Tractable Reliable Communication in Compromised Networks

Giovanni Farina

SORBONNE UNIVERSITÉ
SAPIENZA UNIVERSITÀ DI ROMA

Écoles doctorales: ED130, Ecole Doctorale Informatique, Télécommunications et Electronique, EDITE de Paris *et* PhD school in engineering in computer science at Sapienza Università di Roma

Laboratoires : LIP6, Equipe NPA *et* Dipartimento di Ingegneria Informatica, Automatica e Gestionale (DIAG) "Antonio Ruberti"

TRACTABLE RELIABLE COMMUNICATION IN COMPROMISED NETWORKS

Par Giovanni Farina

Thèse de doctorat de informatique, télécommunications et electronique *et* engineering in computer science.

Dirigée par Prof. Silvia Bonomi et Prof. Sébastien Tixeuil

Présentée et soutenue publiquement le 21 Décembre 2020
Devant un jury composé de :

M. Pierre Sens	Professeur	Président
M. Luca Becchetti	Professeur	Examinateur
Mme. Maria Potop-Butucaru	Professeur	Examinateur
M. Andrea Vitaletti	Professeur	Examinateur
M. Xavier Défago	Professeur	Rapporteur
M. Roy Friedman	Professeur	Rapporteur
Mme. Silvia Bonomi	Professeur	Co-directeur de thèse
M. Sébastien Tixeuil	Professeur	Co-directeur de thèse

Abstract

Reliable communication is a fundamental primitive in distributed systems prone to Byzantine (i.e. arbitrary, and possibly malicious) failures to guarantee the integrity, delivery, and authorship of the messages exchanged between processes. Its practical adoption strongly depends on the system assumptions. Several solutions have been proposed so far in the literature implementing such a primitive, but some lack in scalability and/or demand topological network conditions computationally hard to be verified.

This thesis aims to investigate and address some of the open problems and challenges implementing such a communication primitive. Specifically, we analyze how a reliable communication primitive can be implemented in *1)* a static distributed system where a subset of processes is compromised, *2)* a dynamic distributed system where part of the processes is Byzantine faulty, and *3)* a static distributed system where every process can be compromised and recover. We define several more efficient protocols and we characterize alternative network conditions guaranteeing their correctness.

Contents

List of Symbols and Acronyms

SYMBOLS

t	instant;
n	number of processes;
p_i	process with identifier i;
P	set of all processes;
$\mathcal{P}$	protocol executed by the processes;
G	communication network (static case) or network footprint (dynamic case);
$\mathcal{G}$	communication network (dynamic);
V	vertex set of a graph G;
E	edge set of a graph G;
Δ	bound on the processing delay;
δ	bound on the transmission delay;
f	bound (global or local) on the number faulty processes;
λ	roaming pace of mobile agents;
$\Gamma(v_i)$	neighborhood of node v_i;
$\Pi_{i,j}$	disjoint path solution between nodes v_i, v_j;
K_m	complete graph of size m;
C_l	cycle graph of size l;
$W_{m,l}$	generalized wheel graph;
$MC_{k,l}$	multipartite cycle;

ACRONYMS

UL	Unicast Link
BL	Broadcast Link
TVG	Time Varying Graph
KLO	K-Level Ordering
MKLO	Minimum K-Level Ordering
TMKLO	Temporal Minimum K-Level Ordering
DKLO	Distance-based K Level Ordering
MBF	Mobile Byzantine Faults
KN	Known Neighborhood
UN	Unknown Neighborhood

Chapter 1

Introduction

Communication is "a process by which information is exchanged between individuals through a common system of symbols, signs, or behaviors" [1]. Our lives are ruled by information exchanges: our knowledge has been handed down and has been growing for millennia, our daily decisions, habits, and actions depend on the information we receive; we check the weather forecast, we make calls, we exchange documents, we read the news, we watch television, we interact in social networks, and so on. Information exchanges expand the knowledge available to individuals with respect to the local observation, enabling them to solve increasingly complex problems. Furthermore, communication may not only occur directly among entities, namely between a *sender* and a *receiver* through a *medium*, but it may also take place indirectly via *intermediates*, namely entities as well that take care of relaying information from one sender to a receiver. Non-direct communication further increases the potential interactions between entities, both speeding-up their information exchanges or allowing connections not possible otherwise. All the interactions that may occur between a set of entities define a *communication network*.

Given the countless advantages obtainable from information exchanges, for several decades we have been providing communication abilities (and computation capabilities) to as many devices as possible. We are setting up every day new networks of objects able to communicate, react, and cooperate: sensor networks, smart grids, healthcare systems, vehicular networks, smart cities, etc. We are additionally aiming to interconnect as many distinct networks as possible, again targeting to solve more complex problems.

Although there are innumerable tasks realizable through communication, it all depends on the "quality" of the information exchanged. There are at least three assumptions indeed we take for granted (or at least we would like to) in most communications: the *integrity*, the *authenticity*, and the *delivery* of the information exchanged. When we are on the phone, we would like to be able to clearly speak with who is on the other side and to be sure of his/her identity; when we receive a document, we want to be certain that its content has been sent by its claimed sender; when we listen to the news on tv, we would appreciate if they report exactly what really happened. More precisely, for most of the information we receive, we would like to be sure about its sender, and that it has not been alternated during

the exchange. Such properties, integrity, authenticity, and delivery are generally desired both in communications between humans and equally, if not more so, between devices, given that most of our interactions and businesses currently take place through them. An information exchange guaranteeing all of these three properties is commonly referred to as *reliable communication*.

An example of (generally) reliable communication is the face-to-face speaking between people: while talking with a person, we have visual confirmation of the identity of the speaker and we are certain that what is said is exactly what the speaker wanted to communicate (at least with respect to the words he/she said). When this form of communication is not available, we rely on other means trying to obtain the same guarantees: in a phone call, we associate a personal number to each person and we rely on the telephone network; in postal voting, citizens receive a ballot and then they return it relying on the mail service; in social media, we access the feed of an user and retrieve the information he/she shared, etc.

An example of unreliable communication can be found in social networks, such as Facebook or Twitter, because the enforcement of the authenticity and integrity of the information exchanged is partially (and implicitly) demanded to the users: every participant may easily and directly access to the exact information spread by every user navigating its profile (containing the set of all the information he/she shared), but in the meanwhile, other users may spread rumors, namely information associated to a specific peer but never diffused by it; every participant may rely only on the information directly retrievable from the profile of an user or it may accept the rumors coming from some thirds as valid.

One of the most famous computer science problems, the *Byzantine Generals* [52], provides an example of the challenges and possible solutions providing reliable communication between parties, depending on the set of considered assumptions. The problem is about a set of generals surrounding a city that have to agree on whether or not to attack it, and to make a decision, they need to communicate. As previously motivated, one of the most reliable communication mean between people is the face-to-face speaking, but in this scenario generals may be unable to move from their position and therefore they may need to rely on messengers (intermediates) to communicate with each other. Unfortunately, a messenger can be a traitor and report orders different from the one given by the generals. A possible countermeasure could be to write the orders on letters and to seal them with a personal marker, owned by each general: every general will be able to verify the opinions of the others. The only potential obstacle left could be that messengers may simply destroy the letters, that could be addressed by making multiple letters to entrust to several messengers (and hopefully they will not all betray the generals!).

The described issues may occur in several communication networks: participants may need to rely on intermediates to communicate, but the integrity, authenticity, and delivery of the information exchanged could be lost when part of the peers are not correct.

This thesis investigates the reliable communication problem in a network of devices where some of its components could be compromised. More precisely, it aims to examine and propose solutions enabling processes (i.e. units, devices) of a

distributed system (a set of independent units performing computation and cooperating between each other to pursue a common goal) to exchange *messages* (units of information) guaranteeing their integrity, authenticity, and delivery despite the presence of a subset of processes having an arbitrary behavior (*Byzantine faults*).

The correct functioning of a distributed system depends, most of the time, on the correct message exchanges between processes. The difficulty of guaranteeing reliable communication in a distributed system mostly depends on its features and assumed hypotheses: it is relatively simple to provide in case all processes can directly exchange messages between them, whereas it is more challenging when some peers need to relay on intermediates to communicate, and way more challenging if they could be incorrect and misbehave maliciously. Unfortunately, this last scenario is the most common in networks of devices, in fact, messages exchanged inside a system may travel over several nodes between a sender and a receiver, and the greater the number of devices composing a system, the higher the probability that some of them may experience failures or be compromised.

The literature provides several solutions to the reliable communication problem in distributed systems, that can be mainly partitioned among the ones relying on digital signatures and others based on the information redundancy and how it is propagated in the system. Both the approaches have some weaknesses.
The former requires all processes to be able to digitally sign their messages (a digital signature guarantees authenticity and integrity of a message) and to verify the digital signatures of all other peers; this generally requires either a third party that generates and distributes the cryptographic keys, supporting a digital signature scheme, or that such keys are already known by all processes.
The latter is more general and based on the hypothesis of reliable and authenticated communication channels between processes, namely that all the peers capable to communicate without intermediates can exchange messages guaranteeing their integrity, authorship, and delivery. Unluckily, those solutions may mandate a huge amount of messages to be diffused and may require very high computational power to the nodes. Furthermore, they generally require "more densely connected" networks to be employed with respect to the firsts.

This thesis aims to design and examine protocols enabling all the participants of a communication network to exchange information guaranteeing their integrity, authorship, and delivery. Moreover, we target this problem in one of the most general and challenging scenarios in which it is solvable: *an unknown network with part of the participants arbitrary faulty*. The challenges in this setting are several. The lack of knowledge of the network, specifically of its structure (topology), commends the entities to act opportunistically, namely to blindly rely on information to whatever peer they are able to communicate with, resulting in a high amount of redundant information diffusing in the network. Arbitrarily faulty participants (commonly referred with *Byzantine*) introduce additional difficulties. In fact, they are free to spread whatever information inside the network or block its propagation.

Solutions enabling reliable communication between all processes of a distributed system allow to transparently employ most solutions defined for fully connected systems, such as register abstraction and agreement protocol, on general communication networks.

The work presented in this thesis starts from reviewing the non-cryptographic (i.e. not based on digitally signed messages) solutions available in the literature. All the solutions currently available have some drawbacks: some do not scale, others require complete knowledge on the network topology, and the correctness of several ones cannot be easily verified. Subsequently, the thesis develops in three parts: the first considers a static distributed system, whose communication network does not vary over time, where a subset of processes could be arbitrarily faulty; the second still consider a fixed subset of compromised processes while assuming a communication network evolving over time (therefore processes may be able to communicate with different nodes over time); the third supposes a static communication network where failures are dynamic and all processes can potentially fail over time. In detail:

- in Chapter 2 we provide a survey on the Byzantine tolerant reliable communication problem, and related other problems;
- in Chapter 3 we define all the system model assumptions we consider in this thesis;
- in Chapter 4, we state the specification of the problem variants under investigation in this thesis;
- in Chapter 5, we recall some definitions and results in static and dynamic graph theory;
- in Chapter 6, a more detailed analysis of the available reliable communication solutions for the system models we consider is provided;
- in Chapter 7, we highlight the open problems and challenges available addressing the reliable communication problem in static distributed systems;
- in Chapter 8, we define and analyze a solution to the reliable communication problem for static distributed systems that improves the state-of-art with respect to the number of messages exchanged;
- in Chapter 9, we show that considering some additional assumptions with respect to the minimum required, it is possible to design an optimal reliable communication protocol in static distributed systems, with respect both to the total number of messages exchanged, to the computational complexity in verifying authenticity and integrity of messages, and to the time required by two processes to communicate;
- in Chapter 10, we define and analyze the prototype of a cryptographic reliable communication protocol, that is suitable for a computationally bounded adversary;
- in Chapter 11, we detail the open problems and challenges available addressing the reliable communication problem in dynamic distributed systems;

- in Chapters 12 and 13, we identify additional network conditions enabling reliable communication in dynamic distributed systems;
- in Chapter 14, we examine the reliable communication problem in static distributed systems affected by dynamic failures;
- in Chapter 15, we draw conclusions and perspectives.

The work presented in this thesis is based on the following publications:

- Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast made practical. In *8th Latin-American Symposium on Dependable Computing, LADC 2018*, pages 155–160. IEEE.
- Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Reliable broadcast in dynamic networks with locally bounded byzantine failures. In *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018*, volume 11201 of Lecture Notes in Computer Science, pages 170–185. Springer.
- Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast with honest dealer made practical. *J. Braz. Comput. Soc.*, 25(1):9:1–9:23, 2019.
- Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Une méthode efficace pour éviter la propagation des fake news. In *ALGOTEL 2020–22èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*.
- Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Broadcasting information in multi-hop networks prone to mobile byzantine faults. In *Networked Systems - 8th International Conference, NETYS 2020*, Lecture Notes in Computer Science.
- Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Boosting the efficiency of byzantine-tolerant reliable communication. In *Stabilization, Safety, and Security of Distributed Systems - 22nd International Symposium, SSS 2020*, Lecture Notes in Computer Science.

Chapter 2

State-of-the-art

Byzantine fault tolerance was considered since the early studies in distributed systems, handling part of processes with arbitrary behavior. The Byzantine/arbitrary fault is the hardest kind of failure to address but also the most general one, able to capture most of the misbehaviors happening in a system. Several distributed system problems were defined and investigated considering Byzantine participants, such us *Byzantine Reliable Broadcast* [22], *Byzantine Agreement*, *Register Abstractions*, etc.. Nonetheless, *most of the Byzantine faults tolerant solutions proposed so far in the literature assumed a reliable communication primitive*, namely that all processes in the system are able to exchange messages between them guaranteeing their integrity, authorship, and delivery.

The reliable communication problem investigation was implicitly initiated in complete networks, thus in distributed systems where all processes can directly exchange messages between each other, through the definition of *links*, i.e. communication primitive enabling message exchanges between pairs of processes satisfying certain properties [24].

The reliable communication problem was then studied assuming a general communication network (i.e., not fully connected). Several sets of assumptions were considered: about the kind of failures (omissions, crash, arbitrary, etc.), about their distribution (deterministic or probabilistic), whether they concern links [66], processes, or both. Considering only arbitrary process failures (i.e. *Byzantine faulty* processes), the seminal contribution in general networks was provided by Dolev [30, 31], who identified the necessary and sufficient conditions to solve the reliable communication problem in a general communication network consisting of reliable and authenticated links, assuming an upper bound on the number of faulty processes present in the system. Dolev proposed two protocols working with different assumptions on the knowledge of the network topology by participating processes. More precisely, the lack of topology knowledge impacts both the message complexity (i.e., the number of messages exchanged in the system during a reliable communication instance) and the delivery complexity (i.e., the computational complexity of the procedure used to validate a message) of the protocol. Subsequently, more constrained process failure distributions were studied. Koo [50] analyzed the reliable communication problem assuming that only a frac-

tion of nodes can be compromised in the neighborhood of every process in a radio network. Pelc and Peleg [67] later generalized his results, characterizing a failure model where an upper bound on the number of faulty processes in the neighborhood over every node is assumed, and defining a simple and efficient solution, CPA (Certified Propagation Algorithm), solving the reliable communication problem in the defined failure model. Despite the proposed model and solution were relatively simple, the correctness conditions enabling reliable communication were not immediately identified. Pelc and Peleg initially identified the necessary and a sufficient condition characterizing networks where the reliable communication problem is solvable. Several works followed [43], providing alternative sufficient network conditions till the identification of the necessary and sufficient ones in directed [75] and undirected [54] networks. After that, Pagourtzis et al. [65] further generalized the model of Pelc and Peleg, assuming non-homogeneous local bounds on the number of faulty processes in the neighborhood of every node (namely supposing every process able to estimate an upper bound on the number of potentially faulty peers it is connected to), they showed how the knowledge on the network topology increases, in that specific failure model, the number of tolerable faulty processes, and how such a failure model can be specialized assuming specific sets of processes potentially faulty (*general adversary model*). A recent CPA related contribution [74] analyzes its employment considering a hybrid failure model (where both processes and links could be faulty), random geometric networks, and defining a weaker reliable communication primitive with lower latency. Finally, the possibility of differentiating the processes of the system in compromisable and uncompromisable sets was investigated [76]. This distinction provides several additional benefits: a higher number of tolerable faults and lower latency.
Most of the solutions to the reliable communication problem rely on redundancy mechanisms and thus require highly connected networks to be employed. For this reason, several works investigated weaker Byzantine tolerant reliable communication primitives in loosely connected networks, either accepting that a small minority of correct nodes may deliver invalid messages or allowing that a small minority of correct peers may not deliver genuine messages [56, 58, 59].

Dynamic distributed systems models and analysis were raised in the last two decades thanks to the introduction of computational and communication abilities to more and more devices (e.g. IoT networks). The *static distributed system* models mainly consider *changes as failures*, namely processes not participating in the computation, or link unavailability have been assumed to be unexpected events one must tolerate. In dynamic distributed systems the evolution of the communication network is the main focus. For these reasons, there has been a great effort in defining new protocols solving many fundamental problems in dynamic distributed systems.
Most of the dynamic distributed system models proposed so far can be categorized either as *open* or *closed*. The former considers new processes continuously entering and leaving the system (i.e. *churn*), whereas the latter assumes a fixed set of processes whose links may change over time. The research in closed dynamic distributed systems mainly focused on solutions to fundamental problems in completely correct systems, such as simple broadcast, leader election, token

dissemination, spanning trees, etc. [25]. Open dynamic distributed system contributions gave higher attention to fault-tolerant solutions, targeting several agreement problems. Related to the Byzantine tolerant reliable communication problem in dynamic networks, Maurer et al. [60] identified the necessary and sufficient conditions to solve the problem considering a subset of arbitrary faulty processes.

Another change that occurred in the last decade was the diversification of the kind of failures occurring in distributed systems. The assumptions of faulty participants, unreliable links, etc. properly model *internal malfunctions* indeed. Nevertheless, more and more frequently distributed systems are subject also to *external attackers*, whose aim is to penetrate and compromise them, starting from some weaker machines and then moving in the system till reaching their targets. In complete communication networks, non-static Byzantine faulty processes were considered by Reischuk [68] who proposed an algorithm solving the Byzantine agreement in the case of f malicious agents that remain stationary on f processes only for a given period of time. Later, Ostrovsky and Yung [64] introduced the notion of an adversary that can inject and distribute faults in the system at a constant rate, and they proposed solutions (mixing randomization and self-stabilization) for tolerating the attacks of mobile viruses. Then, Garay [39] considered processes proceeding in synchronous rounds composed of three phases (send, receive, and computation), and Byzantine mobile agents able to move between one process to another during the lifetime of the system. Several subsequent works later specialized his model, making additional hypotheses about the unawareness of processes of being faulty [71], assuming correct processes sending non-equivocal messages [10], channels delays [70], decoupling the system evolution from the agents movements [19], etc.. All aforementioned works for the mobile attacker model addressed either the Byzantine agreement, the approximate Byzantine agreement [21], or the register abstraction [18] problems in complete networks. The only work that considered the reliable communication problem (while solving the mobile Byzantine agreement) in the mobile Byzantine failure model was given by Sasaki et al. [71].

The Byzantine tolerant reliable communication can also be achieved employing cryptography (e.g., digital signatures) [27, 35] that enable all nodes to exchange messages guaranteeing authenticity and integrity. The main advantage of cryptographic protocols is that they allow solving the problem with simpler solutions and weaker conditions (in terms of connectivity requirements). However, on the negative side, the correctness of the protocols is bounded to the crypto-system. Let us note that a common assumption considered by Byzantine tolerant reliable communication protocols is to use authenticated point-to-point channels, which prevents a process from impersonating several others (Sybil attack) [34]. The real difference between cryptographic (authenticated) and non-cryptographic (unauthenticated) protocols for reliable communication is how cryptographic primitives are employed: non-cryptographic protocols may use digital signatures just within neighbors for authentication purposes, whereas cryptographic protocols make use of cryptographic primitives to enable the message verification even between non-directly connected nodes. Let us finally remark that implementing an authenticated channel not necessarily require the use of cryptography [78].

The Byzantine reliable broadcast [22] and Byzantine consensus [52] problems are related to the reliable communication problem with increasing complexity. In the reliable communication problem, if a process delivers a message m, then m was sent by its sender process, and if m was sent by a correct process, then it can be delivered by any correct process. Nonetheless, if a message m is sent by a faulty process, then its delivery is not guaranteed by any correct process. Precisely, a reliable communication primitive does not prevent faulty processes from *equivocate*, namely from sending conflicting messages to different neighbors. A solution to the *Byzantine reliable broadcast* problem [22] addresses this issue, guaranteeing that all correct processes eventually deliver the same set of messages. The Byzantine reliable broadcast problem can be solved in a distributed system if both a reliable communication primitive is available and at most one-third of the processes is Byzantine faulty. Finally, in *Byzantine consensus* problem specifications, all correct processes initially send a message m_i, and they all output the same message m among the various m_i. To solve this problem, a reliable communication primitive, at most one-third of Byzantine faulty processes, and a synchronous systems are required.

Chapter 3

System Model

We provide in this chapter details on all the system model assumptions that are considered in this thesis. Since slightly different hypotheses are made among the contributions, this chapter aims to give a complete overview of all the alternative assumptions that we consider. Every contribution specific system model will be characterized inside the related chapters.

Time

We measure the passage of time by a fictional global clock that spans the set of positive natural numbers $\mathbb{N}^0$. We refer with t to a specific instant, namely $t \in \mathbb{N}^0$.

Processes

A *process* is an independent unit performing computation in a distributed system [24]. We consider a system composed of a fixed set of n processes. Each process is associated with a unique integer identifier i distinguishing it from the others. We use p_i to refer the process whose identifier is i and $P := \{p_1, p_2, \dots p_n\}$ denotes the set of all processes. Each correct process executes a distributed protocol $\mathcal{P}$, stored in a read-only tamper-proof memory.

Links

Processes can communicate by exchanging messages with a subset of their peers through *links*, representing the network components of a distributed system [24]. We either consider *Unicast Links* (*UL*), interconnecting pairs of processes p_i, p_j, or *Broadcast Links* (*BL*) [7, 8, 48], attaching single processes p_i to several others $\{p_a, p_b, p_c \dots\}$. Notice that local broadcast links prevent processes from *equivocating*, namely transmitting conflicting messages over different links. All links are assumed *bidirectional*, meaning that given any two processes connected with a link, any of them can send messages to the other.

Processes and links actions

The are three main actions that can be triggered by processes, send(m, p_j), receive(m, p_i) and deliver(m), and one associated to links, deliver. The send action is triggered by a process p_i (referred with *sender*) to transmit a message m through a link toward process p_j. A link *delivers* a message m if it accomplishes its transmission from a process p_i to another p_j. The receive action is triggered by a process p_j (referred with *receiver*) when a message m is delivered by a link to p_j from process p_i. A processes that successfully validates a message m with respect to a certain specification *delivers* (or *accepts*) such a message.

Communication network

The set of processes P and all of their adjacent links constitute the *communication network*. A communication network can be either *static* or *dynamic*, namely the links in the system can either be always available (perpetually allowing their associated processes to exchange messages) or be accessible only at some instants $t_1, t_2 \ldots$.

We abstract a static communication network using an undirected graph $G := (V, E)$, where the set of nodes V corresponds to the set of processes P, and the set of edges E contains an element $\{p_i, p_j\}$ for each link connecting processes p_i and p_j.

We model a dynamic communication network through a **Time Varying Graph** (**TVG**) [25, 26], that is a tuple $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$ such that:

- V is the set of nodes, corresponding to the of processes of the system ($V = P$);
- $E \subseteq V \times V$ is the set of edges, coinciding with the links of the system;
- $\mathcal{T}$ is the *lifetime* of the network, i.e. a time domain $\mathbb{T}$;
- $\rho : E \times \mathcal{T} \to \{0, 1\}$, called *presence* function, indicates if a given edge is available at a given time;
- $\zeta : E \times \mathcal{T} \to \mathbb{T}$, called *latency* function, indicates the time it takes to cross a given edge at a given start time (in our case, the time a message takes to cross the link).

For simplicity and consistency, we consider dynamic communication networks evolving in discrete time, identifiable with natural integer numbers, namely $\mathcal{T} \subseteq \mathbb{N}^0$. We assume processes instantaneously detecting the presence of a link in case of a dynamic communication network, meaning that if the edge between two nodes is present at t, then any of the related processes can start the transmission of a message (i.e. each such processes can execute the send action at t).

Given the bond we set between the communication network and its models G or $\mathcal{G}$, between processes and nodes of a graph, and between links and edges of a graph, we interchangeably refer respectively with graph and communication network, with processes and nodes, and with links and edges.

We refer with *static* or *dynamic distributed systems* based on the type of communication network assumed.

Messages

Every message exchanged in the system is associated with its *author*, using the identifier of the process that generated it. Precisely, all messages exchanged have format $m := \langle authorID, content \rangle$.

Timing assumptions

We either assume a *synchronous* or an *asynchronous* distributed system [24].
A synchronous distributed system guarantees:

- *synchronous computation*, namely that there is a known upper bound Δ on the *processing delay* (the time it takes a process to execute a computation step), known by all processes;
- *synchronous communication*, namely that there is a known upper bound δ on the message transmission delay (the period between the instant when a message is sent by a process and the moment when it is delivered by the employed link, also known as *link latency*), known by all processes.

An asynchronous distributed system provides none of the aforementioned properties, therefore both process local computations and message exchanges may take an unpredictable amount of time.

We consider several specialized models for synchronous and asynchronous systems:

- *Static_Async*: an asynchronous system with a static communication network.
- *Static_Sync*: a synchronous system with a static communication network evolving in *synchronous rounds*; every round is divided into three phases: *(i) send* where processes send messages through their links for the current round, *(ii) receive* where processes receive all messages sent at the beginning of the current round, and *(iii) computation* where processes execute local computation and generate messages to be sent during the subsequent round. Every round lasts exactly one time instant.
- *Dynamic_Async*: an asynchronous system with a dynamic communication network.
- *Dynamic_FullSync*: a synchronous system with a dynamic communication network. The link latency is always assumed equal to one time instant, namely $\forall e \in E, \forall t \in \mathcal{T}, \zeta(e,t) = 1$. The processing delay Δ is negligible and assumed equal to 0. The computation evolves in *synchronous rounds*; every round is divided into three phases: *(i) send* where processes send messages through their links for the current round, *(ii) receive* where processes receive all messages sent at the beginning of the current round, and *(iii) computation* where processes execute local computation and generate the messages to be sent during the subsequent round. Every round lasts exactly one time instant and the dynamic communication network evolves only during the computation phase, whereas it is stable during the send and receive phases.

- *Dynamic_CompSync*: a dynamic distributed system guaranteeing synchronous computation, specifically Δ is assumed negligible and equal to 0.

All links in the defined synchronous models are assumed having *unbounded capacity*, meaning that their transmission delay is supposed independent from the number of messages that are concurrently transmitted and it only depends on the time t their transmission began. In other words, if the transmission latency of a link is bounded by δ at t, then whatever the number of messages they are all delivered by the specific link in a period lasting at most δ.

Adversary and failure models

We assume an *omniscient adversary* able to control several processes in the system. Precisely, it can turn them into *Byzantine faulty*. A Byzantine faulty process may have a arbitrary behavior, ruled by the adversary. It means that a faulty process may not properly participate in the accomplishment of the goal of the system, not sending messages and/or diffusing invalid or conflicting ones. Notice that the Byzantine fault model is the most general process failure: it includes all others such as crashes, omissions, or eavesdropping [24].

We consider three alternative failures distributions:

- *Global* (globally bounded): up to f processes in the system may permanently (i.e. at every instant t) be Byzantine faulty;
- *Local* (locally bounded): every process in the system may have at most f links toward permanently (i.e. at every instant t) Byzantine faulty processes.
- *Mobile* (mobile globally bounded): at every instant t, there are up to f processes in the system that are Byzantine faulty, and the set of faulty processes may change over time.

The mobile globally bounded failure distribution results from *Mobile Byzantine Faults* (*MBF*) [10, 23, 39, 71]. Specifically, the *MBF* results from f *mobile agents* present in the system and controlled by the adversary. They are initially placed on f distinct processes and able to move on other peers during the lifetime of the system. Whenever an agent is on a process p_i, it forces p_i to behave as Byzantine faulty. An agent can move from a process p_i to a peer p_j only if there exists a link interconnecting them. The movement of an agent is characterized by the *roaming pace* $\hat{\jmath}$, which is the minimum period between two displacements of an agent. Every process reverts back to correct and executes the protocol $\mathcal{P}$ right after an agent moved away. We refer with *cured* process to a correct peer at time t that was Byzantine faulty at $t-1$. We consider cured processes either *aware* or *unaware* of having been faulty in the precedent instant.

A subset of processes $F \subset V$ satisfying a failure distribution is referred as a *corruption set*. Every subset $F \subset V$, $|F| \leq f$ is a valid corruption set for the globally bounded and mobile globally bounded failure model. Every subset $F \subset V$ satisfying the locally bounded constraint is referred as a *f-local set* [67].

Every process that is not (currently or permanently) faulty is *correct* and it executes the distributed protocol $\mathcal{P}$.

Link assumptions

We consider perfect *reliable* and *authenticated* links [24], providing the following features:

- *reliable delivery*: if a correct process p_i sends a message m to a correct process p_j, then p_j eventually receives m;
- *no duplication*: no message is received by a correct process more than once;
- *authenticity*: if some correct process p_j receives a message m from sender p_i, then m was previously sent by p_i to p_j.

Process knowledge

We assume processes know the value of the upper bound f for the global, local, or mobile failure model but they are unaware of the identity of the faulty processes.

