance to both transient and permanent failures, we achieve one of the strongest possible level of fault tolerance.

The chapter is organized as follows. In Section 6.1, we describe the cycle decomposition of the network. In Section 6.2, we describe the problems and present algorithms to solve them. In Section 6.3, we show some topology properties. At last, in Sections 6.4 and 6.5, we prove the correctness of our algorithms.

These results were published in the SRDS conference [45]. Some preliminary elements were also published in the conferences AlgoTel [46] and DISC [43].

6.1 Cycle decomposition

Let us define the notion of *Z-resilient* network (see Definition 6.4). For this purpose, we first give some definitions.

Definition 6.1 (Cycle)**.** A set of nodes C is a *cycle* if there exists a path $(u_1, \ldots, u_n)$ containing all the nodes of C, such that u_1 and u_n are also neighbors. A cycle is *correct* if all its nodes are correct.

Definition 6.2 (Connected set of cycles)**.** An arbitrary set of cycles S of the network is *connected* if, $\forall \{C, C'\} \subseteq S$, there exists a sequence $(C_1, \ldots, C_n)$ of cycles of S such that $C_1 = C$, $C_n = C'$ and, $\forall i \in \{1, \ldots, n-1\}$, C_i and C_{i+1} have at least two nodes in common.

To illustrate, the grey sets of cycles of Figure 6.1 and 6.2 are connected in the sense of Definition 6.2.

Definition 6.3 (Resilient decomposition)**.** An arbitrary set of cycles S of the network is a *resilient decomposition* if, for each pair of nodes p and q, there exists a connected set $S(p, q) \subseteq S$ of at most Δ cycles such that:

1. Each cycle of $S(p, q)$ contains p and not q.
2. Each neighbor of p (distinct from q) belongs to a cycle of $S(p, q)$.

An example of such sets $S(p, q)$ is given in Figure 6.1 and Figure 6.2.

Definition 6.4 (Z-resilient network)**.** A network is *Z-resilient* if there exists an arbitrary set of cycles S of the network such that S is a resilient decomposition, and the diameter of the cycles of S is at most Z.

For instance, in a torus network (see Figure 6.1), the set of square cycles is a resilient decomposition. Thus, a torus is a 2-resilient network. Similarly, in a planar graph network (see Figure 6.2), the set of polygonal cycles is a resilient decomposition. As the diameter of the largest cycle is 2, this network is also 2-resilient.

More generally, in Theorem 6.6 (see Section 6.3), we show that any 3-connected network admits a resilient decomposition.

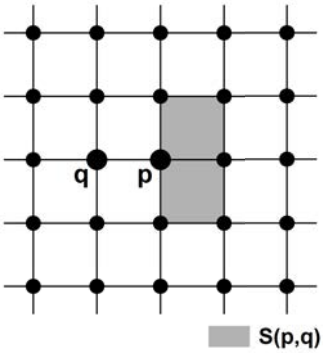

FIGURE 6.1: Example of set of cycles $S(p, q)$ on a torus. Here, the cycles are the elementary squares.

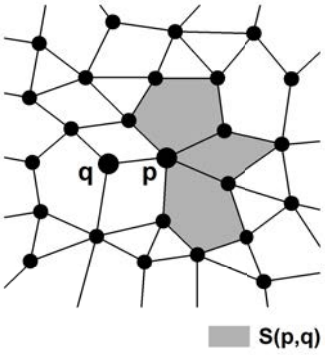

FIGURE 6.2: Example of set of cycles $S(p, q)$ on a planar graph. Here, the cycles are the elementary polygons.

In the following, we are interested in Z-resilient networks where Z is small compared to the network diameter. This is the case of networks where the nodes are homogeneously distributed in a bidimensional or tridimensional space, and are neighbors with the closest nodes – for instance, a network of sensors or mobile robots.

6.2 Algorithms

In this section, we describe our algorithm for reliable communication. Then, after introducing the problem of *self-stabilizing reliable communication*, we describe our second algorithm.

6.2.1 Hypotheses

We consider a Z-resilient network, and we assume that the minimal distance between two Byzantine nodes is strictly greater than $2Z$. Let D be the diameter of the network.

We assume that the time between two activations of a same node has an upper bound T_1. Similarly, the time for a message to cross a communication channel has an upper bound T_2. Let $T = max(T_1, T_2)$. Note that this bound T is unknown to the correct nodes, and that our algorithms do not use any time primitive. This bound is only used to evaluate the global broadcast time.

6.2.2 Algorithm 1: reliable communication

Let us present an algorithm that ensures reliable communication in at most $8D\Delta^2ZT$ time units. The correctness proof of this algorithm is given in Theorem 6.13 (see Section 6.4).

Informal description

Each correct node p initially multicasts $p.m_0$, and its correct neighbors accept $p.m_0$ from p. Then, the nodes send tuples (p, m, X), where p is the supposed initiator of the message m, and X is a set recording the nodes visited by this message. This message can visit at most Z nodes.

When a correct node has received (p, m, X) and (p, m, X') through two disjoint set of nodes X and X', it accepts m from p and multicasts $(p, m, \emptyset)$. Then, the same mechanism is repeated until each correct node accepts m from p.

The underlying idea is that two Byzantine nodes can never cooperate to make a correct node accept a false message, as a message can cross at most Z hops, and the distance between Byzantine nodes is more than $2Z$.

Each correct node p has a memory set $p.Rec$, initially empty, to register the messages received. We say that a node *multicasts* a message when it sends this message to all its neighbors.

Description of Algorithm 1

When p is activated, it executes the following algorithm:

1. If p is activated for the first time: multicast $p.m_0$.
2. If m is received from a neighbor q: accept m, multicast $(q, m, \emptyset)$ and never accept another message from q.
3. If (s, m, X) is received from a neighbor q, with $q \notin X$ and $|X| < Z$: add $(s, m, X \cup \{q\})$ to $p.Rec$ and multicast it.

4. When there exists s, m, X and X' such that $X \cap X' = \emptyset$, $(s, m, X) \in p.Rec$ and $(s, m, X') \in p.Rec$: accept m from s, multicast $(s, m, \emptyset)$ and never accept another message from s.

6.2.3 Self-stabilizing reliable communication

We now consider a stronger version of the previous problem: in addition of tolerating permanent Byzantine nodes, we must also recover from *any* arbitrary initial state (that *e.g.* results from every node being transiently Byzantine). In other words, we now assume that the local memories of nodes and channels can initially contain any element.

More precisely, at time $t = 0$:

- For each correct node, any local memory used in our algorithm can contain *any* element.
- Each communication channel $\{p, q\}$ can contain *any* message sent by p, but not yet received by q.

This arbitrary initial state can represent the temporary violation of the density requirement of permanent Byzantine nodes. We present an algorithm that ensures reliable communication despite any arbitrary initial state in 6.2.4.

6.2.4 Algorithm 2: self-stabilizing reliable communication

Let us present an algorithm that ensures reliable communication despite any arbitrary initial state in at most $12D\Delta^2ZT$ time units. The correctness proof of this algorithm is given in Theorem 6.20 (see Section 6.5).

The interest of Algorithm 1 compared to Algorithm 2 is that it uses shorter messages. Indeed, Algorithm 1 requires the messages to register up to Z node identifiers. Algorithm 2 requires to register up to $K = 4D\Delta^2Z$ node identifiers.

Besides, Algorithm 2 does not terminate: indeed, if it was the case, the initial configuration could be a wrong terminal configuration, and we would not have the self-stabilization property. However, the transient faults only cause a finite number of wrong messages.

In particular, as a torus is a 2-resilient network (see Section 6.1), this algorithm ensures reliable communication in a torus network despite any arbitrary initial state, provided that the distance between Byzantine failures is greater than 4.

Informal description

The previous algorithm cannot work if we assume an arbitrary initial state. Indeed, in the initial state, the broadcast of false messages can already be initiated, and it becomes impossible to distinguish authentic messages from false messages.

We thus adopt a different strategy. First, we remove the limit of Z nodes identifiers. Therefore, when a message broadcast by a correct node p reaches a correct node q, all the nodes visited by the message are registered.

The node q waits until it receives a same message from p through several different paths. Then, it checks if the fusion of these paths can reconstitute a sequence of cycles $(C_1, \ldots, C_n)$ such as described in Definition 6.2. We call this a (p, q)-*valid* set of sequences (see Definition 6.5). If yes, q accepts the message from p.

The interest of this mechanism is the following: as the distance between Byzantine nodes is more than $2Z$, a (p, q)-valid set contains at least one correct path connecting p to q, which ensures that the message is correct. Besides, the fake sequences of node identifiers resulting from the arbitrary initial state are eliminated after a certain time.

Each correct node p has a memory set $p.Rec$, but this set can now initially contain any element. Besides, a correct node q can already have accepted any message from a node p. Later, q can accept another message from p, which replaces the previous one, and so forth.

Definitions

Let $K = 4D\Delta^2 Z$.

Definition 6.5 ((p, q)-valid set of sequences)**.** Let Ω be a set of sequences $(u_1, \ldots, u_n)$ of node identifiers. Let $G(\Omega)$ be the graph (V, E) such that:

- V is the set of node identifiers of the sequences of Ω.
- E is the set of pairs of node identifiers $\{p, q\}$ such that there exists $(u_1, \ldots, u_n) \in \Omega$ and $i \in \{1, \ldots, n-1\}$ such that $p = u_i$ and $q = u_{i+1}$.

We say that Ω is (p, q)-*valid* if:

- $\forall (u_1, \ldots, u_n) \in \Omega$, $u_1 = p$ and $u_n = q$.
- $G(\Omega)$ can be decomposed in a sequence $(C_1, \ldots, C_m)$ of cycles of diameter at most Z, such that $p \in C_1$, $q \in C_m$ and $\forall i \in \{1, \ldots, m-1\}$, C_i and C_{i+1} have at least two elements in common.

- For each path $(u_1, \ldots, u_n)$ of $G(\Omega)$ such that $u_1 = p$, $u_n = q$ and $n \leq K$, $(u_1, \ldots, u_n) \in \Omega$.

Description of Algorithm 2

When a correct node p is activated, it executes the following algorithm:

1. Multicast $(p, p.m_0, (p))$.
2. If $(s, m, (u_1, \ldots, u_n))$ is received from a neighbor q, with $u_n = q$ and $n \leq K$: add $(s, m, (u_1, \ldots, u_n, p))$ to $p.Rec$ and multicast it.
3. When there exists s, m and a (s, p)-valid set Ω such that, $\forall X \in \Omega$, $(s, m, X) \in p.Rec$: accept m from s, and $\forall (m', X')$ such that $(s, m', X') \in p.Rec$, remove (s, m', X') from $p.Rec$.

6.3 Topology properties

In this section, we prove the results that only concern the communication graph, independently of the algorithms.

In 6.3.1, we show that any 3-connected graph admits a resilient decomposition, as claimed in Section 6.1. In 6.3.2, we show that the set of correct cycles is connected, which is a key property fort the correctness proofs of our algorithms.

6.3.1 Resilient decomposition

Theorem 6.6. *If the network is 3-connected, there exists a resilient decomposition.*

Proof. Let us suppose that the communication graph G is 3-connected.

Let p and q be two nodes. Let u and v be two neighbors of p. Let G' be the graph obtained by removing the nodes p and q from G. As G is 3-connected, at least 3 nodes must be removed to disconnect the graph. Thus, G' is connected. Let $(u_1, \ldots, u_n)$ be the shortest path in G' such that $u_1 = u$ and $u_n = v$. Let $C(u, v) = \{p, u_1, \ldots, u_n\}$. According to Definition 6.1, $C(u, v)$ is a cycle in G.

As G is 3-connected, p has at least 2 neighbors distinct from q. Let $(v_1, \ldots, v_m)$ be a sequence containing all the neighbors of p distinct from q, with $m \leq \Delta$. $\forall i \in \{1, \ldots, m-1\}$, let $C_i = C(v_i, v_{i+1})$. At last, let $S(p, q) = \{C_1, \ldots, C_{m-1}\}$.

Let us show that $S(p, q)$ satisfies the properties of Definition 6.3:

- $\forall i \in \{1, \ldots, m-1\}$, C_i and C_{i+1} have the nodes p and v_{i+1} in common. Thus, according to Definition 6.2, $S(p, q)$ is connected.

- By definition, $S(p, q)$ contains $m - 1$ cycles. As $m \leq \Delta$, $S(p, q)$ contains less than Δ cycles

- Let C_i be a cycle of $S(p, q)$. Then, by definition of $C(u, v)$, C_i contains p, and does not contain q.

- Let v_i be a neighbor of p. If $i = m$, v_i belongs to C_{m-1}. Otherwise, v_i belongs to C_i. Thus, each neighbor of p belongs to a cycle of $S(p, q)$.

Therefore, $S = \bigcup_{\{p,q\} \in V} S(p, q)$ is a resilient decomposition, according to Definition 6.3. □

6.3.2 Connected sets of cycles

Let us show that the set of correct cycles is connected in the sense of Definition 6.2. This property in used in the correctness proofs of our algorithms, to show that the correct messages always manage to broadcast between Byzantine nodes.

For this purpose, we first show that each pair of correct nodes is connected by a bounded correct path (Lemmas 6.7 and 6.8). Then, we show that each pair of correct cycles is connected by a bounded set of cycles (Lemmas 6.9 and 6.10).

Lemma 6.7. *Let b be a Byzantine node. Let u and v be two neighbors of b. Then, there exists a correct path of at most ΔZ hops connecting u and v.*

Proof. Let q be any node distinct from b, u and v.

First, let us show that b is the only Byzantine node of the cycles of $S(b, q)$. Indeed, let us suppose the opposite: a cycle of $S(b, q)$ contains a Byzantine node $b' \neq b$. Then, the Byzantine nodes b and b' are on the same cycle, and are distant from at most Z hops: contradiction. Therefore, b is the only Byzantine node of the cycles of $S(b, q)$.

Let C (resp. C') be two cycles of $S(b, q)$ such that $u \in C$ (resp. $v \in C'$). As $S(b, q)$ is connected, according to Definition 6.2, there exists a sequence $(C_1, \ldots, C_n)$ of cycles of $S(b, q)$ such that $C = C_1$, $C' = C_n$ and $\forall i \in \{1, \ldots, n-1\}$, C_i and C_{i+1} have at least two nodes in common. As $S(b, q)$ contains at most Δ cycles, $n \leq \Delta$. Thus, as b is the only Byzantine node of C_i and C_{i+1}, C_i and C_{i+1} have at least one correct node in common. Let u_i be that node. Let $u_0 = u$ and $u_n = v$.

Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{0, \ldots, n\}$: there exists a correct path of at most iZ hops connecting u and u_i.

- $\mathcal{P}_0$ is true, as $u = u_0$.
- Let us suppose that $\mathcal{P}_i$ is true, for $i \in \{0, \ldots, n-1\}$. Then, as $\{u_i, u_{i+1}\} \subseteq C_{i+1}$ and b is the only Byzantine node of C_{i+1}, there exists a path of at most Z nodes of C_{i+1} connecting u_i and u_{i+1}. Thus, $\mathcal{P}_{i+1}$ is true.

Therefore, $\mathcal{P}_n$ is true, and there exists a correct path of at most $nZ \leq \Delta Z$ hops connecting u and $u_n = v$. Thus, the result.

□

Lemma 6.8. *Let p and q be two correct nodes. There exists a correct path of at most $2D\Delta$ hops connecting p and q.*

Proof. As D is the network diameter, let P be a path of n hops connecting p and q, with $n \leq D$. If P is correct, the result follows.

Otherwise, let b be a Byzantine node of P. As the distance between two Byzantine nodes is more than $2Z$, b has two correct neighbors u and v on P. Thus, according to Lemma 6.7, there exists a correct path of at most ΔZ hops connecting u and v. Thus, we can replace b by this path, and we obtain a new path P' that does not contain b.

We repeat this process for each Byzantine node of P, until we get a fully correct path. As the distance between two Byzantine nodes is more than $2Z$, there are at most $1 + D/2Z \leq D/Z$ Byzantine nodes in P. Thus, the final correct path contains at most $D + \Delta Z D/Z \leq 2D\Delta$ hops.

□

Lemma 6.9. *Let p be a correct node. Let u and v be two correct neighbors of p. Then, there exists a connected set X containing at most Δ correct cycles of diameter at most Z, such that u and p (resp. v and p) belong to a cycle of X.*

Proof. First, let us choose a particular node q.

If all the nodes at distance Z or less from p are correct, let q be any node distinct from p, u and v.

Otherwise, let b be a Byzantine node at distance Z or less from p. Let us show that b is the only Byzantine node at distance Z or less from p. Indeed, let us suppose that there

exists another Byzantine node b' in this situation. Then, b (resp. b') is at Z hops or less from p, and the distance between b and b' is at most $2Z$: contradiction. Let $q = b$.

Now, let us show that we can take $X = S(p, q)$:

- By definition, $S(p, q)$ is connected, contains at most Δ cycles, and Z is an upper bound of the diameter of the cycles of $S(p, q)$.
- Let C be a cycle of $S(p, q)$. By definition, C does not contain q. As the diameter of C is at most Z, all the nodes of C are at Z hops or less from p. Therefore, as all the nodes at Z hops or less from p and distinct from q are correct, C is correct. Thus, all the cycles of $S(p, q)$ are correct.
- For each neighbor w of p, there exists a cycle of $S(p, q)$ containing w. Therefore, there exists a cycle C (resp. C') of $S(p, q)$ containing u and p (resp. v and p).

Thus, the result, if we take $X = S(p, q)$. □

Lemma 6.10. *Let p and q be two correct nodes. Then, there exists a connected set Y containing at most $2D\Delta^2$ correct cycles of diameter at most Z, such that p (resp. q) belongs to a cycle of Y.*

Proof. According to Lemma 6.8, there exists a correct path $(u_0, \ldots, u_n)$, with $n \leq 2D\Delta$, such that $u_0 = p$ and $u_n = q$. Let us prove the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1, \ldots, n\}$: there exists a connected set X_i, containing at most $i\Delta$ correct cycles of diameter at most Z, such that p (resp. u_i) belongs to a cycle of X_i.

- First, let us show that $\mathcal{P}_1$ is true. Let v be a correct neighbor of p distinct from u_1. Then, according to Lemma 6.9, there exists a connected set X containing at most Δ correct cycles of diameter at most Z, such that p and u_1 (resp. p and v) belong to a cycle of X. Thus, $\mathcal{P}_1$ is true, if we take $X_1 = X$.
- Now, let us suppose that $\mathcal{P}_i$ is true, for $i \in \{1, \ldots, n\}$. Let C be the cycle of X_i containing u_i. Let v be a correct neighbor of u_i in C. Then, according to Lemma 6.9, there exists a connected set X containing at most Δ correct cycles of diameter at most Z, such that u_i and v (resp. u_i and u_{i+1}) belong to a cycle of X. Let C' be the cycle of X containing u_i and v. Then, as C and C' share two nodes, $X_i \cup X$ is connected, and contains at most $(i + 1)\Delta$ cycles. Thus, $\mathcal{P}_{i+1}$ is true, if we take $X_{i+1} = X_i \cup X$.

Therefore, $\mathcal{P}_n$ is true. Thus, the result, as $n \leq 2D\Delta$. □

6.4 Correctness proof of Algorithm 1

Let us show that Algorithm 1 ensures reliable communication in at most $8D\Delta^2ZT$ time units. For this purpose, we first show that the correct nodes can only accept correct messages (Lemma 6.11). Then, we show that they always eventually accept any correct message (Lemma 6.12 and Theorem 6.13).

Lemma 6.11. *Let p and q be two correct nodes. Then, if q accepts m from p, we necessarily have $m = p.m_0$.*

Proof. Let us suppose the opposite, and let q be the first correct node to accept a message $m \neq p.m_0$ from p. The node q cannot accept m from p in step 2 of our algorithm, as p only multicasts $p.m_0$. Thus, it was in step 4. Implying that there exists X and X' such that $X \cap X' = \emptyset$, $(p, m, X) \in q.Rec$ and $(p, m, X') \in q.Rec$.

Let us show that there exists a Byzantine node $b \in X$ at Z hops or less from q. For this purpose, let us suppose the opposite, and let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{0, \ldots, |X|\}$: there exists a correct node u_i at i hops or less from q and a set $X_i \subseteq X$ such that u_i multicasts (p, m, X_i) and $|X_i| \leq Z - i$.

- As $(p, m, X) \in q.Rec$, (p, m, X) was added to $q.Rec$ in step 3 of our algorithm. It implies that $|X| \leq Z$ and that p multicasts (p, m, X). Thus, $\mathcal{P}_0$ is true if we take $u_0 = q$ and $X_0 = X$.