We suppose processes do not know the topology of the communication network, namely they do not have knowledge of G or $\mathcal{G}$.

We consider the alternative scenarios where processes are aware (*Know Neighborhood* assumption, *KN*) or unaware (*Unknown Neighborhood* assumption, UN) about the set of peers they have a link with. In the latter case, the send link primitive still allows a process to send messages to all of its neighbors.

In case of a mobile failure distribution, we alternatively consider processes *aware* [71] or *unaware* [39] of a mobile agent having affected them. Specifically, in the former case, a process knows when a mobile agent is moving away from it, in the latter it cannot detect the previous presence of an agent.

System models nomenclature

For sake of conciseness, we employ and combine the labels we defined characterizing the timing and failure models assumptions to briefly declare the essential parts of the system model we consider. For example, the label *Static_Async_Global* refers to a static distributed system where the globally bounded failure model is assumed.

Chapter 4

The Reliable Communication Problem

The *reliable communication problem* is the topic under investigation in this thesis. We define its specification in this chapter. We additionally provide the definition of some of the metrics commonly employed in the performance evaluation of reliable communication protocols.

4.1 Reliable communication problem specification

Let us consider processes in a distributed system exchanging *contents*. Let us refer as *source* a process p_s that sends a content and as *target* p_t the peer such a content is addressed to. A solution to the reliable communication problem guarantees:

- ***safety***: if p_t is a *correct* process and it delivers a content c from p_s, then p_s previously sent c;
- ***liveness***: if p_s is a *correct* process and it sends a content c to a correct process p_t, then p_t eventually delivers c from p_s.

The reliable communication problem specification can be specialized through the following versions:

- *one-to-one*: a defined process p_s aims to reliably communicate with a specific target process p_t;
- *one-to-all*: a defined process p_s aims to reliably communicate with every other process;
- *any-to-any*: any process aims to reliably communicate with any other process.

A solution to the reliable communication problem enables content exchanges between correct processes that are not connected by a link, guaranteeing their authorship, integrity, and delivery. A content that has not been sent by its author is referred with *spurious content*, and any message that cannot be sent in an execution

involving only correct processes is indicated as *spurious message*. A solution to the reliable communication problem prevents every correct process from delivering any spurious content.

We say that a solution solves the reliable communication problem at time t if it succeeds assuming the source sending its content at time t.

4.2 Naming clarifications

The specifications of distributed system problems based on message exchanges commonly refer as "message" to the information exchanged between processes (i.e. the payload of their communication). Nevertheless, several distributed system protocols (including the ones presented in this thesis) rely on mechanisms that leverage redundancy, and specifically they diffuse many copies of the same message to pursue their goals.

For this reason, for ease of exposition and aiming to avoid potential ambiguity, we refer with *content* to the information exchanged between processes (i.e. the payload), whereas we refer with *message* to the union of the content and the protocol-specific overhead.

4.3 Performance evaluation metrics

We employ the following metrics to evaluate and compare the protocols presented in this thesis.

Definition 1 [**message complexity**]. The *message complexity* of of a reliable communication protocol is the number of messages that are exchanged in the system to accomplish the task.

Definition 2 [**delivery complexity**]. The *delivery complexity* of a reliable communication protocol is the computational complexity of the procedure locally executed by a process to validate a content.

Definition 3 [**communication latency**]. The *communication latency* is the amount of time required to accomplish reliable communication, namely the length of the period between the moment when the source sends a content and the instant when such a content is delivered by the target process.

Notice that, given the absence of timing guarantees in asynchronous systems, the communication latency can only be evaluated in synchronous ones.

Chapter 5

Definitions and Recalls on Graphs

We recall in this chapter some fundamental definitions and known results in graph theory and time varying graph theory that will be employed and leveraged in the subsequent chapters [1].

5.1 Static graph definitions and recalls

A *graph* is a pair $G := (V, E)$ of sets V, E such that $E \subseteq V \times V$; the elements of V are referred with *vertices* (or *nodes*) of the graph, whereas the elements of E are called *edges* [28]. The graph thus defined is said *undirected*. The value $n := |V|$ is known as *order* of the graph. A vertex v_i is *incident* with the edge $\{v_i, v_j\}$ if it is contained therein.

Definition 4 [**neighbors, neighborhood**]. Two nodes $v_i, v_j \in V$ are said *neighbors* (or *adjacent*) if the edge $\{v_i, v_j\}$ exists in E. Given a node $v_i \in V$, the set of nodes $\Gamma(v_i) =: \{v_j \in V \mid \{v_i, v_j\} \in E\}$ is referred as the *neighborhood* of v_i.

A graph G is *complete* if there exists an edge between any two vertices $v_i, v_j \in V$, namely $G = (V, V \times V)$.

Definition 5 [**path**]. A *path* π is a sequence of nodes with no repetition, $(v_1, v_2, \ldots, v_m), v_i \in V$, such that for each pair of consecutive elements v_i, v_{i+1}, the edge $\{v_i, v_{i+1}\}$ exists in E (a path can alternatively be defined as a sequence of edges joining a sequence of distinct vertices).

The first and last elements of a path are referred as *endpoints*, the others as *internal* nodes. The length of a path π is the value $|\pi| - 1$.
Notice that, in an undirected graph, if there exists a path π in G, then also its *reverse* $\overleftarrow{\pi}$, obtained ordering the sequence π from the last of its elements to the first, is a valid path in G.
The *distance* $d(G, v_i, v_j)$ between two vertices v_i and v_j is the length of a shortest path connecting them. The *diameter* $d(G)$ is the greatest distance between any pair of vertices of the graph.

[1] Basic set theory notations and concepts are assumed as known, the reader can refer to any set theory book for details [45]

Definition 6 [**connected nodes and graph**]. A pair of nodes $v_i, v_j \in V$ is *connected* if there exists a path in G with endpoints v_i, v_j , it is *disconnected* otherwise. A graph is *connected* if there exists a path between any pair of vertices, it is *disconnected* otherwise.

A subgraph G' of a graph $G = (V, E)$ is a graph whose vertex set and edge set are subsets of those of G, namely $G' = (S \subset V,\ \{\{v_i, v_j\} \in E \mid v_i, v_j \in S\})$. Given a subset of nodes $S \subset V$, let us define with $G_{\hat{S}}$ the subgraph of G resulting from the removal of all nodes $v_i \in S$ from G, namely $G_{\hat{S}} := (V \setminus S,\ \{\{v_i, v_j\} \in E \mid v_i, v_j \notin S\})$.

A *k-clique* is a complete subgraph of a graph G of k nodes. Two k-cliques are considered *adjacent* if they share $k - 1$ nodes.

Definition 7 [**vertex cut**]. A subset of nodes $S \subset V$ is a *vertex cut* for a pair of vertices $v_i, v_j \in V - S$ if the removal of S from G disconnects v_i from v_j in the resulting graph (namely v_i and v_j are disconnected in $G_{\hat{S}}$).

Definition 8 [**local node connectivity**]. The *local node connectivity* $\kappa_{i,j}$ between two nodes $v_i, v_j \in V$ is the minimum number of nodes that has to be removed from G to disconnect v_i from v_j.

Definition 9 [**node connectivity**]. The *node connectivity* of a graph is the minimum value k for the local node connectivity $\kappa_{i,j}$.

A graph having node connectivity greater than or equals to k is referred with *k-connected* graph.

Definition 10 [**node disjoint paths**]. Many paths between two nodes are *node disjoint* (or simply disjoint) if they share no vertex except for their endpoints.

Definition 11 [**disjoint path solution**]. A *disjoint paths solution* $\Pi_{i,j}$ between two nodes v_i, v_j is a collection of node disjoint paths between v_i and v_j.

Theorem 1 [**Menger's theorem [61]**]**.** The local node connectivity between two nodes is equal to the maximum number of node disjoint paths that exist between them.

Remark 1. It is possible to compute a disjoint paths solution $\Pi_{i,j}$ between two nodes $v_i, v_j \in V$ of maximum size (namely $\kappa_{i,j}$) with a deterministic algorithm with computational complexity polynomial in the size of the graph [29, 36].

Given two nodes $v_i, v_j \in V$, the metric $D_l(v_i, v_j)$ denotes the maximum number of disjoint paths joining v_i and v_j whose length does not exceed l. Analogously, the value of $C_l(v_i, v_j)$ indicates the minimum number of vertices in G (except v_i and v_j) whose removal destroys all paths joining v_i and v_j that do not exceed l in length.

Remark 2. The relation between the metrics $D_l(v_i, v_j)$ and $C_l(v_i, v_j)$ is the following: $C_l(v_i, v_j) \geq D_l(v_i, v_j)$ (see Figure 5.1). The equality between the two metrics occurs when no constrains are considered for the path length, namely $l = |V| - 1$ (Menger Theorem [61]) [2, 55]. Furthermore, it is a NP-Complete problem to compute both $D_l(v_i, v_j)$ and $C_l(v_i, v_j)$ with $l < |V| - 1$ [2] [3, 41, 44].

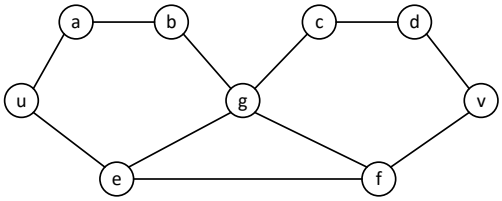


Figure 5.1: A graph where $D_l(v_i, v_j) \neq C_l(v_i, v_j)$: $D_5(u, v) = 1$ and $C_5(u, v) = 2$. In detail, at least two nodes must be removed from the graph to disconnect nodes u and v while considering only paths of length at most 5, whereas no two disjoint paths exist assuming the same constraint.

Definition 12 [**wide diameter [77]**]. Given a k-connected graph G, the wide-diameter $d_k(G)$ is the maximum number l for which there exist k disjoint paths in G of length at most l for any two distinct vertices v_i and v_j in G.

The computation of the wide diameter $d_k(G)$ of a graph in a NP-complete problem [42].

Remark 3. Let G be a k-connected graph with $k \geq 2$. Then $2 \leq d_k(G) \leq |V| - k$. Furthermore, both the upper and the lower bounds are best possible [77].

Definition 13 [**Minimum K Level Ordering (MKLO) [54]**]. The *minimum k-level ordering* $\mathcal{L}_k(G, v_s)$ of a graph $G = (V, E)$ from a given node $v_s \in V$ is the partitioning of V into subsets L_i, called *levels*, such that:

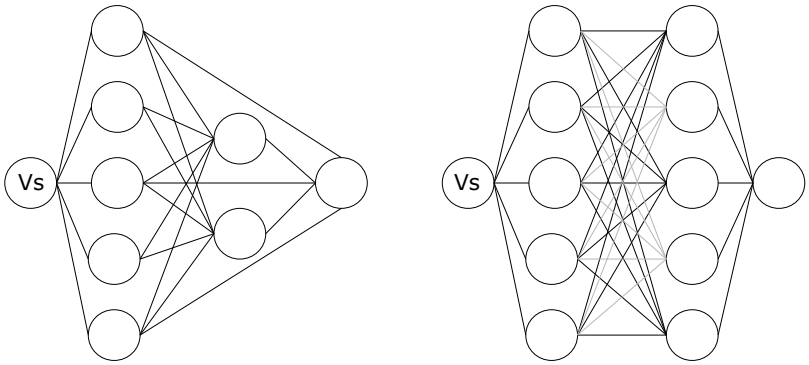
$$\begin{cases} v_i \in L_0 & \text{if } v_i = v_s \\ v_i \in L_1 & \text{if } v_i \in \Gamma(v_s) \\ v_i \in L_{z>1} & \text{if } v_i \in V \setminus (\bigcup_{j=0}^{z-1} L_j) \text{ and } |\Gamma(v_i) \cap (\bigcup_{j=0}^{z-1} L_j)| \geq k \end{cases}$$

Definition 14 [**Distance-based K Level Ordering (DKLO)**]. The *distance-bases k-level ordering* $\mathcal{D}_k(G, v_s)$ of a graph $G = (V, E)$ from a given node $v_s \in V$ is the partitioning of V into subsets L_i, called *levels*, such that:

$$\begin{cases} v_i \in L_0 & \text{if } v_i = v_s \\ v_i \in L_j & \text{if } d(G, v_s, v_i) = j \text{ and } |\Gamma(v_i) \cap L_{j-i}| \geq k \end{cases}$$

Intuitively, a *MKLO* is the arrangement of the nodes of a graph into disjoint levels, with respect to a selected vertex, such that every node has at least k neighbors in the previous levels and belongs to the minimum level for which this property is satisfied for this node. A *DKLO* is a *MKLO* with the additional property that every node (except the one adjacent to the selected node) has at least k neighbors in the level that precedes it. A graphical example is provided in Figure 5.2a.

[2] $D_l(v_i, v_j)$ is polynomially computable, with respect to the size of the graph, for $l \leq 3$ [44].



(a) $\mathcal{L}_5(G, v_s)$ (b) $\mathcal{D}_5(G, v_s)$

Figure 5.2: Example of a *MKLO* and *DKLO* with respect to a node v_s with $k = 5$.

Definition 15 [**parameter** $j_i(G)$ **[54]**]. For a given graph $G = (V, E)$ and node $v_i \in V$

$$j_i(G) := \max \{k \in \mathbb{N} \mid \exists\ \mathcal{L}_k(G, v_i)\}$$

Definition 16 [**parameter** $j(G)$]. Given a graph $G = (V, E)$

$$j(G) := \min_{v_i \in V} j_i(G)$$

Definition 17 [f**-local set [67]**]. A subset of nodes $F \subset V$ is a f-local set if $\forall v_i \in V,\ |\Gamma(v_i) \cap F| \leq f$.

Definition 18 [**parameter** $h_i(G)$ **[54]**]. Given a graph $G = (V, E)$, node $v_i \in V$

$$h_i(G) := \max \{f \in \mathbb{N} \mid \forall \text{ f-local set } F,\ h_i(G_{\hat{F}}) > f + 1\}$$

Definition 19 [**parameter** $h(G)$]. Given a graph $G = (V, E)$

$$h(G) := \min_{v_i \in V} h_i(G)$$

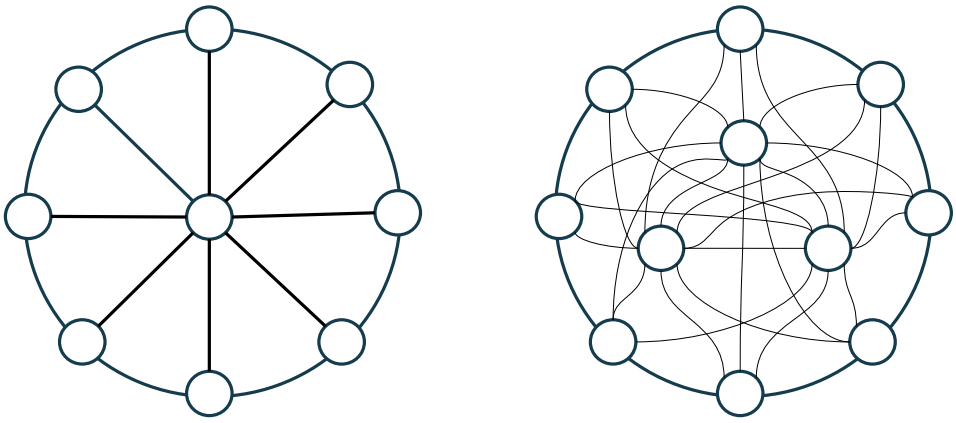
Definition 20 [**parameter** $x(G)$ **[67]**]. For every pair of nodes $v_i, v_j \in V$, the parameter $x(i, j)$ denotes the number of nodes $v_y \in \Gamma(i)$ that are closer to j than i; the parameter $x(G)$ is defined as the minimum $x(i, j)$ between any pair of not incident nodes, namely

$$x(G) := \min\{x(i, j) |\ v_i, v_j \in V, \{v_i, v_j\} \notin E\}$$

5.1.1 Graph topologies

A cycle graph C_l is a graph of l nodes connected in a closed chain.
A complete graph of m nodes is refereed with K_m.

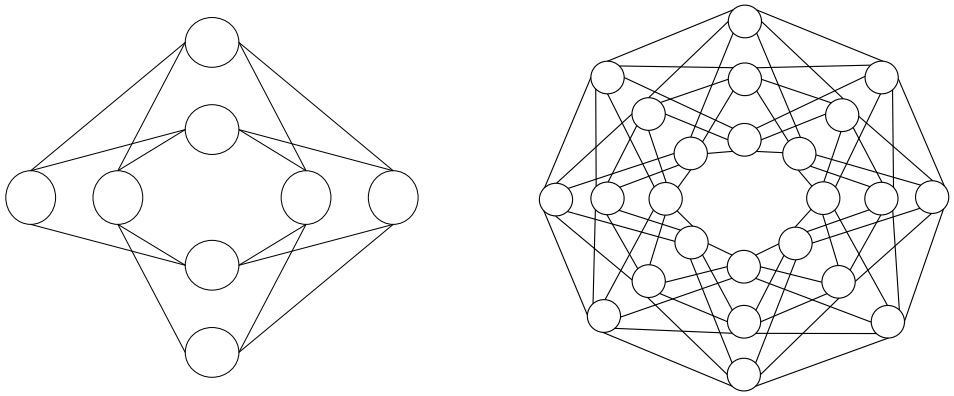
Definition 21 [**generalized wheel [77]**]. A *generalized wheel* , denoted by $W_{m,l}$, is a graph of order $m + l$ obtained from the disjoint union of a complete graph K_m and a cycle graph C_l ($l \geq 3$) by adding edges joining each of vertices of K_m to all nodes of C_l (See Figure 5.3).



(a) $W_{1,8}$ (b) $W_{3,8}$

Figure 5.3: Generalized wheel topologies.

Definition 22 [**multipartite cycle**]. A *multipartite cycle*, denoted by $MC_{k,l}$ is a connected graph composed by l sets of k non adjacent nodes, such that each set is part of exactly two complete bypartite subgraphs of $2k$ nodes (See Figure 5.4).

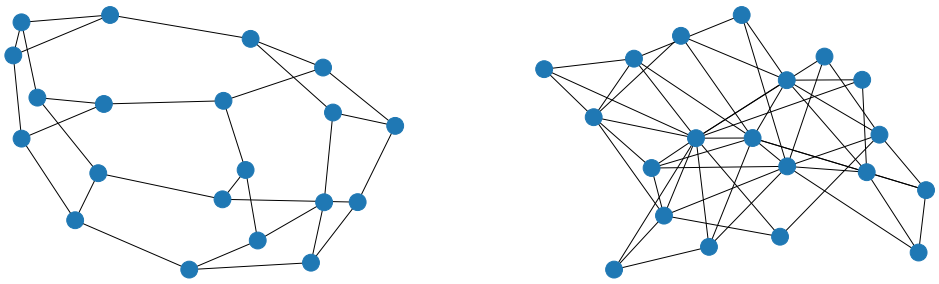


(a) $MC_{2,4}$ (b) $MC_{3,8}$

Figure 5.4: Multipartite cycle topologies.

Definition 23 [**random regular graph [9, 49, 73]**]. A graph G is d-regular if the size of the neighborhood of every node is exactly d, namely $\forall v_i \in V$, $|\Gamma(v_i)| = d$. Let $R(n, d)$ be the set of all d-regular graphs on a set V of n vertices, a random regular graph is obtained by sampling with respect to the uniform distribution from $R(n, d)$ (see Figure 5.5a).

Definition 24 [**Barabási-Albert graph [5, 6]**]. A Barabási-Albert graph is a random one generated using the Barabási–Albert preferential attachment model. Its generation is ruled by two parameters n and m: the former is the order of the graph, the latter corresponds to the initial number of incident edges a node has when connected to the rest of the graph (see Figure 5.5b).



(a) 3-regular random graph of 20 nodes. (b) Barabási-Albert graph with $n = 20, m = 3$.

Figure 5.5: Random regular and Barabási-Albert topologies.

Definition 25 [***k*-clique community graph**]. A k-clique community is a graph composed by the union of k-cliques that can be reached from each other through a series of adjacent k-cliques (see Figure 5.6).

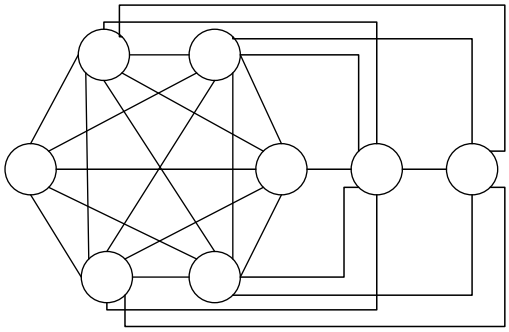


Figure 5.6: 5-clique community graph.

K-pasted-tree and **k-diamond** [4] are logarithmic harary graphs [46], i.e. k-connected topologies, having logarithmic diameter and ensuring link minimality [3]. A graphical representation of their structure is provided in Figure 5.7, whereas their definition is given in [4].

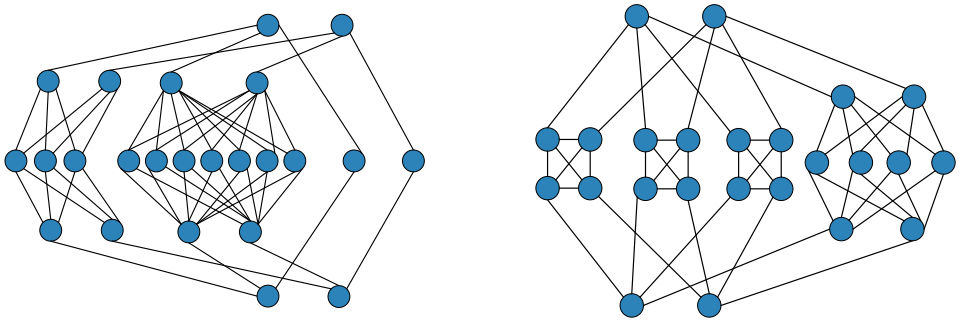
Definition 26 [$G(a,\beta)$ **graphs [71]**]. $G(a,\beta)$ is the set of graphs where all pairs of nodes are connected by a disjoint paths of length at most β.

5.2 Time Varying Graph definitions and recalls

The *Time Varying Graph* (*TVG*) [25, 26] is a formalism to model dynamic networks. It is defined by the tuple $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$ such that:

- V is the set of entities (nodes);
- $E \subseteq V \times V$ is the set of relations (edges);

[3] the removal of any link will reduce the node connectivity of the graph.



(a) 3-pasted tree graph example.

(b) 3-diamond graph example.

Figure 5.7: Examples of k-pasted three and k-diamond graphs.

- $\mathcal{T}$ is the *lifetime* of the network, being a subset of $\mathbb{N}$ (discrete) or $\mathbb{R}^+$ (continuous), and more generally some time domain $\mathbb{T}$;
- $\rho : E \times \mathcal{T} \to \{0, 1\}$, called *presence* function, indicates if a given edge is available at a given time;
- $\zeta : E \times \mathcal{T} \to \mathbb{T}$, called *latency* function, indicates the time it takes to cross a given edge at a given start time (the latency of an edge could itself vary in time).

A subgraph of a time varying graph can be defined in several ways. One is obtained restricting the lifetime of the TVG to a given subinterval $[t_a, t_b] \subseteq \mathcal{T}$, specializing the TVG functions to this new domain without otherwise changing their behavior. Thus, $\mathcal{G}_{[t_a,t_b]}$ is the *temporal subgraph* of $\mathcal{G}$ that restricts the lifetime of the graph to the period $[t_a, t_b]$.
Another subgraph definition extends the one for a static graph to TVG, considering a subset $S \subset V$ of the nodes. Precisely, *subgraph* $\mathcal{G}'$ of a TVG $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ is a TVG whose vertex set and edge set are subsets of those of $\mathcal{G}$, namely $\mathcal{G}' = (S \subset V, \{\{v_i, v_j\} \in E \mid v_i, v_j \in S\}, \mathcal{T}, \rho, \zeta)$ where the domains of the ρ and ζ functions are specialized considering the new vertex and edge set. Accordingly, given a subset of nodes $S \subset V$, let us define with $\mathcal{G}_{\hat{S}}$ the TVG subgraph of $\mathcal{G}$ resulting from the removal of all nodes $v_i \in S$ from $\mathcal{G}$, namely $\mathcal{G}_{\hat{S}} := (V - S, \{\{v_i, v_j\} \in E \mid v_i, v_j \notin S\}, \mathcal{T}, \rho, \zeta)$ where the domains of the ρ and ζ functions are specialized considering the new vertex and edge set.

Given a time varying graph $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, the graph $G := (V, E)$ is identified with *underline graph*, whereas the graph $G_t := (V, \{e \in E \mid \rho(e, t) = 1\})$ is referred with the snapshot of $\mathcal{G}$ at time t.

The *journeys* are the analogue of paths in TVG.

Definition 27 **[journey [25, 26]]**. Given a TVG $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$, a *journey* is a sequence of ordered pairs $\mathcal{J} := ((e_1, t_1), (e_2, t_2), \ldots, (e_m, t_m))$, such that

$(e1, e2, \ldots, e_m)$ is a path [4], $\rho(e_i, t_i) = 1$, and $t_i + 1 > t_i$ [5]. A journey from a node v_i to another v_j is commonly referred with $v_i \rightsquigarrow v_j$.

Notice that, differently from a path, a journey is *not reversible*, namely arranging it from the last of its elements to the first the resulting sequence is not a journey, due to the requirement of increasing times.

The *disjoint journeys* and the *dynamic cut* are respectively the TVG extensions of the disjoint paths and vertex cut notions in graphs.

Definition 28 [**disjoint journeys**]. Given a TVG $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$ and two nodes $v_i, v_j \in V$, a collection of journeys from v_i to v_j are *disjoint* if they share no node except for their first and last elements.

Definition 29 [**dynamic cut**]. Given a TVG $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$ and two nodes $v_i, v_j \in V$, a set of nodes $S \subset V - \{v_i, v_j\}$ is a *dynamic cut* from v_i to v_j if after their removal from $\mathcal{G}$ no journey exists from v_i to v_j in the resulting TVG subgraph $\mathcal{G}_{\hat{S}}$.

Accordingly, the parameters maximum disjoint paths and minimum vertex cut can also be extended to TVG.

Definition 30 [**maximum disjoint journeys**]. Given a TVG $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$ and two nodes $v_i, v_j \in V$, the *maximum disjoint journeys* from v_i to v_j is the maximum number of journeys that exist from v_i to v_j sharing no node except for their first and last elements.

Definition 31 [**dynamic minimum cut**]. Given a TVG $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$ and two nodes $v_i, v_j \in V$, the *dynamic minimum cut* from v_i to v_j is the minimum number of nodes that must be removed from $\mathcal{G}$ so that no journey exists from v_i to v_j in the resulting TVG subgraph.

Notice that, since the journeys are not reversible, the dynamic minimum cut and the maximum disjoint journeys from a node v_i to another v_j are not necessarily equivalent to their respective values from v_j to v_i. Furthermore, the equality between maximum disjoint paths and minimum vertex cut proved in the Menger theorem does not extend to dynamic networks.

Remark 4. Given a TVG $\mathcal{G} := (V, E, \mathcal{T}, \rho, \zeta)$ and two nodes $v_i, v_j \in V$, the dynamic minimum cut from v_i to v_j is greater then or equal to dynamic maximum disjoint journeys between the same vertices (see Figure 5.8).

Remark 5. Given a TVG and two of is nodes $v_i, v_j \in V$, it is NP-Hard to decide whether the dynamic minimum cut from v_i to v_j is at most equal to a certain value x [47] [6].

[4] The definition in [25, 26] assumes a *walk* as base for a journey instead of a path (i.e a sequence of vertices with repetitions). Walks are not useful to our purposes, so we specialize journey definition to extend a path in a graph.

[5] Specifically, this is the definition of *strict* journey that implicitly assumes non-zero edge latency.

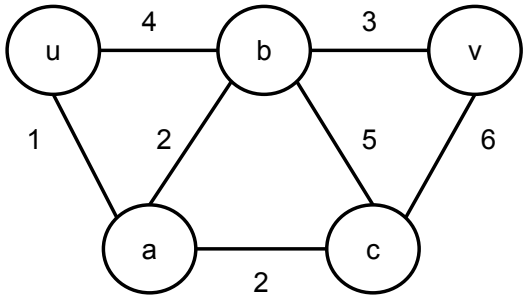


Figure 5.8: An example of a dynamic network where the dynamic minimum cut is higher than the maximum disjoint journeys between two nodes. Let us consider the TVG here depicted, where the edges are present only at the times written above them and that the latency function is always equal to 1 for every edge at every time. It can be noticed that, considering all possible journeys from node u to v, at least 2 nodes must be removed from the topology to guarantee that no journey exists from u to v. On the other hand, there do not exist two journeys from u to v sharing no node except for their endpoints.

Remark 6. Given a TVG and two of is nodes $v_i, v_j \in V$, it is NP-complete to decide whether there exist two disjoint journey from v_i to v_j [47] [6].

The *evolving graph* [37] is an alternative model for dynamic networks, characterized by a sequence of graphs. Precisely, in discrete time domains an evolving graph is a sequence $\mathcal{G} := (G_1, G_2, \dots G_m)$ where each graph G_t (snapshot) is composed by all the edges that are present at time t, whereas the graph G is obtained merging all the snapshots into a single graph called underline graph. It follow that every TVG defined in a discrete time domain $\mathbb{T}$ can be expressed as an evolving graph where $\forall t \in \mathbb{T}, G_t = (V, \{e \in E \mid \rho(e,t) = 1\})$ (additionally keeping the latency function).

In the same way types of graph have been defined in graph theory (trees, planar graphs, grids, complete graphs, etc.), several classes of dynamic graphs have been characterized in the literature. Specifically, a class of dynamic graphs groups all the ones that satisfy a specific set of properties.

Definition 32 [**Class $\mathcal{TC}$ (Temporal Connectivity) [26]**]. $\forall v_i, v_j \in V;\ v_i \rightsquigarrow v_j$ (the class of TVG where every node can reach all the others through a journey).

Definition 33 [**Class $\mathcal{TC}^{\mathcal{R}}$ (Recurrent temporal connectivity) [26]**]. $\forall t \in \mathcal{T}, \mathcal{G}_{[t;*)} \in \mathcal{TC}$ (the class of TVG where, for every time $t \in \mathcal{T}$, the temporal subgraph $\mathcal{G}_{[t;*)}$ satisfies temporal connectivity).

Definition 34 [**Class C^* (always-connected snapshots or 1-interval connectivity [26])**]. $\forall G_t \in \mathcal{G}$, G_t is a connected graph (the class of TVG where every snapshot is a connected graph).

[6] The result of the remark consider a TVG where every edge appears only once. It can be extended to our setting.

Definition 35 [**Class $\mathcal{E}^{\mathcal{R}}$ (Recurrent Edges) [26]**]. $\forall e \in E, \forall t \in \mathcal{T}, \exists t' > t, \rho(e, t') = 1$ (the class of TVG where every edge is present infinitely often).

Definition 36 [**Class $\mathcal{E}^{\mathcal{R}}$ (Bounded edge recurrence) [26]**]. $\exists \delta_{max} \in \mathbb{N}, \forall e \in E, \forall t \in \mathcal{T}, \exists t' \in [t, t+\delta_{max}], \rho(e, t') = 1$ (the class of TVG where every edge re-appears within bounded time).

Chapter 6

Related Works

We present and detail in this section the main solutions available in the literature solving the reliable communication problem in the system models we consider in this thesis.

6.1 Dolev protocol for unknown network, `DolevU`

Dolev [30] provided the seminal contribution to the Byzantine tolerant reliable communication problem, identifying the necessary and sufficient condition to solve the problem in a static distributed system assuming the globally bounded Byzantine failure model, and providing two solutions.

Dolev defined a flooding algorithm whose aim is to spread the content of a source while collecting the identifier of traversed processes. More in detail, the protocol diffuses every content inside messages with a list data structure gathering the identifier of the traversed peers, and such messages are relayed to any node not yet included in the list. A process delivers every content it succeeds in identifying $f + 1$ node disjoint paths among the ones it traversed. We report the complete pseudo-code in Algorithm 1, and we refer to this solution with `DolevU` in the sequel.

For ease of explanation, a graphical execution of the `DolevU` protocol is provided in Figures 6.1 and 6.2.

6.1.1 `DolevU` correctness analysis

Dolev identified the necessary and sufficient conditions to solve the any-to-any reliable communication problem in an asynchronous static distributed system assuming a globally bounded Byzantine failure model.

Remark 7. The any-to-any reliable communication problem can be solved in *Static_Async_Global* if and only if the node connectivity of the graph G modeling the communication network is (strictly) greater than twice the maximum number of assumed faults, namely $k > 2f$ [30, 31].

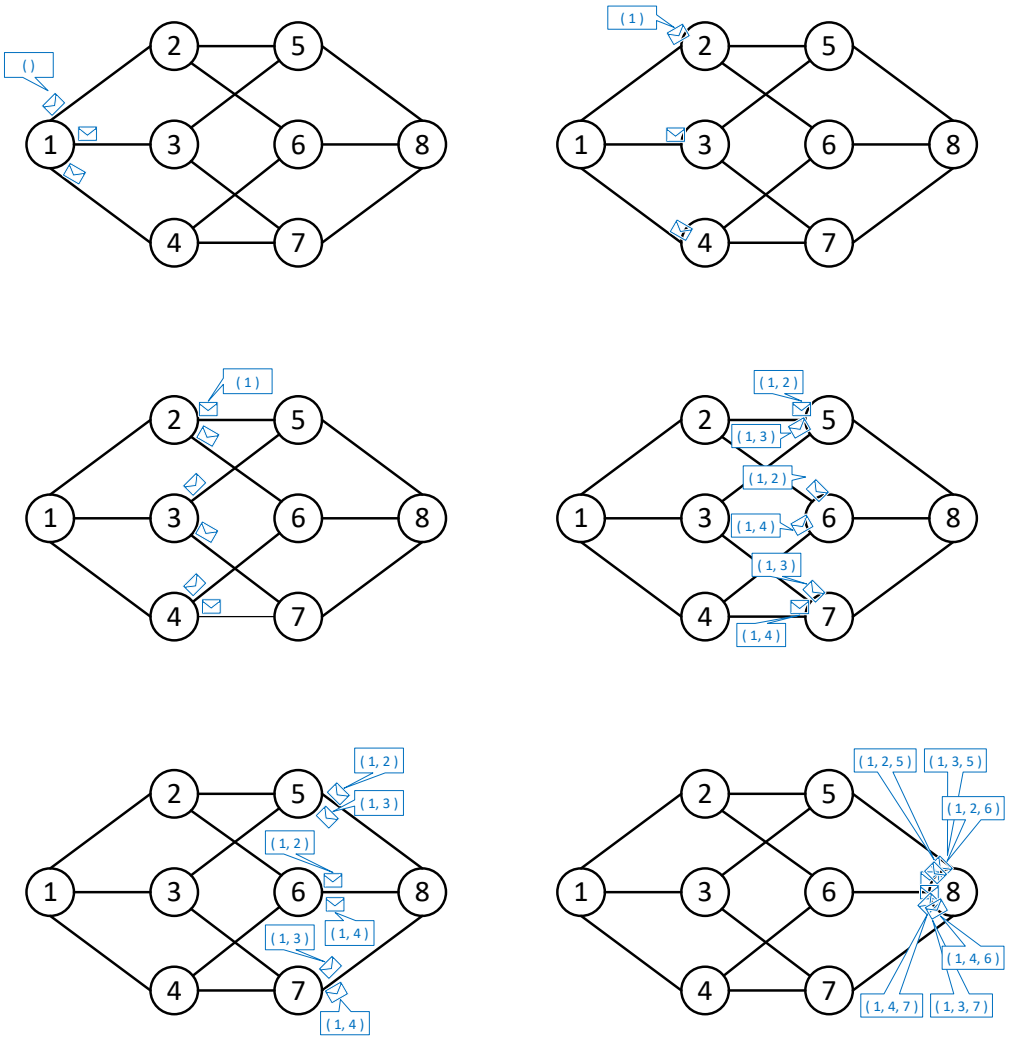


Figure 6.1: Example of content propagation with `DolevU` (only part of the generated messages are shown).

Algorithm 1 DolevU

1: **upon** DolevU_send(c) **do**
2: | **for** $j \in \Gamma(i)$ **do**
3: | | send($\langle i, *, c, \emptyset \rangle, j$)

4: **upon** receive($\langle s, *, c, path \rangle, j$) **do**
5: | $path \leftarrow path \cup \{j\}$
6: | $Paths_{\langle s,c \rangle} \leftarrow Paths_{\langle s,c \rangle} \cup \{path\}$
7: | **for** $j \in \Gamma(i)$ **do**
8: | | **if** $j \notin path$ **then**
9: | | | send($\langle s, c, path \rangle, j$)

10: **upon** max_disjoint_paths($Paths_{\langle s,c \rangle}$) $> f$ **do**
11: | DolevU_deliver($\langle s, c \rangle$)

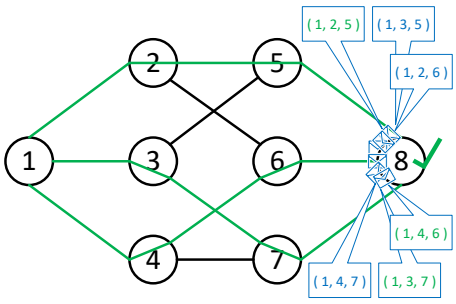


Figure 6.2: Example of content delivery with DolevU.

Remark 8. The `DolevU` protocol solves the any-to-any reliable communication problem in the minimal setting characterized in Remark 7 [30].

The `DolevU` protocol directly solves a more complex problem, the *Byzantine reliable broadcast* [24] one, when local broadcast link are assumed. Briefly, it is guaranteed that all processes eventually deliver the same set of contents.

Theorem 2. Let `DolevU` solve the reliable communication problem in *Static_Async_Global* assuming local broadcast links. Then, a content m is delivered by every correct process if it is delivered by any correct one, namely `DolevU` solves the *Byzantine reliable broadcast* problem.

Proof. When the reliable communication necessary correctness condition is met (Remark 7), the `DolevU` protocol guarantees that if the source p_s of a content c is correct, then any correct target eventually delivers c (Remark 8). This is not guaranteed in case of a faulty source: it may diverge from the protocol and it may prevent some targets from delivering its contents. The local broadcast links provide an additional guarantee: every message a process sends is received by all its neighbors. A correct source p_s multicasts message $\langle s, c, \emptyset \rangle$ to all of its neighbors in `DolevU`. It follows that if a correct process delivered $\langle s, c \rangle$, then message $\langle s, c, \emptyset \rangle$ has been sent to all neighbors of p_s, given the local broadcast links, and the claim follows. □

6.1.2 `DolevU` performance analysis

Message complexity. A source process executing `DolevU` floods its content over the system, starting from diffusing it to all of its neighbors. Every process that receives a content appends the identifier of its sender (neighbor) and then relays the message to all of its neighbors whose label has not been included yet. Such a procedure generates one message for each path in the graph between the source and every other node. Considering that the number of such paths may grow factorially in the size of the network and that the number of paths between two nodes in a complete graph K_n is equal to $\sum_{k=0}^{n-2} \frac{(n-2)!}{k!}$ [69], it follows that a factorial number of messages can be generated solving reliable communication with `DolevU` in the worst-case scenario.

Remark 9. The `DolevU` protocol achieves reliable communication with a factorial message complexity in the size of the system, either considering a synchronous or an asynchronous distributed system.

Delivery complexity. A process executing `DolevU` needs to analyze the paths traversed by a content. Specifically, it has to verify if the content passed through at least $f + 1$ disjoint paths in the network to be delivered, satisfying the safety property. No details are provided in [30] about how to count the number of disjoint paths traversed by a content. The topology of the communication system is unknown to the peers and Byzantine faulty nodes are potentially present in the network diffusing spurious messages, potentially containing non-existing paths of G. Consequently, processes cannot leverage algorithms defined on graphs, having polynomial computational complexity (Remark 1) to identify disjoint paths over

a set of many ones, because spurious messages may contain non-existing paths invalidating the calculus. To the best of our knowledge, the only methodology available to calculate the maximum number of disjoint paths in this setting is to convert such paths (discarding their first endpoint) into sets and to solve the corresponding set problem: *set packing*. The set packing problem considers an universe set of elements $\mathcal{U}$, a set of subsets $\mathcal{S} := \{S_1, S_2, \dots S_n\}, S_i \subset \mathcal{U}$ and a positive integer k, and the aim is to check whether $\mathcal{S}$ contains at least k mutually disjoint sets (i.e. a collection of k sets with no common element). The set packing problem is NP-Complete [40]. Therefore, a process verifying a content with `DolevU` needs to convert the paths traversed by contents into sets and to compute their set packing; if a process succeeds in solving the set packing problem for $k \geq f + 1$, then the associated content can be safely delivered. It follows that an NP-Complete problem has to be solved addressing reliable communication with `DolevU`. To the best of our knowledge, no alternative Byzantine tolerant methodology exists verifying the maximum number of disjoint paths traversed by a process in an unknown network.

Remark 10. The `DolevU` protocol achieves reliable communication with a NP-Complete delivery complexity, either considering a synchronous or an asynchronous distributed system.

Communication latency. Assuming a *Static_Sync_Global* system, the communication latency of `DolevU` is influenced by the topology of the network (more specifically, by its wide diameter) and by the placement of the faulty processes. The available bounds on the wide diameter (Remark 3) provide an upper bound to the communication latency of the protocol.

While spreading a content, `DolevU` generates a message for every possible path available in the network, and it produces them incrementally and synchronously with respect to their length. Specifically, if a source process triggers reliable communication with `DolevU` at t, it sends the content to all of its neighbors at $t + 1$, generating all paths of length one between itself all of its adjacent nodes. All messages are received at time $t + 1$ and then relayed to all the nodes adjacent to the neighbors of the source at time $t+2$. It follows that a content requires an amount of time bounded by the wide diameter of the network to traverse $f + 1$ disjoint paths. Therefore, the communication latency is over bounded by the size of the network n in the worst-case scenario.

Remark 11. The communication latency of `DolevU` is $O(n)$ in *Static_Sync_Global* systems.

6.2 Dolev protocol for routed networks, `DolevR`

Dolev [30, 31] defined an alternative and more efficient solution to the reliable communication problem, in *Static_Async_Global* systems, considering the additional assumption of a *routed communication network*, i.e. a distributed system where messages are relayed only through fixed and known *routes* (i.e. fixed paths): for every pair of processes p_i, p_j, there is a defined set of paths $\{\pi_1, \pi_2, \dots \pi_m\}$ toward which all messages exchanged between p_i and p_j are relayed. More in detail,

Dolev assumed every process knowing a disjoint path solution $\Pi_{i,j}$ of size $2f+1$ (all processes share the same solutions $\Pi_{i,j}$) between every pair of processes p_i, p_j.

Informally, whenever a process p_i aims to reliably communicate with a peer p_j with the protocol defined by Dolev, it routes the content over $\Pi_{i,j}$. Every process that receives a message over a specific path checks whether the inner content has been relayed over one of the routes in $\Pi_{i,j}$ between its source p_i and target p_j: in the positive case it forwards the message to the subsequent hop in the route, otherwise it discards the message. Every process p_j delivers every content from p_i that is received over $f+1$ paths in $\Pi_{i,j}$. The pseudo-code of the protocol is reported in Algorithm 2. We refer to this solution with `DolevR` in the following.

Algorithm 2 `DolevR`

```
1: upon DolevR_send(t, c) do
2:     for π ∈ Π_{i,t} do
3:         send(⟨i, t, c, π⟩, π[1])

4: upon receive(⟨s, t, c, π⟩, j) do
5:     if ∃π ∈ Π_{s,t}, ∃m ∈ ℕ^0, π[m − 1] = j, π[m] = i then
6:         if |π| = m then
7:             Paths_⟨s,c⟩ ← Paths_⟨s,c⟩ ∪ π
8:         else
9:             send(⟨s, t, c, π⟩, π[m + 1])

10: upon |Paths_⟨s,c⟩| > f do
11:     DolevU_deliver(⟨s, c⟩)
```

6.2.1 `DolevR` correctness analysis

The `DolevR` protocol assumes a routed network instead of an unknown one as `DolevU`. Nevertheless, no stronger connectivity requirement needs to be assumed to guarantee the correctness of `DolevR` as a solution to the reliable communication problem.

Remark 12. The `DolevR` protocol solves the any-to-any reliable communication problem in a routed communication network in the setting characterized in Remark 7 [30].

6.2.2 `DolevR` performance analysis

Message complexity. A source process p_i executing the `DolevR` protocol sends its content to a target process p_j inside messages that are forwarded over $2f+1$ disjoint routes $\Pi_{i,j}$. A content relayed over a set of disjoint routes between two peers may traverse every process of the system at most once. Precisely, the source sends one message to $2f+1$ neighbors and all other processes in $\Pi_{i,j}$ but p_i and p_j

transmit one message. Considered the upped bound on the wide diameter (Remark 3), the message complexity of *DolevR* is linear in the number of processes.

Remark 13. The DolevR protocol achieves one-to-one reliable communication with a message complexity $O(n)$, either considering a synchronous or an asynchronous distributed system.

Delivery complexity. Executing DolevR, processes relay only the contents that are exchanged over the fixed routes $\Pi_{s,t}$ between the source p_s and target p_t of a reliable communication instance. The target process p_t has to wait for messages carrying the content from $f + 1$ distinct routes among the one in $\Pi_{s,t}$, i.e. p_t stands by for $f + 1$ copies of the content arriving from distinct neighbors which precede the process in $\Pi_{s,t}$. It follows that a target process executes a procedure having constant computational complexity as many times as the maximum number assumed faults f with DolevR.

Remark 14. The DolevR protocol achieves reliable communication with a delivery complexity $O(f)$, either considering a synchronous or an asynchronous distributed system.

Communication latency. Assuming a *Static_Sync_Global* system, the communication latency of the DolevR protocol is influenced by the length of the longest path $\pi \in \Pi_{s,t}$ defined between a source p_s and a target p_t. Given the upper bound on the wide-diameter (Remark 3), It follows that the communication latency of DolevR is linear in the number of processes in the worst-case scenario.

Remark 15. The communication latency of DolevR is $O(n)$ in *Static_Sync_Global* systems.

6.2.3 DolevR optimality

We prove that the DolevR protocol is asymptotically optimal solving the reliable communication problem, in static distributed systems with globally bounded Byzantine failures, for all evaluation metrics we considered, namely that no other algorithm can solve the reliable communication problem under the same system assumptions with an asymptotically lower message complexity, delivery complexity, or communication latency.

Theorem 3. The DolevR protocol solves the reliable communication problem in static distributed systems, organized in a routed communication network, and assuming the globally bounded Byzantine failure model, with asymptotically optimal message complexity, delivery complexity, and communication latency.

Proof. The DolevR protocol achieves one-to-one reliable communication in a routed static communication network with message complexity $O(n)$, delivery complexity $O(f)$ and communication latency $O(n)$ (Remarks 13,14,15). We show that no algorithm is asymptotically more efficient solving the problem for the evaluation metrics we considered.

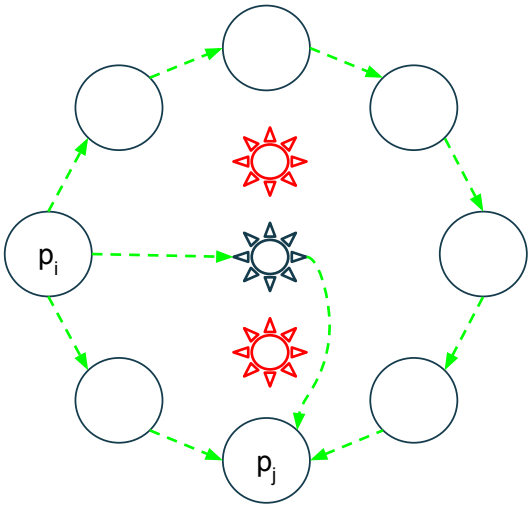


Figure 6.3: Supporting example for Theorem 3: generalized wheel $W(3,8)$, p_i and p_j are respectively source and target of a reliable communication instance, the sun-shaped nodes are the ones in K_3.

Let us consider two processes p_i and p_j, not connected by a link, respectively as the source and target of a reliable communication instance.

The target process relies on the messages it receives from its neighbors to deliver a content. Nevertheless, up to f of its neighbors could be Byzantine faulty and process p_j cannot identify them. Thus, a $O(f)$ procedure is required.

Given that p_i and p_j are not linked, a content must be relayed over fault-free paths (i.e. not including any faulty process) to achieve liveness of reliable communication. In the worst-case scenario, the length of the longest fault-free path is $n-k$. A graphical example is provided in Figure 6.3.

□

6.3 Certified Propagation Algorithm (CPA)

The Certified Propagation Algorithm (CPA) [50,67] is a solution to the reliable communication problem in *Static_Async* systems where the locally bounded Byzantine failure model is assumed.
It is a simple but efficient protocol defined by the following three rules:

- the source process delivers its content and sends it to all of its neighbors;
- a process delivers every content received directly (through a link) from its source and relays it to all of its neighbors;
- a process delivers every content received from $f+1$ distinct neighbors and relays it to all of its neighbors.

6.3.1 CPA correctness analysis

Several graph metrics [43,54,65,67,75] have been defined in the literature characterizing the correctness of CPA as a solution to the reliable communication problem in *Static_Async_Local* systems.
We recall the ones based on the minimum k-level ordering metric.

Remark 16. The one-to-all reliable communication problem can be solved in *Static_Async_Local*, considering process p_i as source, if and only if $h_i(G) > f$ [54].

Remark 17. The one-to-all reliable communication problem can be solved in *Static_Async_Local*, considering process p_i as source, if $j_i > 2f$ [54].

More in detail, the parameter h_i provides a 2-approximation on the maximum number of f local failures a *Static_Async_Local* system can tolerate assuming process p_i as the source. It has to be noticed that the computation of the two parameters $h_i(G)$ and $j_i(G)$ has a different complexity: j_i can be calculated with an algorithm having computational complexity polynomial in the size of the graph, whereas the computation of h_i has been proven to be equivalent to an NP-Hard problem [43].
The conditions recalled in Remarks 16,17 can be generalized to characterize the network requirements enabling any-to-any reliable communication in *Static_Async_Local*.

Corollary 1. The any-to-any reliable communication problem can be solved in *Static_Async_Local* if and only if $h(G) > f$.

Proof. The any-to-any specification requires every process to succeed in reliable communication toward every peer. The claim follows from Remark 16 and from the definition of $h(G)$. □

Corollary 2. The any-to-any reliable communication problem can be solved in *Static_Async_Local* if $j(G) > 2f$.

Proof. The any-to-any specification requires every process to succeed in reliable communication toward every peer. The claim follows from Remark 17 and from the definition of $j(G)$. □

It has to be noted that both the parameters h_i and h are NP-Hard to compute [43,65].

6.3.2 CPA performance evaluation

Message complexity. Every process relays every content it delivers to all of its neighbors only once. It follows that the message complexity of CPA is $O(E)$.

Delivery complexity. Every process waits for $f + 1$ copies of the same content from distinct neighbors to deliver it. It follows that the delivery complexity of CPA is $O(f)$.

Communication Latency. Assuming a *Static_Sync_Local* system, the communication latency of CPA is influenced by the topology of the network (more specifically,

by the parameters $j(G)$ and $h(G)$) and by the actual placement of the faulty processes.

Theorem 4. Given any source process p_i and any target process p_j in a *Static_Sync_Local* system, let x be the level order index of L_x in a *MKLO* $\mathcal{L}_k(G, p_i)$ where p_j is included (i.e. $p_j \in L_x$), let t^{f+1} be the value of x in $\mathcal{L}_{f+1}(G, p_i)$, let t^{2f+1} be the value of x in $\mathcal{L}_{2f+1}(G, p_i)$, and let t^{*f+1} be the minimum value of x in $\mathcal{L}_{f+1}(G_{\hat{F}}, p_i)$ for every possible f-local set F. The communication latency CL of `CPA` solving the any-to-any reliable communication problem is bounded as follows:

$$d(G) \leq t^{f+1} \leq CL \leq t^{*f+1} \leq t^{2f+1}$$

Proof. A *MKLO* from a source process p_i models the spreading of a content sent by p_i with *CPA* assuming specific sets of processes C and F ($C \subset P$, $F \subset P$, $C \cup F = P$, $C \cap F = \emptyset$) respectively as correct and faulty. In detail, $\mathcal{L}_{f+1}(G, p_i)$ exactly characterizes the content spreading from p_i in a *Static_Sync* system assuming $C = P$. Indeed, p_i computes the content at t, and it is sent to, received by and delivered by all neighbors of p_i at time $t + 1$ according to `CPA` and the system assumptions; all neighbors of p_i are placed in L_1 of $\mathcal{L}_{f+1}(G, p_i)$. Subsequently, all processes in L_1 send the content to all of their neighbors at $t + 2$, and all the one that received it from at least $f + 1$ nodes in L_1 delivers it according to *CPA*. Again, these nodes are exactly the one in L_2 of $\mathcal{L}_{f+1}(G, p_i)$, and the reasoning extends to all other processes in the system, thus proving the lower bound $t^{f+1} \leq CL$.

The *MKLO* $\mathcal{L}_{2f+1}(G, p_i)$ assumes every processes having at least $2f + 1$ edges toward nodes in the previous level. A corruption set F containing exactly f nodes of every level $L_{i,i>0}$ in $\mathcal{L}_{2f+1}(G, p_i)$ is not guaranteed to be f-local, thus proving the upper bound $CL \leq t^{*f+1} \leq t^{2f+1}$. □

6.4 `MTD` protocol

Maurer et al. [60] analyzed the reliable communication problem in *Dynamic_CompSync_Global*, namely in dynamic distributed systems assuming the globally bounded failure model. Their work is a complete extension to the seminal contribution of Dolev [30] that identifies the necessary and sufficient condition enabling one-to-one reliable communication and it provides a solution adapting and improving `DolevU` protocol.

The reliable communication protocol proposed by Maurer et al. inherits the diffusion mechanism defined in `DolevU`, namely the contents are flooded over the system while collecting the label of the traversed processes: every process forwards every received message containing a content to all of its neighbors not yet included. Differently from `DolevU`: *i)* it employs a set data structure to collect the label of the processes that are traversed by a content, *ii)* it adopts a verification algorithm that verifies the size of the minimum vertex cut of the set of processes a content traversed (instead of the maximum number of disjoint paths), *iii)* and every process relays every received message to all of it neighbors not yet included in the data structure every time that its neighborhood changes. We report the complete

pseudo-code of the protocol in Algorithm 3 and we refer to this solution with MTD in the following.

Algorithm 3 MTD

```
1: upon MTD_send(c) do
2:     M ← M ∪ {⟨i, ∗, c, ∅⟩}

3: upon receive(⟨s, ∗, c, visited⟩, j) do
4:     visited ← visited ∪ {j}
5:     Visited⟨s,c⟩ ← Visited⟨s,c⟩ ∪ {visited}
6:     M ← M ∪ {⟨s, ∗, c, visited⟩}

7: upon minimum_cut(Visited⟨s,c⟩) > f do
8:     MTD_deliver(⟨s, c⟩)

9: upon local topology Γ(i) changes do
10:    for ⟨s, ∗, c, visited⟩ ∈ M do
11:        for j ∈ Γ(i) do
12:            if j ∉ visited then
13:                send(⟨s, ∗, c, visited⟩, j)
```

Notice that MTD can directly be employed also in *Static_Async_Global*, in fact, a static network can be seen simply as a dynamic network that does not change. It guarantees reliable communication in all *Static_Async_Global* systems where the conditions stated in Remark 7 are satisfied.

6.4.1 MTD correctness analysis

Maurer et al. [60] identified the necessary and sufficient conditions to solve the one-to-one reliable communication problem in *Dynamic_CompSync_Global*.

Remark 18. The one-to-one reliable communication problem can be solved from a process p_i to a process p_j at t in *Dynamic_CompSync_Global* if and only if dynamic minimum cut in $\mathcal{G}_{[t,\infty)}$ from p_i to p_j is (strictly) greater than $2f$ [60].

Differently from the conditions identified for *Static_Async_Global* and *Static_Async_Local* (Remark 7 and Corollaries 1,2), the one defined by Maurer et al. [60] satisfies the solvability condition of a single one-to-one reliable communication instance. It follows that, while the other conditions defined for static networks guarantee the solvability of infinite instances of the reliable communication problem, in this setting the correctness conditions need to be verified for every content sent at time t. Furthermore, the verification of the condition defined in Remark 18 is an NP-Hard problem to solve: given an evolving graph where each edge of it footprint appears exactly one, Kempe et al. [47] proved that it is NP-Hard to compute the dynamic minimum cut between any two nodes. The result extends to general TVGs.

6.4.2 MTD performance analysis

Message complexity. MTD employs the same mechanism of DolevU to diffuse contents, thus inheriting its inefficiency in message complexity. Although MTD improves with respect to DolevU, generating one message for all journeys crossing the same group of processes in the best-case scenario, the number of messages that can be generated in a reliable communication instance may still grow factorially with respect to the size of the system.

Remark 19. The message complexity of MTD is factorial in the size of the system.

Delivery complexity. MTD verified the minimum vertex cut of the collection of sets of processes a content has traversed over the system. The only Byzantine fault-tolerant methodology currently available verifying such a metric is the reduction to a *hitting set problem* instance. More in detail, the hitting set problem takes as input a collection C of subsets of a set S and a positive integer x, $x \leq |S|$, and it verifies whether a subset $S' \subseteq S$, $|S'| \leq x$, such that S' contains at least one element for each subset in C, exits. The hitting set is an NP-Complete problem [40].

Remark 20. MTD achieves reliable communication with a NP-Complete delivery complexity.

Communication latency. The communication latency of MTD strongly depends on the evolution of the dynamic graph. Assuming process p_s and p_t are respectively the source and target of a reliable communication instance that starts at t', let $\mathcal{G}_{t',t''}$ be the temporal subgraph of $\mathcal{G}$ where t'' is the minimum $t \in \mathcal{T}$ such that there exist a set of journeys from p_s to p_t whose dynamic minimum cut is greater than $2f$. The communication latency is upper bounded by the value $t'' - t'$.

Part I

Static Faults, Static Network

Chapter 7

Reliable Communication in Static Networks: Motivations and Challenges

We detail in this chapter some of the challenges and open problems left in the literature addressing the reliable communication problem in static distributed systems with static Byzantine faults.

DolevR, DolevU, and MTD are the solutions available in the literature to the problem in static distributed systems assuming a globally bounded failure model. They are all optimal in terms of the number of tolerated faults, indeed they all solve the reliable communication model under the weakest possible system assumptions (Remark 7). Nevertheless, both DolevU and MTD do not scale, due to their high message complexity and delivery complexity (Remarks 9, 10, 19, 20), and therefore they may not be practically employed on real distributed systems. On the other hand, DolevR shows that additionally assuming processes (partially) knowing the topology of the communication network is possible to obtain an optimal solution in terms of message complexity, delivery complexity, and communication latency. This opens to several questions: is it possible to efficiently achieve reliable communication in case the topology (the routes) of the communication network is unknown to the processes? It is possible to define a solution with lower message complexity and/or delivery complexity with respect to DolevU and MTD?