- Now, let us suppose that $\mathcal{P}_i$ is true, for $i \in \{0, \ldots, |X| - 1\}$. As u_i multicasts (p, m, X_i), according to step 3 of our algorithm, it implies that there exists a node $v \in X_i \subseteq X$ such that u_i receives $(p, m, X_i - \{v\})$ from v. As v is at $i+1 \leq |X| \leq Z$ hops or less from q, according to our hypothesis, v cannot be Byzantine. Thus, v is correct, and $\mathcal{P}_{i+1}$ is true if we take $u_{i+1} = v$ and $X_{i+1} = X_i - \{v\}$.

Thus, $\mathcal{P}_{|X|}$ is true, and $u_{|X|}$ multicasts $(p, m, \emptyset)$, which was necessarily in step 2 of our algorithm. It implies that $u_{|X|}$ accepts m from p before q: contradiction. Thus, there exists a Byzantine node $b \in X$ at Z hops or less from q.

Similarly, there exists a Byzantine node $b' \in X'$ at Z hops or less from q. Thus, as $X \cap X' = \emptyset$, there exists two distinct Byzantine nodes b and b' distant from $2Z$ hops or less: contradiction. Thus, the result. □

Lemma 6.12. *Let p be a correct node, and let C be a correct cycle of diameter at most Z in the communication graph. Let us suppose that two nodes u and v of C accept $p.m_0$ from p before date t. Then, all nodes of C accept $p.m_0$ from p before date $t + 4ZT$.*

Proof. According to Lemma 6.11, a correct node can only accept $p.m_0$ from p.

Let w be a node of C such that, if one of the 3 nodes $\{u, v, w\}$ is removed, there still exists a path of nodes of C connecting the two remaining nodes in at most Z hops. Thus, there exists two internally disjoint paths $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_m)$ of nodes of C, with $u_1 = u$, $v_1 = v$, $u_n = v_m = w$, $n \leq Z$ and $m \leq Z$. Let $X_0 = Y_0 = \emptyset$, and for $i \neq 0$, let $X_i = \{u_1, \ldots, u_i\}$ and $Y_i = \{v_1, \ldots, v_i\}$.

Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1, \ldots, n\}$: u_i multicasts $(p, p.m_0, X_i)$ before date $t + 2iT$.

- As $u_1 = u$ accepts $p.m_0$ from p before date t, according to our algorithm, u multicasts $(p, p.m_0, \emptyset)$ before date t. Thus, $\mathcal{P}_0$ is true.

- Let us suppose that $\mathcal{P}_i$ is true, for $i \in \{0, \ldots, n-1\}$. As u_i multicasts $(p, p.m_0, X_i)$ before $t + 2iT$, u_{i+1} receives $(p, p.m_0, X_i)$ from u_i before $t + 2(i+1)T$. Thus, as $|X_i| < n \leq Z$, according to step 3 of our algorithm, u_{i+1} multicasts $(p, p.m_0, X_{i+1})$ before $t + 2(i+1)T$, and $\mathcal{P}_{i+1}$ is true.

Therefore, $\mathcal{P}_n$ is true, and $u_n = w$ multicasts $(p, p.m_0, X_n)$ before date $t+2nT \leq t+2ZT$. According to step 3 of our algorithm, it implies that we have $(p, p.m_0, X_{n-1}) \in w.Rec$ before $t + 2ZT$. Similarly, we have $(p, p.m_0, Y_{m-1}) \in w.Rec$ before $t + 2ZT$. Thus, as $Y_{m-1} \cap X_{n-1} = \emptyset$, according to step 4 of our algorithm, w accepts $p.m_0$ from p before $t + 2ZT$.

Now, let w' be any correct node of C. If $w' \in \{u, v, w\}$, the result follows. Otherwise, according to the choice of w, we can take two nodes $\{u', v'\} \subseteq \{u, v, w\}$ such that there exists two internally disjoint paths $(u'_1, \ldots, u'_{n'})$ and $(v'_1, \ldots, v'_{m'})$ of nodes of C, with $u'_1 = u$, $v'_1 = v$, $u'_{n'} = v'_{m'} = w'$, $n' \leq Z$ and $m' \leq Z$. Then, by a perfectly similar reasoning, w' accepts $p.m_0$ from p before $t + 4ZT$. Thus, the result.

□

Theorem 6.13. *Let p and q be two correct nodes. Then, q accepts $p.m_0$ from p before date $8D\Delta^2 ZT$, and never accepts another message from p.*

Proof. According to Lemma 6.11, a correct node can only accept $p.m_0$ from p.

If p and q are neighbors, according to our algorithm, q accepts $p.m_0$ from p before date $2T$, and then never accepts another message from q. Thus, the result. Now, let us suppose that p and q are not neighbors.

According to Lemma 6.10, there exists a sequence of correct cycles $(C_1, \ldots, C_n)$ of diameter at most Z in the communication graph, such that $p \in C_1$, $q \in C_n$, $n \leq 2D\Delta^2$ and $\forall i \in \{1, \ldots, n-1\}$, C_i and C_{i+1} have at least two nodes in common.

Let us show the following property $\mathcal{P}_i$ by induction, for $i \in \{1, \ldots, n\}$: all the nodes of C_i accept $p.m_0$ from p before date $2T + i4ZT$.

- Let u and v be the two neighbors of p in C_1. Then, according to step 1 and 2 of our algorithm, u and v accept $p.m_0$ from p before date $2T$. Thus, according to Lemma 6.12, $\mathcal{P}_1$ is true.

- Let us suppose that $\mathcal{P}_i$ is true, for $i \in \{1, \ldots, n-1\}$. Then, as C_i and C_{i+1} have two nodes in common, according to Lemma 6.12, $\mathcal{P}_{i+1}$ is true.

Therefore, $\mathcal{P}_n$ is true, and q accepts $p.m_0$ from p before date $2T + n4ZT \leq 2T + 8D\Delta^2 ZT \leq 9D\Delta^2 ZT$. Besides, according to our algorithm, q never accepts another message from p. Thus, the result.

□

6.5 Correctness proof of Algorithm 2

Let us show that Algorithm 2 ensures reliable communication despite any arbitrary initial state in at most $12D\Delta^2 ZT$ time units. In 6.5.1, we show that the effects of the arbitrary initial state dissipate after a certain time. In 6.5.2, we show that messages are regularly accepted. At last, in 6.5.3, we show that we always achieve reliable communication.

6.5.1 Self-stabilization

Let us show that any problematic message resulting from the arbitrary initial state is eliminated after $2KT$ time units.

For this purpose, we introduce the notion of *real sequence* (see Definition 6.14). A real sequence is a sequence of nodes that corresponds to a correct path after the last Byzantine node. In Lemma 6.15, we show that after $2KT$ time units, all the sequences attached to messages are real. Therefore, after $2KT$ time units, the lies can only come from the permanent Byzantine nodes, and not from the arbitrary initial state.

Definition 6.14 (Real sequence)**.** A sequence of node identifiers $(u_1, \ldots, u_n)$ is *real* if $n \in \{2, \ldots, K-1\}$ and:

- Either $(u_1, \ldots, u_n)$ is a correct path is the communication graph.
- Or there exists $k \in \{1, \ldots, n-1\}$ such that u_k is a Byzantine node, u_{k+1} is a correct neighbor of u_k, and $(u_{k+1}, \ldots, u_n)$ is a correct path in the communication graph.

Lemma 6.15. *Let p and q be two correct nodes. Let us suppose that (p, m, X) is added to $q.Rec$ after date $2KT$. Then, X is necessarily a real sequence. Besides, if X is a correct path in the communication graph, $m = p.m_0$.*

Proof. Let us suppose the opposite: X is not a real sequence. As (p, m, X) is added to $q.Rec$, according to our algorithm, X is necessarily a sequence $(u_1, \ldots, u_n)$, with $n \in \{2, \ldots, K\}$ and $u_n = q$. For $i \in \{1, \ldots, n-1\}$, let $X_i = (u_1, \ldots, u_{n-i})$ and $Y_i = (u_{n-i}, \ldots, u_n)$.

Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1, \ldots, n-1\}$: Y_i is a correct path in the communication graph and u_{n-i} multicasts (p, m, X_i) after date $2(K-i)T$.

- As (p, m, X) is added to $q.Rec$ after date $2KT > 0$, according to our algorithm, $q = u_n$ necessarily receives (p, m, X_1) from u_{n-1}. It implies that u_{n-1} multicasts (p, m, X_1) after date $2(K-1)T > 0$. If u_{n-1} is Byzantine, according to Definition 6.14, X is real: contradiction. Thus, u_{n-1} is correct, and $Y_1 = (u_{n-1}, u_n)$ is a correct path. Thus, $\mathcal{P}_1$ is true.

- Now, let us suppose that $\mathcal{P}_i$ is true, for $i \in \{1, \ldots, n-2\}$. As u_{n-i} multicasts (p, m, X_i) after date $2(K-i)T > 0$, according to our algorithm, u_{n-i} necessarily receives (p, m, X_{i+1}) from u_{n-i-1}. It implies that u_{n-i-1} multicasts (p, m, X_{i+1}) after date $2(K-i-1)T > 0$. If u_{n-i-1} is Byzantine, according to Definition 6.14, X is real: contradiction. Thus, u_{n-i-1} is correct. Therefore, as Y_i is a correct path and u_{n-i-1} is a correct neighbor of u_{n-i}, Y_{i+1} is also a correct path. Thus, $\mathcal{P}_{i+1}$ is true.

Therefore, $\mathcal{P}_{n-1}$ is true, and $X = Y_{n-1}$ is a correct path in the communication graph. Thus, according to Definition 6.14, X is a real sequence: contradiction. Thus, the first part of the result. Besides, in the case where X is a correct path in the communication graph, as $\mathcal{P}_{n-1}$ is true, $u_1 = p$ multicasts $(u_1, m, X_{n-1}) = (p, m, (p))$ before date $2(K-n+1)i > 0$. Thus, according to our algorithm, we necessarily have $m = p.m_0$.

□

6.5.2 Regular acceptance

Let us show that, for each pair of correct nodes p and q, q accepts a message from p each $2KT$ time units (whether or not this message is correct).

For this purpose, we show that q receives messages from p through each path of length K (Lemma 6.16). Then, we show that these paths reconstitute the sequence of cycles required by our algorithm for acceptance (Lemma 6.17).

Lemma 6.16. *Let $X = (u_1, \ldots, u_n)$ be a correct path in the communication graph, with $n \in \{2, \ldots, K-1\}$, and let t be a date. Then, $(u_1, u_1.m_0, X)$ is added to $u_n.Rec$ between t and $t + (2K-1)T$.*

Proof. Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1, \ldots, n-1\}$: u_{i+1} multicasts $(u_1, u_1.m_0, X_i)$ between t and $t + (2i+1)T$, with $X_i = (u_1, \ldots, u_{i+1})$.

- First, let us show that $\mathcal{P}_1$ is true. As u_1 is activated between t and $t+T$, u_1 multicasts $(u_1, u_1.m_0, (u_1))$. Thus, as u_2 receives $(u_1, u_1.m_0, (u_1))$ between t and $t+3T$, according to step 2 of our algorithm, u_2 multicasts $(u_1, u_1.m_0, (u_1, u_2))$. Thus, $\mathcal{P}_1$ is true.

- Now, let us suppose that $\mathcal{P}_i$ is true, for $i \in \{1, \ldots, n-2\}$. As u_{i+1} multicasts $(u_1, u_1.m_0, X_i)$ between t and $t + (2i+1)T$, u_{i+2} receives $(u_1, u_1.m_0, X_i)$ between t and $t + (2(i+1)+1)T$. Thus, as $|X_i| \leq n \leq K$, according to step 2 of our algorithm, u_{i+2} multicasts $(u_1, u_1.m_0, X_{i+1})$. Thus, $\mathcal{P}_{i+1}$ is true.

Therefore, $\mathcal{P}_{n-1}$ is true, and u_n multicasts $(u_1, u_1.m_0, X)$ between t and $t + (2(n-1)+1)T \leq t + (2K-1)T$. Implying that $(u_1, u_1.m_0, X)$ is added to $u_n.Rec$ between t and $t + (2K-1)T$. Thus, the result. □

Lemma 6.17. *Let p and q be two correct nodes, and let t be a date. Then, between t and $t + 2KT$, q accepts a message from p.*

Proof. According to Lemma 6.10, there exists a sequence $(C_1, \ldots, C_m)$ of correct cycles of diameter at most Z, such that $p \in C_1$, $q \in C_m$, $m \leq 2D\Delta^2$ and $\forall i \in \{1, \ldots, n-1\}$, C_i and C_{i+1} have at least two nodes in common.

Let Ω be the set of paths $(u_1, \ldots, u_n)$ such that $u_1 = p$, $u_n = q$, $n \leq K$, and forall $i \in \{1, \ldots, n\}$, u_i belongs to a cycle of $(C_1, \ldots, C_m)$. As $2Zm \leq K$, each node and each edge of a cycle of $(C_1, \ldots, C_m)$ is contained by at least one path of Ω. Therefore, according to Definition 6.5, Ω is (p, q)-valid.

According to Lemma 6.16, $\forall X \in \Omega$, $(p, p.m_0, X)$ is added to $q.Rec$ between t and $t + (2K-1)T$. Thus, two possibilities:

- Either one of these tuples $(p, p.m_0, X)$ has been removed between t and $t+(2K-1)T$. According to step 3 of our algorithm, it implies that q accepts a message from p between t and $t + 2KT$.
- Or, at date $t+(2K-1)T$, $\forall X \in \Omega$, we have $(p, p.m_0, X) \in q.Rec$. As q is activated between $t + (2K-1)T$ and $t + 2KT$, according to step 3 of our algorithm, as Ω is valid, q accepts $p.m_0$ from p.

Thus, in all cases, q accepts a message from p between t and $t + 2KT$.

□

6.5.3 Reliable communication

At last, let us show that our algorithm ensures reliable communication. For this purpose, we show that after a certain time, a message accepted is necessarily correct (Lemmas 6.18 and 6.19). Then, we show that the correct messages are eventually and definitively accepted (Theorem 6.20).

Lemma 6.18. *Let p and q be two correct nodes. Let Ω be a (p,q)-valid set such that, $\forall X \in \Omega$, X is a real sequence. Then, at least one of these sequences is a correct path in the communication graph.*

Proof. Let us suppose the opposite: Ω does not contain any sequence that corresponds to a correct path in the communication graph.

As the sequences of Ω are real, according to Definition 6.14, it implies that $\forall X = (u_1, \ldots, u_n) \in \Omega$, there exists $k \in \{1, \ldots, n-1\}$ such that u_k is a Byzantine node, u_{k+1} is a correct neighbor of u_k, and $(u_{k+1}, \ldots, u_n)$ is a correct path in the communication graph. Thus, these Byzantine nodes u_k form a node cut on $G(\Omega)$, and the subgraph G' containing q after the cut is a subgraph of the communication graph.

As Ω id (p,q)-valid, $G(\Omega)$ can be decomposed in a sequence $(C_1, \ldots, C_m)$ of cycles of diameter at most Z, such that $p \in C_1$, $q \in C_m$ and $\forall i \in \{1, \ldots, m-1\}$, C_i and C_{i+1} have at least two elements in common. Let $i \in \{1, \ldots, n\}$ be the smallest integer such that a node of C_i belongs to G'. As $G(\Omega)$ is 2-connected, C_i contains at least two Byzantine nodes. As the diameter of C_i is at most Z, the distance between these two Byzantine

nodes in G' is at most $2Z$. Therefore, as G' is a subgraph of the communication graph, the distance between these two Byzantine nodes is at most $2Z$: contradiction. Thus, the result.

□

Lemma 6.19. *Let p and q be two correct nodes. After date $4KT$, if q accepts m from p, then we necessarily have $m = p.m_0$.*

Proof. According to Lemma 6.15, after date $2KT$, if a tuple (p, m, X) is added to $q.Rec$, then X is a real sequence. According to Lemma 6.17, between dates $2KT$ and $4KT$, q accepts a message from p. According to our algorithm, it implies that $\forall(m', X')$ such that $(p, m', X') \in q.Rec$, $(p, m'X')$ was removed from $q.Rec$. Therefore, after date $4KT$, $\forall X$ such that $(p, m, X) \in q.Rec$, X is a real sequence.

As q accepts m from p, according to our algorithm, there exists a (p, q)-valid set Ω such that $\forall X \in \Omega$, $(p, m, X) \in q.Rec$. As all the sequences of Ω are real, according to Lemma 6.18, Ω contains a least one sequence $X_0 = (u_1, \ldots, u_n)$ that is a correct path in the communication graph. Thus, as $(p, m, X_0) \in q.Rec$, according to Lemma 6.15, $m = p.m_0$. □

Theorem 6.20. *Let p and q be two non-neighbors correct nodes. Then, after date $6KT = 12D\Delta^2ZT$, q has accepted $p.m_0$ from p, and never accepts another message from p.*

Proof. According to Lemma 6.19, after date $4KT$, if q accepts a message from p, this is necessarily $p.m_0$. Thus, according to Lemma 6.17, between dates $4KT$ and $6KT$, q accepts $p.m_0$ from p, and then never accepts another message from p. □

6.6 Conclusion

In this chapter, we proposed a distance-based approach for Byzantine resilience. Both regular and self-stabilizing reliable communication can be achieved within a linear time.

An open question would be to show the tightness of this distance criteria. Also, another interesting extension would be to combine self-stabilization and tolerance to dynamic changes in the network topology.

Chapter 7

Tolerating critical pairs

In the Chapter 6, we proposed solutions assuming a minimal distance between two Byzantine nodes. In other words, there must be no *critical pair*, a critical pair being a pair of Byzantine nodes that are too close from each other.

In this chapter, we show that it is also possible to tolerate critical pairs, provided that there is a minimal distance between two critical pairs. We provide an algorithm and show its correctness on a lattice topology.

We present the motivation in Section 7.1, the setting in Section 7.2, the algorithm in Section 7.3, and the correctness proof in Section 7.4.

This work is a collaboration with Pr. Toshimitsu Masuzawa from Osaka University (Japan).

7.1 Motivation

We consider a 8-connected lattice network, as illustrated in Figure 7.1.

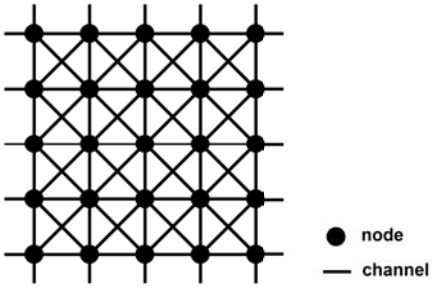

FIGURE 7.1: A 5×5 8-connected lattice.

According to Chapter 6, we can ensure reliable communication provided that the distance between two Byzantine nodes is strictly greater than 2. In this setting, this protocol corresponds to the simple Certified Propagation Algorithm [32], where a message must be received from 2 neighbors to be forwarded.

Yet, if two Byzantine nodes are at distance 2 or less, they can cooperate to make a correct node accept and forward a false message. Let us call such a pair of Byzantine nodes a *critical pair*.

In this chapter, we propose an algorithm that tolerates critical pairs, provided that the critical pairs themselves are sufficiently distant. This is illustrated in Figure 7.2. The previous algorithm can tolerate the Byzantine nodes of case A. This new algorithm can tolerate both cases A and B.

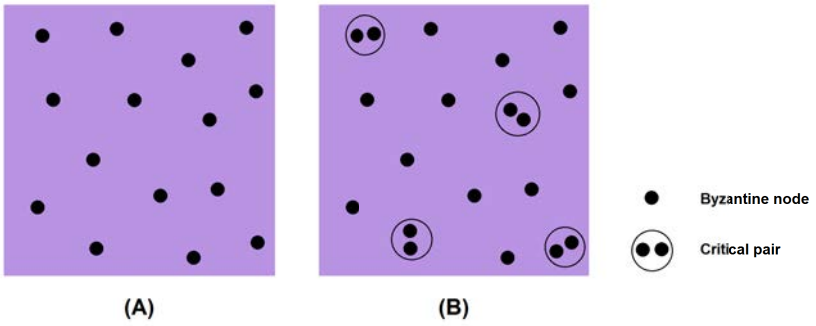

FIGURE 7.2: Improvement of the algorithm.

7.2 Setting

Let $N \geq 10$. We consider a network where:

- Each node corresponds to a tuple (i, j), with $1 \leq i \leq N$ and $1 \leq j \leq N$.
- Two nodes (i_1, j_1) and (i_2, j_2) are neighbors if $|i_1 - i_2|$ (resp. $|j_1 - j_2|$) equals 0, 1 or N.