Open problems

Considered the analysis provided in Chapter 6 on protocols solving the reliable communication problem in static distributed systems, it is possible to deduce that there exists a solution, CPA, efficiently solving the reliable communication problem, in the locally bounded Byzantine failure model, on distributed systems with unknown network topology.

On the other hand, there is DolevR which is asymptotically optimal solving the problem in the globally bounded failure model, for all the performance metrics we

evaluate, but it assumes a partial knowledge on the communication network given to the processes (the defined routes), and DolevU and MTD that work in unknown networks instead but they are quite unrealistic to employ, given the high message and delivery complexities.

Despite the two failure distributions differ, one can include the other: a globally bounded distribution of f failures in a network is also a locally bounded distribution: f faulty processes arbitrarily spread in the system cannot aggregate being more than f in the neighborhood of any node. What prevents one from directly use CPA as reliable communication protocol in unknown networks are its topological correctness conditions. We saw that CPA requires the existence of certain *MKLOs* to ensure reliable communication, and to verify whether the problem can be solved with CPA assuming a certain amount of failures f may be equivalent to solve an NP-Complete problem (Section 6.3.1). The network conditions required by DolevU to solve the reliable communication problem are generally weaker with respect the one of *CPA* ($j(G) > 2f$ implies node connectivity greater than $2f$), and they are testable with a polynomial algorithm (Remark 1). A graphics comparison between the two topological conditions is provided in Figure 7.1.

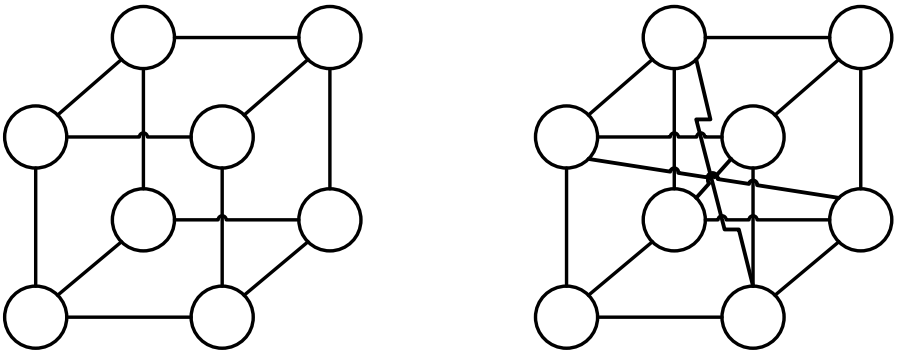


Figure 7.1: Comparative example among the correctness conditions of DolevU and CPA. Assuming a single Byzantine faulty process present in the system, the network on the left supports reliable communication with DolevU but CPA does not; the network on the right allows reliable communication with both DolevU and CPA.

Contributions

In Chapter 8, we analyze and optimize the DolevU protocol, aiming to improve its message complexity and delivery complexity.
In Chapter 9, we combine a reliable communication protocol with a topology reconstruction procedure, aiming to define an optimal reliable communication primitive in unknown networks.
In Chapter 10, we define a prototype of a cryptographic reliable communication protocol.

Chapter 8

Modified Dolev Protocol (BFT)

In this chapter, we define and analyze a modified version of the DolevU protocol, seeking for a practically employable solution to the reliable communication problem, in the globally bounded failure model, without introducing further assumptions with respect to the necessary one identified by Dolev (Remark 7). Specifically, we take as base the DolevU protocol as it is and we further dig inside several of its aspects, aiming to design a more efficient solution.

The results presented in this chapter were published in [11, 14, 17].

8.1 System model

We consider either a *Static_Async_Global* or *Static_Sync_Global* system, where the node connectivity k of the communication network G is assumed greater than twice the maximum number f of faulty processes (in order to enable reliable communication, Remark 7 in the globally bounded failure model), i.e. $k > 2f$.

8.2 Digging into the verification algorithm

We saw in Section 6.1.2 that no fault-tolerant solution besides solving the NP-Complete set packing problem is available at the moment verifying a content with DolevU. MTD employs an alternative verification procedure that computes the minimum cut of the paths (precisely, sets of processes) traversed by a content. As shown in Section 6.4.2, on the delivery complexity point of view, the two verification procedures are asymptotically equivalent: they both require a reduction to a NP-Complete problem to be solved. On the other hand, we recalled in Remark 2 that, given a pair of nodes in a graph and assuming an upper bound to the length of the paths, the local node connectivity may be greater than or equal to the maximum number of disjoint paths available between the vertices. It is possible to take advantage of this graph property designing a more efficient protocol with respect to the message complexity and communication latency. For these reasons, we adopt as verification algorithm the one of MTD.

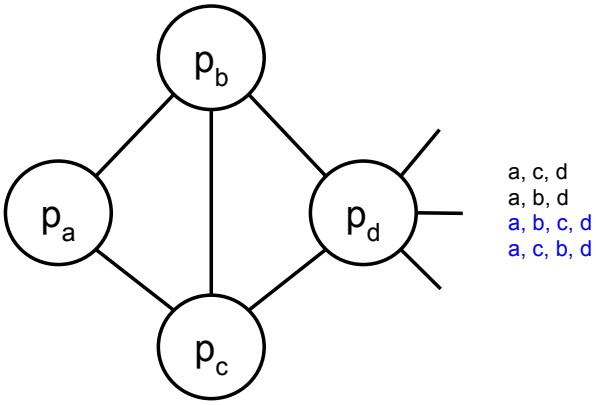


Figure 8.1: Example of a *Static_Sync_Global* system where the propagation algorithm of MTD allows to save messages with respect to DolevU. Assuming a source process p_a sending a content at t, process p_d receives the first two paths in Figure at time $t+2$ and the others at $t+3$. Process p_d relays to its neighbors only one among the former two with MTD, all paths are forwarded with DolevU otherwise.

8.3 Digging into the message format

DolevU and MTD slightly differ in the message format: the latter employs a set data structure, whereas a list one is used in the former to collect the identifier of the processes traversed by a content. However, the information on the order in which the processes are traversed is neglected while verifying a content in both protocols: both verification algorithms take as input a collection of sets validating a content. Moreover, the discarding of the traversing order information enables further saving in message complexity: several paths on the same processes are captured with a single set data structure in the best-case scenario. A clarifying example is provided in Figure 8.1.

For the reasons we listed, we adopt a set data structure to collect the identifier of processes traversed by a content.

8.4 BFT protocol

We propose modifications to the DolevU (MTD) protocol aiming to reduce its message complexity achieving reliable communication.

Modification 1. If a process p_t receives a content c directly from its source p_s (i.e., the source and the sender of a received content coincide), specifically if p_t receives a message $\langle s, c, \emptyset \rangle$ from p_s, then p_t delivers $\langle s, c \rangle$ [60, 67].

Modification 2. If a process p_t delivers a content c from p_s, then p_t discards all the *visited* sets it received associated to $\langle s, c \rangle$ and it relays $\langle s, c \rangle$ to all of its neighbors with an empty *visited*, namely it multicasts $\langle s, c, \emptyset \rangle$ [63].

Modification 3. A process p_i relays messages associated to $\langle s, c\rangle$ only to the neighbors that have not yet delivered $\langle s, c\rangle$.

Modification 4. A process p_i stops relaying further messages associated to $\langle s, c\rangle$ after it has delivered $\langle s, c\rangle$ and relayed $\langle s, c, \emptyset\rangle$.

We define the BFT protocol having

- the message format of MTD;
- the propagation algorithm of DolevU;
- the verification algorithm of MTD;
- the Modifications 1-4 employed.

For ease of exposition, let us refer with a−BFT to the BFT protocol where Modifications 1-4 are not adopted.

We detail the effects the Modifications 1-4 have on a−BFT and we prove the correctness of BFT in *Static_Async_Global*, namely that it solves the reliable communication under the weakest assumption identified by Dolev.

The a−BFT protocol correctly solves the reliable communication problem in *Static_Async_Global*.

Corollary 3. The a−BFT protocol solves the reliable communication problem in *Static_Async_Global*.

Proof. The a−BFT protocol generates one *visited* for every possible path traversable from the source toward every other process. Given the assumption on the node connectivity, every correct process will eventually receive a collection of *visited* related to a content such that their minimum cut is at least $2f + 1 - f$. □

Starting from the claim in Corollary 3, we plug the defined Modifications into a−BFT, proving they preserve the correctness of the protocol in the assumed system model.

Modification 1 adds another delivery policy to the protocol: the neighbors of a source can directly deliver its contents at their reception over the dedicated links. This modification results in a speed-up solving the reliable communication problem: the neighbors of a source directly deliver its content without waiting for messages carrying further *visited* collections. A clarifying example is provided in Figure 8.2.

Lemma 1. The a−BFT protocol solves the reliable communication problem in *Static_Async_Global* while adopting Modification 1.

Proof. The Modification 1 only affects the verification algorithm of a−BFT, introducing an additional delivery policy. It follows that only the safety property of reliable communication has to be verified for the investigated protocol.

The *Static_Async_Global* system model considers reliable and authenticated channels. If a process p_t received message $\langle s, c, \emptyset\rangle$ from p_s it had been sent by p_s, given the properties guaranteed by the links, and the claim follows. □

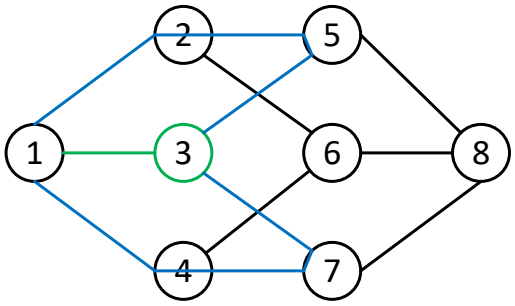


Figure 8.2: Speed-up introduced in a−BFT by Modification 1: assuming process p_1 as source and $f = 1$, process p_3 can deliver a content from p_1 when it receives the content over the directed link with p_1, without waiting for further copies over $(1, 2, 5)$ and $(1, 4, 7)$.

Modifications 2 and 3 aim to reduce the message complexity of a−BFT: while all possible *visited* sets in G are generated by a−BFT, these two Modifications prevent from generating many of them.

Lemma 2. The a−BFT protocol solves the reliable communication problem in *Static_Async_Global* while adopting Modification 2.

Proof. The information about the processes that relayed a content c is used by a peer p_i to decide whether such a content can safely be accepted. Once that c is delivered by p_i, the information about all the *visited* sets it received associated to c is not useful to any other process, because c has been already verified as safe. □

Modification 2 provides also a transparent way to get the neighbors of a process p_i know that a content c has been delivered by p_i. Indeed, a content is relayed with an empty *visited* set only by the processes that delivered it.

Lemma 3. The a−BFT protocol solves the reliable communication problem in *Static_Async_Global* while adopting Modifications 2 and 3.

Proof. Modification 3 acts only on the propagation algorithm of a−BFT, thus only the liveness of reliable communication needs to be proved.

Let us consider three processes p_x, p_y and p_z interconnected by the links $\{p_x, p_y\}$ and $\{p_y, p_z\}$, among the other processes and links of the system. Let us assume that process p_y delivered a content c whereas p_x and p_z did not, and that process p_x has a messege m carrying c to relay. Message m is not useful to process p_y, because it already delivered c. It follows that the only reason to relay m from p_x to p_y is to enable a third process p_z to accept c. Modification 2 imposes process p_y to relay the content with an empty *visited* to all of its neighbors after its delivery. It follows that any *visited* set coming from a process p_x will not increase the minimum cut of all the *visited* related to c received by process p_z with respect to the empty

visited forwarded by p_y, and the claim follows. A graphical example is provided in Figure 8.3. □

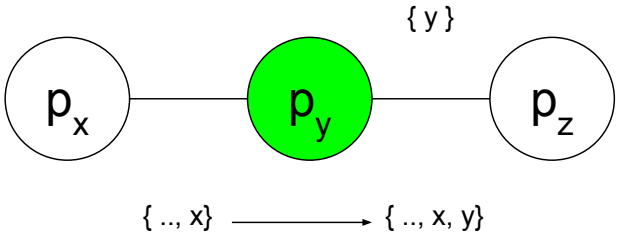


Figure 8.3: Supporting example for Lemma 3.

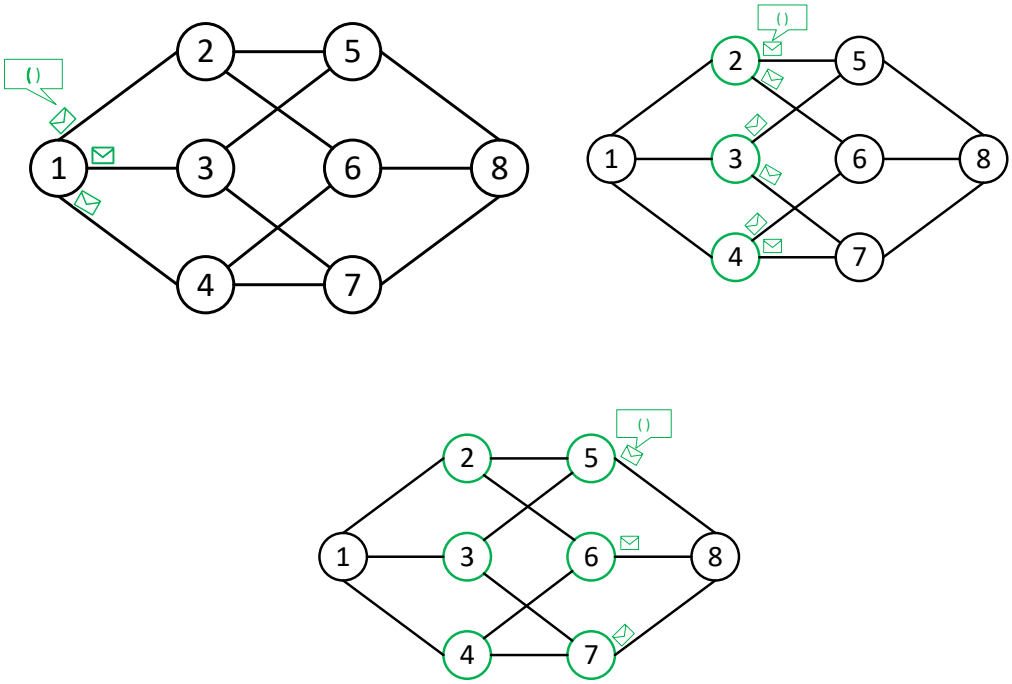


Figure 8.4: Example of content propagation with BFT.

Modification 4 is a sort of specular version of Modification 3 that introduces a halting condition for a process in relaying messages related to a content (that is, after its delivery).

Lemma 4. The a–BFT protocol solves the reliable communication problem in *Static_Async_Global* while adopting Modifications 2 and 4.

Proof. Modification 4 affects only the propagation algorithm of a–BFT, thus only the liveness property of reliable communication must be proved.

Let us assume two processes p_i and p_j connected by a link $\{i,j\}$, such that the former has delivered a content c while the latter has not and let us suppose, for the ease of contradiction, that p_j never delivers c. Modification 2 imposes process p_i to send the message $\langle s, c, \emptyset\rangle$ to p_j. It has to be noted that any further message related to c sent by p_i would contain at least one element in *visited*. It implies that any further message relayed by p_i will not increase the minimum cut of all *visited* sets received by p_j related to c. It follows that process p_j do not deliver c with a–BFT with Modification 2, but this contradicts Lemma 2, and the claim follows. □

For ease of completeness, the pseudo-code of BFT is provided in Algorithm 4, whereas a graphical execution can be found in Figure 8.4.

Algorithm 4 BFT

```
1: function DELIVER_AND_RELAY(⟨s, c⟩)
2:     Delivered ← Delivered ∪ {⟨s, c⟩}
3:     BFT_DELIVER(⟨s, c⟩)
4:     for k ∈ Γ(i) do
5:         if {k} ∉ Visited⟨s,c⟩ then
6:             send(⟨s, c, ∅⟩, k)

7: function BFT_SEND(c)
8:     for k ∈ Γ(i) do
9:         send(⟨i, ∗, c, ∅⟩, k)

10: upon receive(⟨s, ∗, c, visited⟩, j) do
11:     if ⟨s, c⟩ ∉ Delivered then
12:         visited ← visited ∪ {j}
13:         Visited⟨s,c⟩ ← Visited⟨s,c⟩ ∪ {visited}
14:         if j = s then
15:             DELIVER_AND_RELAY(⟨s, c⟩)
16:         else
17:             for k ∈ Γ(i) do
18:                 if k ∉ visited and {k} ∉ Visited⟨s,c⟩ then
19:                     send(⟨s, c, visited⟩, k)

20: upon minimum_cut(Visited⟨s,c⟩) > f do
21:     DELIVER_AND_RELAY(⟨s, c⟩)
```

8.5 Partial quiescency

One would expect any reliable communication protocol to eventually stop sending messages related to a content. We refer to a Byzantine tolerant reliable communica-

tion solution that guarantees such a property with *quiescent* protocol. Notice that a quiescent protocol slightly differs from one that terminates (i.e. that guarantees the liveness property): the latter ensures that the task is accomplished eventually, but messages may continue to circulate in the system.

In [30] Dolev assumed processes knowing the identifier of all peers in the system while solving the reliable communication (and Byzantine agreement) problem. Such an assumption enables any source process to target its contents to specific processes, namely, it allows any source to send specific contents to definite target processes. But mainly, it makes the `DolevU` protocol quiescent. Indeed, for every content diffused in the system, at most all possible *visited* sets (the power set of P) can be generated and diffused: the Byzantine processes cannot introduce further *visited* sets beyond those for every diffused content.

`BFT` is a not quiescent protocol in the system model we assumed: faulty processes may diffuse infinite messages related to spurious contents (because they are never accepted by the correct processes and the participant in the system are unknown). Nevertheless, it is more robust with respect to `DolevU` against certain Byzantine misbehavior, assuming processes not knowing the peers of the system: `BFT` guarantees that processes eventually stop sending the messages related to contents diffused by correct source, due to Modification 4. We refer to a protocol with such a guarantee with *partial quiescent*.

An additional assumption, weaker than the knowledge on the participants in the system, is probably required to define a quiescent solution, e.g. bounded namespace for the identifiers.

8.6 Selection policies

The main aim of the Modifications introduced in `BFT` is to reduce the message complexity of the state-of-art protocols, `DolevU` and `MTD`, solving the reliable communication in *Static_Async_Global*. However, not all possible *visited* sets that `BFT` may produce are required to a process in the system for delivering a specific content, as pointed out in the following example.

Let us consider a synchronous distributed system organized as in Figure 8.5: a multipartite cycle topology $MC_{8,3}$ (the network is 6-connected network, thus up to 2 faulty processes are tolerated). Let us suppose process p_1 as source of a content c generated at round r_0, processes p_4 and p_{22} as Byzantine faulty and let us focus on the processes in $L4$ as potential targets of the content. Assuming faulty processes simply relaying no message, it can be noticed that at round r_1 all correct processes in $L1$ and $L7$ delivers c with `DolevU` or `BFT`, and they forward it respectively to all processes in $L2$ and $L6$. At round r_2 no process in $L2$ or $L6$ is able to deliver the content, because the minimum cut associated to messages carrying c is f. Thus, all *visited* sets associated to the content received by such processes are forwarded to the subsequent levels $L5$ and $L3$. Also, processes in these levels are not able to deliver c, and they relay all the received *visited* sets to processes in $L4$. Looking at the generated *visited* sets, it can be noticed that they grew exponentially in both sides of the cycle: initially, two messages were received by every process in $L6$

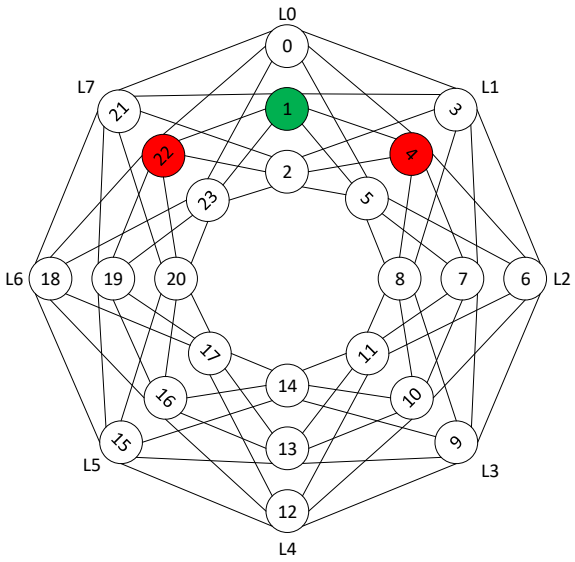


Figure 8.5: Example motivating Selection Policies (Section 8.6)

or $L2$, then each of them relayed both to every process in the subsequent level, generating 2^3 *visited* per side, then again all *visited* received by processes in $L5$ or $L3$ are forwarded to all processes in $L4$, generating $(2^3) * 3$ messages per side. From the point of view of processes in $L4$, many not useful *visited* are received, because they all contain one label of a process either in $L7$ or $L1$. For example, the *visited* sets $\{21, 18, 15\}$, $\{23, 20, 17\}$, $\{3, 6, 9\}$, $\{5, 8, 12\}$ would have been enough to any process in $L4$ to deliver c. However, no other Modification has been identified yet to further reduce the number of generated messages while guaranteeing the liveness of reliable communication.

Nonetheless, BFT solves the problem in an asynchronous system and messages may be relayed with an unpredictable delay. It implies that a process can potentially wait before forwarding all the *visited* sets it has received related to a content. Specifically, it may relay a bounded number of *visited* sets in a time window, namely fixing the rate of *visited* sets to relay per round. It follows that processes may adopt specific policies relaying their content, giving priority to certain *visited* sets.

We propose two forwarding policies, namely, we define two alternative procedures that iteratively select a bounded number of *visited* per content to relay. They are named *multi-random* and *multi-shortest*, and they are detailed in Algorithm 5. The difference between the two policies is on how messages are selected among the ones to relay: the *multi-random* randomly picks *visited* sets to relay, the *multi-shortest* gives priority to the smallest sets. A process p_i executing BFT relays a message $\langle s, c, visited \rangle$ to all of its neighbors p_j such that $j \notin visited$: they are the only ones that potentally allows a process p_j to deliver. Therefore, the forwarding policies select one *visited* set (randomly or one of the shortest) and check if it is useful to some of it neighbors that have not delivered yet the related content. This

selection contiues till either *i)* at least one useful *visited* has been selected for every neighbor p_i, or *ii)* a fixed threshold has been reached.

Notice that these selection policies additionally mitigate Byzantine faulty processes flooding the system: although spurious contents are never delivered by correct processes, messages carrying them can still circulate being relayed by all processes. The selection policies reduce in some sense the number of spurious messages Byzantine processes may diffuse.

Algorithm 5 `Selection Policies`

```
1: function GET_NEIGHBORS_DELIVERED(⟨s, c⟩)
2:     for k ∈ Γ(i) do
3:         if {k} ∈ Visited_⟨s,c⟩ then
4:             Neighbors_delivered ← Neighbors_delivered ∪ {k}
       return Neighbors_delivered

5: function SEND(⟨s, c, visited⟩, k)
6:     ToRelay_⟨s,c⟩ ← ToRelay_⟨s,c⟩ ∪ visited

7: function SELECT_MESSAGES(⟨s, c⟩)
8:     if MULTIRANDOM_POLICY then
9:         ToRelay_list_⟨s,c⟩ ← SHUFFLE(LIST(ToRelay_⟨s,c⟩))
10:    else if MULTISHORTEST_POLICY then
11:        ToRelay_list_⟨s,c⟩ ← SORT(LIST(ToRelay_⟨s,c⟩))      ▷ order visited by their size
12:    Nodes_to_relay ← Γ(i) − GET_NEIGHBORS_DELIVERED(⟨s, c⟩)
13:    for visited ∈ ToRelay_list_⟨s,c⟩ do
14:        number_missing ← |Nodes_to_relay|
15:        if number_missing = 0 or |Selected_visited_⟨s,c⟩| = threshold then
16:            break
17:        Nodes_to_relay ← Nodes_to_relay ∩ visited
18:        if |Nodes_to_relay| ≠ number_missing then
19:            Selected_visited_⟨s,c⟩ ← Selected_visited_⟨s,c⟩ ∪ visited
20:            ToRelay_⟨s,c⟩ ← Selected_visited_⟨s,c⟩ \ ToRelay_⟨s,c⟩
       return Selected_visited_⟨s,c⟩
```

8.7 BFT performance analysis

We simulate executions of BFT in a *Static_Sync_Global* in order to evaluate its performances and to have a comparison with the state-of-art solutions.

We consider the following parameters to setup our simulations:

- n, *i.e.* the size of the network considered;
- k, *i.e.* the vertex connectivity of the network considered;
- *topology*, *i.e.* the topology of network considered;

- *channel capacity*, *i.e.* the maximum number of messages related to a content that a process can send in a link per round;
- *selection policy*, *i.e.* one among *multi-shortest* and *multi-random*.

In order to carry an analysis as complete as possible, we consider the following network topologies:

- k-regular k-connected random graph [73];
- k-pasted-tree [4];
- k-diamond [4];
- multipartite cycle;
- Barabási-Albert graph [5].

We execute our simulations either considering the maximum number of tolerable faulty processes, thus for every k-connected network we assume $f = \lfloor (k-1)/2 \rfloor$ failures (Remark 7), or we test all possible values for f between 0 and $\lfloor (k-1)/2 \rfloor$. In any case, processes deliver a content only when the related *visited* have a minimum vertex cut greater than $\lfloor (k-1)/2 \rfloor$.
We move from considering the scenario where all processes are correct to the case in which f Byzantine processes act as crash failures. We made use of the implementation provided by Gainer-Dewar and Vera-Licona [38] for the algorithm defined by Murakami and Uno [62] to solve the minimum cut reduction to the hitting set problem.
We consider two configurations for the channel capacity: *bounded* and *unbounded*. The former constrains processes to send a limited number of messages per link in every round, the latter imposes no restriction. For the bounded case, we assume a bound for the channel capacity equal to $f + 1$ messages. We simulate one-to-all reliable communication instances where the source and the faulty processes are randomly selected.

For all the results we present, we directly plot all the measures we got as points (except for Figures 8.6,8.12 where the mean of the measures is shown) in order to show their distributions, and we accordingly increase the size of the points with higher density.

The source code of the simulations is available in [13].

Comparison with the state-of-art

We start comparing the message complexity of the state-of-art solutions with BFT. We consider k-regular k-connected random graphs, we assume the node connectivity k either equal to 3 or 5 and we simulate DolevU, MTD and BFT supposing unbounded channels and all correct processes and we vary the size of the network from $n = 6$ to $n = 20$.

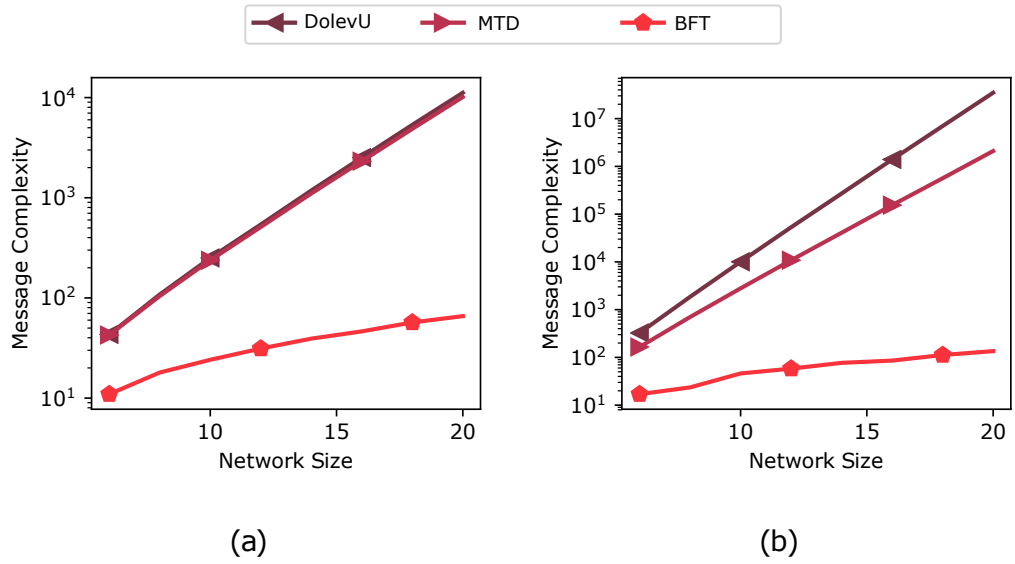


Figure 8.6: **State-of-art protocols VS BFT, message complexity.** Random regular graphs, unbounded channel capacity, $f = 0$. (a) $k = 3$, (b) $k = 5$.

We already remarked about the lack in the state-of-art protocols of a halting condition, in fact, they generate all source-to-all *paths/visited* in every execution. It can be noticed in Figure 8.6 that the modifications we defined have a remarkable impact on the message complexity even in a small and all-correct scenario. It can also be noticed the advantages gained by choosing a set data structure to collect the label of the traversed processes with respect to a list one (the former employed in MTD, the latter in DolevU).

Multi-random VS multi-shortest

We proposed as a countermeasure against the capability of Byzantine faulty processes to flood the network a constraint on the channel capacity, limiting the number of messages that every process can send over a link per round, and we set this bound equal to $f + 1$ for each content. We then defined two forwarding policies to select which *visited* relay in the actual round. Assuming bounded channels, we compare the *multi-random* and *multi-shortest* policies, considering networks of size $n = 100$, random regular graphs, multipartite wheel, k-diamond and k-pasted-tree, and f Byzantine failures. The results are presented in Figures 8.7 and 8.8 (notice that the scale of the graphics in Figure 8.7 is logarithmic). Starting with the multi-random policy, it can be seen in Figure 8.7 that, while for some graphs the multi-random policy acts smoothly, the random regular, the multipartite wheel graphs and the k-diamond, there exist topologies where the communication latency and message complexity may conspicuously increase (k-pasted-tree). This leads us to discard such a policy to be one generally employable. Contrarily, the performance achieved by adopting the multi-shortest policy appears to not suffer from such a weakness (Figures 8.8). Therefore, we further investigate such a selection policy while increasing the size of the network.

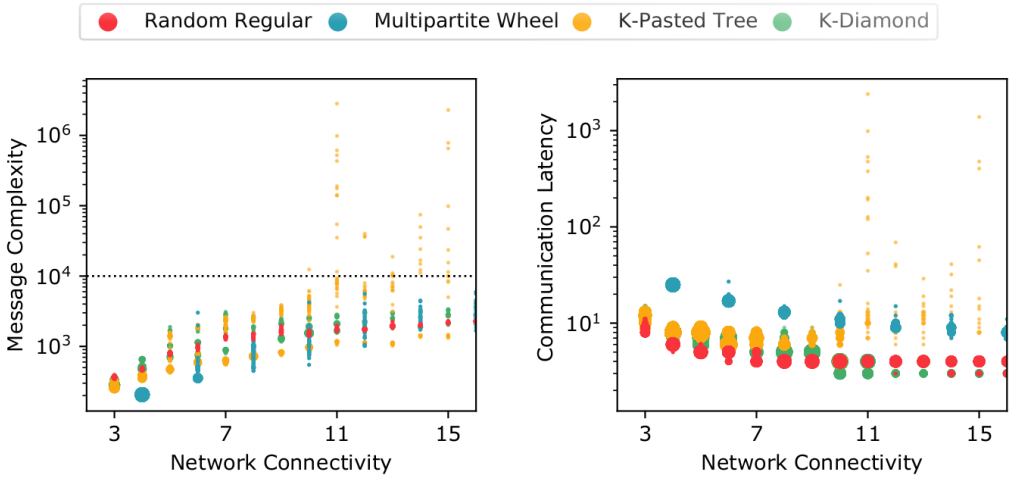


Figure 8.7: **Multi-Random selection policy, message complexity and communication latency.** Bounded channel capacity, $n = 100$, $f = \lfloor (k-1)/2 \rfloor$.

Multi-shortest selection policy evaluation

We carry further tests on BFT with the multi-shortest selection policy. We consider bounded channel capacity, topologies random regular, multipartite wheel, k-diamond and k-pasted-tree of size $n = 150$ and $n = 200$, and f Byzantine failures.

First results are presented in Figures 8.9 and 8.10. It is possible to see that the trends of the message complexity and broadcast latency keep defined employing our protocol joined with the multi-shortest policy while increasing the size of the network. Specifically, the message complexity keeps always close to or below the n^2 boundary. It can also be deduced that a regular network does not necessarily results in optimal performances employing our protocol, indeed there are notable differences in the results obtained considering different regular topologies.

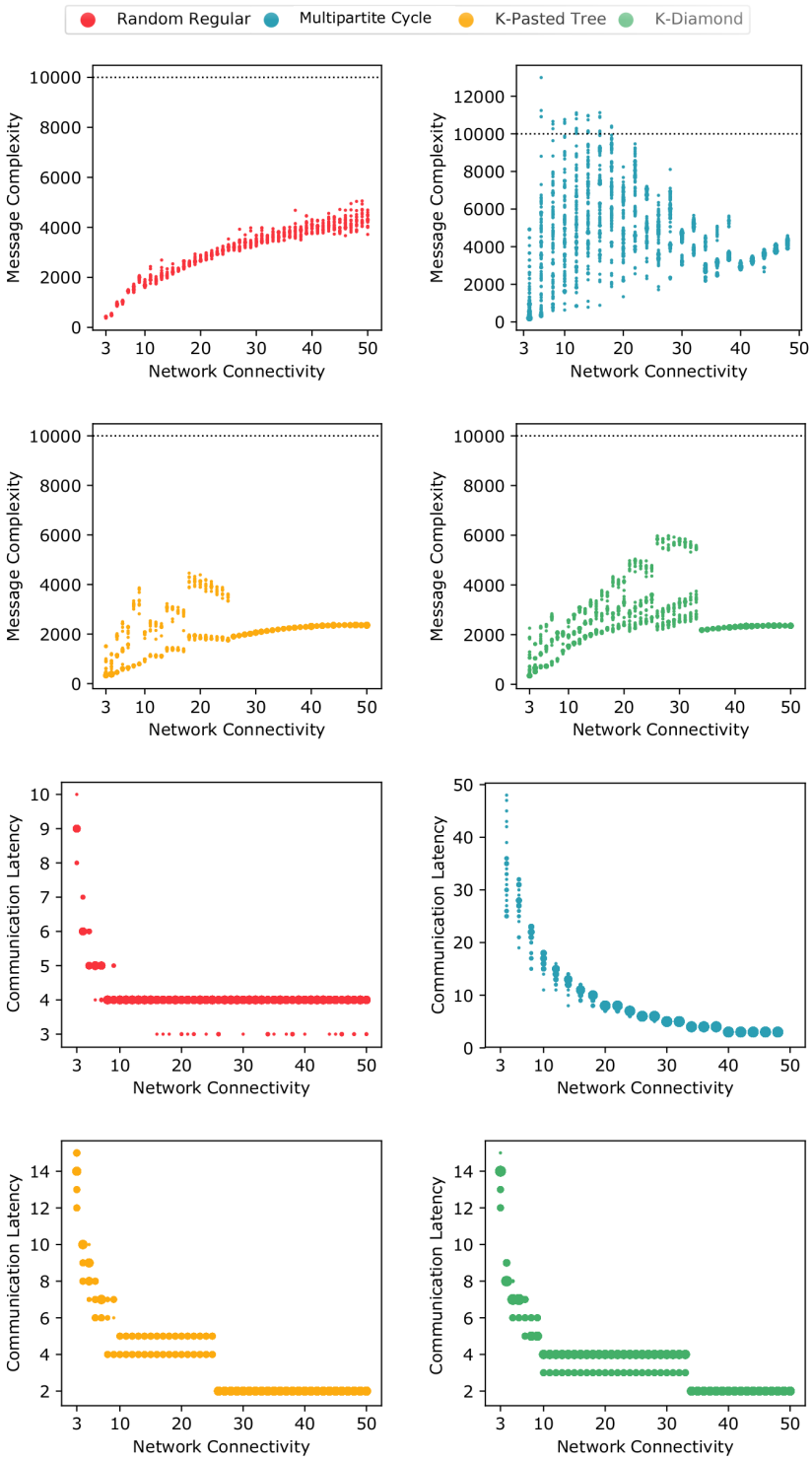


Figure 8.8: **Multi-Shortest selection policy, message complexity and communication latency, n = 100.** Bounded channel capacity, $f = \lfloor (k-1)/2 \rfloor$.

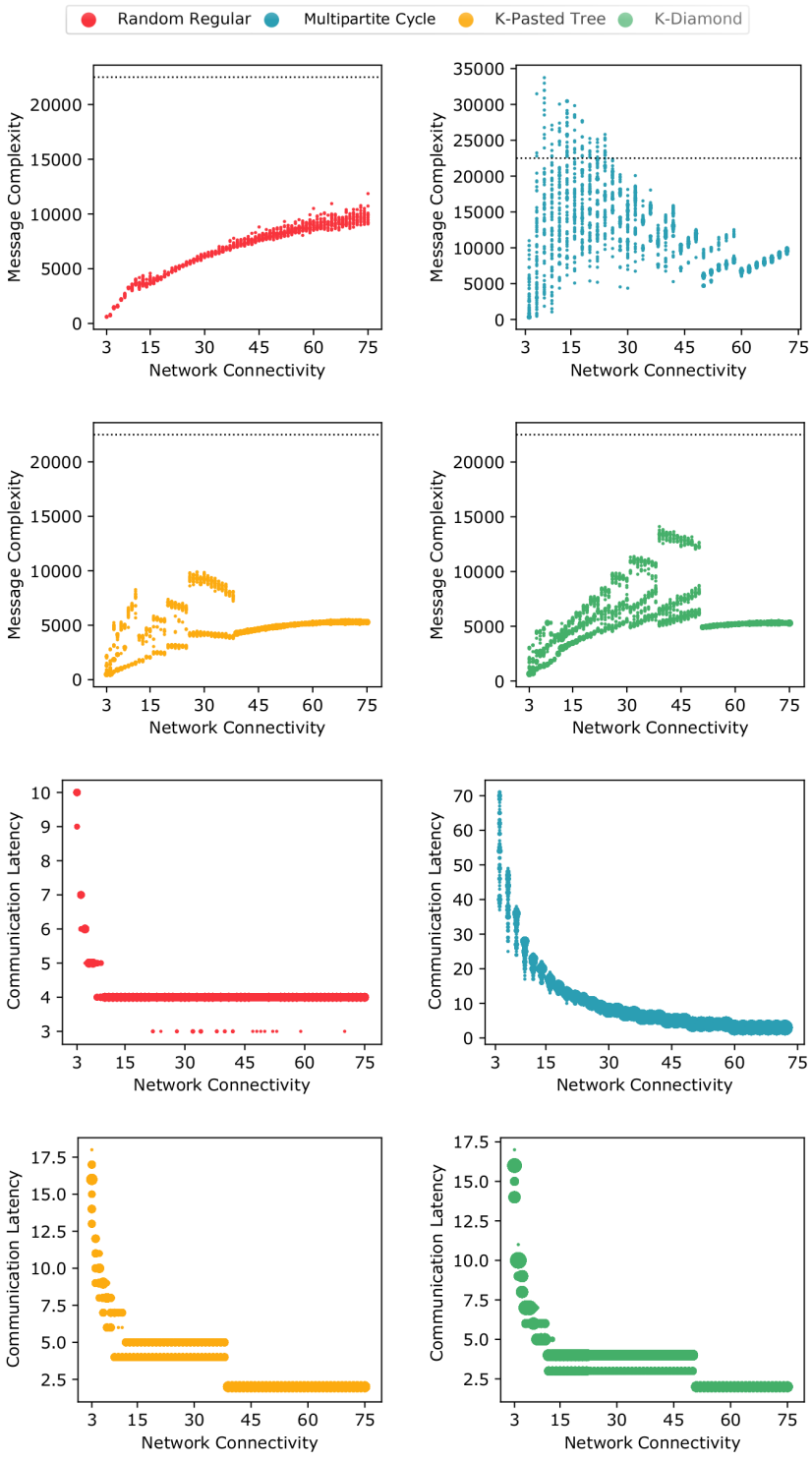


Figure 8.9: **Multi-Shortest selection policy, message complexity and communication latency, n = 150.** Bounded channel capacity, $f = \lfloor (k-1)/2 \rfloor$.

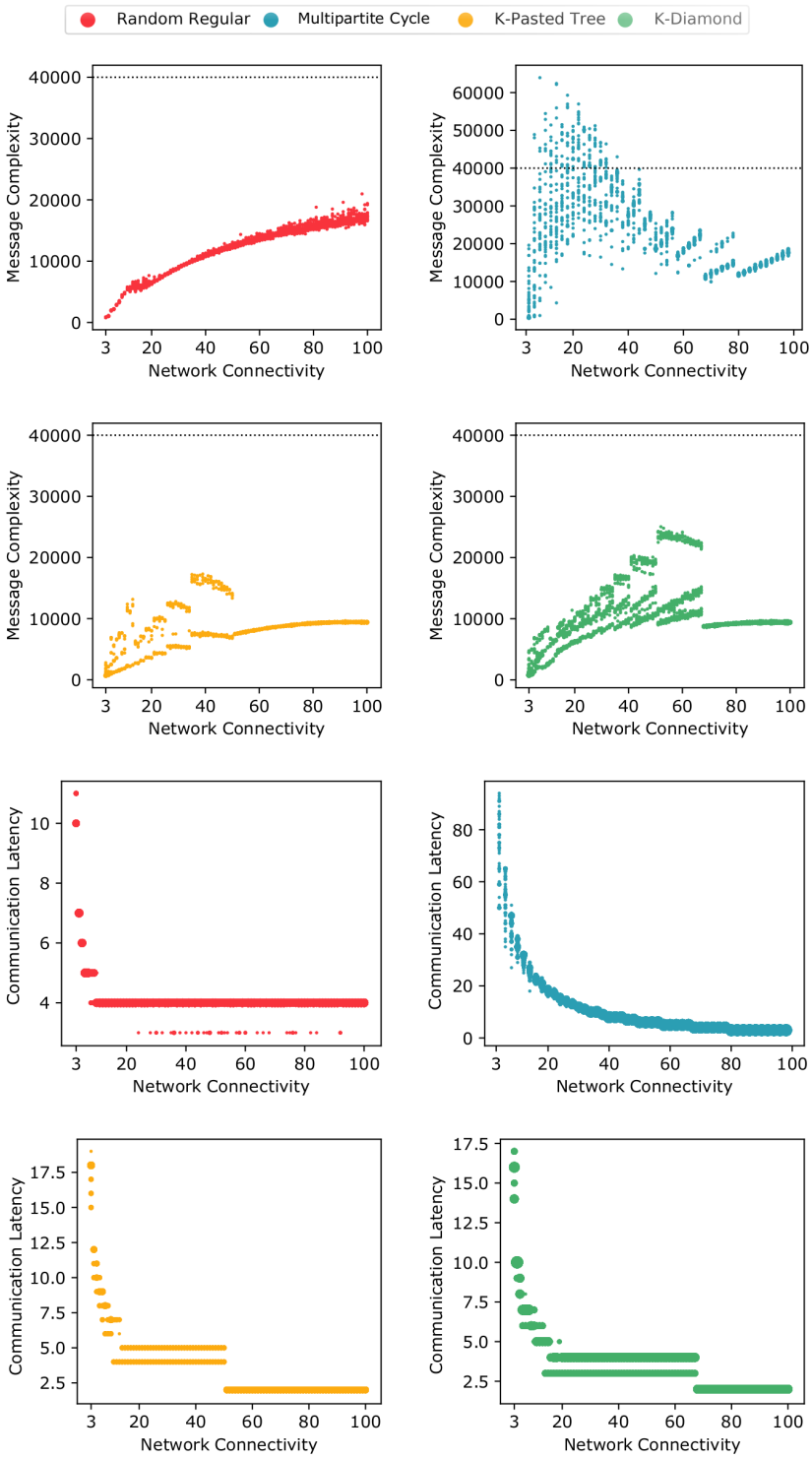


Figure 8.10: **Multi-Shortest selection policy, message complexity and communication latency, n = 200.** Bounded channel capacity, $f = \lfloor (k-1)/2 \rfloor$.

It can also be noticed from the distribution of the measures that there are sev-

eral topologies (k-pasted-tree, k-diamond, and especially multipartite cycle) where the placement of the source and the Byzantine failures play a remarkable impact on the message complexity.

In order to evaluate the effects of the multi-shortest selection policy on the communication latency of the protocol, we evaluate BFT both considering a bounded and unbounded channel capacity. The obtained results are depicted in Figure 8.11, where it can be deduced that the policy we defined introduces a negligible additional delay.

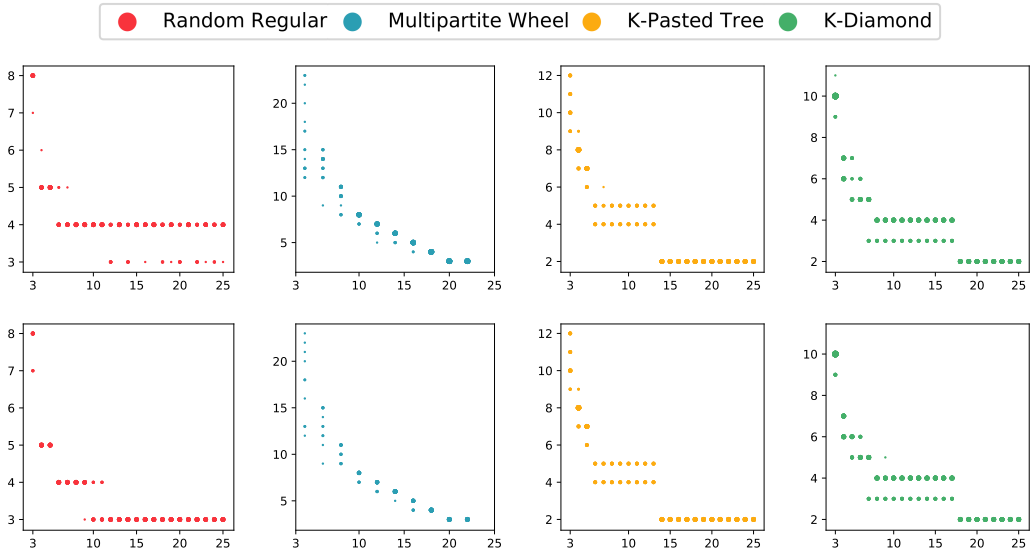


Figure 8.11: **Communication Latency, Multi-Shorted selection policy with bounded capacity VS no selection policy with unbounded capacity.** *X-axis*: network connectivity; *Y-axis*: broadcast latency; *first row*: Multi-Shortest bounded channel; *second row*: unbounded channel. $n = 50$, $f = \lfloor (k-1)/2 \rfloor$.

Varying the number of failures

We evaluate how the message complexity of BFT evolves when the number of faulty processes is not maximal.

We consider random regular, multipartite cycle, k-pasted-tree, and k-diamond graphs. We suppose different values for the network connectivity k, ranging from 3 to 50, testing all possible values for f from 0 to $\lfloor (k-1)/2 \rfloor$ while imposing every process to deliver a content only when the associated minimum cut is greater than $\lfloor (k-1)/2 \rfloor$.

We report the results we obtained in Figure 8.12. It is possible to deduce the number of faults effectively present in a network differently impacts the message complexity of the protocol depending on the considered topology. There are settings where the message complexity remains constant independently from the number of actual failures (random regular and k-pasted-tree) or even slightly reduces (in k-diamond, due to the lower graph density). Whereas on multipartite cycle graphs,

it has a relevant impact in lower connected instances that reduces while increasing the node connectivity.

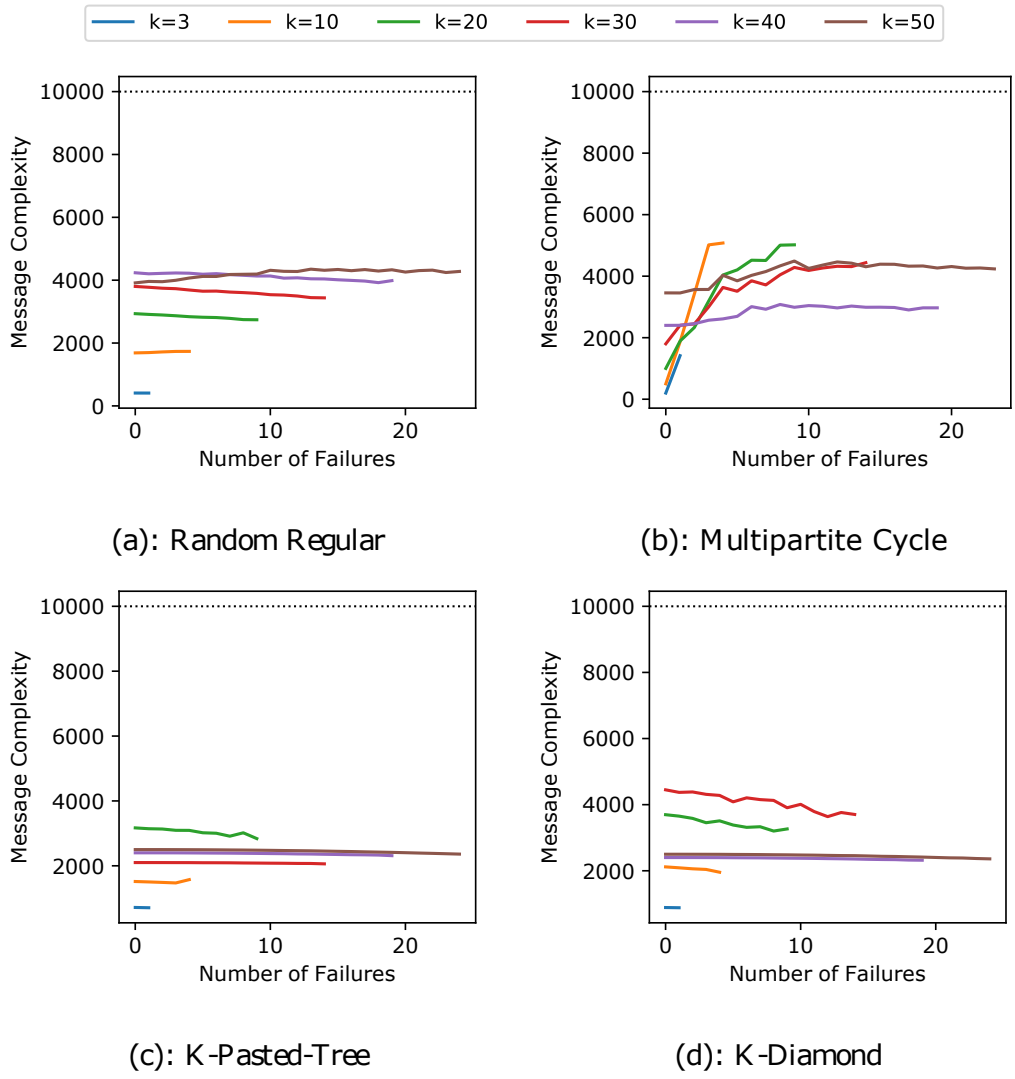


Figure 8.12: **Multi-Shortest selection policy, varying the number of actual faults, message complexity.** $n = 100$.

Barabási-Albert graph

We separately evaluate BFT in a Barabási-Albert graph while varying the attachment parameter m, in order to analyze our protocol on a topology having different degree distribution with respect to the previously analyzed. The results are reported in Figure 8.13. The BFT protocol and the multi-shortest selection policy keep performing in the same manner as previous scenarios. In order to have a comparison with the topologies analyzed before, we plot in Figure 8.14 the relation between the attachment parameter m and the network node connectivity.

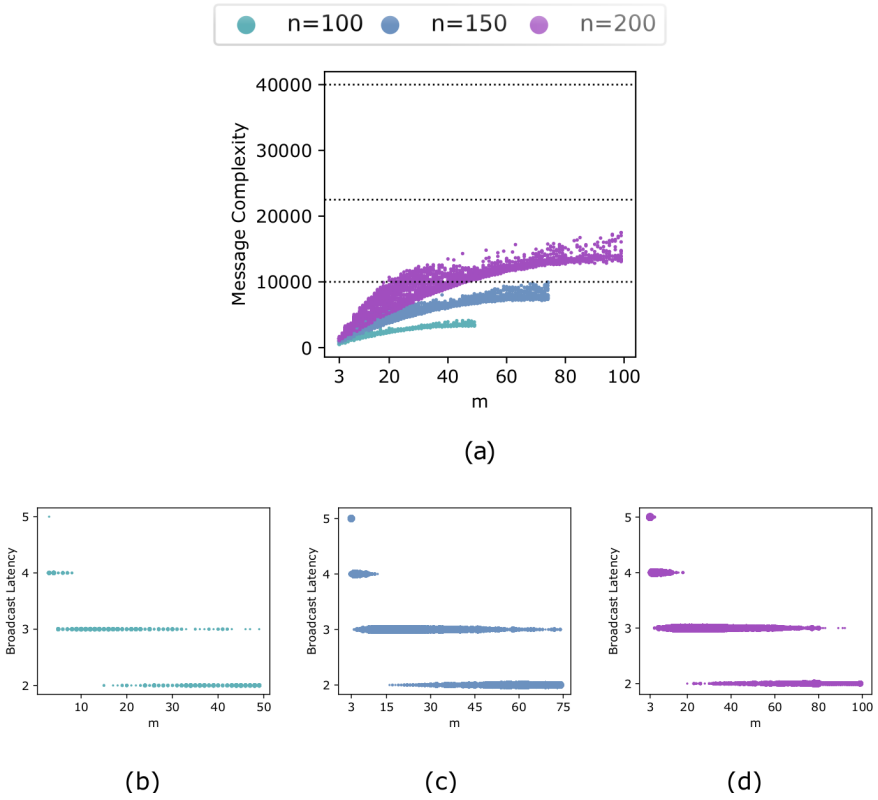


Figure 8.13: **Multi-shortest selection policy, Barabasi-Albert graph, message complexity and communication latency.** $f = \lfloor (k-1)/2 \rfloor$, bounded channel capacity, $n = 100, 150, 200$ (a) message complexity, (b,c,d) communication latency.

8.8 Limitations

We showed through simulations that the BFT protocol and the multi-shortest selection policy allow to significantly reduce the message complexity of the state-of-art protocols solving the reliable communication problem in unknown networks assuming the globally bounded Byzantine failure model. Such simulations considered a synchronous system, which by construction guarantees a timed evolution of the executed distributed protocol. Despite the proposed modifications do not generate more messages than the best performing state-of-art MTD, they may lose efficiency moving to an asynchronous system. In fact, the Modifications we introduced leverage process deliveries to reduce the amount of generated messages. In an asynchronous system, processes and links may introduce unbounded delays in the computations and message transmissions. Specifically, in the scenario where some of the peers are very slow, preventing any process from delivering the content, the message complexity may continue to increase till new nodes accept the content.

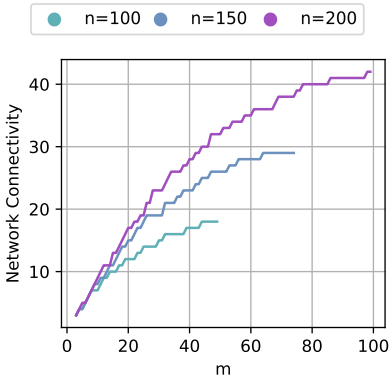


Figure 8.14: **Barababasi-Albert Network, relation between the attachment parameter m and the network connectivity**.

8.9 Conclusion

We revisited the available solutions to the reliable communication problem in general networks hit by up to f arbitrarily distributed Byzantine failures and proposed modifications following performance-related observations. Although the delivery complexity of our protocol remains unchanged with respect to the state-of-art solutions, our experiments show that it is possible to drastically reduce the message complexity (from factorial to polynomial in the size of the network), practically enabling reliable communication in larger systems assuming only the required assumptions identified by Dolev (Remark 7) There are several open problems that may follow: is it possible to define a solution to the hitting set problem suited for the specific input generated by our protocol? Is it possible to remove from the system the contents generated by Byzantine processes? And under which assumption? Which are the graph parameters that govern the message complexity of our protocol? Nevertheless, the obtained gain in performance may be lost when assuming an asynchronous system experiencing delays. It remains an open problem the possibility of defining an efficient reliable communication protocol for *Static_Async_Global* under the strict assumptions identified by Dolev. The answer to this question will provide several insights on the reliable communication problem in dynamic distributed systems.

Chapter 9

CombinedRC

We show in this chapter that considering some additional assumptions further than the necessary ones identified by Dolev (Remark 7), it is possible to efficiently solve the reliable communication problem in *Static_Async_Global* having an unknown network with optimal message complexity and optimal delivery complexity (Section 6.2.3). Specifically, we show how to combine Byzantine fault-tolerant topology reconstruction with a reliable communication primitive to make it efficient, and we characterize the set of assumptions that makes such an objective achievable.

The results presented in this Chapter were published in [15].

9.1 System model

We assume a *Static_Async_Global* system in this chapter, where the node connectivity k of the communication network G is greater than twice the maximum number f of faulty processes (in order to enable reliable communication, Remark 7 in the globally bounded failure model), i.e. $k > 2f$.
We alternatively consider *unicast* (*UL*) or *local broadcast* (*LBL*) links.
We either suppose the *known neighborhood* assumption (*KN*) or the *unknown neighborhood* assumption (*UN*).

9.2 The topology reconstruction problem

Given a distributed system of correct and Byzantine faulty processes interconnected by an unknown communication network G, the aim of a distributed protocol addressing the *topology reconstruction problem* is to enable all correct processes p_i to reconstruct a subset of the topology of the communication network, namely to output a subgraph G_i of G.

The Byzantine fault-tolerant topology reconstruction problem has been analyzed by Nesterenko and Tixeuil [63] in *Static_Async_Global* systems. Then, transient faults (that occur for a bounded period) have been considered by Dolev et al., defining a self-stabilizing Byzantine-tolerant solution [32, 33].

A topology reconstruction G_i is expected to be as close as possible to the actual topology of G. In order to characterize its accuracy with respect to the real network,

let us note that the nodes (processes) of the communication network G can be partitioned in *correct* and *faulty*, and its edges in *correct*, *one-faulty* and *two-faulty*, respectively interconnecting two correct processes, a correct processes and a faulty one, and two Byzantine processes. Likewise, the nodes and edges of a topology reconstruction G_i can either be *real* or *spurious*, respectively mapping or not nodes and edges in G.