Let $d(p, q)$ be the distance between two nodes p and q.

Definition 7.1 (Critical pair)**.** A *critical pair* is a pair $\{b, b'\}$ of Byzantine nodes such that $d(b, b') \leq 2$.

The distance between a node p and a critical pair $\{b, b'\}$ is $\min(d(p, b), d(p, b'))$, and the distance between two critical pairs $\{b_1, b'_1\}$ and $\{b_2, b'_2\}$ is $\min(d(b_1, b_2), d(b'_1, b_2), d(b_1, b'_2), d(b'_1, b'_2))$.

Let $H = 21$. In the following, we assume that the distance between two critical pairs is at least $H + 3$.

7.3 Algorithm

7.3.1 Informal description

The principle of algorithm [32] is the following: to forward a message, a node must receive it from two neighbors. This is illustrated in Figure 7.3.

Our algorithm can be seen as a recursion of the previous algorithm. Let a *voting path* be a sequence of nodes along which a message can broadcast w.r.t. algorithm [32]. Here, to forward a message, a node must receive it from two disjoint voting paths of a bounded length. Thus, a single pair of Byzantine nodes is not sufficient to initiate the broadcast of a false message. This is illustrated in Figure 7.4.

FIGURE 7.3: Principle of the previous algorithm.

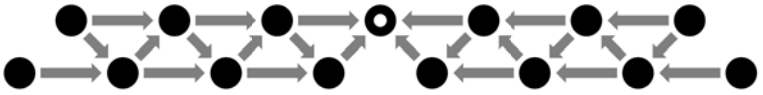

FIGURE 7.4: Principle of our algorithm.

7.3.2 Description of the algorithm

Each correct node has two memory sets Rec and Rec', initially empty.

Each correct node p executes the following algorithm.

1. Initially, accept $p.m_0$ from p and multicast $(p, p.m_0, \emptyset)$.
2. When (s, m, Ω) is received from a neighbor q, with $q \notin \Omega$:
   - If $q = s$, accept m from s and multicast $(s, m, \emptyset)$.
   - Add $(s, m, \Omega \cup \{q\}, q)$ to Rec.
3. When there exists s, m, Ω, Ω', q and q' such that $q \neq q'$, $(s, m, \Omega, q) \in Rec$, $(s, m, \Omega', q') \in Rec$ and $|\Omega \cup \Omega'| \leq H$:

- Multicast $(s, m, \Omega \cup \Omega')$.
- Add $(s, m, \Omega \cup \Omega')$ to Rec'.

4. When there exists s, m, Ω and Ω' such that $(s, m, \Omega) \in Rec'$, $(s, m, \Omega') \in Rec'$, $\Omega \cap \Omega' = \emptyset$ and $|\Omega| + |\Omega'| \leq H$:
   - Accept m from s.
   - Multicast $(s, m, \emptyset)$.

7.4 Correctness proof

7.4.1 Definitions

Definition 7.2 (Voting path)**.** A *voting path* is a sequence $(u_1, \ldots, u_n)$ of distinct correct nodes, with $n \in \{3, \ldots, H+1\}$, such that $\forall i \in \{3, \ldots, n\}$, there exists $j \in \{1, \ldots, i-2\}$ such that u_i is neighbor with both u_{i-1} and u_j.

Definition 7.3 (Reliable node)**.** Let s be a correct node. A correct node p is *reliable for* s if p always eventually accepts $s.m_0$ from s.

Definition 7.4 (Square)**.** A set of four nodes $\{p, q, r, s\}$ is a *square* if there exists i and j such that $p = (i, j)$, $q = (i+1, j)$, $r = (i, j+1)$ and $s = (i+1, j+1)$. A square is *correct* if all its nodes are correct.

Definition 7.5 (Square path)**.** A sequence of squares $(S_1, \ldots, S_n)$ is a *square path* connecting S_1 and S_n if $\forall i \in \{1, \ldots, n-1\}$, $|S_i| \cap |S_{i+1}| = 2$.

Definition 7.6 (Square path)**.** A sequence of squares $(S_1, \ldots, S_n)$ is a *square path* connecting S_1 and S_n if $\forall i \in \{1, \ldots, n-1\}$, $|S_i| \cap |S_{i+1}| = 2$.

Definition 7.7 (Connected set of squares)**.** A set of squares X is connected if, for each pair of squares (S, S') of X, there exists a square path of X connecting S and S'.

7.4.2 Proof

In Theorem 7.15, we show that for any correct nodes s and p, p is reliable for s.

Lemma 7.8. *Let s be a correct node. If a correct node accepts a message m from s, then $m = s.m_0$.*

Proof. Let us suppose the opposite. Let p be the first correct node to accept a message m from s such that $m \neq s.m0$. According to the algorithm, it implies that there exists Ω and Ω' such that $(s, m, \Omega) \in p.Rec'$, $(s, m, \Omega') \in p.Rec'$, $\Omega \cap \Omega' = \emptyset$ and $|\Omega| + |\Omega'| \leq H$.

Let us suppose that there exists no critical pair $\{b, b'\} \subseteq \Omega$ at distance $|\Omega| + 1$ or less from p. Let $p_0 = p$ and $\Omega_0 = \Omega$. Let us prove the following property $\mathcal{P}_i$ by induction, $\forall i \in \{0, \ldots, |\Omega| + 1\}$: there exists a correct node p_i at distance i or less from p and a set $\Omega_i \subseteq \Omega$ such that $(s, m, \Omega_i) \in p_i.Rec'$ and $|\Omega_i| \leq |\Omega| - i$.

- $\mathcal{P}_0$ is true, as $(s, m, \Omega) \in p.Rec'$.
- Let us suppose that $\mathcal{P}_i$ is true, for $i \in \{0, \ldots, |\Omega|\}$. As $(s, m, \Omega_i) \in p_i.Rec'$, according to step 3 of the algorithm, there exists s, m, X, X', q and q' such that $q \neq q'$, $(s, m, X, q) \in p_i.Rec$, $(s, m, X', q') \in p_i.Rec$ and $\Omega_i = X \cup X'$. As $(s, m, X, q) \in p_i.Rec$, according to step 2 of the algorithm, it implies that p_i received (s, m, Y) from q, with $q \notin Y$ and $X = Y \cup \{q\}$. The same is also true for q' and X'. Besides, $\{q, q'\} \subseteq \Omega_i \subseteq \Omega$.

  As p_i is at distance $i \leq |\Omega|$ or less from p, q and q' are at distance $|\Omega| + 1$ or less from p. Thus, as there exists no critical pair $\{b, b'\} \subseteq \Omega$ at distance $|\Omega| + 1$ or less from p, it is impossible that both q and q' are Byzantine. Therefore, there exists a correct node $v \in \{q, q'\}$ that sent (s, m, Z), with $v \notin Z$ and $Z \cup \{v\} \subseteq \Omega_i$. Thus, $Z \subseteq \Omega$ and $|Z| \leq |\Omega_i| - 1 \leq |\Omega| - (i + 1)$. Besides, as p is the first correct node to accept m from s, (s, m, Z) was necessarily sent in step 3 of the algorithm, implying that $(s, m, Z) \in v.Rec'$.

  Therefore, $\mathcal{P}_{i+1}$ is true, if we take $p_{i+1} = v$ and $\Omega_{i+1} = Z$.

Thus, $\mathcal{P}_{|\Omega|+1}$ is true, implying that there exists a set $\Omega_{|\Omega|+1}$ such that $|\Omega_{|\Omega|+1}| \leq -1$: contradiction. Therefore, there exists a critical pair $\{b_1, b'_1\} \subseteq \Omega$ at distance $|\Omega| + 1$ or less from p.

By a similar reasoning, there exists a critical pair $\{b_2, b'_2\} \subseteq \Omega'$ at distance $|\Omega'| + 1$ or less from p. As $\Omega \cap \Omega' = \emptyset$, $\{b_1, b'_1\}$ and $\{b_2, b'_2\}$ are not the same critical pair. As $|\Omega| + |\Omega'| \leq H$, the distance between $\{b_1, b'_1\}$ and $\{b_2, b'_2\}$ is at most $|\Omega| + |\Omega'| + 2 \leq H + 2$: contradiction. Thus, the result.

□

Lemma 7.9. *Let s be a correct node, and let $(u_1, \ldots, u_n)$ be a voting path. If u_1 and u_2 are reliable for s, then we eventually have $(s, s.m_0, \Omega) \in u_n.Rec'$, with $\Omega \subseteq \{u_1, \ldots, u_{n-1}\}$.*

Proof. Let us prove the following property $\mathcal{P}_i$ by induction, $\forall i \in \{3, \ldots, n\}$: we eventually have $(s, s.m_0, \Omega_i) \in u_i.Rec'$, with $\Omega_i \subseteq \{u_1, \ldots, u_{i-1}\}$.

- As u_1 and u_2 are reliable for s, according to the algorithm, u_1 and u_2 eventually multicast $(s, s.m_0, \emptyset)$. According to Definition 7.2, u_3 is neighbor with u_1 and u_2. Thus, according to step 2 of the algorithm, we eventually have $(s, s.m_0, \{u_1\}, u_1) \in u_3.Rec$ and $(s, s.m_0, \{u_2\}, u_2) \in u_3.Rec$. Thus, according to step 3, we eventually have $(s, s.m_0, \{u_1, u_2\}) \in u_3.Rec'$, and $\mathcal{P}_3$ is true. Besides, u_3 eventually multicasts $(s, s.m_0, \{u_1, u_2\})$. Thus, we eventually have $(s, s.m_0, \{u_1, u_2, u_3\}, u_3) \in u_4.Rec$ and $(s, s.m_0, \{u_j\}, u_j) \in u_4.Rec$, with $j \in \{1, 2\}$. Therefore, for the same reason, $\mathcal{P}_4$ is true.
- Let $i \in \{4, \ldots, n-1\}$, and let us suppose that $\forall k \in \{1, \ldots, i\}$, $\mathcal{P}_k$ is true. Let $j \in \{1, \ldots, i-1\}$ be such that u_{i+1} and u_j are neighbors. As $\mathcal{P}_j$ and $\mathcal{P}_i$ are true, we eventually have $(s, s.m_0, \Omega_j) \in u_j.Rec'$ and $(s, s.m_0, \Omega_i) \in u_i.Rec'$, with $\Omega_i \subseteq \{u_1, \ldots, u_{i-1}\}$ and $\Omega_j \subseteq \{u_1, \ldots, u_{j-1}\}$. According to the algorithm, it implies that u_j eventually multicasts $(s, s.m_0, \Omega_j)$ and that u_i eventually multicasts $(s, s.m_0, \Omega_i)$. Thus, we eventually have $(s, s.m_0, \Omega_j \cup \{u_j\}) \in u_{i+1}.Rec$ and $(s, s.m_0, \Omega_i \cup \{u_i\}) \in u_{i+1}.Rec$. Let $\Omega_{i+1} = \Omega_i \cup \Omega_j \cup \{u_i, u_j\} \subseteq \{u_1, \ldots, u_i\}$. As $|\Omega_{i+1}| \leq i \leq n-1 \leq H$, according to step 3, we eventually have $(s, s.m_0, \Omega_{i+1}) \in u_{i+1}.Rec'$, and $\mathcal{P}_{i+1}$ is true.

Thus, $\mathcal{P}_n$ is true. Thus, the result.

□

Lemma 7.10. *Let s be a correct node. Let $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_m)$ be two voting paths such that $u_n = v_m$, $n + m \leq H + 2$, $\{u_1, \ldots, u_{n-1}\} \cap \{v_1, \ldots, v_{m-1}\} = \emptyset$, and u_1, u_2, v_1 and v_2 are reliable for s. Then, u_n is also reliable for s.*

Proof. According to Lemma 7.9, we eventually have $(s, s.m_0, \Omega) \in u_n.Rec'$ and $(s, s.m_0, \Omega') \in u_n.Rec'$, with $\Omega \subseteq \{u_1, \ldots, u_{n-1}\}$ and $\Omega' \subseteq \{v_1, \ldots, v_{m-1}\}$. Thus, $\Omega \cap \Omega' = \emptyset$ and $|\Omega| + |\Omega'| \leq n + m - 2 \leq H$. Therefore, according to step 4 of the algorithm, u_n eventually accepts $s.m_0$ from s. Thus, the result. □

Lemma 7.11. *The set of correct squares is connected in the sense of Definition 7.7.*

Proof. Let S and S' be two correct squares. Let $(S_1, \ldots, S_n)$ be a square path connecting S and S' (this path may contain Byzantine nodes). Let us show that we can fix this path to obtain a correct square path.

Let k be the first integer such that S_k is not a correct square, and let $k' > k$ be the first integer such that $S_{k'}$ is a correct square. Let b be a Byzantine node of S_k.

- If b does not belong to a critical pair, then all the nodes at distance 2 or less from b are correct. Therefore, there exists a connected set of correct squares X such as represented in Figure 7.5.
- If b belongs to a critical pair $\{b, b'\}$, then all the nodes at distance 2 or less from either b or b' are correct (otherwise we would have two critical pairs that do not respect the distance hypothesis). Therefore, there exists a connected set of correct squares X such as represented in Figure 7.6 (a similar set exists for each possible relative placement of b and b').

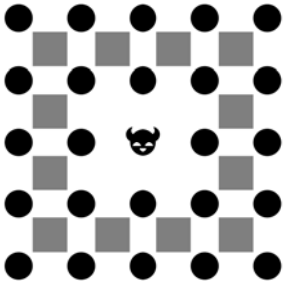

FIGURE 7.5: Connected set of correct squares around a single Byzantine node.

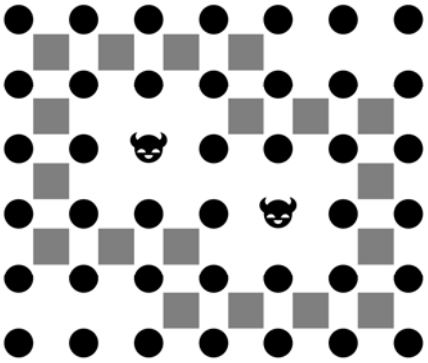

FIGURE 7.6: Connected set of correct squares around a critical pair.

Therefore, there exists a square path $(Z_1, \ldots, Z_m)$ of squares of X such that $Z_1 = S_k$ and $Z_m = S_{k'}$. Therefore, we can fix the square path $(S_1, \ldots, S_n)$ into $(S_1, \ldots, S_{k-1}, Z_1, \ldots, Z_m, S_{k'+1}, \ldots, S_n)$, so that this new path does not contain b. We repeat the process until we obtain a square path that does not contain any Byzantine node. □

Lemma 7.12. *Let s be a correct node. There exists a correct square S such that all the nodes of S are reliable for s.*

Proof. If s belongs to a correct square, let $S = \{s, p, q, r\}$ be this correct square. As p, q and r are neighbors of s, according to the algorithm, all the nodes of S are reliable for s.

Now, let us suppose that s does not belong to a correct square. Then, we are in a situation such as represented in Figure 7.7 (or a symmetrical situation), involving a critical pair $\{b, b'\}$. Let us suppose that the node s corresponds to the coordinates $(0, 0)$ on the lattice. Let:

- $P_1 = ((1, 0), (1, -1), (2, 0))$
- $P_2 = ((-1, 0), (-1, 1), (-2, 1), (-1, 2), (0, 2), (0, 3), (1, 2), (1, 1), (2, 1), (2, 0))$
- $P_3 = ((1, 0), (1, -1), (2, -1))$
- $P_4 = ((-1, 0), (-1, -1), (-2, -1), (-1, -2), (0, -2), (0, -3), (1, -3), (1, -2), (2, -2), (2, -1))$

According to our hypothesis, all the nodes at distance 2 or less from b or b' are correct. Therefore, according to Definition 7.2, P_1, P_2, P_3 and P_4 are voting paths. Besides, all the correct neighbors of s are reliable for s. Thus, according to Lemma 7.10, the nodes $(2, 0)$ and $(2, 1)$ are reliable for s. Thus, all the nodes of the square $\{(1, 0), (1, -1), (2, 0), (2, -1)\}$ are reliable for s. Thus, the result.

□

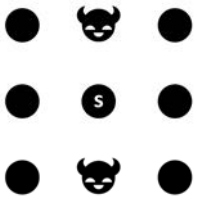

FIGURE 7.7: Configuration where a node s does not belong to a correct square.

Lemma 7.13. *Let S be the square formed by the nodes of coordinates $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$. Let s be a correct node, and let us suppose that all the nodes of S are correct and reliable for s. Let us suppose that the nodes of coordinates $(0, 2)$ and $(1, 2)$ are also correct. Then, the node $(0, 2)$ is reliable for s.*

Proof. First, let us suppose that the nodes $(-1, 1)$ and $(2, 1)$ are correct. Let:

- $P_1 = ((0, 0), (0, 1), (-1, 1), (0, 1))$

- $P_2 = ((1,0),(1,1),(2,1),(1,2),(0,1))$

Then, for the same argument as in the proof of Lemma 7.12, the node $(0,2)$ is reliable for s.

Now, let us suppose that there exists a Byzantine node b which is either $(-1,1)$ or $(2,1)$. Then, for the same argument as in the proof of Lemma 7.11, there exists a connected set X of correct squares such as represented in Figure 7.5 or Figure 7.6 (depending on whether b belongs to a critical pair $\{b,b'\}$ or not), which contains S, $(0,2)$ and $(1,2)$. Thus, there exists a voting path $P_3 = (u_1,\dots,u_n)$ such that $\{u_1,u_n\} = \{(0,0),(1,0)\}$, $u_n = (0,2)$ and $\forall i \in \{0,\dots,n\}$, u_1 belongs to a square of X. By analyzing all possible placements of b and possibly b', it is possible to have $n \leq 20$

Thus, as $P_4 = ((1,1),(0,1),(0,2))$ is also a voting path, for the same argument as in the proof of Lemma 7.12, the node $(0,2)$ is reliable for s. Thus, the result. □

Lemma 7.14. *Let s be a correct node, and let p be a correct node that belongs to a correct square S. Then, p is reliable for s.*

Proof. According to Lemma 7.12, there exists a correct square S' such that all the nodes of S' are reliable for s. According to Lemma 7.11, there exists a square path $(S_1,\dots,S_n)$ of correct squares such that $S_1 = S'$ and $S_n = S$. Let us prove the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1,\dots,n\}$: all the nodes of S_i are reliable for s.

- $\mathcal{P}_1$ is true, as $S_1 = S'$.
- Let us suppose that $\mathcal{P}_i$ is true, for $i \in \{1,\dots,n-1\}$. Then, S_i (resp. S_{i+1}) is equivalent to the square $\{(0,0),(0,1),(1,0),(1,1)\}$ (resp. $\{(1,0),(1,1),(2,0),(2,1)\}$) of Lemma 7.13, with the appropriate translations, rotations and symmetries. Thus, according to Lemma 7.13, the node corresponding to $(2,0)$ is reliable for s, and so is the node corresponding to $(2,1)$ (by symmetry). Thus, $\mathcal{P}_{i+1}$ is true.

□

Theorem 7.15. *Let s and p be two correct node. Then, p is reliable for s.*

Proof. If p belongs to a correct square, according to Lemma 7.14, p is reliable for s.

Otherwise, we are in a situation such as represented in Figure 7.7 (if we replace s by p), or a symmetrical situation. Let us suppose that p corresponds to the coordinates $(0,0)$ on Figure 7.7. Then, the nodes $(-1,0)$, $(-1,1)$, $(1,0)$ and $(1,1)$ belong to correct

squares, and are reliable for s according to Lemma 7.14. Thus, $((-1,1),(-1,0),(0,0))$ and $((1,1),(1,0),(0,0))$ are voting paths. Therefore, according to Lemma 7.10, $p = (0,0)$ is reliable for s. □

7.5 Conclusion

In this chapter, we proposed an algorithm tolerating pairs of close Byzantine nodes. Thus, we have a higher probability to tolerate a given placement of Byzantine nodes.

An interesting extension would be to keep applying this principle recursively to tolerate close critical pairs, and so forth. Also, we elaborated on the case where 2 votes are required to forward a message. Yet, in the case where $k > 2$ votes are required, this idea could be generalized to tolerate critical groups of k Byzantine failures.

Part III

Extensions

Chapter 8

Dynamic networks

In this chapter, we give the condition for reliable communication in a *dynamic* [47] network, in the presence of up to k arbitrarily placed Byzantine failures.

In a static network, the condition is the existence of $2k + 1$ disjoint paths between the source ad the sink. However, this result relies on Menger's theorem [38], which ensures the equivalence between node cut and connectivity. Unfortunately, this theorem cannot be generalized to dynamic networks [48].

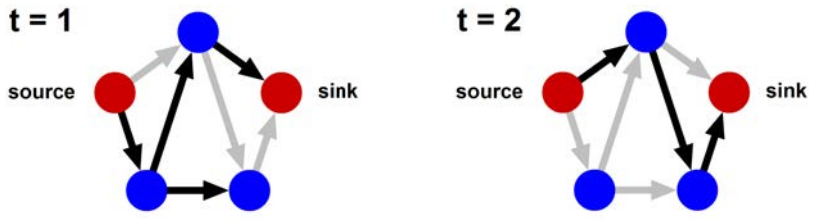

FIGURE 8.1: Counterexample to Menger's theorem in dynamic graphs. Black arrows represent arcs that are present at that time.

A simple counterexample is given in Figure 8.1, where at least two nodes must be removed in order to disconnect the source from the sink: for example, the two nodes that are adjacent to the source. However, it is impossible to find two node-disjoint paths between the source and the sink: there exist one path between the source and the sink at time 1, one path at time 2, and one dynamic path that spans two edges at time 1 and one edge at time 2; yet, any two of those three paths share at least one node.

The chapter is organized as follows:

- In Section 8.1, we present the model and give basic definitions.

- In Section 8.2, we describe our algorithm, then prove the necessary and sufficient condition for reliable communication.
- In Section 8.3, we apply this condition to several case studies.

This work is a collaboration with Pr. Xavier Defago from Japan Advanced Institute of Science and Technology.

8.1 Preliminaries

8.1.1 Network model

We consider a continuous temporal domain $\mathbb{R}^+$ where dates are positive real numbers. We model the system as a time varying graph, as defined by Casteigts, Flocchini, Quattrociocchi and Santoro [47], where vertices represent the processes and edges represent the communication links (or channels). A time varying graph is a dynamic graph represented by a tuple $\mathcal{G} = (V, E, \rho, \zeta)$ where:

- V is the set of *nodes*.
- $E \subseteq V \times V$ is the set of *edges*.
- $\rho : E \times \mathbb{R}^+ \to \{0, 1\}$ is the *presence* function: $\rho(e, t) = 1$ indicates that edge e is present at date t.
- $\zeta : E \times \mathbb{R}^+ \to \mathbb{R}^+$ is the *latency* function: $\zeta(e, t) = T$ indicates that a message sent at date t takes T time units to cross edge e.

The discrete time model is a special case, where time and latency are restricted to integer values.

8.1.2 Definitions

Informally, a *dynamic path* is a sequence of nodes a message can traverse, with respect to network dynamicity and latency.

Definition 8.1 (Dynamic path)**.** A sequence of distinct nodes $(u_1, \ldots, u_n)$ is a *dynamic path* from u_1 to u_n if and only if there exists a sequence of dates $(t_1, \ldots, t_n)$ such that, $\forall i \in \{1, \ldots, n-1\}$ we have:

- $e_i = (u_i, u_{i+1}) \in E$: there exists an edge connecting u_i to u_{i+1}.
- $\forall t \in [t_i, t_i + \zeta(e_i, t_i)]$, $\rho(e_i, t) = 1$: u_i can send a message to u_{i+1} at date t_i.
- $\zeta(e_i, t_i) \leq t_{i+1} - t_i$: the aforementioned message is received by date t_{i+1}.

We now define the *dynamic minimal cut* between two nodes p and q as the minimal number of nodes (besides p and q) one has to remove from the network to prevent the existence of a dynamic path between p and q. Formally:

- Let $Dyn\,(p, q)$ be the set of node sets $\{u_1, \ldots, u_n\}$ such that $(p, u_1, \ldots, u_n, q)$ is a dynamic path.
- For a set of node sets $\Omega = \{S_1, \ldots, S_n\}$, let $Cut\,(\Omega)$ be the set of node sets C such that, $\forall i \in \{1, \ldots, n\}$, $C \cap S_i \neq \emptyset$ (C contains at least one node from each set S_i).
- Let $MinCut\,(\Omega) = \min_{C \in Cut(\Omega)} card(C)$ (the size of the smallest element of $Cut\,(\Omega)$). If $Cut(\Omega)$ is empty, we assume that $MinCut\,(\Omega) = +\infty$.
- Let $DynMinCut\,(p, q) = MinCut\,(Dyn\,(p, q))$.

8.2 Algorithm and condition for reliable communication

In this section, we describe our Byzantine-resilient multihop broadcast protocol. This algorithm is used as a constructive proof for the sufficient condition for reliable communication. We then prove the necessary and sufficient condition for reliable communication.

8.2.1 Informal description

Let us suppose that a node p wants to broadcast a message m. To each message, we attach the set of nodes that have been visited by this message since it was sent (that is, we use (p, m, S), where S is a set of nodes already visited by m since p sent it).

As the Byzantine nodes can send any message, in particular, they can forward false tuples (p, m, S). Therefore, a correct node only accepts a message when it has been received through a collection of dynamic paths that cannot be cut by k nodes (where k is a parameter of the algorithm, and supposed to be an upper bound on the total number of Byzantine nodes in the network).

Focusing on minimal cut instead of node-disjoint paths (unlike [37]) makes this approach robust to high network dynamicity, as demonstrated in Lemma 8.6.

8.2.2 Variables

Each correct node u maintains the following variables:

- $u.m_0$, the message that u wants to broadcast.
- $u.\Omega$, a dynamic set registering all tuples (s, m, S) received by u.
- $u.Acc$, a dynamic set of confirmed tuples (s, m). We assume that whenever $(s, m) \in u.Acc$, u accepts m from s.

Initially, $u.\Omega = \{(u, u.m_0, \emptyset)\}$ and $u.Acc = \{(u, u.m_0)\}$.

8.2.3 Algorithm

Each correct node u obeys the three following rules:

1. Initially, and whenever $u.\Omega$ or the local topology of u change: multicast $u.\Omega$.
2. Upon reception of Ω' through channel (v, u): $\forall (s, m, S) \in \Omega'$, if $v \notin S$ then append $(s, m, S \cup \{v\})$ to $u.\Omega$.
3. Whenever there exist s, m and $\{S_1, \ldots, S_n\}$ such that $\forall i \in \{1, \ldots, n\}$, $(s, m, S_i \cup \{s\}) \in u.\Omega$ and $MinCut\,(\{S_1, \ldots, S_n\}) > k$: append (s, m) to $u.Acc$.

8.2.4 Main theorem

Let us consider a given dynamic graph, and two given correct nodes p and q. Our main result is as follows:

Theorem 8.2. *For a given dynamic graph, A k-Byzantine tolerant reliable communication from p to q is feasible if and only if $DynMinCut\,(p, q) > 2k$.*

Proof. The proof of the "only if" part is in Lemma 8.3. The proof of the "if" is in Lemma 8.6. □

Lemma 8.3 (Necessary condition)**.** *For a given dynamic graph, let us suppose that there exists an algorithm ensuring reliable communication from p to q. Then, we necessarily have $DynMinCut\,(p, q) > 2k$.*

Proof. Let us suppose the opposite: there exists an algorithm ensuring reliable communication from p to q, and yet, $DynMinCut\,(p,q) \leq 2k$. Let us show that it leads to a contradiction.

As we have $DynMinCut\,(p,q) = MinCut\,(Dyn\,(p,q)) \leq 2k$ and $MinCut\,(Dyn\,(p,q)) = \min_{C \in Cut(Dyn(p,q))} card(C)$, there exists an element C of $Cut\,(Dyn\,(p,q))$ such that $card(C) \leq 2k$. Let C_1 be a subset of C containing k' elements, with $k' = \min(k, card(C))$. Let $C_2 = C - C_1$. Thus, we have $card(C_1) \leq k$ and $card(C_2) \leq k$.

According to the definition of $Cut\,(Dyn\,(p,q))$, C contains a node of each possible dynamic path from p to q. Therefore, the information that q receives about p are completely determined by the behavior of the nodes in C.

Let us consider two possible placements of Byzantine nodes, and show that they lead to a contradiction:

- First, suppose that all nodes in C_1 are Byzantine, and that all other nodes are correct. This is possible since $card(C_1) \leq k$.

  Suppose now that p broadcasts a message m. Then, according to our hypothesis, since the algorithm ensures reliable communication, q eventually accepts m from p, regardless of what the behavior of the nodes in C_1 may be.

- Now, suppose that all nodes in C_2 are Byzantine, and that all other nodes are correct. This is also possible since $card(C_2) \leq k$.

  Then, suppose that p broadcasts a message $m' \neq m$, and that the Byzantine nodes have exactly the same behavior as the nodes of C_2 had in the previous case.

  Thus, as the information that q receives about p is completely determined by the behavior of the nodes of C, from the point of view of q, this situation is indistinguishable from the previous one: the nodes of C_2 have the same behavior, and the behavior of the nodes of C_1 is unimportant. Thus, similarly to the previous case, q eventually accepts m from p.

Therefore, in the second situation, p broadcasts m, and q eventually accepts $m' \neq m$. Thus, the algorithm does not ensure reliable communication, which contradicts our initial hypothesis. Hence, the result. □

Lemma 8.4 (Safety)**.** *Let us suppose that all correct nodes follow our algorithm. If* $(p,m) \in q.Acc$*, then* $m = p.m_0$.

Proof. As $(p, m) \in q.Acc$, according to rule 3 of our algorithm, there exists $\{S_1, \ldots, S_n\}$ such that, $\forall i \in \{1, \ldots, n\}$, $(p, m, S_i \cup \{p\}) \in q.\Omega$, and $MinCut\left(\{S_1, \ldots, S_n\}\right) > k$.

Suppose that each node set $S \in \{S_1, \ldots, S_n\}$ contains at least one Byzantine node. If C is the set of Byzantine nodes, then $C \in Cut\left(\{S_1, \ldots, S_n\}\right)$ and $card(C) \leq k$. This is impossible because $MinCut\left(\{S_1, \ldots, S_n\}\right) > k$. Therefore, there exists $S \in \{S_1, \ldots, S_n\}$ such that S does not contain any Byzantine node.

Now, let us use the correct dynamic path corresponding to S to show that $m = m_0$. Let $n' = card(S \cup \{p\})$. Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{0, \ldots, n'\}$: there exists a correct node u_i and a set of correct nodes X_i such that $(p, m, X_i) \in u_i.\Omega$ and $card(X_i) = card(S \cup \{p\}) - i$.

- As $S \in \{S_1, \ldots, S_n\}$, $(p, m, S \cup \{p\}) \in q.\Omega$. Thus, $\mathcal{P}_0$ is true if we take $u_0 = q$ and $X_0 = S \cup \{p\}$.
- Let us now suppose that $\mathcal{P}_{i+1}$ is true, for $i < n'$. As $(p, m, X_i) \in u_i.\Omega$, according to rule 2 of our algorithm, it implies that u_i received Ω' from a node v, with $(p, m, X) \in \Omega'$, $v \notin X$ and $X_i = X \cup \{v\}$. Thus, $card(X) = card(X_i) - 1 = card(S \cup \{p\}) - (i + 1)$.

  As $v \in X_i$ and X_i is a set of correct nodes, v is correct and behaves according to our algorithm. Then, as v sent Ω', according to rule 1 of our algorithm, we necessarily have $\Omega' \subseteq v.\Omega$. Thus, as $(p, m, X) \in \Omega'$, we have $(p, m, X) \in v.\Omega$. Hence, $\mathcal{P}_{i+1}$ is true if we take $u_{i+1} = v$ and $X_{i+1} = X$.

By induction principle, $\mathcal{P}_{n'}$ is true. As $card(X_{n'}) = 0$, $X_{n'} = \emptyset$ and $(p, m, \emptyset) \in u_{n'}$. As $u_{n'}$ is a correct node and follows our algorithm, the only possibility to have $(p, m, \emptyset) \in u_{n'}.\Omega$ is that $u_{n'} = p$ and $m = p.m_0$. Thus, the result. □

Lemma 8.5 (Communication)**.** *Let us suppose that* $DynMinCut\left(p, q\right) > 2k$*, and that all correct nodes follow our algorithm. Then, we eventually have* $(p, p.m_0) \in q.Acc$.

Proof. Let $\{S_1, \ldots, S_n\}$ be the set of node sets $S \in Dyn\left(p, q\right)$ that only contain correct nodes. Similarly, let $\{X_1, \ldots, X_{n'}\}$ be the set of node sets $X \in Dyn\left(p, q\right)$ that contain at least one Byzantine node.

Let us suppose that $MinCut\left(\{S_1, \ldots, S_n\}\right) \leq k$. Then, there exists $C \in Cut\left(\{S_1, \ldots, S_n\}\right)$ such that $card(C) \leq k$. Let C' be the set containing the nodes of C and the Byzantine nodes. Thus, and $C' \in Cut\left(\{S_1, \ldots, S_n\} \cup \{X_1, \ldots, X_{n'}\}\right) = Cut\left(Dyn\left(p, q\right)\right)$, and $card(C') \leq 2k$. Thus, $MinCut\left(Dyn\left(p, q\right)\right) \leq 2k$, which contradicts our hypothesis. Therefore, $MinCut\left(\{S_1, \ldots, S_n\}\right) > k$.

In the following, we show that $\forall S \in \{S_1, \ldots, S_n\}$, we eventually have $(p, p.m_0, S \cup \{p\}) \in q.\Omega$, ensuring that q eventually accepts $p.m_0$ from p.

Let $S \in \{S_1, \ldots, S_n\}$. As $S \in Dyn\,(p, q)$, let $(u_1, \ldots, u_N)$ be the dynamic path such that $p = u_1$, $q = u_N$ and $S = \{u_2, \ldots, u_{N-1}\}$. Let $(t_1, \ldots, t_N)$ be the corresponding dates, according to Definition 8.1. Let us show the following property $\mathcal{P}_i$ by induction, $\forall i \in \{1, \ldots, N\}$: at date t_i, $(p, p.m_0, X_i) \in u_i.\Omega$, with $X_i = \emptyset$ if $i = 1$ and $\{u_1, \ldots, u_{i-1}\}$ otherwise.

- $\mathcal{P}_1$ is true, as we initially have $(p, p.m_0, \emptyset) \in p.\Omega$.
- Let us suppose that $\mathcal{P}_i$ is true, for $i < N$. According to Definition 8.1, $\forall t \in [t_i, t_i + \zeta(t_i, u_i)]$, $\rho(e_i, t) = 1$, e_i being the edge connecting u_i to u_{i+1}.
  - Let $t_A \leq t_i$ be the earliest date such that, $\forall t \in [t_A, t_i + \zeta(t_i, u_i)]$, $\rho(e_i, t) = 1$.
  - Let $t_B \leq t_i$ be the date where (p, m, X_i) is added to $u_i.\Omega$.
  - Let $t_C = max(t_A, t_B)$.

  Then, at date t_C, either $u_i.\Omega$ or the local topology topology of u_i changes. Thus, according to rule 1 of our algorithm, u_i multicasts $\Omega' = u_i.\Omega$ at date t_C, with $(p, p.m_0, X_i) \in \Omega'$.

  As $\zeta(e_i, t_i) \leq t_{i+1} - t_i \leq t_{i+1} - t_C$, u_{i+1} receives Ω' from u_i at date $t_C + \zeta(e_i, t_i) \leq t_{i+1}$. Then, according to rule 2 of our algorithm, $(p, p.m_0, X_i \cup \{u_i\})$ is added to $u_{i+1}.\Omega$.

  Thus, $\mathcal{P}_{i+1}$ is true if we take $X_{i+1} = X_i \cup \{u_i\}$.

By induction principle, $\mathcal{P}_N$ is true. As $u_1 = p$, $X_N = \{u_1, \ldots, u_{N-1}\} = S \cup \{p\}$, and we eventually have $(p, p.m_0, S \cup \{p\}) \in q.\Omega$.

Thus, $\forall S \in \{S_1, \ldots, S_n\}$, we eventually have $(p, p.m_0, S \cup \{p\}) \in q.\Omega$. Then, as $MinCut\,(\{S_1, \ldots, S_n\}) > k$, according to rule 3 of our algorithm, $(p, p.m_0)$ is added to $q.Acc$. □

Lemma 8.6 (Sufficient condition)**.** *Let there be any dynamic graph. Let p and q be two correct nodes, and k denote the maximum number of Byzantine nodes. If $DynMinCut\,(p, q) > 2k$, our algorithm ensures reliable communication from p to q.*

Proof. Let us suppose that the correct nodes follow our algorithm, as described in Section 8.2. First, according to Lemma 8.4, if $(p, m) \in q.Acc$, then $m = p.m_0$. Thus, when q accepts a message from p, p is necessarily the author of this message. Then,

according to Lemma 8.5, we eventually have $(p, p.m_0) \in q.Acc$. Thus, q eventually receives and accepts the message broadcast by p. Therefore, our algorithm ensures reliable communication from p to q. □

8.3 Case Studies

In this section, we apply the aforementioned condition to several case studies: cyclic network, participants interacting in a conference, robots moving on a grid and agents in the subway. We thus show the benefit of using a multihop algorithm for reliable communication.

8.3.1 A deterministic dynamic toy network

Let $n > 0$, and let $(p_1, \ldots, p_n)$ and $(q_1, \ldots, q_n)$ be two sequences of nodes. We consider the dynamic network $\mathcal{T}_n$ where, at date $t \in \{0, 1, 2, \ldots\}$, p_i is connected to $q_{i+t \mod n}$. This is illustrated in Figure 8.2. Using our main theorem (Theorem 8.2), we are able to exactly characterize the Byzantine resilience of $\mathcal{T}_n$.

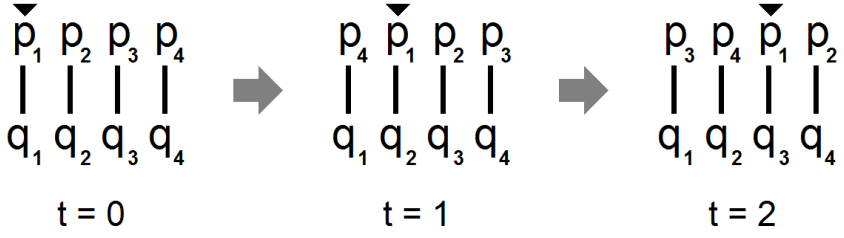

FIGURE 8.2: Case study: a deterministic dynamic toy network $\mathcal{T}_4$.

Theorem 8.7. *In $\mathcal{T}_n$, to ensure reliable communication between any two pairs of correct nodes, it is necessary and sufficient that $n > 2k$ and $t \geq 2k + n - 1$, where k denote the maximum number of Byzantine nodes in the network.*

Proof. Let $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_n\}$. Let u and v be two nodes.

- If $u \in P$ and $v \in Q$, let i and d be such that $u = p_i$ and $v = q_{i+d \mod n}$. Thus, $DynMinCut(u, v) = 0$ if $t < d$, and $+\infty$ otherwise. The same holds if $u \in Q$ and $v \in P$ (by symmetry).
- If $u \in Q$ and $v \in Q$, let i and d be such that $u = q_i$ and $v = q_{i+d \mod n}$. Thus, $DynMinCut(u, v) = 0$ if $t < d$, and $min(t - d + 1, n)$ otherwise. The same holds if $u \in P$ and $v \in P$ (by symmetry).

Thus, as the maximal value of d is $n-1$ (*e.g.*, when $u = q_1$ and $v = q_n$), $m = min_{(u,v)\in V\times V}\ DynMinCut\,(u,v) = 0$ if $t < n-1$, and $min(t-n+2, n)$ otherwise. Now, according to Theorem 8.2, $m > 2k$ is necessary and sufficient to enable reliable communication between any pair of correct nodes.

First, let us show that the condition of Theorem 8.7 is necessary. Let us suppose the opposite: $n \leq 2k$ or $t < 2k+n-1$, and $m > 2k$. Then, if $n \leq 2k$, as $m \leq n$, we get $m \leq 2k$: a contradiction. If $t < 2k+n-1$ and $k = 0$, then $t < n-1$ and $m = 0$: a contradiction. Hence, the condition is necessary.

Then, let us show that the condition of Theorem 8.7 is sufficient. As $t \geq 2k+n-1 \geq n-1$, we have $m = min(t-n+2, n)$. Besides, as $t \geq 2k+n-1$, it foàllows that $t-n+2 \geq 2k$. Thus, as $n > 2k$, we have $m > 2k$, and the condition is sufficient. □

In particular, with $t = 2n$, we can tolerate roughly one fourth of Byzantine nodes.