9.3 Explorer [63]

Nesterenko and Tixeuil [63] analyzed the Byzantine fault-tolerant topology reconstruction problem and they provided a solution, Explorer, assuming a *Static_Async_Global* system, the known neighborhood assumption and unicast links.

Among the results they identified in [63], the following ones hold for any solution to the Byzantine fault-tolerant topology reconstruction problem.

Remark 21. No algorithm can decide whether a two-faulty edge exists [63].

Comment: the only processes aware of a link $\{p_i, p_j\}$ are only its endpoints in an unknown network. It follows that, in case they are both faulty, they can arbitrarily decide whether to declare it or not to the others. □

Remark 22. No algorithm can compute a reconstruction of only real nodes and edges while including both all correct and all one-faulty edges [63].

Comment: in unknown networks, the only process aware about a link are its endpoints. The two may claim the opposite about it, namely that it exists and it does not between them and there is no way to decide which one is lying. □

The Explorer protocol they defined is specified by the following two procedures:

- every process p_i broadcasts its neighborhood $\Gamma(i)$ (namely it broadcasts the identifier of processes it has a link with);
- every process p_i stores all neighborhoods $\Gamma(j)$ delivered with a reliable communication primitive in a dictionary data structure $cTop_i := \bigcup\langle j, \Gamma(j) \rangle$.

Every process p_i then infers a reconstruction G_i of G from *cTop* (there are alternative ways to do it). Nevertheless, the topology reconstructions computed with Explorer on different processes p_i can differ among each others, mainly due the assumption of unicast link allowing faulty processes to equivocate.

Nesterenko and Tixeuil briefly envisioned a potential extension to their protocol to the *unknown neighborhood* scenario. Nonetheless, such solution was not formally stated and verified in their work [63]. Furthermore, it has been later shown [53] that the reliable communication protocol adopted in Explorer fails in guaranteeing the liveness property.

9.4 Explorer2

We define a revised version of the Explorer protocol, naming it Explorer2. Such a protocol:

- it employs BFT as reliable communication primitive;
- it handles the weaker scenario in which processes are initially unaware about the peers they have a link with (*unknown neighborhood* assumption);
- it implicitly introduces a failure detector that enables all correct processes to eventually output the same topology reconstruction under certain assumptions.

The unknown neighborhood scenario is coped by a simple *neighborhood discovery procedure*, defined by the following actions:

- every process p_i multicasts a *HELLO* message at the beginning of the execution (basically a message with no payload);
- every process that receives a *HELLO* message adds the identifier of its sender to its neighborhood $\Gamma(i)$.

It has to be noticed that in the system model we assumed processes are unaware about the number of peers they may have a link with; they only know that each of them will receive at least $f + 1$ *HELLO* message for distinct neighbors (due to the node connectivity assumption). It follows that the set variable $\Gamma(i)$ collecting the identifier of the neighbors of a process p_i may grow over time.
Consequently, the Explorer protocol must be updated to handle such evolving $\Gamma(i)$. Specifically, in Explorer2:

- every process p_i broadcasts with BFT its neighborhood $\Gamma(i)$ every time that it grows;
- every process p_i that delivers $\Gamma(j)'$ from p_j with a reliable communication primitive substitutes the entry $\langle j, \Gamma(j) \rangle \in cTop_i$ with $\langle j, \Gamma(j)' \rangle$ if $\Gamma(j) \subset \Gamma(j)'$.

Notice that the neighborhood $\Gamma(i)$ broadcast by a correct processes only grows over time. Furthermore, every process p_i may broadcast many neighborhood $\Gamma(i)_1, \Gamma(i)_2, \ldots, \Gamma(i)_{n-1}$ such that for every pair of $\Gamma(i)_x, \Gamma(i)_y$ either $\Gamma(i)_x \subset \Gamma(i)_y$ or $\Gamma(i)_y \subset \Gamma(i)_x$. Such evolution of the $\Gamma(i)$ can be leveraged to prevent certain Byzantine misbehavior and to enable all correct processes to eventually output the same reconstruction.

We stated in Theorem 2 that DolevU (and thus BFT) solves the Byzantine reliable broadcast problem in *Static_Async_Global* if local broadcast links are assumed.
We additionally adopt in Explore2 the following procedure that acts as failure detector in case local broadcast links are assumed:

- every process p_i that delivers two neighborhoods $\Gamma(j)$ and $\Gamma(j)'$ from p_j, such that $\Gamma(j)' \not\subset (\Gamma(j) \in cTop_i)$ and $(\Gamma(j) \in cTop_i) \not\subset \Gamma(j)'$, do not consider node j for the reconstruction G_i.

The topology reconstruction $G_i(V_i, E_i)$ is computed in `Explorer2` by every process p_i as follows:

1. $\forall \langle u, \Gamma(u) \rangle \in cTop_i \Rightarrow \exists u \in V_i$;
2. $\forall v \in \Gamma(u), \langle u, \Gamma(u) \rangle \in cTop : X \leftarrow \bigcup u, |X| > f \Rightarrow \exists v \in V_i$.
3. $\forall \langle v, \Gamma(v) \rangle \in cTop_i,\ u \in \Gamma(v),\ u \in V_i \Rightarrow\ \exists (v, u) \in E_i$.

In other words: a node v is inserted in V_i if *1)* the related entry is in $cTop_i$ or *2)* v is declared as neighbor by at least $f + 1$ nodes; *3)* an edge $\{v, u\}$ is added in E_i if both nodes are in V_i and v declares u in its neighborhood.

Notice that a reconstruction $G_i(V_i, E_i)$ can be computed from p_i in alternative ways. For example, rule 3 can strengthened adding an edge $\{u, v\}$ in E_i only if the related link is declared both in $\Gamma(v)$ and $\Gamma(u)$, or rule 2 may ignore processes that do not declare their neighborhood (thus faulty). Nonetheless, the proposed procedure aims to maximize the number of real one-faulty edges detected, and thus the node connectivity of the subgraph of G_i composed only by real edges. This fact will be later leveraged defining an efficient reliable communication primitive.

Every topology reconstruction G_i computed with `Explorer2` guarantees the following properties.

Property 1. $\forall p_j \in P, p_j \in Correct \Rightarrow j \in V_i$; $\forall \{j, k\} \in E,\ p_j, p_k \in Correct \Rightarrow \exists \{j, k\} \in E_i$ (all correct processes and all correct edges are eventually contained in G_i).

Proof. All sets of assumptions considered enable process p_i to reconstruct a topology G_i that contains all correct nodes and edges. In the weakest scenario of *UL* and *UN* assumptions, every correct process p_j is eventually included in the $\Gamma(k)$ of all of its correct neighbors p_k. Subsequently, such $\Gamma(k)$ are broadcast through a reliable communication primitive, and thus it is delivered by p_i and all of its included edges will be added due to rule 2. □

Property 2. $p_j \notin P \Rightarrow j \notin V_i$ (G_i contains no non-real node).

Proof. `Explorer2` relies on a reliable communication primitive (`BFT`) to exchange the $\Gamma(i)$. A node j can be included in V_i either because the associated process p_j has broadcast, with a reliable communication primitive, its $\Gamma(j)$ or it has been included in least $f + 1$ $\Gamma(k)$ of distinct processes p_k. In the former case, the neighborhood $\Gamma(j)$ associated with a not existing process p_j can initially be diffused only by Byzantine faulty processes; it follows that, due to the safety property guaranteed by any reliable communication primitive, no correct process delivers the $\Gamma(j)$ of a not existing process. In the latter case, the identifier of a not existing process must be included in more then f $\Gamma(k)$. No correct process p_k adds the identifier of a not existing process inside its $\Gamma(k)$, due to the authenticated link. □

Property 3. Assuming the *unknown neighborhood* assumption (*UN*), G_i may never include some Byzantine processes.

Proof. In the unknown neighborhood assumption processes are unaware about the peers they have a link with. It follows that a Byzantine neighbor of a process may make itself not discoverable, simply remaining silent. Furthermore, it may adopt such a behavior only with part of its neighbors. It follows that some reconstruction G_i may never include some Byzantine processes. □

Property 4. Assuming the *known neighborhood* assumption (*KN*), eventually $j \in V_i \Leftrightarrow p_j \in P$ (Property 2 + all real nodes are eventually detected).

Proof. All correct nodes p_j broadcast their neighborhood $\Gamma(j)$ through a reliable communication primitive. It follows that eventually all correct processes p_i deliver $\Gamma(j)$ from p_j. The degree of every node i in G is greater than $2f$, and thus its identifier is included in the $\Gamma(j)$ of at least f+1 processes. □

Property 5. $\forall \{u, v\} \in E_i,\ \{u, v\} \notin E \Rightarrow u \in Byzantine$ (every spurious edge contains at least one Byzantine process).

Proof. A spurious edge can only be declared by a Byzantine processes. Indeed, due to the authenticated channels, correct processes only declare the neighbors they have a link with. Given that neighborhood information is exchanged through a reliable communication primitive, only a Byzantine endpoint of a spurious edge can declared it. □

Property 6. Assuming the *unknown neighborhood* (*UN*) assumption and *unicast links* (*UL*), $\forall \{u, v\} \in E,\ u \in Correct,\ v \in Byzantine \not\Rightarrow \exists \{u, v\} \in E_i$ and $\forall \{u, v\} \in E_i,\ u \in Correct,\ v \in Byzantine \not\Rightarrow \exists \{u, v\} \in E$.

Proof. Rule 2 requires a link $\{u, v\}$ to be declared in at least one among $\Gamma(u), \Gamma(v) \in cTop_i$ to be included in E_i. In the *UN-UL* scenario, Byzantine processes may not declare some of their links (i.e. they may not send the *HELLO* message to some of their neighbors) or they may declare some spurious links. □

Property 7. Assuming the *known neighborhood* assumption (*KN*), eventually $\forall \{u, v\} \in E,\ u \in Correct,\ v \in Byzantine \Rightarrow \exists \{u, v\} \in E_i$ (all one-faulty edges will eventually be present in any G_i).

Proof. The rule 2 requires a link $\{u, v\}$ to be included in one among $\Gamma(u)$ and $\Gamma(v)$ to be included in E_i and the correct endpoint in a one-faulty edge always declare the link. □

Property 8. Assuming *local broadcast links* (*LBL*), all one-faulty edges between a Byzantine process and all of its correct neighbors are eventually either all or none present in G_i.

Proof. In the weakest scenario of *unknown neighborhood*, the *local broadcast links* guarantee that if a neighbors of a process p_i receives a message, then it is received by all correct neighbors of p_i. It follows that if a correct process detects p_i (namely it

receives the *HELLO* message from p_i), the same occurs also for all the other correct neighbors of p_i. □

Property 9. Assuming local broadcast links (*LBL*), all correct processes eventually share the same topology reconstruction.

Proof. The BFT protocol solves the Byzantine reliable broadcast problem when *local broadcast* links are assumed (Theorem 2). Thus, all correct processes deliver the same set of messages, even if in a potentially different order. The failure detector introduced in Explorer2 prevent faulty processes p_i from broadcasting many $\Gamma(i)$ such that they are not one subset of another: every process either declare many $\Gamma(i)$ satisfying inclusion dependency or it is detected as faulty. All correct processes eventually set the biggest $\Gamma(i)$ they delivered from p_i or do not consider i in P_i. Thus, all reconstruction G_i eventually coincide. □

Property 10. No reconstruction G_i computed assuming *local broadcast links* (*LBL*) will ever contain more real edges than one obtained assuming the *know neighborhood assumption* (*KN*).

Proof. It follows from Properties 7 and 8: any reconstruction built assuming *KN* eventually includes all correct and one-faulty edges, whereas some one-faulty edges may miss assuming *UN* and *LBL*. □

Notice that in case of unicast links and unknown neighborhood, faulty processes may equivocate, and therefore send different neighborhood information to different processes. Such a misbehavior can be prevented solving the Byzantine reliable broadcast problem (Section 2). Local broadcast links directly enables BFT solving this problem (Theorem 2), otherwise a Byzantine reliable broadcast solution [22] can be deployed on top of BFT.

9.5 Explorer2 analysis

Explorer2 relies on a reliable communication primitive to address the topology reconstruction problem. Therefore, to evaluate its general complexity as distributed system protocol, we analyze in this section the amount of reliable communication instances that are executed solving the topology reconstruction problem.

All correct processes p_i in Explorer2 broadcast their neighborhood $\Gamma(i)$. In case of *know neighborhood* assumption (*KN*), every process broadcasts such an information only once. It follows that Explorer2 requires $O(n)$ reliable communication executions to enable all correct processes computing G_i. In case of *unknown neighborhood* (*UN*), every process has to perform neighborhood discovery and then to broadcast its $\Gamma(i)$. Nonetheless, no process p_i knows how many peers have to be detected before diffusing $\Gamma(i)$. Thus, they may broadcast their neighborhood many times, $n-f-2$ in the worst case scenario (the assumptions on the node connectivity of G and the failure model guarantee that every process has at least $f+1$ links with correct peers). It follows that Explorer2 with neighborhood discovery executes $O(n^2)$ reliable communication instances to address topology reconstruction.

9.6 Fault-free disjoint path solution

The Explorer2 protocol enables processes to partially reconstruct the topology of the communication network G, and we showed that different sets of assumptions provide more or less accurate reconstructions (Properties 1-10 of Explorer2).

We reported in Section 6.1.2 the DolevR protocol that leverages known disjoint routes, defined between all pairs of processes, to achieve reliable communication in *Static_Async_Global*. We showed that the resulting reliable communication primitive is optimal for all the evalution metrics we considered, namely message complexity, delivery complexity and communication latency (Theorem 3). We highlighted how f Byzantine faulty processes may compromise at most f paths of any disjoint path solution $\Pi_{i,j}$ in DolevR, and that the liveness of such a protocol is guaranteed by the existence of disjoint path solutions of size greater than $2f$ between all pairs of processes, where at least $f + 1$ of paths in every $\Pi_{i,j}$ cannot be compromised by faulty processes.

It follows that, if every pair of correct processes p_i, p_j is able to identify a disjoint path solution $\Pi_{i,j}$ interconnecting them where at least $f + 1$ paths are *faults-free* (i.e. they do not include any Byzantine faulty process), *real* (namely composes only by real edges) and *disjoint* (*FF_R_D*), then they are able to optimally achieve reliable communication with respect the evaluation metrics we considered.

We analyze several sets of assumptions enabling all pairs of correct processes p_i, p_j to compute a disjoint path solutions $\Pi_{i,j}$ in G_i containing at least $f + 1$ FF_R_D paths. We start by considering the stronger assumption with respect the node connectivity of G, subsequently we attempt to weaken such a constraint while considering more powerful links and knowledge assumptions.

Theorem 5 identifies a sufficient condition on the node connectivity of G that allows Explorer2 to compute disjoint path solutions between correct nodes that always contain at least $f + 1$ FF_R_D paths, while considering the weakest assumption for the links and the neighborhood knowledge.

Theorem 5. The assumption $\mathbf{k > 3f}$ **enables** every correct process p_i to compute a disjoint paths solution $\Pi_{i,j}$ toward any correct process p_j that contains at least $f + 1$ faults-free, real and disjoint paths.

Proof. Let us consider the weakest setting concerning the link and neighborhood knowledge, namely *UL* and *UN*. Let us assume processes employing Explorer2 and that all messages it generated have been already delivered by the processes. The unknown neighborhood and unicast links assumptions allow Byzantine faulty processes to decide which one-faulty and two-faulty edges to declare (Remark 21 and Property 6), thus the local connectivity between any two node i, j in a reconstructed topology may be reduced by at most f. It follows that any disjoint paths solution $\Pi_{i,j}$ computed between correct processes will have size greater than $2f$ paths (Property 1). Given that at most f paths of any $\Pi_{i,j}$ may contain faults the claim follows. □

Theorem 6 shows that the node connectivity assumption identified in Theorem 5 is minimal to achieve the specified target while considering the weakest assumptions for the links and the neighborhood knowledge.

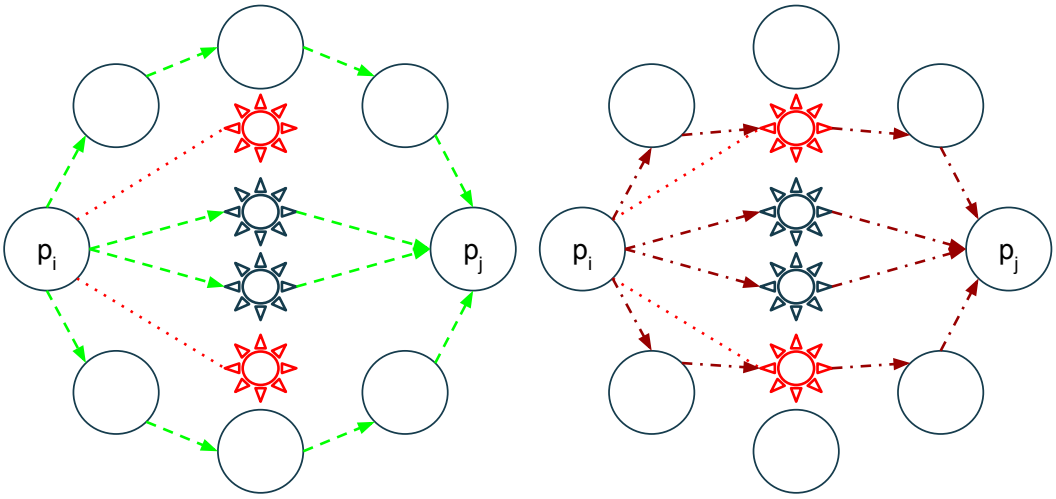


Figure 9.1: Explanatory example of Theorem 6.

Theorem 6. The assumption $\mathbf{k \leq 3f}$ **is not sufficient** to enable every correct process p_i to compute a disjoint paths solution $\Pi_{i,j}$ toward every correct process p_j containing at least $f + 1$ faults-free, real and disjoint paths with any protocol.

Proof. Let us consider the weakest setting concerning the link and neighborhood knowledge, namely *UL* and *UN*. The unknown neighborhood assumption and the unicast links allow faulty processes to decide which one-faulty and two-faulty edges are detectable by correct processes. It follows that the faulty processes may potentially be able to reduce the local connectivity between some pairs of correct processes p_i, p_j by f, i.e. the local connectivity $\kappa_{i,j}$ in G_i may be lower than $2f$ and thus a disjoint path solution of size at most $2f - 1$ will be identifiable between p_i and p_j, whatever algorithm is envisioned for the reconstruction. Then, up to f paths in $\Pi_{i,j}$ may include faulty processes and the claim follows.

A clarifying graphical example is provided in Figure 9.1. Let us consider a $W(4, 8)$ generalized wheel (a 6-connected graph, the "sun-shaped" nodes are the ones in K_4) as communication network. Let us suppose that two nodes in K_4 are Byzantine faulty (red nodes, thus $f = 2$ and $k = 6 \leq 3f = 6$), let us select two not adjacent processes p_i and p_j in C_8 respectively as source and target of a reliable communication instance and let us assume that the Byzantine processes hide their one-faulty edges with p_i (dashed edges). Process p_i, after reconstructing the entire $W(4, 8)$ topology (except for the dashed edges) with `Explorer2`, can compute a solution $\Pi_{i,j}$, but it could either be the green one or the violet one depicted in Figure, and only the former contains $f + 1$ FF_R_D paths. □

Theorem 7 provides an alternative set of sufficient assumptions, with respect to the one identified in Theorem 5, that enable all correct processes to identify $f + 1$ FF_R_D paths between all pair of correct processes through `Explorer2`.

Theorem 7. The set of assumptions *a)* $\mathbf{k > 2f + \lfloor f/2 \rfloor}$ and *b)* **local broadcast** links **enables** every correct process p_i to compute a disjoint paths solution $\Pi_{i,j}$ toward any correct process p_j containing at least $f + 1$ faults-free, real and disjoint paths.

Proof. Given Property 8 of `Explorer2`, let us suppose that $f_d \leq f$ Byzantine processes decide to be detected by their neighbors and they send the *HELLO* message, whereas $f - f_d$ ones do not. Let us assume that all messages exchanged by `Explorer2` have been already delivered and let us consider $\Pi_{i,j}$ as the disjoint path solution computed on G_i between a pair of correct processes p_i and p_j. The assumption on the node connectivity of G guarantees that at least $2f + \lfloor f/2 \rfloor + 1$ disjoint paths exist between p_i and p_j in the communication network. The undeclared Byzantine processes may reduce the local connectivity between p_i and p_j by $f - f_d$ in G_i. Let us temporarily assume, for the purpose of the proof, that the declared Byzantine processes behave as correct ones. It follows, from Property 1 and 8 of `Explorer2`, that the size of $\Pi_{i,j}$ would be at least equal to:

$$2f + \lfloor f/2 \rfloor + 1 \ \ -(\mathbf{f - f_d}) = f + \lfloor f/2 \rfloor + 1 + f_d$$

Specifically, all paths between p_i and p_j that contain only correct or declared Byzantine processes existing in G would be present in G_i.

Let us now consider the declared Byzantine processes not reporting the edges existing between them (i.e. the two-faulty edges, Remark 21). It follows that the paths in G containing two-faulty edges are not present in G_i. Therefore, pairs of Byzantine processes may cause a reduction to the maximum size of $\Pi_{i,j}$: every couple may decrease by one the number of available disjoint paths in G_i between p_i and p_j. It follows that the size of $\Pi_{i,j}$ would be at most reduced to:

$$f + \lfloor f/2 \rfloor + 1 + f_d - \lfloor \mathbf{f_d/2} \rfloor$$

namely, f_d declared Byzantine faulty processes may reduce the local connectivity between p_i and p_j in G_i by at most $\lfloor f_d/2 \rfloor$. The f_d declared Byzantine processes may also be selected among the paths in $\Pi_{i,j}$. Specifically, in the worst case scenario f_d paths in $\Pi_{i,j}$ may contain Byzantine processes. It follows that at most f_d paths would not be fault-free, and thus the remaining fault-free ones in $\Pi_{i,j}$ would be:

$$f + \lfloor f/2 \rfloor + 1 + f_d - \lfloor f_d/2 \rfloor - \mathbf{f_d} = f + 1 + \lfloor f/2 \rfloor - \lfloor f_d/2 \rfloor$$

Therefore, at least $f + 1$ paths in $\Pi_{i,j}$ are faults-free, real and disjoint. □

Corollary 4. The set of assumptions *a)* $\mathbf{k > 2f + \lfloor f/2 \rfloor}$ and *b)* **known neighborhood enables** every correct process p_i to compute a disjoint paths solution $\Pi_{i,j}$ toward any correct process p_j containing at least $f + 1$ faults-free, real and disjoint paths.

Proof. It follows from Theorem 7 considering Property 10 of `Explorer2`. □

We conjecture that the set of assumptions identified in Theorem 7 is minimal to identify a disjoint path solution containing at least $f + 1$ faults-free, real and disjoint paths, namely that none of them can be weakened targeting the specified target.

Conjecture 1. The set of assumptions *a)* $\mathbf{k \leq 2f + \lfloor f/2 \rfloor}$ and *b)* **known neighborhood is not sufficient** to enable every correct process p_i to compute a disjoint paths solution $\Pi_{i,j}$ toward every correct process p_j containing at least $f + 1$ faults-free, real and disjoint paths with any protocol.

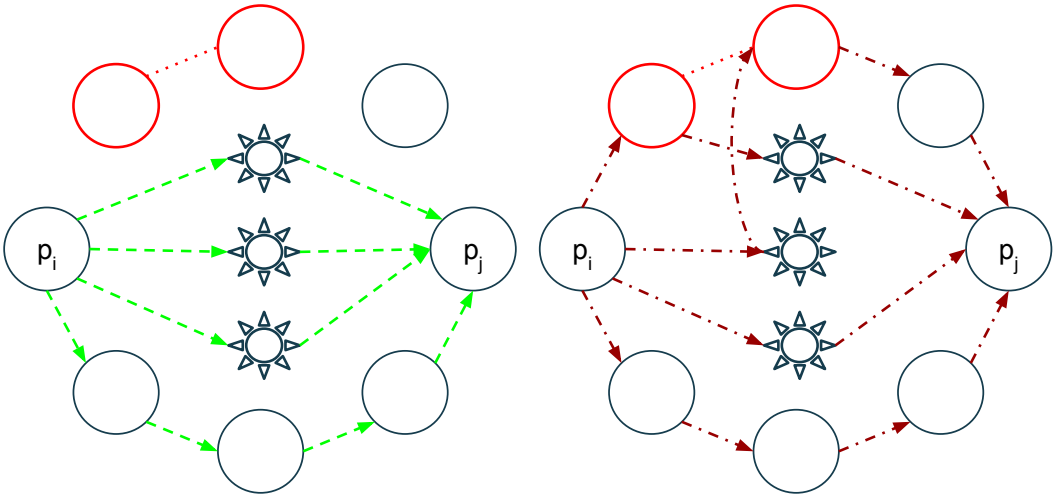


Figure 9.2: Explanatory example of Conjecture 1.

Argumentation. The claim could be motivated using the same argumentation provided for Theorem 7: even considering the *known neighborhood* assumption, the Byzantine process may not declare the two-faulty edges between them (Remark 21). It follows that the node connectivity of a topology reconstruct G_i could be reduced by $\lfloor f/2 \rfloor$, and thus the maximum size of a disjoint path solution $\Pi_{i,j}$ could be reduced to $2f$. Among the path selected for $\Pi_{i,j}$, up to f may contain Byzantine faulty processes. Nevertheless, it is not clear whether a disjoint path solution containing faulty processes is always returnable by a deterministic algorithm computing disjoint paths in a network. Taking the example in Figure 9.2, the solution on the right is never output employing [29, 36] as algorithm, because it always return a solution minimizing the total number of edges.

Dolev et al. [32] additionally considered in their work the setting in which the Byzantine faulty processes are spaced between each other. Specifically, they assumed that no pair of Byzantine processes is connected by a link. Such an assumptions ensures that no two-faulty edge exists in G. In turns, it lowers the node-connectivity requirement enabling correct processes to identify a disjoint path solution with at least $f + 1$ faults-free and real paths.

Corollary 5. The set of assumptions *a)* $\mathbf{k > 2f}$, *b)* **local broadcast** links and *c)* **no two-faulty edge** in G **enables** every correct process p_i to eventually compute a disjoint paths solution $\Pi_{i,j}$ toward any correct process p_j containing at least $f + 1$ faults-free, real and disjoint paths.

Proof. Following the same argumentation provided for Theorem 7, the Byzantine faulty processes may decide either to be detected by all of their neighbors or by none. Let us assume that f_d ($f_d \leq f$) faulty processes decide to declare them self (sending the *HELLO* message). The undeclared Byzantine processes may reduce the local connectivity between two processes p_i and p_j by $f - f_d$ in G_i. All edges with a declared Byzantine endpoints are eventually present in G_i (Property 8). Given

that no two-faulty edge is present in G, it follows that $\Pi_{i,j}$ has size at least equals to:

$$2f + 1 - (\mathbf{f} - \mathbf{f_d}) = f + 1 + fd$$

in which at most f_d may contain Byzantine processes. □

Corollary 6. The set of assumptions *a)* $\mathbf{k} > \mathbf{2f}$, *b)* **known neighborhood** and *c)* **no two-faulty edge** in G **enables** every correct process p_i to eventually compute a disjoint paths solution $\Pi_{i,j}$ toward any correct process p_j containing at least $f + 1$ faults-free, real and disjoint paths.

Proof. It follows from Corollary 5 considering Property 10 of `Explorer2`. □

Lastly, we conjecture that also the set of assumption identified in Corollary 5 is minimal.

Conjecture 2. The set of assumptions $\mathbf{k} > \mathbf{2f}$, *b)* **unicast** links, *c)* **unknown neighborhood**, and *d)* **no two-faulty edge** in G **is not sufficient** to enable every correct process p_i to eventually compute a disjoint paths solution $\Pi_{i,j}$ toward every correct process p_j containing at least $f + 1$ faults-free, real and disjoint paths with any protocol.

Argumentation. Same of Conjecture 1, considering that no two-faulty edge is present in G.

9.7 CombinedRC, an optimal reliable communication protocol

We combine `Explorer2`, `BFT` and `DolevR` protocols to design an efficient reliable communication primitive. We call such a protocol `CombinedRC`.

The `Explorer2` protocol is used to partially reconstruct the network topology, and then to enable processes to compute disjoint paths solutions through which relay contents. The `BFT` protocol is adopted as reliable communication subprimitive by `Explorer2` and `CombinedRC` during the initialization. Lastly, the `DolevR` protocol is employed as actual reliable communication primitive in `CombinedRC`, leveraging the routes computed and communicated using `Explorer2` and `BFT`.

We showed in Section 9.6 that `Explorer2`, under certain conditions, enables every correct process p_i to identify a disjoint paths solution $\Pi_{i,j}$ interconnecting it with any other correct process p_j, such that at least $f + 1$ paths in the solution are faults-free, real and disjoint. It follows that once the solution $\Pi_{i,j}$ is known to both of its endpoints p_i and p_j they can efficiently communicate.

We claimed in Property 9 of `Explorer2` that all correct processes eventually obtain the same topology reconstruction in case local broadcast links are employed. Thus, under such an assumption, processes p_i and p_j eventually compute the same solution $\Pi_{i,j}$. Under the weaker setting of unicast links, the reconstructed topology may differ on distinct processes, thus a source process p_i has additionally to reliably communicate the computed solution $\Pi_{i,j}$ to a target process p_j.