8.3.2 A real-life dynamic network: the Infocom 2005 dataset

In this section, we consider the Infocom 2005 dataset [49], which is obtained in a conference scenario by iMotes capturing contacts between participants. This dataset can represent a dynamic network where each participant is a node and where each contact is a (temporal) edge. We consider an 8-hour period during the second day of the conference. In this period, we consider the dynamic network formed by the 10 most "sociable" nodes (our criteria of sociability is the total number of contacts reported). We assume that at most one on these nodes may be Byzantine (that is, $k = 1$).

Let p and q be two correct nodes. Let us suppose that p wants to transmit a message to q within a period of 10 minutes. After 10 minutes, three types of communication can be achieved:

- *Simple communication*: there exists a dynamic path from p to q.
- *Reliable communication*: the condition for reliable communication from p to q identified in Theorem 8.2 is satisfied.
- *Direct communication*: p meets q directly.

If we want to ensure reliable communication despite one Byzantine node, the simplest strategy is to wait until p meets q directly. Let us show now that relaying the message (*e.g.* using our algorithm as presented in Section 8.2) is usually beneficial and that our approach realizes a significant gain of performance.

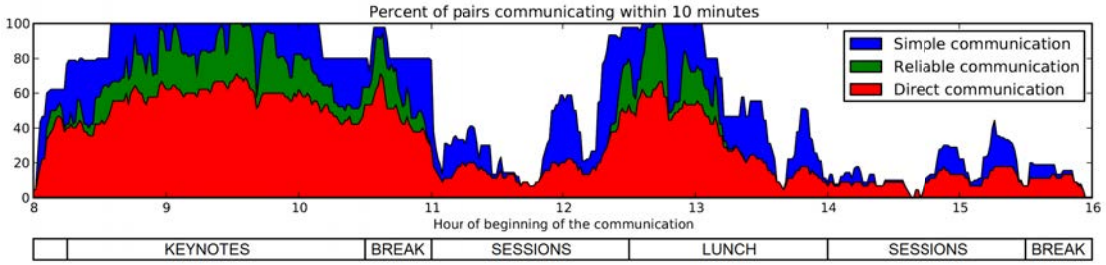

FIGURE 8.3: Reliable communication between 10 most sociable nodes of the Infocom 2005.

Figure 8.3 represents the percentage of pairs of nodes (p, q) that communicate within 10 minutes, according to the date of beginning of the communication. We can correlate the peaks with the program of the conference: the first period corresponds to morning arrivals during the keynotes; the peak between 10:30 and 11:00 corresponds to the morning break; the peak starting at 12:30 corresponds to the end of parallel sessions and the departure for lunch. As it turns out, many pairs of nodes are able to communicate reliably, even though they are unable to meet directly. For instance, at 9:30, all pairs of nodes are effectively able to reliably exchange information, even though only two thirds of them come into direct contact. This means that relaying the information is actually effective and desirable.

8.3.3 Probabilistic mobile robots on a grid

We consider a network of 10 mobile robots that are initially randomly scattered on a square grid.

At each time unit, a robot randomly moves to a neighbor vertex, or does not move (the new position is chosen uniformly at random among all possible choices). Let $position(u, t)$ be the current vertex of the robot u at date t. We consider that two robots can communicate if and only if they are on the same vertex. Our setting induces the following dynamic graph $\mathcal{G} = (V, E, \rho, \zeta)$: $V = \{u_1, \ldots, u_{10}\}$, $E = V \times V$, $\rho((u, v), t) = 1$ when $position(u, t) = position(v, t)$ and $\zeta((u, v), t) = 0$.

Let p and q be two correct robots, and suppose that up to k other robots are Byzantine. We aim at evaluating the *communication time*, that is: the mean time to have $DynMinCut\,(p, q) > 2k$ (Our condition for reliable communication established in Theorem 8.2). For this purpose, we ran more than 10000 simulations, and represented the results on Figure 8.4, 8.5, 8.6 and 8.7. Let us comment on these results.

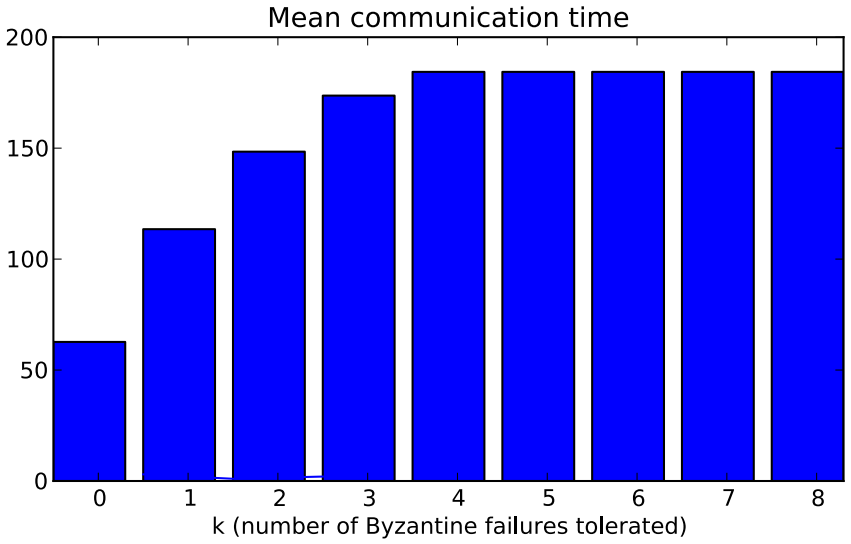

FIGURE 8.4: Mean communication time (10×10 grid)

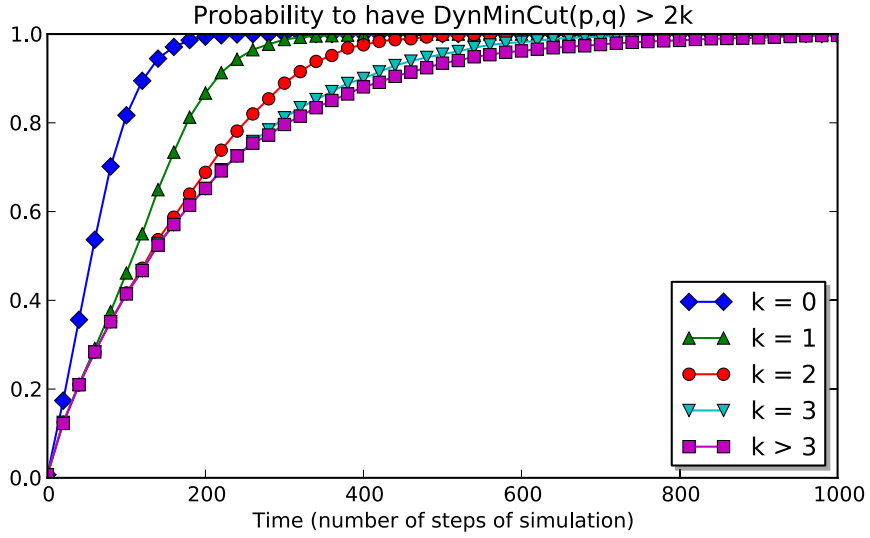

FIGURE 8.5: Probability to satisfy our condition for reliable communication (10×10 grid)

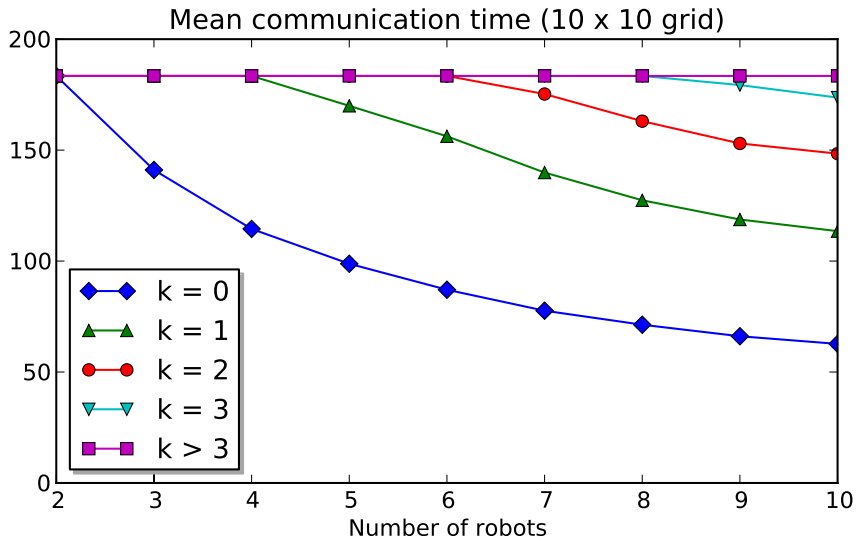

FIGURE 8.6: Mean communication time depending on the number of robots

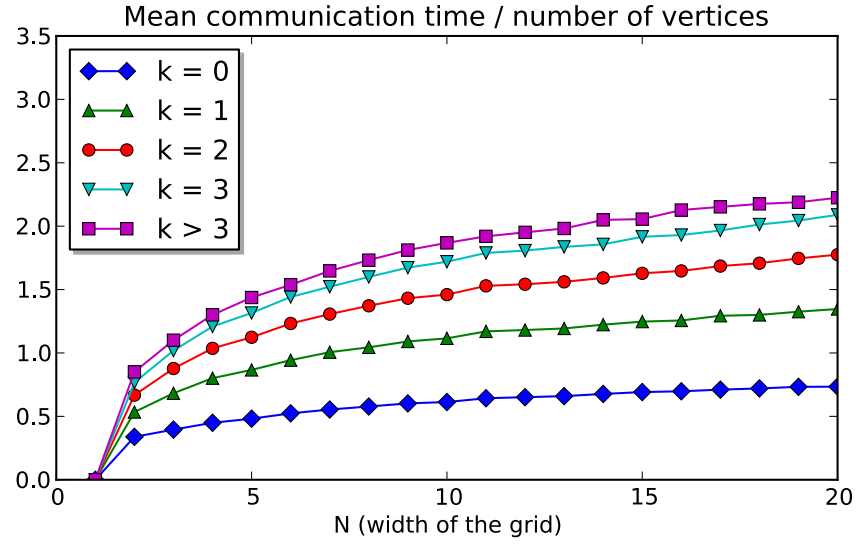

FIGURE 8.7: Mean communication time divided by the number of vertices

First, Figure 8.4 represents the mean communication time on a 10×10 grid, for all possible values of k. The time first increases with k, then stabilizes for $k > 3$. Indeed, for $k > 3$, due to the number of robots, the condition $DynMinCut\,(p, q) > 2k$ is satisfied if and only if p and q are on the same vertex: reliable *multihop* communication is impossible and only source to destination *direct* communication is feasible.

Then, Figure 8.5 represents the cumulative probability to satisfy our reliable communication condition on a 10×10 grid, with respect to time. As expected, this probability decreases when k increases. We also notice that this probability increases linearly at first with respect to time.

Also, Figure 8.6 represents the mean communication time according to the number of robots. With only 2 robots, we must wait for the source to meet the sink directly. However, when the number of robots increases, reliable multihop communication becomes increasingly more interesting. Also, we notice that, for every two robots that we add, it becomes possible to tolerate one more Byzantine fault in multihop communication. This illustrates the condition $DynMinCut\,(p, q) > 2k$.

Finally, we study the influence of the size of the grid. We observe that the mean communication is roughly proportional to the number of vertices in the grid (that is, N^2 for a grid of width N). Figure 8.7 represents the ratio between the communication time and the number of vertices. This value seems to converge, or at least to increase very slowly with the size of the grid.

As we can see, the reliable multihop communication approach can be an interesting compromise. For instance, let us consider a 10×10 grid. The basic communication time is 63 time units. Now, let us suppose that we want to tolerate one Byzantine failure. If we wait for the source to meet directly with the sink, the mean communication times increases by 194% from the fault-free case. If we use our algorithm instead, it increases by only 81%.

8.3.4 Mobile agents in the Paris subway

We consider a dynamic network consisting of 10 mobile agents randomly moving in the Paris subway. The agents can use the classical subway lines (we exclude tramways and regional trains). Each agent is initially located at a randomly chosen junction station – that is, a station that connects at least two lines. Then, the agent randomly chooses a neighbor junction station, waits for the next train, moves to this station, and repeats the process. We use the train schedule provided by the local subway company (`http://data.ratp.fr`). The time is given in minutes from the departure of the first

train (*i.e.*, around 5:30). We consider that two agents can communicate in the two following cases:

1. They are staying together at the same station.
2. They cross each other in trains. For instance, if at a given time, one agent is in a train moving from station A to station B while the other agent moves from B to A, then we consider that they can communicate.

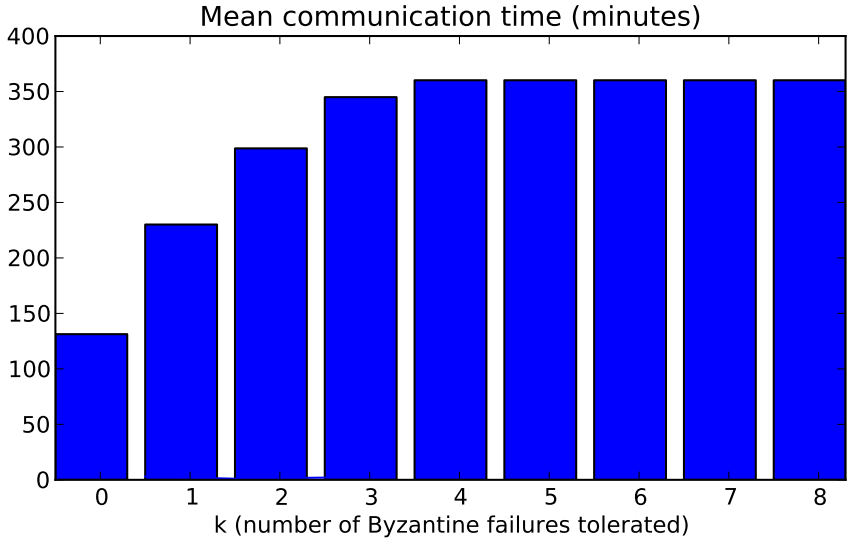

FIGURE 8.8: Mean communication time (subway)

We provide the same plots as in 8.3.3: the mean communication time (see Figure 8.8) and the probability to satisfy the condition for reliable communication (see Figure 8.9). The results are very similar to those of 8.3.3, which suggests that the topology used for the simulations has only a minor qualitative influence.

The basic communication time is 131 minutes. Again, let us suppose that we want to tolerate one Byzantine failure. If we wait for the source to meet the sink directly, the mean communication time increases by 174%. If we use our algorithm, it increases only by 75%, which shows that there is a clear benefit in terms of latency.

8.4 Conclusion

In this chapter, we gave a necessary and sufficient condition for reliable communication in a dynamic network that is subject to Byzantine failures. Unlike in static networks, it

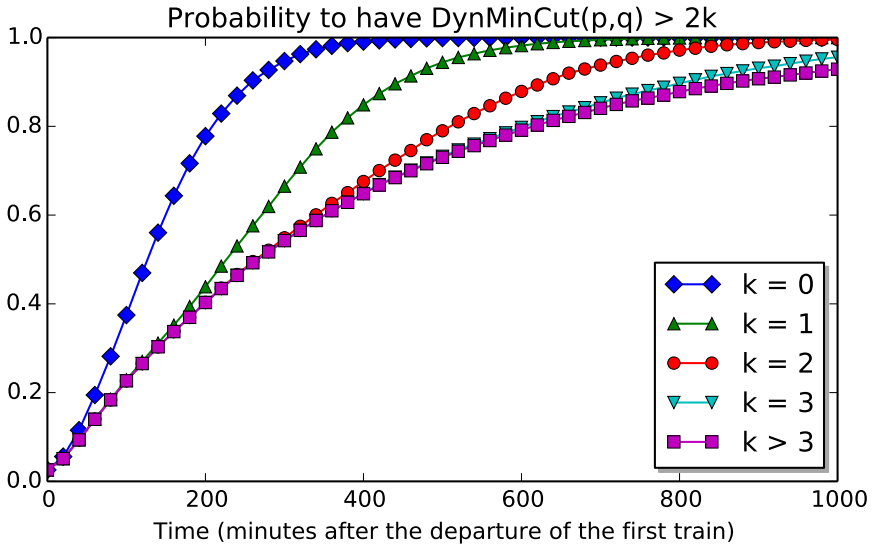

FIGURE 8.9: Probability to satisfy the condition for reliable communication (subway)

turns out the the existence of dynamic paths that are not node-disjoint is not necessarily harmful, as long as the dynamic minimal cut remains sufficiently high. The sufficiency part of our condition is constructive, as we provide an algorithm for optimally broadcasting a message in this context (with respect to the number of Byzantine nodes tolerated). We demonstrated the benefits of this protocol in several case studies, both in synthetic example and in real dynamic networks.

Our result implicitly considers a worst-case placement of the Byzantine nodes, which is the classical approach when studying Byzantine failures in a distributed setting. Studying variants of the Byzantine node placement, and the associated necessary and sufficient condition for enabling multihop reliable communication, constitutes an interesting path for future research.

Chapter 9

Fractal Byzantine tolerance

In this chapter, we propose the very first algorithm tolerating a uniform rate of Byzantine failures in an unbounded network.

Let us consider the following setting: a $N \times N$ grid network where each node has a probability λ to be Byzantine. In this setting, all previous solutions (including those presented in this dissertation) have the same weakness: when N approaches infinity, the probability that two correct nodes communicate reliably (*communication probability*) approaches zero.

We present the first algorithm having the following property: for $\lambda < 10^{-5}$, the communication probability is greater than $1 - 4\lambda$, independently of the size N of the grid. Therefore, this algorithm is scalable in term of Byzantine resilience.

The chapter is organized as follows:

- In Section 9.1, we describe our algorithm.
- In Section 9.2, we prove the claims.
- In Section 9.3, we extend our result to a 3-dimensional grid.

A first version on these results has been published in the ICDCN conference [50].

9.1 Algorithm

9.1.1 Informal description

Our protocol is defined on a sequence of grid networks $G_1, G_2, G_3 \ldots$ of increasing size, G_n being a $10^n \times 10^n$ grid. The main idea is to use the protocol on G_n to define the protocol on G_{n+1}, and so forth.

On G_1, we use an existing protocol that can tolerate one Byzantine failure on a 10×10 grid. Then, to each node p of G_n, we associate a cluster of nodes $G(p)$ on G_{n+1}, $G(p)$ being a 10×10 grid (see Figure 9.1, where this principle is illustrated with 3×3 grids). The idea is that $G(p)$ must simulate the behavior of p, and always tolerate at least one Byzantine failure.

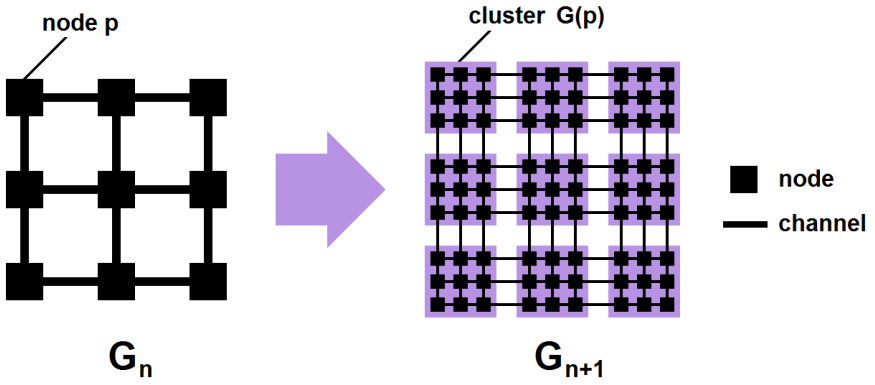

FIGURE 9.1: Principle of the protocol (with 3×3 grids)

For this purpose, we use the following mechanism. First, on each cluster $G(p)$, we use the same protocol as G_1, so that the nodes can locally broadcast messages in $G(p)$. In parallel, each node of $G(p)$ simulates the behavior of p. When the algorithm of p requires to send a message to a neighbor q, the border nodes of $G(p)$ send the message to their neighbor in $G(q)$. Then, each border node of $G(q)$ locally broadcasts the message. The nodes of $G(q)$ accept this message from p when they locally receive it from at least 6 border nodes.

Figure 9.2 shows how a message passing in G_n is simulated on G_{n+1}, with the aforementioned mechanism. The 6 confirmation messages ensure that the message passing is reliable, despite the presence of at most one Byzantine node per cluster (see Section 9.2).