Summarizing, in CombinedRC any source process p_i routes contents through the computed $\Pi_{i,j}$ and any target process p_j waits for messages over $f + 1$ paths among the ones in $\Pi_{i,j}$. The pseudo-code of CombinedRC is presented in Algorithm 6. Every process relays its contents over the computed routes if available, otherwise they are queued for a subsequent transmission (lines 1-5).
Every process p_i attempts to compute a solution $\Pi_{i,j}$ toward every other process p_j of the system. In case of local broadcast links, the reconstructed topology G_i is eventually the same on every process. Therefore, a source process has to relay its contents over the computed disjoint routes every time they change (a finite number of times). In case of unicast links, once that the local connectivity toward a target p_j reaches a value greater than $2f$, the source process p_i communicates the computed solution $\Pi_{i,j}$ via BFT (lines 6-21).
Every process relays contents or computed disjoint solution following the path attached to messages (lines 22-33).
Every process that delivers a disjoint paths solution with BFT adopts it to verify contents (lines 34-35) using DolevR (lines 36-37).

9.7.1 CombinedRC correctness analysis

We prove in this section the correctness of CombinedRC as a solution to the reliable communication protocol.

Theorem 8. CombinedRC provides safety of reliable communication.

Proof. A process that receives a *CNT* message checks the identity of its sender through authenticated link. It follows that any spurious *CNT* message will contain in the *visited* field at least one identifier of the Byzantine processes. It follows that the faulty processes cannot diffuse *CNT* messages over more than f disjoint paths. □

Theorem 9. CombinedRC provides liveness of reliable communication in all settings where Explorer2 succeeds in identifying a disjoint path solution between any two correct processes p_i, p_j that contains at least $f+1$ FF_R_D paths (Theorems 5, 7, Corollaries 4, 5, 6).

Proof. Let us assume that all messages exchanged by Explorer2 have been delivered and that a process p_i aims to reliably communicate with a correct process p_j. In case local broadcast links are assumed, processes p_i and p_j eventually share the same topology reconstruction, thus also the disjoint path solution $\Pi_{i,j}$ will eventually be the same both on p_i and p_j. Process p_i relays the contents through $\Pi_{i,j}$ every time such a solution changes. The assumption of $f + 1$ FF_R_D paths in $\Pi_{i,j}$ guarantees reliable communication. In case of unicast links, the solution $\Pi_{i,j}$ is diffused via BFT and contents are then routed over $\Pi_{i,j}$. The assumption of $f + 1$ FF_R_D paths in $\Pi_{i,j}$ guarantees reliable communication. □

9.7.2 CombinedRC performance analysis

CombinedRC provides reliable communication with optimal message complexity, delivery complexity and communication latency (Theorem 3). Specifically, it routes

Algorithm 6 CombinedRC

```
1: upon RC_send(m, target) do
2:     Sent ← Sent ∪ ⟨m, target⟩
3:     if Π_{i,target} ≠ ∅ then
4:         for path ∈ Π_{i,target} do
5:             send(⟨CNT, i, target, m, path⟩, path[1])

6: upon G_i changes do
7:     for j ∈ G_i such that i ≠ j do
8:         if LB then
9:             if local_conn(G_i, i, j) > f + ⌊f/2⌋ and disj_paths(G_i, i, j) ≠ |Π_{i,j}| then
10:                Π_{i,j} ← disj_paths(G_i, i, j)
11:                for path ∈ Π_{i,j} do
12:                    for ⟨m, target⟩ ∈ Sent such that j = target do
13:                        send(⟨CNT, i, j, m, path⟩, path[1])
14:                Π_{j,i} ← disj_paths(G_i, j, i)
15:        else if UC then
16:            if Π_{i,j} = ∅ and local_conn(G_i, i, j) > 2f then
17:                Π_{i,j} ← disj_paths(G_i, i, j)
18:                for path ∈ Π_{i,j} do
19:                    send(⟨ROU, i, j, Π_{i,j}, path⟩, path[1])
20:                    for ⟨m, target⟩ ∈ Sent such that j = target do
21:                        send(⟨CNT, i, j, m, path⟩, path[1])

22: upon receive(⟨CNT, s, t, m, path⟩, j) do
23:     if predecessor(path, i) = j then
24:         if t = i then
25:             Paths_cnt[⟨m, s⟩] ← Paths_cnt[⟨m, s⟩] ∪ {path}
26:         else
27:             send(⟨CNT, s, t, m, path⟩, successor(path, i))

28: upon receive(⟨ROU, s, t, Π, path⟩, j) do
29:     if predecessor(path, i) = j then
30:         if t = i then
31:             Paths_rou[⟨s, Π⟩] ← Paths_uRts[⟨s, Π⟩] ∪ {path}
32:         else
33:             send(⟨ROU, s, t, Π, path⟩, successor(path, i))

34: upon DolevU_deliver(Paths_rou[⟨s, Π⟩], s) do
35:     Π_{s,i} ← Π

36: upon DolevR_deliver(Paths_cnt[⟨m, s⟩], s) do
37:     RC_deliver(m, s)
```

contents over computed disjoint routes as DolevR, thus $O(n)$ messages per content are exchanged, an $O(f)$ procedure is executed to verify every content, and the communication latency is $O(n)$.

CombinedRC requires an initialization phase where the network topology is partially reconstructed and the solutions containing $f + 1$ FF_R_D paths are computed between every pair of correct processes. We showed in Section 9.5 that Explorer2 requires at most $O(n^2)$ reliable communication instances to partially reconstruct the network topology. For every pair of processes p_i, p_j, assuming local broadcast links, the same solution $\Pi_{i,j}$ is eventually computed by both p_i and p_j without additional message exchanges, because the topology reconstruction will eventually be the same on every process and the disjoint paths solutions can computed through a deterministic algorithm. On the other hand, employing unicast links, every couple of processes p_i, p_j must agree on a solution $\Pi_{i,j}$. Thus, an addition content exchange (with payload $\Pi_{i,j}$) using a reliable communication primitive has to be performed for each pair of correct processes. It follows that the initialization phase of CombinedRC requires the execution of $O(n^2)$ reliable communication instances. Notice that, in case of known neighborhood and local broadcast links, the cost of the initialization phase reduces to $O(n)$ instances, indeed each process diffuses its neighborhood only once and all correct processes eventually share the same reconstruction.

9.8 Conclusion

In this chapter, we combined a topology reconstruction protocol with two reliable communication solutions, aiming to overcome the potential limitations imposed by the available solutions to the reliable communication problem in *Static_Async_Global* (Chapter 7 and Section 8.8). The designed protocol, Explorer2, allows to optimally solve the reliable communication problem with respect all the evaluation metrics we are considering. It requires stronger assumptions (Theorems 5, 7, Corollaries 4, 5, 6) with respect to the necessary ones to solve the problem (Remark 7), and an initialization phase of non optimal reliable communication instances to setting it up. Therefore, after executing a limited number of non optimal reliable communication instances (order of $O(n^2)$), unlimited contents can be exchanged efficiently, whereas the cost of every reliable communication instance would have been remained always not optimal otherwise.

Chapter 10

Cryptographic Reliable Communication Protocols

In this chapter, we present an alternative to `CombinedRC` putting in place an efficient reliable communication protocol in *Static_Async_Global*. Applying a known idea in security and networking, we leverage a reliable communication protocol to establish cryptographic keys that will be employed to reliably exchange contents [1].

We do not focus on the security aspects that should be taken into account designing a robust cryptographic solution, such as the chiphering scheme, keys renewals, etc [72]. We detail instead a high level cryptographic protocol prototype solving the reliable communication problem in *Static_Async_Global*. The solution we propose does not require stronger assumptions on the topology of the communication network, nor on its links, nor on the process a priori knowledge with respect the minimal one (Remark 7) to be employed. It considers instead a weaker adversary: all other solutions reviewed or presented so far in this thesis assume an adversary with unbounded capabilities (on the computational point of view) able to control part of the processes in the system. In this chapter instead, the adversary has a bounded computational power and it is unable to compromise the adopted cryptographic primitives.

We start presenting a simple cryptographic reliable communication protocol, `AuthRC` [60], solving the problem in *Static_Async_Global* where all processes are able to digitally sign their message and verify signatures [72].
Subsequently, we define `CryptoRC` that, following the same approach of `CombinedRC`, initially leverages a non efficient reliable communication solution to set up an efficient primitive (cryptographic in this case).
Finally, the topology reconstruction protocol `Explorer2` (Chapter 9, Section 9) is additionally plugged in order to define an optimal primitive, `CombinedCryptoRC`.

[1] Recalls on cryptography and digital signature schemes can be found in [72].

10.1 AuthRC, an authenticated reliable communication protocol [60]

We analyze in this section an authenticated solution to the reliable communication problem proposed in [60]. Authenticated protocols leverage digital signature primitives instead of authenticated links to achieve message authenticity and integrity (and thus safety of reliable communication), namely they assume all processes in the systems able to digitally sign messages and to verify their signatures [2].

10.1.1 System model

We consider in this section an asynchronous distributed system organized in a static communication network G of *perfect unicast links* [24] (i.e. reliable not-authenticated). We assume the *globally bounded* Byzantine failure model and a communication network G having node connectivity greater than the number of maximum assumed faults, namely $\kappa > f$. All processes have access to a digital signature primitive enabling them to digitally sign contents and to verify signatures. We assume that no process, included the Byzantine faulty ones, can compromise the digital signature primitive, namely to generate any spurious content with a valid digital signature.

Notice that such a system model is minimal solving the reliable communication model, namely that none of the assumptions considered can be weakened without making the problem unsolvable: perfect links are required to guarantee that messages are not lost and the node connectivity is minimal to prevent faulty processes from being a cut in the communication network.

10.1.2 AuthRC

`AuthRC` is a simple authenticated reliable communication protocol defined by the followin procedures:

- the source process p_s computes the digital signature $sign_{\langle s,c \rangle}$ of the content c and it multicasts the message $\langle s, c, sign_{\langle s,c \rangle} \rangle$ to all of its neighbors;
- a process p_i relays to all of its neighbors every message $\langle s, c, sign_{\langle s,c \rangle} \rangle$ the first time it is received;
- if a process p_i receives message $\langle s, c, sign_{\langle s,c \rangle} \rangle$ such that $sign_{\langle s,c \rangle}$ is valid signature for the content c with author process p_s then it delivers c from p_s.

10.1.3 AuthRC correctness analysis

Remark 23. `AuthRC` ensures safety of reliable communication [60].

[2] In its original definition [60] `AuthRC` employs an alternative and equivalent approach verifying integrity and delivery of the contents based on asymmetric cryptography as well.

Proof. It follows from the assumption of an adversary with bounded capacity, specifically from its inability in breaking the digital signature primitive: no process can generate a valid digital signature for a content c excepts for its author. □

Remark 24. AuthRC ensures liveness of reliable communication if $k > f$ [60].

Proof. Assuming f faulty processes acting as cut in the communication network, the graph remains connected. It follows that there always exist a path between every two correct processes and the propagation procedure adopted in AuthRC guarantees the content forwarding. □

10.1.4 AuthRC performance analysis

Delivery Completity. The delivery complexity of the protocol depends on the adopted digital signature scheme, it is equivalent to the complexity of the procedure verifying a digital signature of a content indeed, and it is independent from the size of the system.

Message Complexity. Every process relays a received content to all of its neighbors only at its first reception. It follows that the message complexity of the proposed solution is $O(E)$, and thus $O(n^2)$ in case of a complete network topology.

Communication Latency. The length of a fault-free path between two processes can be up to $n - f - 1$ hops long (given the bound on the wide-diameter, Remark 3). It follows that the communication latency is $O(n)$.

10.2 CryptoRC, a cryptographic reliable communication protocol

We present in this section a cryptographic protocol solving the reliable communication problem in *Static_Async_Global*. The design pattern is similar to the one adopted for CombinedRC: it leverages a bounded number of non optimal reliable communication instances to set up an efficient primitive.

10.2.1 System model

We consider in this section a *Static_Async_Global* system with a communication network G having node connectivity greater than twice the number of maximum assumed faults, namely $\kappa > 2f$. We assume an adversary with bounded capacity, unable to break the digital signature primitive, namely that no Byzantine process can generate any spurious content with a valid digital signature.

10.2.2 CryptoRC

CryptoRC is a prototype of an authenticated reliable communication protocol. It is a two phases algorithm that initially execute some non optimal reliable communication instances to then set-up an efficient primitive. In detail, every process p_i generates its own pair of private and public keys $\langle key_i^{pub}, key_i^{pri} \rangle$ supporting asymmetric encryption on which digital signatures are based upon. Subsequently, every

process p_i distributes with a reliable communication protocol (e.g. BFT) its public key key_i^{pub}, that is associated to its source p_i by every process p_j that delivers it. Lastly, the AuthRC protocol is employed as actual reliable communication protocol, using the generated and distributed cryptographic keys.

10.2.3 CryptoRC correctness analysis

Theorem 10. CryptoRC ensures safety of reliable communication.

Proof. Assuming that no two processes can generate the same pair of $\langle key_i^{pub}, key_i^{pri} \rangle$ and that the adversary is unable to break the digital signature scheme, the claim follows from Remark 23. □

Theorem 11. CryptoRC ensures liveness of reliable communication if $k > f$.

Proof. The public keys exchanged during the first phase of the protocol are diffused through a reliable communication solution (BFT correctly works under the assumption considered), it follows all the keys diffused by correct processes are eventually delivered by all the correct processes in the system. The claim follows from Remark 24, given that the link and node connectivity assumption here considered are stronger. □

10.2.4 CryptoRC performance analysis

CryptoRC requires an initialization phase where all processes exchange their public key through a reliable communication protocol (like BFT). It follows that $O(n)$ non optimal reliable communication instances need to be executed to setup the more efficient authenticated reliable communication primitive.

Delivery Completity. The delivery complexity of the protocol depends on the adopted digital signature scheme, it is equivalent to the complexity of the procedure verifying digital signature indeed, and it is independent from the size of the system.

Message Complexity. After the initialization phase, every process relays a received content to all of its neighbors only at its first reception. It follows that the message complexity of the proposed solution is $O(E)$, and thus $O(n^2)$ in case of complete network topology.

Communication Latency. The length of a fault-free path between two process can be up to $n - f - 1$ hops long (given the bound on the wide-diameter, Remark 3). It follows that the communication latency is $O(n)$.

10.3 Optimizing CryptoRC: CryptoCombinedRC

We showed in Theorem 3 that DolevR optimally solves the one-to-one reliable communication problem in *Static_Async_Global*. We reported AuthRC, that achieves reliable communication with $O(n^2)$ message complexity and $O(n)$ communication latency. It follows that its message complexity is non optimal while fixing a specific target for a reliable communication instance.

In this section, we combine CryptoRC and Explorer2 in order to design an optimal cryptographic any-to-any reliable communication protocol.

10.3.1 System model

We consider in this section a *Static_Async_Global* system with a communication network G having node connectivity greater than twice the number of maximum assumed faults, namely $\kappa > 2f$. We assume an adversary with bounded capacity, unable to break the digital signature scheme, namely that no Byzantine process can generate any spurious content with a valid digital signature. We alternatively consider the known neighborhood and unknown neighborhood assumption.

10.3.2 CryptoCombinedRC

CombinedCryptoRC is obtained from the combination of CryptoRC and Explorer2. It is a three phase protocol. During the first one, every process generates its pair of keys $\langle key_i^{pub}, key_i^{pri} \rangle$ and diffuses key_i^{pub} through BFT, setting up the CryptoRC primitive. At the second phase, processes employs CryptoRC as reliable communication primitive of Explorer2 to partially reconstruct the network topology. In the last phase, once that the local node connectivity between a source and a target processes in the reconstructed topology is greater than $f + 1$, a disjoint path solution is computed, and content digitally signed and then routed over the computed paths.

The Explorer2 protocol enables all correct processes to partially reconstruct the topology of G existing between only correct processes (Property 1 of Explorer2). It follows that the local connectivity between every pair of correct processes in G_i will be greater than f and thus all the correct peers p_i, p_j can compute a disjoint path solution $\Pi_{i,j}$ of size at least $f + 1$. A reconstruction G_i may still contain spurious edges, but they all include at least one Byzantine endpoint (Property 5). Consequently, all correct processes can compute disjoint path solutions toward all correct peers such that at least one of its paths is fault-free and thus they can achieve reliable communication between them.

10.3.3 CryptoCombinedRC correctness analysis

Theorem 12. CryptoCombinedRC ensures safety of reliable communication.

Proof. Assuming that no two processes can generate the same pair of $\langle key_i^{pub}, key_i^{pri} \rangle$ and that the adversary is unable to break the digital signature scheme, the claim follows from Remark 23. □

Theorem 13. CryptoCombinedRC ensures liveness of reliable communication.

Proof. The Explorer2 protocol enables all correct processes to completely reconstruct the topology of G existing between only correct processes (Property 1 of Explorer2). It follows that any correct process is eventually able to compute $f + 1$ disjoint paths in G_i toward any other. Given that at most f faulty processes are present in the system, at least one among the computed routes is always traversable by a content and the claim follows. □

10.3.4 CryptoCombinedRC performance analysis

CombinedCryptoRC solves the one-to-one reliable communication problem in *Static_Global_Async* with asymptotically optimal message complexity and communication latency (Theorem 3).

CryptoCombinedRC requires an initialization phase where all processes exchange they neighborhood and public keys through a reliable communication protocol (BFT or CryptoRC). Depending on whether the known or unknown neighborhood assumption is considered, $O(n)$ or $O(n^2)$ CryptoRC (AuthRC) instances are required to enable correct processes to partially reconstruct the network topology. Moreover, every process has to diffuse its generated public key to setup an efficient reliable communication protocol. Therefore, additional $O(n)$ non optimal reliable communication instances with BFT are required to initialize CryptoCombinedRC.

Message Complexity. The message complexity of CryptoCombinedRC is optimal solving the one-to-one reliable communication protocol. Every correct processes eventually computes $f + 1$ disjoint routes over the reconstructed topology G_i where to relay its contents, generating one message per edge.

Delivery Complexity. The delivery complexity of the protocol depends on the adopted digital signature scheme, it is equivalent to the complexity of the procedure verifying digital signature indeed, and it is independent from the size of the system.

Communication Latency. The length of a fault-free path between two process can be up to $n - f - 1$ hops long (given the bound on the wide-diameter, Remark 3). It follows that the communication latency is $O(n)$.

10.3.5 Comparison between CombinedRC, CryptoRC and CombinedCryptoRC

CombinedRC, CryptoRC and CombinedCryptoRC all aim to setup an efficient reliable communication protocol in *Static_Async_Global* systems. All require an initialization phase where a certain number of messages has to be exchanged through a non efficient reliable communication solution.

On one hand, the number of reliable communication instances required to initialize CryptoRC is asymptotically either lower than or equal to the one of CombinedRC and CryptoCombinedRC . Moreover, CryptoRC and CryptoCombinedRC requires no further system assumption beyond the necessary ones identified by Dolev (Remark 7).

On the other hand CryptoRC and CryptoCombinedRC assumes a weaker adversary than CombinedRC, namely with bounded computation capabilities and thus unable to compromised the cryptographic primitive CryptoRC and CryptoCombinedRC are based upon.

10.4 Conclusion

We presented in this section how to set up an efficient (or even optimal) authenticated reliable communication protocol in *Static_Async_Global* system with unknown network topology. Differently from `CombinedRC`, the proposed protocols `CryptoRC` and `CombinedCryptoRC` requires not further assumption with respect to the necessary ones to be properly employed but assume a weaker adversary with bounded capacity unable to break the cryptographic pritimive the defined protocol are based upon.

Part II

Static Faults, Dynamic Network

Chapter 11

Reliable Communication in Dynamic Networks: Motivations and Challenges

We move forward in the analysis of the Byzantine-tolerant reliable communication problem considering a dynamic network, namely one that changes over time.

Currently, there is only the solution of Maurer et al., `MTD`, available in the literature addressing the problem considering a dynamic network. Nevertheless, several limitations are preventing from its practical employment: *i)* its message complexity and delivery complexity does not allow the protocol to scale; *ii)* the correctness condition defined for its use applies to single instances occurring between a pair of processes, assuming the communication starting at a specific time, and its verification is an NP-Hard problem to solve. This implies that the knowledge of the complete evolution of the network, namely $\mathcal{G}$, would not help processes to achieve reliable communication more efficiently, like it happens with `DolevR`: specific journeys where to route the contents would be NP-Hard to calculate and would have to be recalculated for each new content to exchange.

The `BFT` protocol reduces the message complexity solving the reliable communication problem in static networks, and actually, such a solution can directly be adapted and employed rather than `MTD`. However, as pointed out in Section 8.8, its efficiency may be lost when a specific scenario arises: as long as no new process delivers the content, the number of messages may continue to grow.

On static distributed systems, the `CPA` protocol showed as an efficient solution to the reliable communication problem against locally bounded Byzantine failures and also tolerating globally bounded failure considering stronger network assumptions. Nevertheless, its employment has not been analyzed yet in dynamic networks.

Another aspect that requires further attention on dynamic networks is the assumption of reliable links. Although they are reasonable to assume on static distributed systems, it is less so on dynamic networks whose nature is the continuous evolution.

Reliable links in dynamic networks

The protocols previously analyzed in static distributed systems consider perfect reliable links, where each message sent by any correct sender is eventually received by its receiver if correct. Nevertheless, the implementation of a reliable link in a distributed system may send several "lower level" messages to provide reliable delivery. In the system model we assumed for dynamic distributed systems, processes can instantaneously detect the presence of a communication link but they have no knowledge about its stability, namely whether it will enable a process to reliably transmit messages. It follows that several transmissions may occur before that a message is delivered by a link. Nonetheless, the focus of this thesis is fault-tolerant reliable communication protocols at a high level, without a detailed focus on the actual implementation of the assumptions considered. Therefore, as we did for static protocols analysis, we will count only the messages that are successfully exchanged by the processes evaluating the message complexity of the solutions.

TVG is a powerful model that properly characterizing the evolution of several real dynamic networks. Among the aspects it captures, there is the punctual "latency" of its edges. Specifically, TVG allows to characterize the temporal availability of a connection between its nodes with the presence function ρ, but which can either allow or not an interaction at a certain time, ruled by the latency function ζ. For this reason, given a TVG $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, we define the predicate $\mathsf{RL}(v, u, \hat{t})$ asserting whether the reliable delivery property is achievable by link between two nodes exchanging a message at certain time $\hat{t}$.

Definition 37 [**Reliable Link Predicate** *RL*$(v, u, \hat{t})$]. Let $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ be a TVG. The predicate *Reliable Link* $\mathsf{RL}(v, u, \hat{t})$ for nodes $v, u \in V$ and $\hat{t} \in \mathcal{T}$ is defined as follows:

$$\mathsf{RL}(v, u, \hat{t}) = \begin{cases} \mathsf{true} & \textit{if } \forall t \in [\hat{t}, \hat{t} + \zeta(\langle v, u \rangle, \hat{t})],\ \rho(\langle v, u \rangle, t) = 1. \\ \mathsf{false} & \textit{otherwise}. \end{cases}$$

Given that processes are only aware of the availability of a link but they do not know a priori whether it will allow or not the reliable delivery, to maximize the possibility of its occurrence they need to relay their messages infinitely often (see MTD, Section 6.4), hoping for eventual reliable delivery of the messages exchanged. It follows that we may not evaluate the actual message complexity of a reliable communication protocol under such assumptions. To overcome this issue, we additionally assume in our subsequent analysis processes able to detect whether a message is delivered by a link, and we suppose an additional logical layer acting between the distributed protocol and the link implementation, limiting the number of messages relayed by a process, the *no-duplication layer*.

The *no-duplication layer* guarantees that, if a process p_i sends a message m to a process p_j many times, then such a message is relayed to p_j at most once. Specifically, if a process p_i sends a message m to process p_j at $\hat{t}$, the predicate *RL*$(i, j, \hat{t})$ is verified and m was not delivered in by the link $\{p_i, p_j\}$ at any $t < \hat{t}$, then it is received by p_j.

Notice that those two additional assumptions are not required to solve the reliable communication problem in dynamic networks, but they enable to evaluate the message complexity of solutions. Moreover, in the case predicate $RL(i,j,\hat{t})$ is verified for every edge $\{i,j\}$ present at $\hat{t}$ we get back the perfect reliable link assumption.

Contributions

In Chapter 12, we identify alternative dynamic network conditions enabling any-to-any reliable communication at any time t assuming the globally bounded Byzantine failure model.

In Chapter 13, we extend the `CPA` protocol to dynamic distributed systems and we characterize the dynamic network conditions enabling one-to-all and any-to-any reliable communication in the locally bounded Byzantine failure model.

Chapter 12

Reliable Communication with Globally Bounded Byzantine Failures

We seek in this chapter for alternative dynamic network conditions enabling reliable communication in the globally bounded Byzantine failure model.

The characterization currently available in the literature is strict (Remark 18), namely it identifies all the instances where the problem is solvable from a process p_s to a peer p_t at time t, but it is not general, in the sense it does not guarantee reliable communication infinitely often between every pair of processes, as the condition identified for static distributed systems (Remark 7). Furthermore, such a condition is NP-Hard to verify (Section 6.4.1).

We identify the class of TVG where the any-to-any reliable communication problem is solvable at every time t.

12.1 System model

We alternatively consider a *Dynamic_FullSync_Global* or a *Dynamic_CompSync_Global* distributed system. We assume processes able to instantaneously detect link presence.

12.2 Any-to-any reliable communication solvability

Given two processes p_u, p_v, the strict condition enabling reliable communication from p_u to p_v at t in the globally bounded failure model requires the existence of a set of journeys $u \rightsquigarrow v$ in $\mathcal{G}_{[t,\infty)}$ such that their dynamic cut k is greater than $2f$ (Remark 18). We refer to such collection of journeys with $u \rightsquigarrow_k v$.

The class $\mathcal{TC}$ of TVG (Definition 32) as been identified has the minimal one where it is possible to solve any-to-any reliable communication at least once in *Dynamic_CompSync* assuming all correct processes [25]. We define the following sub-class $\mathcal{TC}_k$ of $\mathcal{TC}$.

Definition 38 [**Class $\mathcal{TC}_k$ (Temporal k Connectivity)**]. $\forall u, v \in V;\ u \rightsquigarrow_k v$ (every node can reach all the others through journeys with dynamic minimum cut at least k at least once).

In $\mathcal{TC}_k$ TVGs every node can accomplish any-to-any reliable communication in *Dynamic_ComplSync_Global* at least once if $f < k/2$.

Class $\mathcal{TC}^{\mathcal{R}}$ (Definition 33) is a sub-class of $\mathcal{TC}$ in which temporal connectivity occurs infinitely often, thus enabling any-to-any reliable communication in *Dynamic_CompSync* assuming all correct nodes at any time t.
We define the following sub-class $\mathcal{TC}^{\mathcal{R}}_k$ of $\mathcal{TC}^{\mathcal{R}}$.

Definition 39 [**Class $\mathcal{TC}^{\mathcal{R}}_k$ (Recurrent k Temporal Connectivity)**]. $\forall t \in \mathcal{T}$, $\mathcal{G}[t; +\infty) \in \mathcal{TC}_k$ (starting from any time t, the TVG is eventually temporally k connected).

The class $\mathcal{TC}^{\mathcal{R}}_k$ characterizes the TVGs where the any-to-any reliable communication problem in *Dynamic_CompSync_Global* is solvable at any time t if $f < k/2$ by construction. Furthermore, it is the minimal class of TVGs where it is possible.

Class $\mathcal{C}^*$ (Definition 34) has been identified as a sub-class of $\mathcal{TC}^{\mathcal{R}}$ where simple broadcast terminates in $O(n)$ times. Specifically, assuming a *Dynamic_FullSync* system of all correct nodes, every any-to-any reliable communication instance terminates in $O(n)$ times [51].
We define the sub-class $\mathcal{CK}^*_k$ of $\mathcal{C}^*$.

Definition 40 [**Class $\mathcal{CK}^*_k$ (1-interval k-connectivity)**]. $\forall G_i \in \mathcal{G}$, G_i is a k-connected graph (the node connectivity of every snapshot is greater or equal than k).

We prove in the following Theorem 14 that the class $\mathcal{CK}^*_k$ provides two useful properties:

- there always exist journeys between every pair of nodes with dynamic minimum cut at least k;
- there always exist journeys between every pair of nodes with dynamic minimum cut at least k in every temporal subgraph defined in $[t, t + n - k]$.

Theorem 14. Given an evolving graph $\mathcal{G}$, if $\mathcal{G} \in \mathcal{CK}^*_k$ then $\mathcal{G}[t, t + n - k] \in \mathcal{TC}_k$ for any $t \in \mathcal{T}$.

Proof. Let us consider a pair of nodes $u, v \in V$. By definition, the dynamic minimum cut from node v to u is the minimum number of nodes (besides v and u) that have to be removed from the network to prevent the existence of a journey from v to u. It follows that, if it always exists at least one journey from v to u removing whatever set of $k - 1$ nodes, then the dynamic minimum cut is at least k. The removal of whatever set of $k - 1$ vertices makes the resulting subgraph 1-interval 1-connected ($\mathcal{C}^*$). It has been proven that $n - 1$ instants are sufficient to traverse a journey between any two nodes in a 1-interval connected network [51]. The dynamic network we are considering has size $n - (k - 1)$ nodes after the removal

and it is 1-interval connected. It is possible to conclude that a journey always exists between every pair of nodes and it can be traversed in at most $n-(k-1)-1 = n-k$ instants. □

Corollary 7. The any-to-any reliable communication problem is always solvable in *Dynamic_FullSync_Global* if $\mathcal{G} \in \mathcal{CK}_k^*$ and $k > 2f$. Furthermore, it always terminates in $n-k$ rounds with `MTD`.