The interest of this fractal definition lies in its probabilistic guarantees. Indeed, with a uniform rate of Byzantine failures, the communication probability takes the form of a

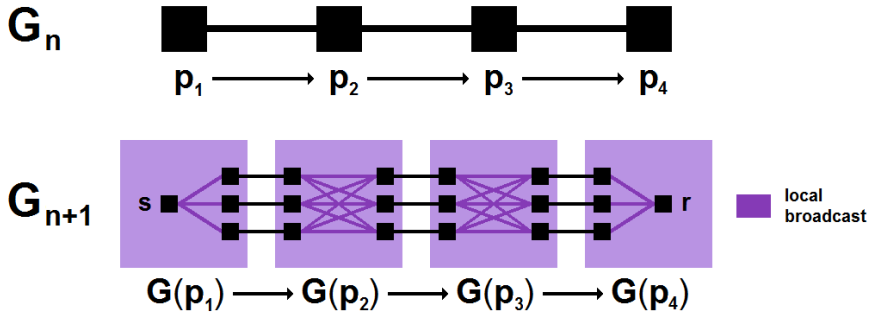

FIGURE 9.2: Example of message passing

n-factors product (see Theorem 9.2). This product is converging for a sufficiently low Byzantine rate.

9.1.2 Description of the algorithm

In our setting, each node knows its coordinates (i, j) on the grid.

Let G_n be a $10^n \times 10^n$ *grid* (with $n \geq 1$). To each node p of G_n located at (i, j), we associate a cluster of nodes $G(p)$ of G_{n+1}, such that the node (x, y) of $G(p)$ corresponds to the node $(10i + x, 10j + y)$ of G_{n+1}. Thus, each node of G_{n+1} belongs to a cluster.

As $G(p)$ is a 10×10 grid, the nodes of $G(p)$ use the protocol of G_1 to locally broadcast messages. On G_1, we use the protocol of Chapter 5 with setting $(1, 2)$. Now, let us explain how to construct the protocol of G_{n+1} with the protocol of G_n.

Inductive definition. When a node s_0 of G_{n+1} wants to broadcast a message m_0, it locally broadcasts m_0 in its cluster $G(p_0)$. The nodes of $G(p_0)$ accepting m_0 from s_0 consider that p_0 wants to broadcast $M_0 = (s_0, m_0)$. Then, in each cluster $G(p)$ of G_{n+1}, each correct node $u \in G(p)$ simulates the behavior of p as follows:

- When p wants to send M to q, if a neighbor v of u belongs to $G(q)$, u sends (q, M) to v.
- When u has accepted (q, M) from at least 6 nodes of $G(p)$ having a neighbor in $G(q)$, u considers that p received M from q.
- When p accepts (s_0, m_0) from p_0, u accepts m_0 from s_0.

9.2 Correctness proof

In this section, we prove the probabilistic guarantees of the algorithm.

For this purpose, we first provide a methodology that, for a given placement of Byzantine failures, returns a reliable node set. Then, we use this methodology to prove the probabilistic guarantees.

Similarly to the protocol, this methodology is defined by induction. To determine a reliable node set on G_1, we use the methodology described in Chapter 5. Then, assuming that we can determine a reliable node set on G_n, we explain how to determine a reliable node set on G_{n+1}.

Let us suppose that, for a given set $Corr$ of correct nodes on G_n, we have a function $Rel_n(Corr)$ returning a reliable node set. Let us use the function Rel_n to define a function Rel_{n+1}.

Lemma 9.1. *For a given set $Corr$ of correct nodes of G_{n+1}, the following function returns a reliable node set:*

$Rel_{n+1}(Corr) = \bigcup_{p \in Rel_n(Corr')} Rel_{G(p)}$, *where ...*

- *$Rel_{G(p)}$ is a reliable node set on $G(p)$.*
- *$Corr'$ is the sets of nodes p of G_n such that $G(p)$ contains at most one Byzantine node.*

Proof. First, let us show that the behavior of $Rel_n(Corr')$ is perfectly simulated by the sets $Rel_{G(p)}$, $p \in Rel_n(Corr')$. Indeed, let us suppose the opposite. Then, there exists a node $u \in Rel_{G(p)}$ such that ...

1. Either p reaches a state that u cannot reach.
2. Or u reaches a state that p cannot reach.

Let us show that both cases lead to a contradiction.

(1) Let p be the first node to reach a state that a node $u \in Rel_{G(p)}$ cannot reach. It implies that p received a message m from a neighbor q that u cannot receive. As u is the first node in this situation, all the nodes of $Rel_{G(q)}$ having a neighbor v in $G(p)$ eventually send m to v. As $G(p)$ and $G(q)$ contain at most one Byzantine node, by analyzing all possible cases, we show that we always have at least 6 nodes of $Rel_{G(q)}$

with a neighbor in $Rel_{G(p)}$. Thus, at least 6 nodes of $Rel_{G(p)}$ locally broadcast (q, m). Then, as $Rel_{G(p)}$ is a reliable node set, according to the protocol, u eventually receives m from q: contradiction.

(2) Let $u \in Rel_{G(p)}$ be the first node to reach a state that p cannot reach. It implies that u received a message that p cannot receive. Thus, according to the protocol, u accepted a message from at least 6 nodes of $G(p)$ having a neighbor in a cluster $G(q)$. As $G(p)$ and $G(q)$ contain at most one Byzantine node, by analyzing all possible cases, we show that we always have at least 6 nodes of $Rel_{G(p)}$ with a neighbor in $Rel_{G(q)}$. Thus, as only 10 nodes of $G(p)$ have a neighbor in $G(q)$, at least one node of $Rel_{G(q)}$ sent a message that q cannot send. So u is not the first node in this situation: contradiction.

Now, let s_0 and r_0 be two nodes of $Rel_{n+1}(Corr)$. Let $G(p_0)$ and $G(q_0)$ be the clusters containing theses nodes. As $Rel_n(Corr')$ is a reliable node set, q_0 accepts (s_0, m_0) from p_0 if and only if p_0 broadcasts (s_0, m_0). Thus, as the behavior of $Rel_n(Corr')$ is perfectly simulated by $Rel_{n+1}(Corr)$, r accepts m_0 if and only if s broadcasts m_0. Thus, the result.

□

Theorem 9.2. *For a Byzantine rate $\lambda < 10^{-5}$, the communication probability is greater than $1 - 4\lambda$.*

Proof. Let $\mu = 1 - \lambda$. First, let us evaluate the probability $P(\mu)$ that a correct node of a cluster $G(p)$ belongs to $Rel_{G(p)}$. According to Lemma 9.1, $Rel_{G(p)}$ exists if at most one node of $G(p)$ is Byzantine. Thus, as $G(p)$ contains 100 nodes, $P(\mu) = \mu^{100} + 100\alpha(1 - \mu)\mu^{99}$, α being the probability that a correct node belongs to $Rel_{G(p)}$ when $G(p)$ contains exactly one Byzantine node. Let us consider all possible placements of this Byzantine node: in 64 cases (resp. 32 and 4), $Rel_{G(p)}$ contains 99 nodes (resp. 98 and 96). Thus, we have $\alpha = (64 \times 99 + 32 \times 98 + 4 \times 96)/(99 \times 100)$.

Now, let us evaluate the fraction $F_n(\mu)$ of correct nodes that belong to $Rel_n(Corr)$. Let us show by induction that $F_n(\mu) \geq \prod_{i=1}^{i=n} P^i(\mu)$, P^i being the i^{th} application of the function P. The property is true for $n = 1$, as $F_1(\mu) = P(\mu)$. Now, let us suppose that the property is true at rank n. According to Lemma 9.1, a correct node u belongs to $Rel_{n+1}(Corr)$ if and only if $u \in Rel_{G(p)}$ and $p \in Rel_n(Corr')$. The event $u \in Rel_G(p)$ happens with probability $P(\mu)$, and if so, the event $p \in Rel_n(Corr')$ happens with a probability greater than $F_n(P(\mu))$. Thus, $F_{n+1}(\mu) \geq P(\mu)F_n(P(\mu)) = \prod_{i=1}^{i=n+1} P^i(\mu)$, and the property is true at rank $n + 1$.

At last, let us give a lower bound of this n-factors product. The function $P(\mu) = \mu^{100} + 100\alpha(1-\mu)\mu^{99}$ is convex for $\mu > 1-10^{-5}$: $P''(\mu) < 0$ (where P'' is the second derivative of P), and thus $P(\mu) > f(k,\mu) = 1 - k(1-\mu)$, with $k = (1 - P(1-10^{-5}))/(1-10^{-5})$. Then, by induction, $P^n(\mu) \geq f(k^n, \mu)$, and therefore $F_n(\mu) \geq \prod_{i=1}^{i=n} f(k^n, \mu)$.

As $\forall n \geq 6$, $k^n \leq 1/n^2$, we have $F_n(\mu)$ ...

- $\geq \prod_{i=1}^{i=6} f(k^n,\mu) \prod_{i=7}^{i=n}(1-\lambda/i^2)$
- $\geq \prod_{i=1}^{i=6} f(k^n,\mu)\ sin(\pi\sqrt{\lambda})/\sqrt{\lambda}$ (Wallis formula [51])
- $\geq (1-\sqrt{2}\lambda)(1-\sqrt{2}\lambda)$
- $\geq 1 - 2\lambda$

A sufficient condition for two correct node to communicate reliably is that they belong to $Rel_n(Corr)$. Thus, the communication probability is greater than $F_n(\mu)^2 \geq 1 - 4\lambda$.

□

For instance, if $\lambda < 10^{-5}$, the communication probability if greater than 0.9999.

9.3 The 3D grid case

The aforementioned result assumes a 2-dimensional grid topology. However, our scheme can easily be extended to other regular topologies, where the structure repeats itself at each new scale. We illustrate this using a 3-dimensional grid topology.

Definition 9.3 (3-dimensional grid network)**.** A $N \times N \times N$ grid is a network such that:

- Each node has a unique identifier (i,j,k) with $0 \leq i < N$, $0 \leq j < N$ and $0 \leq k < N$.
- Two nodes (i_1,j_1,k_1) and (i_2,j_2,k_2) are neighbors if and only if one of these three conditions is satisfied:
  - $i_1 = i_2$, $j_1 = j_2$ and $|k_1 - k_2| = 1$.
  - $j_1 = j_2$, $k_1 = k_2$ and $|i_1 - i_2| = 1$.
  - $k_1 = k_2$, $i_1 = i_2$ and $|j_1 - j_2| = 1$.

Here, let G_n be a $3^n \times 3^n \times 3^n$ grid (with $n \geq 1$). To each node p of G_n located at (i, j, k), we associate a cluster of nodes $G(p)$ of G_{n+1}, such that the node (x, y, z) of $G(p)$ corresponds to the node $(3i + x, 3j + y, 3k + z)$ of G_{n+1}. We keep using the same protocol.

Let us show that we obtain even better results with this new topology (the requirement about λ is $\lambda < 10^{-3}$ instead of $\lambda < 10^{-5}$).

Theorem 9.4. *In this new case, for a Byzantine rate $\lambda < 10^{-3}$, the communication probability is greater than $1 - 2\lambda$.*

Proof. Lemma 9.1 also remains valid: again, by analyzing all possible cases, we show that we always have at least 6 nodes of $Rel_{G(q)}$ with a neighbor in $Rel_{G(p)}$.

The proof of Theorem 9.2 should be modified as follows:

- As $G(p)$ contains 27 nodes, $P(\mu) = \mu^{27} + 27\alpha(1 - \mu)\mu^{26}$.
- In all cases, $Rel_{G(p)}$ contains 26 nodes. Thus, $\alpha = 1$ here.
- As $P(\mu)$ is convex for $\mu > 1-10^{-3}$, $P(\mu) > f(k, \mu)$, with $k = (1-P(1-10^{-3}))/(1-10^{-3})$.
- As $\forall n \geq 1$, $k^n \leq 1/n^2$, we have $F_n(\mu) \geq \prod_{i=1}^{i=n}(1 - \lambda/i^2) \geq 1 - \sqrt{2}\lambda$. Thus, the communication probability is greater than $1 - 2\lambda$.

□

9.4 Conclusion

In this chapter, we proved that it is possible to tolerate a uniform Byzantine rate in an unbounded network. Our approach is constructive and has been exemplified in 2D and 3D grid networks. As an open problem, we have the strong intuition that our approach could be generalized to less regular topologies, such as planar graphs. Also, this fractal approach for Byzantine broadcast could also be transposed to other distributed problems, such as consensus, leader election, mutual exclusion, etc.

Chapter 10

Conclusion

In this chapter, we summarize the contributions of this thesis, and discuss about future perspectives. We also make the list of our publications during this thesis.

10.1 Summary of contributions

Part I: Quantitative Byzantine tolerance

In this part, we proposed a quantitative approach to bypass the difficulty of Byzantine tolerance in sparse network. The idea is that, since we accept that some nodes may be Byzantine, we can reasonably accept that some correct nodes do not deliver the correct messages, provided that a majority of correct nodes achieve reliable communication.

We considered two settings. In the first case (Chapter 4), the nodes have a sufficient level of topology knowledge to compute *control zones* that filter Byzantine messages. In the second case (Chapter 5), we released this hypothesis, and proposed a strategy based on multiple confirmation paths.

In both cases, we gave a theoretical methodology that, for a given set on Byzantine nodes, returns a set of nodes communicating reliably. This methodology is necessary to correctly evaluate the algorithms, as we cannot make hypotheses on the behavior of Byzantine nodes.

Finally, we made a statistical evaluation of both solutions, and showed that they significantly outperform previous solutions in the case of randomly distributed Byzantine failures. Also, we notice that having a certain degree of topology knowledge (control zones versus fixed disjoint paths) is an important advantage in terms of Byzantine resilience.

Part II: Qualitative Byzantine tolerance

In this part, we came back to the classical approach where all correct nodes are required to communicate reliably. An existing criteria is on the *density* of Byzantine failures, which may be represented by the fraction of Byzantine neighbors per correct node. However, in sparse network, this criteria may be difficult or impossible to satisfy. Thus, we considered a more tolerant density criteria: the *distance* between Byzantine failures.

In Chapter 6, we studied the relationship between this distance and a particular cycle decomposition of the network. We showed that reliable communication was possible when the minimal distance D between two Byzantine failures is twice the diameter of the largest cycle (for instance, $D > 4$ in a torus network). We then generalized this algorithm to make it self-stabilizing, that is: resilient to both transient and permanent failures.

In Chapter 7, we considered the case where this distance criteria is locally violated by some *critical pairs* of Byzantine nodes. We showed that such critical pairs can still be tolerated, provided that they are themselves sufficiently distant. This recursive strategy is a first step towards the fractal algorithm proposed in Chapter 9.

Part II: Extensions

In this part, we extended our scope in two directions.

In Chapter 8, we considered the case of dynamic networks, where the topology can vary with time. We generalized the condition for reliable communication in the presence of k arbitrarily placed Byzantine failures to this new setting. Then, with several case studies, we showed the interest of multihop reliable communication, instead of waiting that the source meets the sink directly (assuming that it is possible).

In Chapter 9, we considered the case of a network which size is not bounded, and can grow indefinitely. We proposed the first algorithm that can tolerate a uniform rate of Byzantine failures in this setting. For this purpose, we gave a fractal structure to our algorithm, which, we believe, could also be used to solve several other distributed problems.

10.2 Perspectives

At last, let us discuss about future research perspectives raised by our works.

Application to real life networks. In an important part of our works, we consider regular network topologies: grids, lattices, planar graphs... Indeed, the regularity of these topologies is an advantage to prove theoretical results and to easily perform simulations. Such regular networks exist in several applications (such as hub grids for distributed computation). However, an important class of modern networks are less regular: sensor networks, social networks...

Yet, such networks still have a certain level of regularity. For instance, fire sensors in a forest are scattered in a roughly homogeneous way, and the average members of a social network have roughly the same number of friends. Thus, it may be possible to grasp this regularity and extend our approaches to such graphs.

Probabilistic analysis. Most of our works (except Chapter 9) provide deterministic theoretical results (probabilistic guarantees are obtained by simulation). Indeed, lattice networks are not an easy setting for elegant probabilistic proofs. However, it would be interesting to consider networks where the topology itself obeys to probabilistic parameters. This way, we could be able to directly obtain probabilistic guarantees, varying parameters such as the diameter, the average node degree...

Combining strategies. We have investigated several strategies of fault tolerance in this dissertation, and a lot of other strategies exist. Thus, an interesting idea would be to combine two or more strategies to improve their guarantees. For instance, we could try to combine the local vote and the vote on multiple paths.

Increasing the resilience of the network. In Chapter 6 and 8, we have seen that it was possible to combine Byzantine tolerance with self-stabilization (in other words, tolerance to transient failures) and with dynamic changes in the network topology. To go further, we could combine both aforementioned settings, and also add other constraints: nodes joining and leaving to network, homonymy, anonymity... Overcoming all these difficulties together would achieve the highest possible level of fault tolerance.

10.3 List of publications

Here is the current list of works published during this PhD (November 2014). All these publications have gone through a peer review process (except for the invited paper).

10.3.1 International journals

- Alexandre Maurer and Sébastien Tixeuil, *Containing Byzantine Failures with Control Zones*. In IEEE Transactions on Parallel and Distributed Systems (TPDS), February 2014.
- Alexandre Maurer and Sébastien Tixeuil, *Byzantine Broadcast with Fixed Disjoint Paths*. In Journal of Parallel and Distributed Computing (JPDC), November 2014.
- Alexandre Maurer and Sébastien Tixeuil, *Tolerating random Byzantine failures in an unbounded network*. In Parallel Processing Letters (PPL), accepted with minor revisions.

10.3.2 International conferences

- Alexandre Maurer and Sébastien Tixeuil, *Limiting Byzantine Influence in Multihop Asynchronous Networks*. In IEEE International Conference on Distributed Computing Systems (ICDCS 2012).
- Alexandre Maurer and Sébastien Tixeuil, *On Byzantine Broadcast in Loosely Connected Networks*. In International Symposium on Distributed Computing (DISC 2012).
- Alexandre Maurer and Sébastien Tixeuil, *A Scalable Byzantine Grid*. In International Conference on Distributed Computing and Networking (ICDCN 2013).
- Alexandre Maurer and Toshimitsu Masuzawa, *Edge coloring despite transient and permanent faults*. In International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2014).
- Alexandre Maurer and Sébastien Tixeuil, *Self-Stabilizing Byzantine Broadcast*. In IEEE Symposium on Reliable Distributed Systems (SRDS 2014).

10.3.3 French conferences

- Alexandre Maurer and Sébastien Tixeuil, *Confinement de fautes Byzantines dans les réseaux multi-sauts asynchrones*. In Rencontres Francophones pour les Aspects Algorithmiques des Télécommunications (AlgoTel 2012).
- Alexandre Maurer and Sébastien Tixeuil, *Tolérer les fautes Byzantines dans les graphes planaires*. In Rencontres Francophones pour les Aspects Algorithmiques des Télécommunications (AlgoTel 2013).

10.3.4 Invited paper

- Alexandre Maurer and Sébastien Tixeuil, *Dependable Information Broadcast in Sparsely Connected Networks.* In International Conference on Latin American Dependable Computing (LADC 2013)

Chapter 11

Résumé de la thèse en français

Ce chapitre est un résumé de la thèse en français. Il suit le plan général de la thèse, on peut donc se référer aux parties correspondantes pour davantage de détails.

Introduction

Dans le monde actuel, les réseaux informatiques prennent de plus en plus d'importance. On pense bien sûr à Internet, mais il existe bien d'autres applications : réseaux de capteurs, grilles de calcul, réseaux opportunistes de téléphones cellulaires, réseaux de robots explorant des environnements dangereux...

Cependant, à mesure que les réseaux s'étendent, ils deviennent de plus en plus susceptibles de défaillir. En effet, les nœuds du réseau peuvent être sujets à des pannes, attaques, corruptions de mémoire... On est face à un problème de taille, car les systèmes complexes sont souvent "aussi fragiles que leur plus petit composant".

Pour faire face à cette difficulté, le concept de *tolérance aux fautes* a été introduit [1]. L'idée est de concevoir des systèmes qui continuent de fonctionner normalement même si un ou plusieurs éléments défaillent.

De nombreux modèles de fautes existent, mais afin de tous les englober, nous avons choisi d'étudier le modèle de faute le plus général possible: le modèle *Byzantin* [5], où les nœuds fautifs ont un comportement totalement arbitraire (donc imprévisible et potentiellement malveillant). Tolérer des nœuds Byzantins implique donc de garantir qu'il n'existe *aucune* stratégie, aussi improbable soit-elle, permettant à ces nœuds de déstabiliser le réseau.

Dans cette thèse, notre objectif est de permettre aux nœuds corrects (non-Byzantins) de disséminer une information de façon fiable à travers le réseau. Cette information peut être la mesure d'un capteur, la position d'un robot, le résultat d'un calcul local... Cette dissémination est illustrée en Figure 11.1.

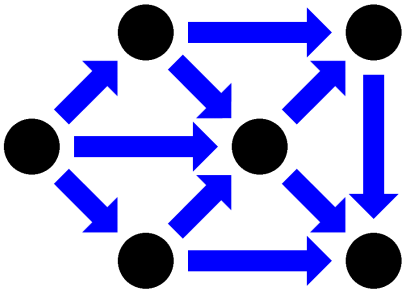

FIGURE 11.1: Exemple de dissémination d'information dans un réseau multi-sauts. Le nœud de gauche envoie un message "bleu" à ses voisins, qui le transmettent à leurs propres voisins, et ainsi de suite, jusqu'à ce que tous les nœuds reçoivent le message.

Dans ce contexte, les fautes Byzantines sont extrêmement dangereuses. En effet, une seule faute Byzantine, si elle n'est pas neutralisée, peut potentiellement mentir à l'ensemble du réseau, en diffusant ou retransmettant de fausses informations. Cela est illustré dans la Figure 11.2.

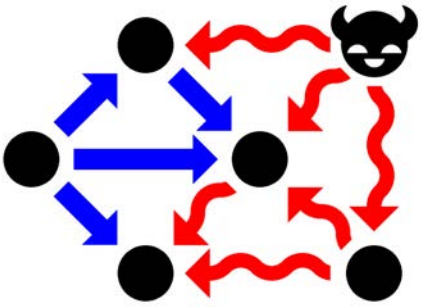

FIGURE 11.2: Exemple de dissémination d'information avec un nœud Byzantin. Ici, le nœud en haut à droite est Byzantin, et diffuse un message "rouge" afin de faire croire au réseau que le nœud de gauche a envoyé ce message (ce qui n'est pas le cas).

Par conséquent, il nous faut concevoir des algorithmes de communication *tolérants* aux fautes Byzantines. L'approche classique pour concevoir des algorithmes est de les implémenter puis de les tester. Cependant, dans le cas de fautes Byzantines, cela impliquerait de faire des hypothèses restrictives sur le comportement des nœuds Byzantins, et rien ne garantirait alors que l'on prend en compte les pires situations possibles.

On adopte donc une approche plus théorique : l'*algorithmique distribuée* [6–8]. L'idée est de prouver des propriétés mathématiques sur les algorithmes que l'on propose – par

exemple, montrer qu'aucun message incorrect ne sera jamais accepté, quelle que soit la configuration. On peut ainsi obtenir des garanties très fortes sur la fiabilité du réseau.

Dans la section suivante, nous passons en revue les solutions existantes et expliquons notre contribution. Nous présentons ensuite nos résultats dans les Parties I, II et III.

Travaux existants et notre contribution

Une manière de neutraliser des nœuds malveillants est de recourir à la *cryptographie* [9, 10] : les nœuds utilisent des signatures numériques pour authentifier l'émetteur à travers les multiples retransmissions. La cryptographie a de nombreux avantages, mais nous avons choisi de la laisser de côté pour plusieurs raisons.

Tout d'abord, la cryptographie n'est pas inconditionnellement fiable, comme l'a montré le récent bug *Heartbleed* [18]. Il est donc intéressant de combiner des couches de communication cryptographiques avec des couches non-cryptographiques (paradigme de la "défense en profondeur"). Par ailleurs, la cryptographie implique un élément central qui distribue les clés cryptographiques. Par conséquence, si cet élément défaille, l'ensemble du réseau défaille. Or nous voulons un système où chaque élément peut défaillir indépendamment des autres, sans pour autant compromettre le fonctionnement global.

On s'intéresse donc aux solutions non-cryptographiques. On peut les regrouper en deux familles : le *vote local* et le *vote sur chemins disjoints*.

- Le *vote local* [32–35] consiste à ne transmettre un message que si on l'a reçu de k voisins différents. Ainsi, si il y a moins de k voisins Byzantins par nœud correct, ils ne peuvent jamais collaborer pour initier la diffusion d'un message mensonger. Cela est illustré en Figure 11.3.

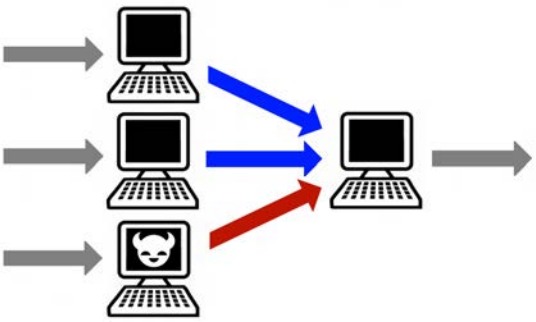

FIGURE 11.3: Vote local avec $k = 2$.

- Le *vote sur chemins disjoints* [36, 37] consiste à envoyer un message de l'émetteur au récepteur en empruntant plusieurs chemins disjoints, puis à effectuer un vote à la réception. Ainsi, si une minorité de chemins sont corrompus par un nœud Byzantin, cela est sans importance, car la majorité de votes corrects l'emportera. Cela est illustré en Figure 11.4.

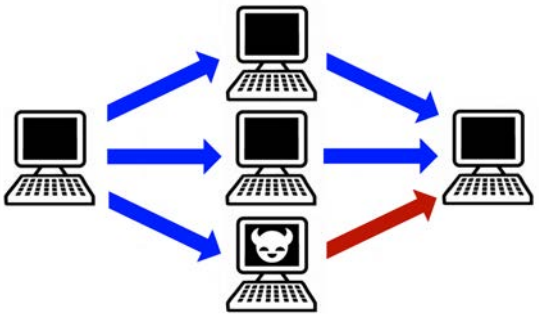

FIGURE 11.4: Exemple de vote sur chemins disjoints.

Ces solutions ont cependant des limites. En effet, elles nécessitent des réseaux fortement *connectés*, où chaque nœud possède un grand nombre de voisins. On peut donc légitimement se poser la question suivante : qu'advient-il de ces solutions dans un réseau plus faiblement connecté ? On prendra l'exemple de la *grille* (voir Figure 11.5), où chaque nœud a au plus 4 voisins.

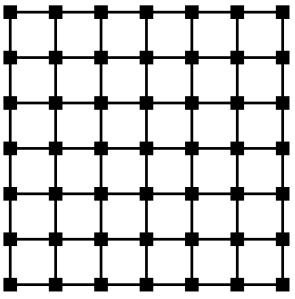

FIGURE 11.5: Une grille 7×7.

Sur une grille :

- Le vote local ne fonctionne pas. En effet, même dans la version la moins exigeante de cet algorithme ($k = 2$), seuls les 8 nœuds entourant l'émetteur peuvent accepter le message. Au-delà, il n'y a pas assez de canaux pour obtenir les 2 confirmations requises.

- Le vote sur chemins disjoints fonctionne, mais tolère au plus une faute Byzantine. En effet, à partir de 2 fautes Byzantines, le récepteur est susceptible de recevoir 2 votes corrects et 2 votes mensongers, et ne peut plus décider.

Cela nous amène au fil rouge de cette thèse : concevoir des algorithmes qui tolèrent les fautes Byzantines dans des réseaux faiblement connectés, où les solutions existantes ont de faibles performances.

Cette thèse est organisée en 3 parties :

- La Partie I présente des algorithmes qui tolèrent les fautes Byzantines de manière *quantitative* : on accepte la possibilité que certains nœuds corrects soient "sacrifiés" dans la foulée, afin de pouvoir tolérer un grand nombre de fautes Byzantines en contrepartie.
- La Partie II revient à une approche plus classique (tolérance *qualitative*), où tous les nœuds corrects doivent communiquer fiablement. On considère ici un critère de densité de fautes adapté aux réseaux faiblement connectés : la *distance* entre les nœuds Byzantins.
- Enfin, la Partie III étend des résultats existants à de nouveaux contextes : les réseaux dynamiques, et les réseaux dont la taille tend vers l'infini.

Partie I : Tolérance quantitative

Dans cette partie, nous proposons des solutions permettant de tolérer un grand nombre de fautes Byzantines dans des réseaux faiblement connectés. Afin d'y parvenir, nous faisons la concession suivante : on accepte qu'une minorité de nœuds corrects échouent à communiquer fiablement. On cherche donc à optimiser la *probabilité de communication* – c'est-à-dire, la probabilité d'avoir une communication fiable entre deux nœuds corrects.

Nous proposons deux solutions :

- La première utilise des ensembles de nœuds appelés *zones de contrôle*, dont le rôle est de "filtrer" les messages Byzantins. Cette solution tolère un grand nombre de fautes Byzantines, mais nécessite que les nœuds aient un certain degré de connaissance de la topologie du réseau.
- La seconde s'intéresse au cas où cette connaissance n'est pas disponible, et où les nœuds ignorent leur position dans le réseau. On propose alors une alternative basée sur l'utilisation de chemins disjoints bornés.

Zones de contrôle

Cette section résume les résultats du Chapitre 4. Ces résultats ont été publiés dans les conférences AlgoTel [40], ICDCS [41] et dans le journal TPDS [42].

Nous proposons ici un algorithme basé sur des *zones de contrôle*. Une zone de contrôle est constituée d'un ensemble de nœuds connexe, le "cœur", et d'une "frontière" qui isole le cœur du reste du réseau. L'idée est donc que tout nœud du cœur qui veut communiquer avec l'extérieur doit passer par la frontière. Un exemple de zone de contrôle est donné en Figure 11.6.

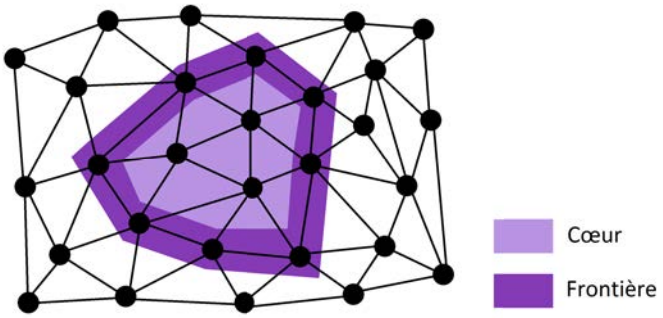

FIGURE 11.6: Exemple de zone de contrôle.

Le fonctionnement d'une zone de contrôle est illustré en Figure 11.7. Supposons qu'un nœud correct retransmette un message (p, m_0) – c'est-à-dire, un message affirmant qu'un nœud p a diffusé l'information m_0. Lorsque ce message pénètre dans la zone de contrôle, deux choses se produisent. D'une part, le message continue à se diffuser dans le cœur de la zone de contrôle. D'autre part, une *autorisation* pour ce message est diffusée sur la frontière. Lorsque ce message voudra sortir du cœur, cette autorisation sera requise.

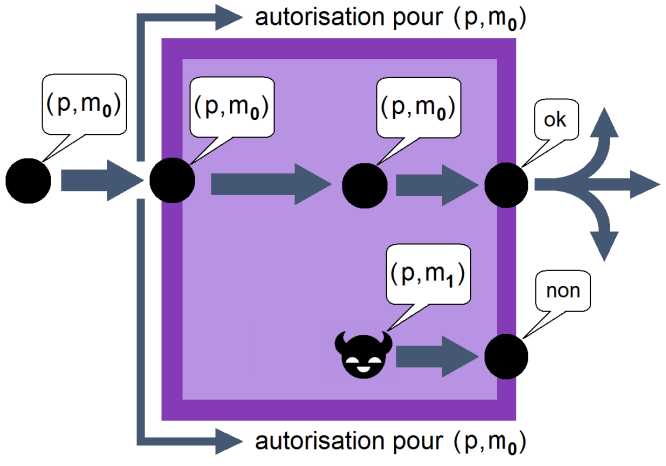

FIGURE 11.7: Principe de fonctionnement d'une zone de contrôle.

Supposons maintenant qu'un nœud Byzantin situé dans le cœur diffuse un message mensonger (p, m_1) – c'est-à-dire, un message prétendant que p a diffusé $m_1 \neq m_0$, ce qui n'est pas le cas. Comme ce message n'est pas issu de p mais du cœur de la zone de contrôle, il n'a jamais pénétré dans la zone, et aucune autorisation n'a été diffusée. Ainsi, ce message mensonger ne pourra jamais s'échapper du cœur. L'idée est bien sûr de définir, non pas une, mais un grand nombre de zones de contrôle, afin de minimiser la diffusion de messages mensongers.

On souhaite à présent évaluer les performances de cet algorithme. Comme dit plus haut, il n'est pas possible de simuler directement un algorithme en présence de fautes Byzantines, car cela implique de faire des restrictions hasardeuses sur le comportement des nœuds Byzantins. Pour surmonter cette difficulté, nous adoptons donc une approche hybride :

- Dans un premier temps, nous démontrons une formule qui, pour un réseau donné, un ensemble de zones de contrôle donné et un ensemble de nœuds Byzantins donné, détermine précisément l'ensemble des nœuds corrects qui communiqueront toujours fiablement quoiqu'il arrive.
- Dans un second temps, nous utilisons cette formule pour faire une évaluation statistique de notre algorithme : on génère plusieurs distributions aléatoires de nœuds Byzantins, et pour chacune, on détermine la fraction de nœuds corrects qui communiquent fiablement. Sur un grand nombre d'expériences, on peut ainsi estimer la probabilité de communication avec une bonne précision.

Nous appliquons cette méthodologie à une grille 100×100, avec des zones de contrôle carrées. Nous mettons en lumière un phénomène intéressant : si il y a trop peu de zones de contrôle, les performances sont mauvaises, mais si il y en a trop, c'est également le cas. Il semble donc exister un nombre optimal de zones de contrôle.

Nous comparons ensuite cet algorithme aux solutions précédentes dans le même contexte. Si l'on souhaite obtenir une probabilité de communication de 99%, on peut tolérer plus de 120 fautes Byzantines avec des zones de contrôle, contre moins de 5 avec les solutions existantes.

Chemins disjoints bornés

Cette section résume les résultats du Chapitre 5. Ces résultats ont été publiés dans la conférence DISC [43] (version partielle) et dans le journal JPDC [44] (version complète).

Nous considérons ici un contexte où les nœuds ignorent leur position dans le réseau, et ne peuvent donc pas constituer de zones de contrôle. C'est par exemple le cas dans un réseau de robots ou de capteurs éparpillés aléatoirement sur un terrain inconnu.

Nous proposons donc l'algorithme suivant : pour accepter et retransmettre un message, un nœud doit le recevoir de plusieurs nœuds distincts, par plusieurs chemins disjoints et de longueur bornée. Par exemple, sur la Figure 11.8, pour accepter et retransmettre le message, le nœud de droite doit le recevoir via un premier chemin d'au plus $H_1 = 3$ sauts, un second chemin d'au plus $H_2 = 4$ sauts et un troisième chemin d'au plus $H_3 = 2$ sauts.

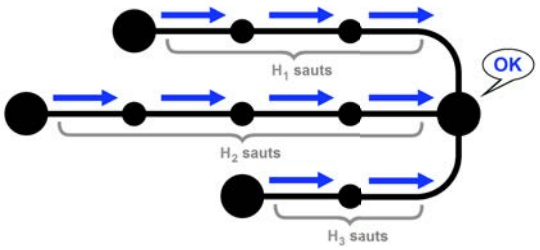

FIGURE 11.8: Principe de l'algorithme.

Ce paramétrage particulier peut être décrit par le triplet $(3, 4, 2)$. Pour être plus général, nous considérons par la suite un paramétrage quelconque $(H_1, \ldots, H_n)$. Bien entendu, si ces n nœuds sont Byzantins, alors ils peuvent collaborer pour initier la diffusion d'un message mensonger. Mais s'ils sont suffisamment distants les uns des autres, cela ne se produira jamais.

De la même façon que pour les zones de contrôle, nous prouvons tout d'abord une formule permettant de déterminer les nœuds corrects qui communiquent fiablement. Puis nous effectuons une évaluation statistique sur des tores (un tore étant une grille "continue"). Nous montrons un phénomène similaire à celui observé dans le cas des zones de contrôle : il semble exister un paramétrage optimal, ni trop simple, ni trop complexe.

Nous effectuons finalement une comparaison générale des solutions disponibles sur un tore 50×50. Si l'on veut une probabilité de communication de 99%, alors on peut tolérer un *taux de Byzantins* (probabilité qu'un nœud soit Byzantin) $\lambda = 4 \times 10^{-6}$ avec un algorithme de diffusion simple et non-sécurisé. Ce taux passe à 5×10^{-5} avec les solutions précédentes, et à 2×10^{-3} avec notre algorithme (soit une amélioration de facteur 40).

Bien entendu, on obtient de meilleurs résultats avec les zones de contrôle ($\lambda = 8 \times 10^{-3}$), mais cela requiert des hypothèses supplémentaires.

Partie II : Tolérance qualitative

Dans cette partie, nous proposons des algorithmes qui garantissent une communication fiable entre *tous* les nœuds corrects.

De tels algorithmes ont été proposés dans [32–35]. Afin de prouver leurs garanties, ces solutions font l'hypothèse que la "densité" des fautes Byzantines est limitée de la manière suivante : la fraction de voisins Byzantins par nœud correct est bornée. Cependant, comme nous l'avons vu, ce critère est trop exigeant pour des réseaux faiblement connectés. Nous considérons donc un critère de densité plus souple : la distance minimale entre 2 nœuds Byzantins.

Nos contributions sont les suivantes :

- Nous proposons une décomposition particulière d'un réseau quelconque en cycles élémentaires, puis nous donnons un algorithme qui garantit une communication fiable lorsque la distance entre fautes Byzantines est plus de deux fois le diamètre du plus grand cycle. Nous rendons également cette solution *auto-stabilisante* [3].
- Nous étudions ensuite la possibilité de tolérer des "paires critiques", c'est-à-dire, des paires de nœuds Byzantins qui ne respectent pas la distance minimale. Nous montrons que nous pouvons également tolérer des telles paires critiques, pourvu qu'elles soient elles-mêmes suffisamment distantes.

Décomposition en cycles

Cette section résume les résultats du Chapitre 6. Ces résultats ont été publiés dans les conférences AlgoTel [46], DISC [43] (version partielle) et SRDS [45] (version complète).

Nous remarquons tout d'abord que tout graphe 3-connexe admet une décomposition en cycles élémentaires qui vérifie certaines propriétés. Un exemple simple est le tore, où les cycles élémentaires sont les "carrés" qui le composent. Un exemple plus général est le graphe planaire, où les cycles sont alors les polygones délimités par les arêtes (voir Figure 11.9). Mais une décomposition similaire existe également pour des graphes non-planaires.

Nous faisons alors l'hypothèse suivante sur le placement des nœuds Byzantins : la distance minimale entre deux nœuds Byzantins doit être plus de deux fois le diamètre du plus grand cycle. Par exemple, dans le cas du tore, cette distance doit être strictement supérieure à 4.

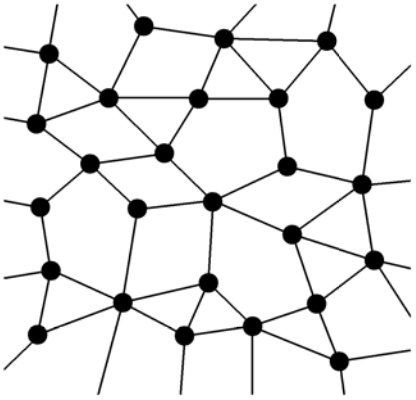

FIGURE 11.9: Exemple de graphe planaire.

Cette hypothèse nous permet d'avoir la propriété suivante : entre deux nœuds corrects, il existe toujours un "chemin cyclique correct" – c'est-à-dire, une suite de cycles formés de nœuds corrects telle que chaque cycle a au moins deux nœuds en commun avec le suivant. Cette propriété est centrale dans la preuve des algorithmes qui suivent.

Nous proposons donc un premier algorithme qui garantit une communication fiable dans ce contexte. Le principe de l'algorithme est qu'un nœud doit recevoir un message via deux chemins disjoints d'au plus Z sauts pour le retransmettre, Z étant le diamètre du plus grand cycle. Ainsi, la distance entre les nœuds Byzantins étant supérieure à $2Z$, ils ne peuvent jamais collaborer pour initier la diffusion d'un message mensonger. Par ailleurs, la propriété précédente sur les chemins cycliques assure la diffusion des messages corrects.

Nous rendons ensuite cet algorithme *auto-stabilisant* [3] : en plus de tolérer les fautes Byzantines permanentes, notre algorithme doit garantir une communication fiable à partir de n'importe quel état initial des nœuds corrects. Autrement dit, en plus de tolérer des fautes Byzantines permanentes, notre algorithme tolère également un nombre illimité de fautes *transitoires*, dont l'effet est représenté par l'état initial arbitraire du réseau. On atteint ainsi l'un des plus haut niveau possible de tolérance aux fautes.

Tolérer des paires critiques

Cette section résume les résultats du Chapitre 7. Ce travail est une collaboration avec le Pr. Masuzawa de l'université d'Osaka (Japon).

On considère un tore 8-connexe, où chaque nœud est voisin avec les 8 nœuds qui l'entourent. Dans un tel réseau, l'algorithme précédent garantit une communication fiable lorsque la distance minimale entre deux fautes Byzantines est supérieure à 2.

Cependant, cette garantie s'effondre si nous avons une "paire critique", c'est-à-dire, une paire de nœuds Byzantins qui sont distants de 2 sauts ou moins.

Nous proposons ici un algorithme capable de tolérer de telles paires critiques, pourvu qu'elles soient elles-mêmes suffisamment distantes. Cela est illustré sur la Figure 11.10. L'algorithme précédent ne garantit une communication fiable que dans le cas A. Celui que nous proposons ici garantit une communication fiable dans les cas A et B.

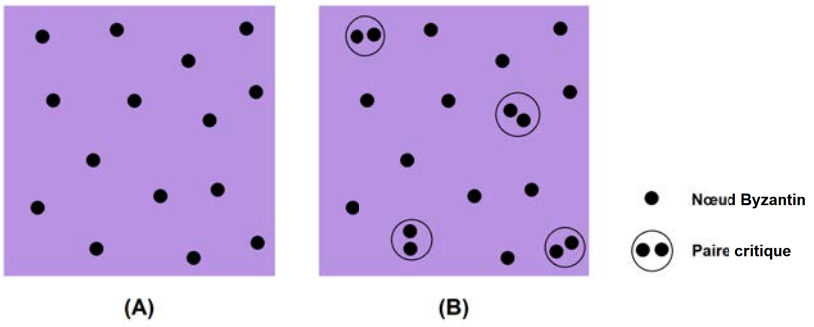

FIGURE 11.10: Amélioration de l'algorithme.

Le principe de l'algorithme peut être vu comme une récursion du vote local. Dans le cas le plus simple du vote local, un message est retransmis s'il est reçu de la part de deux voisins distincts. Ici, un message est retransmis s'il est reçu par deux parcours disjoints, de longueur bornée, qui correspondent chacun au mécanisme de propagation du vote local. Cela est illustré en Figure 11.11.

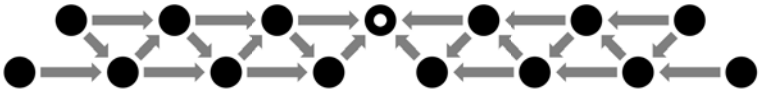

FIGURE 11.11: Principe de l'algorithme.

Ainsi, pour qu'un message mensonger se diffuse, il faut la collaboration de deux paires critiques suffisamment proches. Nous montrons que si la distance minimale entre deux paires critiques est supérieure à 23, nous avons une communication fiable. Par la suite, nous pouvons imaginer des algorithmes qui continuent d'appliquer ce principe récursivement, ou utilisent le même principe avec plus de deux votes.

Partie III : Extensions

Dans cette dernière partie, nous élargissons des solutions existantes à de nouveaux contextes. Nos contributions sont les suivantes :

- Nous généralisons la condition de communication fiable en présence de k fautes Byzantines aux réseaux *dynamiques* [47], où la topologie varie au cours du temps. Nous appliquons ensuite cette condition à divers cas d'étude : participants qui interagissent dans une conférence, robots se déplaçant sur une grille et agents dans le métro parisien.

- Nous considérons le cas d'un réseau dont la taille peut être augmentée à l'infini. Dans un tel contexte, aucun algorithme existant ne permet de tolérer un taux de Byzantins constant. Nous proposons un mécanisme fractal qui permet de résoudre ce problème pour des taux de Byzantins suffisamment faibles.

Réseaux dynamiques

Cette section résume les résultats du Chapitre 8. Ce travail est une collaboration avec le Pr. Defago du Japan Advanced Institute of Science and Technology.

La condition pour tolérer k fautes Byzantines arbitrairement placées dans un réseau statique est qu'il existe $2k+1$ chemins disjoints entre l'émetteur et le récepteur [36, 37]. La preuve de ce résultat repose sur le théorème de Menger [38], qui garantit l'équivalence entre coupe minimale et connectivité. Cependant, ce théorème n'est pas généralisable aux réseaux dynamiques [48]. Cela est illustré sur la Figure 11.12, où il faut supprimer au minimum 2 nœuds pour déconnecter l'émetteur du récepteur, mais où il est impossible de trouver 2 chemins disjoints.

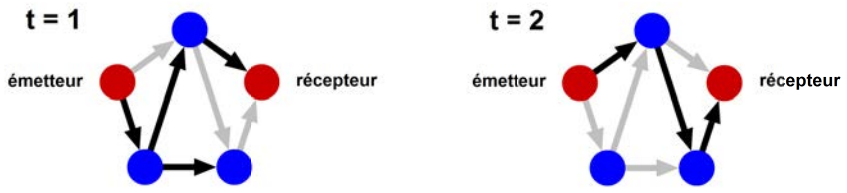

FIGURE 11.12: Contre-exemple du théorème de Menger dans les graphes dynamiques.

Nous généralisons ici cette condition aux réseaux dynamiques. Pour cela, nous introduisons la notion de *chemin dynamique* : un chemin le long duquel un message peut se propager, en prenant en compte les délais de transmission et la dynamicité du réseau. La condition est alors l'existence d'un ensemble de chemins dynamiques entre l'émetteur et le récepteur, de telle sorte qu'il soit impossible d'interrompre tous ces chemins en supprimant $2k$ nœuds. Cette condition est nécessaire et suffisante. Nous donnons également un algorithme de communication fiable pour le cas où cette condition est satisfaite.

Dans un second temps, nous appliquons cette condition à plusieurs cas d'étude. Nous considérons tout d'abord le graphe des interactions entre participants de la conférence Infocom 2005 [49]. On s'intéresse aux dix participants les plus "sociables", et à leur capacité à communiquer dans un intervalle de 10 minutes. Les résultats sont représentés en Figure 11.13.

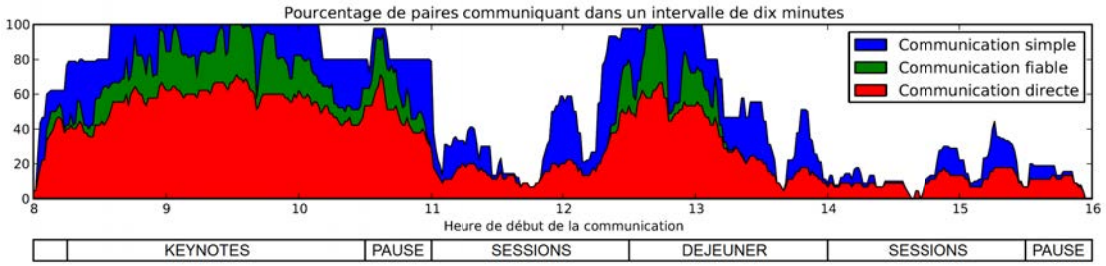

FIGURE 11.13: Communication fiable lors de la conférence Infocom 2005.

La communication est dite *directe* lorsque deux participants se rencontrent, *simple* lorsqu'ils peuvent échanger un message indirectement, et *fiable* lorsque cette communication indirecte tolère une faute Byzantine. Ainsi, si on veut tolérer une faute Byzantine, il est intéressant d'utiliser notre algorithme plutôt que d'attendre que l'émetteur et le récepteur se rencontrent en personne. Par exemple, à 9h30, tous les participants parviennent à communiquer fiablement de cette façon, alors que seulement deux tiers se rencontrent directement.

On considère ensuite un réseau de 10 robots mobiles se déplaçant aléatoirement sur une grille 10×10. Il faut en moyenne 63 unités de temps pour une communication simple. Pour une communication directe, ce délai augmente de 194%, alors qu'il n'augmente que de 81% pour une communication fiable. On obtient des résultats similaires avec des agents se déplaçant aléatoirement dans le métro parisien, en prenant en compte les horaires des rames. Dans les deux cas, notre algorithme permet un gain de temps notable par rapport à la communication directe.

Tolérance fractale

Cette section résume les résultats du Chapitre 9. Une partie de ces résultats ont été publiés dans la conférence ICDCN [50].

Nous considérons ici une grille où chaque nœud a une probabilité λ d'être Byzantin. Cela peut représenter, par exemple, le taux de défaillance de chaque nœud. Dans ce contexte, toutes les solutions précédentes, y compris celles proposées dans cette thèse, ont la même faiblesse : pour un λ donné, quand la taille de la grille augmente, la probabilité de communication tend vers 0.

En effet, pour chacune de ces solutions, il existe un "motif critique" de nœuds Byzantins qui, sur une grille, entraîne une diffusion incontrôlée de messages mensongers. Pour un λ constant, lorsque la taille de la grille augmente, la probabilité d'avoir un tel motif critique tend vers 1, et par conséquent, la probabilité de communication tend vers 0.

Nous proposons ici le premier algorithme qui surmonte ce problème. Sa propriété est la suivante : pour $\lambda < 10^{-5}$, la probabilité de communication est toujours supérieure à $1 - 4\lambda$, quelle que soit la taille de la grille. Nous avons ainsi une solution scalable en terme de tolérance aux Byzantins.

Pour cela, nous utilisons un algorithme fractal. Nous considérons une série de grilles G_n de taille croissante, G_n étant une grille $10^n \times 10^n$. Ainsi, à chaque nœud p de G_{n+1}, on peut associer une sous-grille $G(p)$ de taille 10×10 sur G_{n+1}. Cela est illustré sur la Figure 11.14, avec des grilles 3×3.

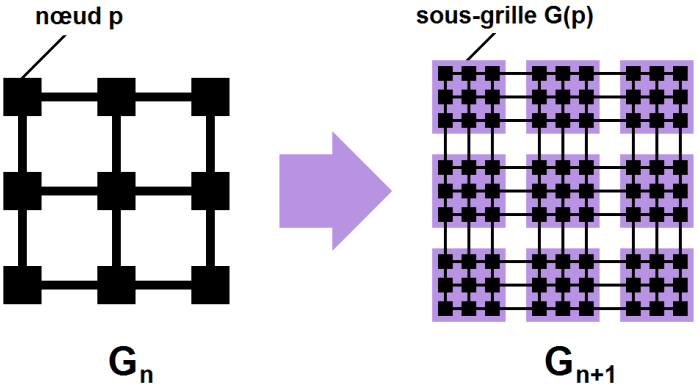

FIGURE 11.14: Découpage fractal de la grille.

L'algorithme est défini de la façon suivante. Sur G_1, nous utilisons un algorithme existant. Puis, nous utilisons l'algorithme sur G_n pour définir l'algorithme sur G_{n+1}. Cette définition fractale nous permet d'avoir une probabilité de communication sous la forme d'un produit infini convergent, d'où les résultats. On montre également des résultats similaires sur une grille 3D.

Conclusion

Dans cette thèse, nous nous sommes intéressés au problème de la communication fiable en présence de fautes Byzantines dans les réseaux faiblement connectés. Nous avons tout d'abord proposé des solutions quantitatives qui tolèrent un grand nombre de fautes Byzantines et assurent une communication fiable entre une majorité de nœuds corrects. Nous avons ensuite proposé des solutions qualitatives basées sur la distance minimale

entre les nœuds Byzantins. Enfin, nous avons généralisé des résultats existants aux réseaux dynamiques et aux réseaux de taille non-bornée.

Ces résultats conduisent à plusieurs perspectives de recherche. Tout d'abord, nous pouvons adapter ces solutions à des réseaux réels, à la topologie moins régulière. Nous pouvons également envisager une analyse probabiliste théorique de ces algorithmes. Enfin, nous pouvons combiner ces approches entre elles et ajouter de nouvelles contraintes afin d'atteindre un degré de tolérance aux fautes encore plus élevé.

Bibliography

[1] Peter Alan Lee and Thomas Anderson. Fault tolerance, principles and practice. *Dependable Computing and Fault-Tolerant Systems*, 3:51–77, 1990.

[2] Flavin Cristian. Understanding fault-tolerant distributed systems. *Communications of the ACM CACM Homepage archive*, 34(2):56–78, 1991.

[3] Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000.

[4] P.A. Santeler, K.A. Jansen, and S.P. Olarig. Fault tolerant memory, April 24 2001. URL http://www.google.com/patents/US6223301. US Patent 6,223,301.

[5] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[6] Gerard Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2000.

[7] Nicola Santoro. *Design and Analysis of Distributed Algorithms*. Wiley, 2007.

[8] Michel Raynal. *Distributed Algorithms for Message-Passing Systems*. Springer, 2013.

[9] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186, 1999.

[10] Vadim Drabkin, Roy Friedman, and Marc Segal. Efficient byzantine broadcast in wireless ad-hoc networks. In *DSN*, pages 160–169. IEEE Computer Society, 2005. ISBN 0-7695-2282-3.

[11] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18(3):219–246, 1982.

[12] R. Perlman. *Network layer protocols with byzantine robustness*. CSAIL Publications and Digital Archive, 1989.

[13] Nikodin Ristanovic, Panos Papadimitratos, George Theodorakopoulos, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Adaptive message authentication for multi-hop networks. *International Conference on Wireless On-Demand Network Systems and Services*, 2011.

[14] John Talbot and Dominic Welsh. *Complexity and Cryptography: An Introduction*. Cambridge University Press, 2006.

[15] Peter Alan Lee and Thomas Anderson. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[16] G. de Meulenaer, F. Gosset, O.-X. Standaert, and O. Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *IEEE International Conference on Wireless and Mobile Computing*, 2008.

[17] G.T. Sibley, M.H. Rahimi, and G. Sukhatme. Robomote: a tiny mobile robot platform for large-scale ad-hoc sensor networks. In *IEEE International Conference on Robotics and Automation*, 2002.

[18] The Heartbleed Bug (http://heartbleed.com).

[19] R. Lippmann, K. Ingols, C. Scott, and K. Piwowarski. Validating and restoring defense in depth using attack graphs. *IEEE Military Communications Conference*, 2006.

[20] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. McGraw-Hill Publishing Company, New York, May 1998. ISBN 0-07-709352. URL http://www.cs.technion.ac.il/~hagit/DC/. 6.

[21] D. Malkhi, Y. Mansour, and M.K. Reiter. Diffusion without false rumors: on propagating updates in a Byzantine environment. *Theoretical Computer Science*, 299(1–3):289–306, April 2003. ISSN 0304-3975.

[22] D. Malkhi, M. Reiter, O. Rodeh, and Y. Sella. Efficient update diffusion in byzantine environments. In *The 20th IEEE Symposium on Reliable Distributed Systems (SRDS '01)*, pages 90–98, Washington - Brussels - Tokyo, October 2001. IEEE. ISBN 0-7695-1366-2.

[23] Y. Minsky and F.B. Schneider. Tolerating malicious gossip. *Distributed Computing*, 16(1):49–68, 2003.

[24] Toshimitsu Masuzawa and Sébastien Tixeuil. Stabilizing link-coloration of arbitrary networks with unbounded byzantine faults. *International Journal of Principles and Applications of Information Science and Technology (PAIST)*, 1(1):1–13, December 2007. URL http://210.119.33.7/paist/paper/2008_12/TOC2008_12.pdf.

[25] Mikhail Nesterenko and Anish Arora. Tolerance to unbounded byzantine faults. In *21st Symposium on Reliable Distributed Systems (SRDS 2002)*, pages 22–29. IEEE Computer Society, 2002.

[26] Yusuke Sakurai, Fukuhito Ooshita, and Toshimitsu Masuzawa. A self-stabilizing link-coloring protocol resilient to byzantine faults in tree networks. In *Principles of Distributed Systems, 8th International Conference, OPODIS 2004*, volume 3544 of *Lecture Notes in Computer Science*, pages 283–298. Springer, 2005.

[27] Toshimitsu Masuzawa and Sébastien Tixeuil. Bounding the impact of unbounded attacks in stabilization. In Ajoy Kumar Datta and Maria Gradinariu, editors, *SSS*, volume 4280 of *Lecture Notes in Computer Science*, pages 440–453. Springer, 2006. ISBN 978-3-540-49018-0.

[28] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. Maximum metric spanning tree made byzantine tolerant. In David Peleg, editor, *Proceedings of DISC 2011*, Lecture Notes in Computer Science (LNCS), Rome, Italy, September 2011. Springer Berlin / Heidelberg. URL http://arxiv.org/abs/1104.5368.

[29] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. Bounding the impact of unbounded attacks in stabilization. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2011.

[30] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. On byzantine containment properties of the min+1 protocol. In *Proceedings of SSS 2010*, Lecture Notes in Computer Science, New York, NY, USA, September 2010. Springer Berlin / Heidelberg. URL http://hal.inria.fr/inria-00487091/PDF/DuboisMasuzawaTixeuil.pdf.

[31] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. The impact of topology on byzantine containment in stabilization. In *Proceedings of DISC 2010*, Lecture Notes in Computer Science, Boston, Massachusetts, USA, September 2010. Springer Berlin / Heidelberg. URL http://hal.inria.fr/inria-00481836/en/.

[32] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In Soma Chaudhuri and Shay Kutten, editors, *PODC*, pages 275–282. ACM, 2004. ISBN 1-58113-802-4.

[33] Vartika Bhandari and Nitin H. Vaidya. On reliable broadcast in a radio network. In Marcos Kawazoe Aguilera and James Aspnes, editors, *PODC*, pages 138–147. ACM, 2005.

[34] Andrzej Pelc and David Peleg. Broadcasting with locally bounded byzantine faults. *Inf. Process. Lett.*, 93(3):109–115, 2005.

[35] C. Litsas, A. Pagourtzis, G. Panagiotakos, and D. Sakavalas. On the resilience and uniqueness of CPA for secure broadcast (http://eprint.iacr.org/2013/738.pdf). 2013.

[36] D. Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.

[37] Mikhail Nesterenko and Sébastien Tixeuil. Discovering network topology in the presence of Byzantine faults. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 20(12):1777–1789, December 2009. URL http://doi.ieeecomputersociety.org/10.1109/TPDS.2009.25.

[38] T. Böhme, F. Göring, and J. Harant. Menger's theorem. *Journal of Graph Theory*, 37(1):35–36, 2001.

[39] Ian T. Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *IJHPCA*, 15(3):200–222, 2001.

[40] Alexandre Maurer and Sébastien Tixeuil. Confinement de fautes byzantines dans les réseaux multi-sauts asynchrones. In *14èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel)*, 2012.

[41] Alexandre Maurer and Sébastien Tixeuil. Limiting byzantine influence in multihop asynchronous networks. In *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS 2012)*, pages 183–192, June 2012.

[42] Alexandre Maurer and Sébastien Tixeuil. Containing byzantine failures with control zones. In *IEEE Transactions on Parallel and Distributed Systems*, 2014.

[43] Alexandre Maurer and Sébastien Tixeuil. On byzantine broadcast in loosely connected networks. In *Proceedings of the 26th International Symposium on Distributed Computing (DISC 2012)*, volume 7611 of *Lecture Notes in Computer Science*, pages 183–192. Springer, 2012.

[44] Alexandre Maurer and Sébastien Tixeuil. Byzantine broadcast with fixed disjoint paths. In *Journal of Parallel and Distributed Computing*, 2014.

[45] Alexandre Maurer and Sébastien Tixeuil. Self-stabilizing byzantine broadcast. In *33rd IEEE Symposium on Reliable Distributed Systems (SRDS 2014)*.

[46] Alexandre Maurer and Sébastien Tixeuil. Tolérer les fautes byzantines dans les graphes planaires. In *15èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel)*, 2013.

[47] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[48] David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4): 820–842, 2002.

[49] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *TMC*, 6(6):606–620, 2007.

[50] Alexandre Maurer and Sébastien Tixeuil. A scalable byzantine grid. In *Proceedings of the 14th International Conference on Distributed Computing and Networking (ICDCN 2013)*, volume 7730 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2013.

[51] Donat K. Kazarinoff. On wallis formula. In *Edinburgh Mathematical Notes*, volume 40, pages 19–21, 1956.