Proof. It follows from Remark 18 and Theorem 14 □

Notice that, differently for the condition in Remark 18, give a dynamic network expresses as evolving graph, 1-interval k-connectivity can be verified polynomially in the number of nodes and in the lifetime of the system.

Chapter 13

Reliable Communication with Locally Bounded Byzantine Failures, DynCPA

We investigate in this chapter the potential employment of CPA as a solution to the reliable communication problem in dynamic distributed systems. We introduce the required modifications to let CPA work in dynamic networks, defining DynCPA, and we identify the conditions a dynamic communication network has to satisfy enabling reliable communication in the locally bounded failure model.

Part of the results presented in this Chapter was published in [12]

13.1 System model

We alternatively consider a *Dynamic_CompSync_Local* or a *Dynamic_FullSync_Local* system. We assume processes able to instantaneously detect the presence of a link. While evaluating the message complexity of a protocol, we suppose processes being aware of the message deliveries of their links, thus employing the no-duplication layer.

13.2 DynCPA

The DynCPA protocol extends CPA to cope with a dynamic environment. In CPA processes relay all the delivered contents right after their delivery. Differently from static networks, the neighborhood of processes changes over time in dynamic ones. Furthermore, the peers are not aware of the processes they eventually have a link with. Porting CPA to dynamic distributed systems, we adopt the same spreading policy employed in MTD: diffuse all messages contained in a specific collection every time that the local neighborhood of a process changes. Specifically, we define the DynCPA protocol as follows:

- every process sends to all of its neighbors all the contents $\langle s, c \rangle$ it delivered every time its neighborhood changes;

- the source p_s delivers its content $\langle s, c\rangle$ (*acceptance policy 0, AP0*);
- a process p_i delivers every content $\langle s, c\rangle$ received from its source p_s (*acceptance policy 1, AP1*);
- a process p_i delivers every content $\langle s, c\rangle$ received from $f+1$ distinct neighbors $p_j \neq p_s$ (*acceptance policy 2, AP2*);

13.3 DynCPA correctness analysis

The *MKLO* (Section 5.1) allows to characterize the instances where the reliable communication protocol can be solved assuming the locally bounded failure model in static distributed systems. We extend the concept of *MKLO* to dynamic graphs, defining the *TMKLO* metric, and we use it to define dynamic network conditions enabling reliable communication.

Definition 41 [**Function** $\mathcal{A}_k(v,t)$]. Given a TVG $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, a node $s \in V$ and a time $\hat{t} \in \mathcal{T}$. The function $\mathcal{A}_k(v,t)$ defined for $t \geq \hat{t} \in \mathcal{T}$ for every node $v \in V$ as follows:

$$\mathcal{A}_k(v,t) = \begin{cases} 1 & \text{if } v = s \qquad \text{(AK0)} \\ 1 & \text{for } t \geq t' + \zeta(\langle s, v\rangle, t') \text{ if } \exists\ t' \geq \hat{t}\ :\ \mathsf{RL}(s, v, t') = \mathsf{true} \qquad \text{(AK1)} \\ 1 & \text{for } t \geq \max(t_i + \zeta(\langle v, v_i\rangle, t_i)) \quad \text{if } \exists\ S \subset V, |S| \geq k : \\ & \qquad \forall v_i \in S,\ \mathcal{A}_k(v_i, t_i) = 1,\ \mathsf{RL}(v, v_i, t_i) = \mathsf{true} \qquad \text{(AK2)} \\ 0 & \textit{otherwise} \end{cases}$$

Definition 42 [**Temporal Minimum K-Level Ordering (*TMKLO*)**]. The *Temporal Minimum K-Level Ordering* $\mathcal{M}_k(\mathcal{G}, s, \hat{t})$ of a TVG $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ for a given node $s \in V$ and a time $\hat{t} \in \mathcal{T}$ is the partitioning of V into subsets L_i, called *levels*, such that:

$$v \in L_{t_i} \textit{ iff } t_i = \min\ t \geq \hat{t} \in \mathcal{T} \textit{ such that } \mathcal{A}_k(v, t_i) = 1$$

The Function $\mathcal{A}_k(v,t)$ and the related *TMKLO* characterize a specific possible message propagation pattern in a *Dynamic_CommSync* systems. In particular, given a time $\hat{t} \in \mathcal{T}$ and $s \in \mathcal{G}$, the Function $\mathcal{A}_k(v,t)$ is equal to 1 for a node v at time t if either $v = s$, or it was possible to establish a reliable link between v and s at $t' \geq \hat{t}$, or it was possible to establish reliable links between v and k nodes already having their associated value in the acceptance function equal to 1. Basically, the message propagation patter required by `DynCPA` to accomplish reliable communication, assuming specific values for k and f. The *TMKLO* collects the minimum times t_i when the $\mathcal{A}_k(v_i, t)$ turns to 1 for every process $v \in V$.

Lemma 5. `DynCPA` solves the one-to-all reliable communication problem from a process p_s at $\hat{t}$ in *Dynamic_CompSync_Local* if all processes are correct and there exists the *TMKLO* $\mathcal{M}_k(\mathcal{G}, p_s, \hat{t})$ with $k > f$.

Proof. The *TMKLO* $\mathcal{M}_k(\mathcal{G}, p_s, \hat{t})$ characterizes a specific message pattern in a *Dynamic_CompSync_Local* system. We need to prove that such a message pattern enables one-to-all reliable communication from a process p_s at $\hat{t}$. Every process is associated to a specific level L_{t_i} in $\mathcal{M}_k(\mathcal{G}, p_s, \hat{t})$ where $t_i \geq \hat{t} \in \mathcal{T}$ is the first instant the acceptance function $\mathcal{A}_k(v, t)$ given nodes s and $\hat{t}$ is equal to 1 for v. The acceptance function associated with a specific node v turns to 1 due to the verification of one among the *AK*0, *AK*1, and *AK*2 conditions. Node s satisfies *AK*0 and the associated process p_s sent a content. It follows that it delivered the content according to acceptance policy 0 of DynCPA. After that, it relays the content at every time $t \geq \hat{t}$ given the assumption of negligible local computation latency. The predicate $RL(v, u, \bar{t})$ guarantees the reliable delivery of a message sent from process p_v to p_u at $\bar{t}$. Every process p_i inserted in $\mathcal{M}_k(\mathcal{G}, p_s, \hat{t})$ due to *AK*1 had the predicate $RL(s, i, \bar{t})$ verified for $\bar{t} \geq \hat{t}$. It follows that it was possible to establish a reliable link between processes p_s and p_i at time $\bar{t}$. Process p_s continuously relay its content, it follows that every process p_i received the content directly from p_s and delivered it according to AP1. Every process p_i inserted in $\mathcal{M}_k(\mathcal{G}, p_s, \hat{t})$ due to *AK*2 had the predicate $RL(*, i, *)$ verified with a set of k processes already in the *TMKLO*. At least one of such p_i verified $RL(*, i, *)$ with a set of k processes which verified *AK*1, and thus delivered the content and continuously relay it. It follows that also p_i delivers the content according to AC2 if $k > f$. The same reasoning extends to the other processes verifying *AK*2 with the predicate $RL(*, i, *)$ with other k nodes that already verified the acceptance function. □

Theorem 15. DynCPA solves the one-to-all reliable communication problem from a process p_s at $\hat{t}$ in *Dynamic_CommSync_Local* if there exists the *TMKLO* $\mathcal{M}_k(\mathcal{G}, p_s, \hat{t})$ with $k > 2f$.

Proof. The claim follows from Lemma 5 considering that $k > 2f$ and at most f faulty processes can be present in the neighborhood over every node over the lifetime of the network. □

Theorem 16. DynCPA solves the one-to-all reliable communication problem from a process p_s at $\hat{t}$ in *Dynamic_CompSync_Local* only if there exists the *TMKLO* $\mathcal{M}_k(\mathcal{G}, p_s, \hat{t})$ with $k > f$.

Proof. It follows from the definition of $\mathcal{A}_k(v, t)$ function and the related *TMKLO*. In particular, $\mathcal{A}_k(v, t)$ models a specific message propagation pattern in *Dynamic_CompSync* systems and *TMKLO* collects the first times t_i when certain conditions verifies.

Specifically, it models the pattern where every process in the temporal minimum level ordering either is the source, or it had a reliable link available with the source, or had at least k reliable link available with processes already in the *TMKLO*. If a process is not in the partitioning it implies that none of the following conditions verified, thus either it has not delivered the content or we get a contradiction.

□

Theorem 17. DynCPA solves the one-to-all reliable communication problem from a process p_s at $\hat{t}$ in *Dynamic_CompSync_Local* if and only if there exists the *TMKLO* $\mathcal{M}_k(\mathcal{G}_F, p_s, \hat{t})$ with $k > f$ considering any possible f-local corruption set F.

Proof. It follows from Lemma 5 and Theorem 16 considering that a *TMKLO* with $k > f$ exists removing from the network any possible corruption set. □

Given a TVG $\mathcal{G}$ expressed as evolving graph, a *TMKLO* $\mathcal{M}_k(\mathcal{G}, s, \hat{t})$ can be computed polynomially with respect to the number of nodes n and the lifetime of the network $\mathcal{T}$.

Theorem 18. Given a TVG $\mathcal{G}$ with finite lifetime $\mathcal{T}$ expressed as evolving graph, a *TMKLO* $\mathcal{M}_k(\mathcal{G}, s, \hat{t})$ can be computed polynomially with respect to the size and the lifetime of the graph, specifically, its computational complexity is $O(|V| + |\mathcal{T}||E|)$.

Proof. Given a TVG $\mathcal{G}$ modeled with an evolving graph, a *TMKLO* $\mathcal{M}_k(\mathcal{G}, s, \hat{t})$ can be computed polynomially as follows.

Node s is placed in level $L_{\hat{t}}$ of the *TMKLO*. Then, the snapshots characterizing the TVG have to be analyzed sequentially from the one at time $\hat{t}$. In particular, it has to be verified if the edges with only one endpoint already included in some level of the *TMKLO* are up enough to satisfy RL() and if they allow a node j to be part of the *TMKLO* according to one of the conditions AK1 and AK2. The algorithm ends when a *TMKLO* is found or when all the snapshots have been analyzed (and in the latter case we can infer that the TMKLO does not exist). Assuming that $\mathcal{G}$ spans over a finite lifetime $\mathcal{T}$, the complexity of this algorithm is:

$$O(|\mathcal{T}||E|) + O(|V| + |E|) = O(|V| + |\mathcal{T}||E|)$$

□

It follows that the sufficient condition identified in Theorem 15 can be verified polynomially for a reliable communication instance from a source p_s to a target p_t at time $\hat{t}$, whereas the exact condition reported in Theorem 17 is NP-Hard to verify [43].

13.4 DynCPA performance analysis

Message Complexity. Every process executing DynCPA sends all the delivered content to all of its neighbors every time its neighborhood changes. It follows that it is not possible in general to overbound the message complexity of DynCPA.

Assuming the no-duplication layer and counting only the succeeding message transmissions, every message m from a process p_i to a process p_j is exchanged only once. Every process that delivers a content in DynCPA relays only one message to each of its neighbors (over the time). It follows that the message complexity of DynCPA is $O(n^2)$ under such an assumption.

Delivery Complexity. The delivery complexity of DynCPA is equivalent to CPA, namely every process either receives a content directly from its source or it waits

for $f + 1$ copies of the same content sent by distinct peers. It follows that the delivery complexity of DynCPA is $O(f)$.

Communication Latency. The communication latency of DynCPA strongly depends on the evolution of the dynamic communication network and on the placement of the Byzantine processes. The *TMKLO*, as the *MKLO* in static networks (Theorem 4), allows to establish upper and lower bounds on the communication latency achieving reliable communication with DynCPA.

Theorem 19. Let us consider a source process p_s succeeding in one-to-all reliable communication at $\hat{t}$ with DynCPA in *Dynamic_CompSync_Local*. Let t_{max}^{f+1} be the time associated to the last level L_{t_x} of $\mathcal{M}_{f+1}(\mathcal{G}, p_s, \hat{t}) = \{L_{t_0}, L_{t_1} \dots L_{t_x}\}$. Let t_{max}^{*f+1} the maximum time associated to the last level L_{t_x} of $\mathcal{M}_{f+1}(\mathcal{G}_{\mathcal{F}}, p_s, \hat{t}) = \{L_{t_0}, L_{t_1} \dots L_{t_x}\}$ considering every f-local set F. Let t_{max}^{2f+1} be the time associated to the last level L_{t_x} of $\mathcal{M}_{2f+1}(\mathcal{G}, p_s, \hat{t}) = \{L_{t_0}, L_{t_1} \dots L_{t_x}\}$ (if it exists). The Communication Latency (CL) is bounded as follows:

$$t_{max}^{f+1} - \hat{t} \leq CL \leq t_{max}^{*f+1} - \hat{t} \leq t_{max}^{2f+1} - \hat{t}$$

Proof. Lower Bound: Let us assume by contradiction that CL can be lower than $t_{max}^{f+1} - \hat{t}$. It follows that the last process p_i delivering the content does it at a time $t_i < t_{max}^{f+1}$. By definition of TMKLO, a process p_i is inserted in a level L_{t_x} at the lowest time t_x one of the following conditions is satisfied: $p_i = p_s$, p_s, p_t can establish a reliable link, or p_i can establish a reliable link with k processes that are already included in the TMKLO. It follows that if $t_i < t_{max}^{f+1}$ it has delivered the content without a reliable link with the source or reliable links with $f + 1$ other processes that already delivered the content. This is in contradiction with the acceptance policies *AC0*,*AC1* and *AC2* of DynCPA.

Upper Bounds: Let us assume by contradiction that CL can be greater than $t_{max}^{*f+1} - \hat{t}$. It follows that the last process p_i delivering the content does it at a time $t_i > t_{max}^{*f+1}$. By definition of TMKLO, a process p_i is inserted in a level L_{t_x} at the lowest time t_x one of the following conditions is satisfied: $p_i = p_s$, p_s, p_t can establish a reliable link, or p_i can establish a reliable link with k processes that are already included in the TMKLO; Given that t_{max}^{*f+1} is the maximum time associated to a level in $\mathcal{M}_{f+1}(\mathcal{G}_{\mathcal{F}}, p_s, \hat{t})$, every possible corruption set has been consider and $k > f + 1$, the delivery off all correct processes occurred at t_{max}^{*f+1}, leading to a contradiction.

The inequality $t_{max}^{*f+1} - \hat{t} \leq t_{max}^{2f+1} - \hat{t}$ is motivated by the fact a f-local do not necessarily distribute among all levels in a TMKLO. □

13.5 Recurrent dynamic networks

In this section, we start from a specific class of dynamic networks and we identify an additional set of assumptions enabling reliable communication in *Dynamic_Async_Local*. Specifically, we start from the $\mathcal{E}^{\mathcal{R}}$ recurrent edges TVG class (Definition 35), that guarantees every edge present infinitely often over time. As we highlighted several times, a communication link is useful to processes only if it allows reliable message delivery. Thus, we further assume that every edge

infinitely satisfies RL() over time. The model just characterized aims to simulate a *Static_Local_Async* system, indeed even if all the communication links do not provide any latency guarantee, they infinitely ensure reliable delivery. It follows that we can extend in $\mathcal{E}^{\mathcal{R}}$ all the correctness conditions defined for CPA (Section 6.3.1).

Theorem 20. DynCPA protocol solves the one-to-all reliable communication problem from a process p_s at any time t in *Dynamic_Async_Local* if the communication network $\mathcal{G}$ is a $\mathcal{E}^{\mathcal{R}}$ recurrent edges TVG and the parameter $j_s(G)$ associated to its footprint G is greater then twice the value of f.

Theorem 21. DynCPA protocol solves the one-to-all reliable communication problem from a process p_s at any time t in *Dynamic_Async_Local* if the communication network $\mathcal{G}$ is a $\mathcal{E}^{\mathcal{R}}$ recurrent edges TVG and the parameter $h_s(G)$ associated to its footprint G is greater then the value of f.

Theorem 22. DynCPA protocol solves the any-to-any reliable communication problem at any time t in *Dynamic_Async_Local* if the communication network $\mathcal{G}$ is a $\mathcal{E}^{\mathcal{R}}$ recurrent edges TVG and the parameter $j(G)$ associated to its footprint G is greater then twice the value of f.

Theorem 23. DynCPA protocol solves the any-to-any reliable communication problem at any time t in *Dynamic_Async_Local* if the communication network $\mathcal{G}$ is a $\mathcal{E}^{\mathcal{R}}$ recurrent edges TVG and the parameter $h(G)$ associated to its footprint G is greater then the value of f.

Proof. All the conditions identified in Theorems 20,21,22 and 23 follow from Remarks 16 and 17 and from Corollaries 1 and 2 (in Section 6.2.3) considering that every edge in the footprint infinitely guarantees reliable delivery. □

The conditions stated in Theorems 20,21,22 and 23 can also be reinterpreted from the process perspective. Let us suppose the processes knowing that the communication network evolves as a recurrent edge TVG (but they ignore their precise evolution, namely $\mathcal{G}$), and having knowledge only on the topology of the footprint G. Under such assumptions, processes are able to verify whether the reliable communication problem is solvable: they can test the conditions defined for CPA (Section 6.3.1) on G. Their validity in the dynamic contest are guaranteed by the fact that *1)* processes are able to instantaneously detect their neighbors, *2)* they continuously aims to relay delivered contents to all the processes they contact, and *3)* links infinitely provide reliable delivery.

The set of assumptions identified in Theorems 20,21,22 and 23 do not allow to establish bound to the communication latency. The reason is that there is no guarantee on the time required for any link to provide reliable delivery. On the other hand, considering the $\mathcal{E}^{B}$ class of TVG (Definition 36), we can characterize an alternative set of assumptions enabling reliable communication with bounded communication latency, extending the results of Theorem 4.

Theorem 24. Given a source process p_s succeeding in one-to-all reliable communication at $\hat{t}$ with DynCPA in *Dynamic_CompSync_Local* having as communication

network a $\mathcal{E}^B$ TVG with $\zeta_{max} = \max(\zeta(e,t))$ and δ_{max} as maximum delay between two consecutive appearances of any edge. Let l^{*f+1} the maximum number of levels associated to the *MKLO* $\mathcal{L}_{f+1}(G_F, p_s) = \{L_0, L_1 \dots L_x\}$ considering every f-local set F. Let l^{2f+1} be the size of the *MKLO* $\mathcal{L}_{2f+1}(G, p_s) = \{L_0, L_1 \dots L_x\}$ (if it exists). The Communication Latency (CL) is upper bounded as follows:

$$CL \leq l^{*f+1}(\zeta_{max} + \delta_{max}) \leq l^{2f+1}(\zeta_{max} + \delta_{max})$$

Proof. Given the assumptions on the dynamic communication network we know that every edge reappears in δ_{max} and it guarantees reliable delivery with latency ζ_{max}. The worst-case scenario with respect to the message propagation in the system model considered is the one where every node has to wait for δ_{max} to forward a message. The bound on the communication latency follows from Remarks 16,17 (Section 6.2.3) noting that every node in level L_i delivers in at most $\delta_{max} + \zeta_{max}$. □

13.6 1-interval dynamic networks

We consider in this section a *Dynamic_FullSync_Local* system in which the latency function $\zeta(e,t)$ of the associated TVG is always equal to 1 for every edge at every time t. We define two sub-class $\mathcal{CJ}^*_{\mathsf{j}}$ and $\mathcal{CH}^*_{\mathsf{h}}$ of $\mathcal{C}^*$, 1-interval dynamic networks (Definition 34), in order to define alternative dynamic network conditions enabling reliable communication. The former assumes snapshots having $j(G_i) \geq \mathsf{j}$, the latter considers snapshots with $h(G_i) \geq \mathsf{h}$.

Class $\mathcal{CJ}^*_{\mathsf{j}}$ (**1-interval** j **snapshots**). $\forall G_i \in \mathcal{G}$, $j(G_i) \geq \mathsf{j}$ (the j parameter of every snapshot is greater than or equal to j).

Class $\mathcal{CH}^*_{\mathsf{h}}$ (**1-interval** h **snapshots**). $\forall G_i \in \mathcal{G}$, $h(G_i) \geq \mathsf{h}$ (the h parameter of every snapshot is greater than or equal to h).

Theorem 25. The any-to-any reliable communication problem is always solvable in *Dynamic_FullSync_Local* if $\mathcal{G} \in \mathcal{CH}^*_{\mathsf{h}}$ and $\mathsf{h} > f$. Furthermore, the communication latency is upper bounded by $n - f - 1$.

Proof. The assumption of $\mathcal{G} \in \mathcal{CH}^*_{\mathsf{h}}$ guarantees all the snapshots G_i having $h(G_i) > \mathsf{h}$. It follows that for all the snapshots G_i there exists the Minimum K-Level Ordering $\mathcal{L}_k(G_{i,\bar{F}}, s)$ with $k = \mathsf{h} > f$ considering any source s and removing any f-local set F from G_i.

We show that `DynCPA` succeeds in any-to-any reliable communication problem under the assumptions considered. Let us assume the nodes of any compromised set F removed from $\mathcal{G}$. Let us collect in a set D the processes that delivered a specific content. When the reliable communication instance is initiated, the source delivers its content and sends it to all of its neighbors in the subsequent round.
At the successive round, the source has a reliable link available with at least $f + 1$ processes. Therefore, they all deliver the content of the source. It follows that the set D contains at the second round the source and at least $f + 1$ processes.
In the third round, the network may change but it ensures $h(G_i) > f$. If a node

$v \in V, v \notin D$ is currently neighbor of the source, it delivers the content and is inserted in D. If all the current neighbors of the source are in D, all nodes in L_2 of $\mathcal{L}_{f+1}(G_{*,\bar{F}}, s)$ are currently linked to at least $f+1$ nodes that are relaying the content. Thus, if a process in $L2$ did not delivered the content yet, it delivers at that round. If all nodes in $L1$ and L_2 are already in D then every process p_i in L_3 of $\mathcal{L}_{f+1}(G_{*,\bar{F}}, s)$ not in D has at least $f+1$ reliable link with nodes in D, and delivers the content. The same reason extends till including all nodes in G, noting that at least 1 node is inserted in D at every round. □

Theorem 26. The any-to-any reliable communication problem is always solvable in *Dynamic_FullSync_Local* if $\mathcal{G} \in \mathcal{CJ}^*_{\mathsf{j}}$ and $\mathsf{j} > 2f$. Furthermore, the communication latency is upper bounded by $n - f - 1$.

Proof. It follows from the same reasoning provided in Theorem 25 considering that at most f faulty processes may be linked to every process during all lifetime of the network. □

13.7 Conclusion

We analyzed in this chapter the reliable communication problem in a dynamic distributed system assuming the locally bounded failure model. We showed that the `CPA` protocol can be extended and properly work in dynamic networks, defining `DynCPA`. We provided several dynamic network conditions enabling one-to-all and any-to-any reliable communication at a specific time t or infinitely often. All of them either assume a synchronous system or one that guarantees only bounded local computation latency. It remains open the possibility of defining dynamic network conditions enabling to solve our problem in an asynchronous dynamic system.

Part III

Dynamic Faults, Static Networks

Chapter 14

Reliable Communication with Mobile Byzantine Faults

We investigate in this chapter the reliable communication problem in static distributed systems affected by Mobile Byzantine Faults. Given that the reliable communication specification does not consider the state of the processes potentially changing over time (between correct and faulty), we define a new specification to the reliable communication problem considering Mobile Byzantine Faults. We show that the problem so defined cannot be solved in the case of an asynchronous system when mobile Byzantine agents are present. We, therefore, identify the necessary and several sufficient network conditions enabling reliable communication in the considered environment.

The results presented in this Chapter were published in [16].

14.1 System model

We consider a static distributed system affected by Mobile Byzantine Faults, resulting from a set of f mobile agents present in the system and capable of moving between one process to another making them Byzantine faulty. We alternatively assume a synchronous or asynchronous distributed systems. In the former case, we consider one evolving in synchronous rounds of three phases: *send*, *receive*, and *computation*. We suppose that every process is equipped with a tamper-proof read-only memory where the code of the distributed protocol $\mathcal{P}$ is stored. We assume the mobile agents having roaming pace $\lambda = 1$ and able to move only in between the computation and the send phase of two consecutive rounds. We alternatively consider either an *aware* or *unaware* mobile Byzantine failure model. We assume every cured process wiping all of its local variables at the beginning of the round.

14.2 Reliable communication with MBF specification

The standard specification of the reliable communication problem between a *source* process p_s and a *target* process p_t requires the following properties to be satisfied:

safety - if a correct process p_t delivers a content c from p_s, then c has been sent by p_s; *liveness*: if a correct process p_s sends a content c to a correct process p_t, then c is eventually delivered by p_t.
In the environment here considered, the failure state of every process may continuously change over time and no process is permanently correct. Furthermore, processes can be compromised while they are exchanging messages, namely between the computation and send phase when contents are generated, enqueued for the transmission, and then diffused. It follows that every process which aims to reliably communicate with another peer must remain correct for at least two consecutive rounds to diffuse any message. Therefore, we define as *correct source* a process p_s that is correct for two consecutive round r_x and r_{x+1}, and computes a content c at r_x.

Another aspect to take into account is that a message may require several rounds to reach a target process, due to the network topology and to the protocol employed diffusing it. Nevertheless, the state of every process may change over time and a target process must be not permanently faulty to deliver a content sent by a source. Therefore, we say that a process p_i is *not permanently faulty* if for every round r_x there always exists a round $r_y \geq r_x$ where p_i is correct.

Given the considerations stated above, we define a specification for the reliable communication problem with Mobile Byzantine Faults.

Reliable communication with MBF specification. Given a *correct source* process p_s and *target* process p_t, a reliable communication primitive guarantees that:

- *safety*: if p_t is correct at r_x and it delivers a content c from p_s, then c has been sent by p_s;
- *liveness*: if a correct source p_s sends a content c to a not permanently faulty process p_t, then p_t eventually delivers c from p_s.

14.3 Reliable communication in asynchronous systems

We show in this section that is impossible to design a protocol $\mathcal{P}$ able to solve the reliable communication problem between a correct source p_s and a target p_t when the assumed distributed system is asynchronous and there is only one mobile Byzantine agent.

When considering an asynchronous system, correct processes still execute a deterministic distributed protocol $\mathcal{P}$, but there is no known upper bound on the time demanded for local computation, neither on the time required to deliver a message through a link.

Theorem 27. There exists no distributed protocol $\mathcal{P}$ that is able to solve the reliable communication problem specification with Mobile Byzantine Faults in an asynchronous system even if *(i)* the source process p_s is permanently correct, *(ii)* there exists only one mobile Byzantine agent, and *(iii)* processes are aware of their failure state.

Proof. The reliable communication specification requires both safety and liveness property to be satisfied. We show that no protocol $\mathcal{P}$ can ensure the liveness property, even assuming an always correct source, only one mobile Byzantine agent, and the aware failure model.

The reason is that every reliable link may never guarantee reliable delivery in the assumed system model. The reliable delivery property of a reliable link is guaranteed only between correct processes. Given the no constraints supposed on the link transmission delay, even assuming a permanently correct source continuously sending messages relative to a content c, such messages may never be delivered by the links of the source, because any receiver p_i may be continuously and temporarily compromised by a mobile agent during each message transmission. Thus no process in the system may receive a message from the source, even being permanently not faulty. □

On the other hand, we highlight the solvability of safe communication (i.e. enforcing only the safety property of reliable communication) in the case of an asynchronous system. Specifically, it is possible to design a "best-effort" protocol that ensures safety while trying to maximize the number of delivered contents.

Theorem 28. Safe communication can be achieved with a non-degenerated protocol in an asynchronous distributed system in the aware mobile Byzantine failure model.

Proof. We show a "best-effort" solution for the safe communication problem. Let us assume every process p_i having access to a local clock T_i. It is reasonable to assume that a Byzantine agent which is forcing a process p_k to send a message m must remain on p_k till the end of its transmission to guarantee the link message delivery. Let us consider the following protocol:

- the source process p_s continuously sends the message $\langle s, t, c\rangle$;
- every process p_k stores and continuously relays any message $\langle s, t, c\rangle$ received from p_s;
- every process p_k stores every message $\langle s, t, c\rangle$ received from a process $p_i \neq p_s$ jointly with the timestamp $\mathrm{t}^x_{\langle s,t,c\rangle}$ containing the value of T_k at the reception of $\langle s, t, c\rangle$;
- every process p_k that stores a set of $2f + 1$ tuples $M := [\langle\langle s, t, c\rangle, \mathrm{t}^1_{\langle s,t,c\rangle}\rangle, \langle\langle s, t, c\rangle, \mathrm{t}^2_{\langle s,t,c\rangle}\rangle, \ldots, \langle\langle s, t, c\rangle, \mathrm{t}^{2f+1}_{\langle s,t,c\rangle}\rangle]$ received from distinct neighbors such that $\forall_{x<y}, \mathrm{t}^x_{\langle s,t,c\rangle} < \mathrm{t}^y_{\langle s,t,c\rangle}$ and $\mathrm{t}^{2f+1}_{\langle s,t,c\rangle} - \mathrm{t}^1_{\langle s,t,c\rangle} < \lambda$ continuously relays $\langle s, t, c\rangle$;
- if process p_t relays $\langle s, t, c\rangle$ then it delivers c.

We show that the protocol defined above guarantees the safety property of reliable communication in an asynchronous system. Let us consider a single agent initially placed on a process $p_1 \neq p_i$, that starts the transmission of a spurious message $\langle s, t, \hat{c}\rangle$ to a process p_i at time t_1^{start}, and concludes at time t_1^{end} when $\langle s, t, \hat{c}\rangle$

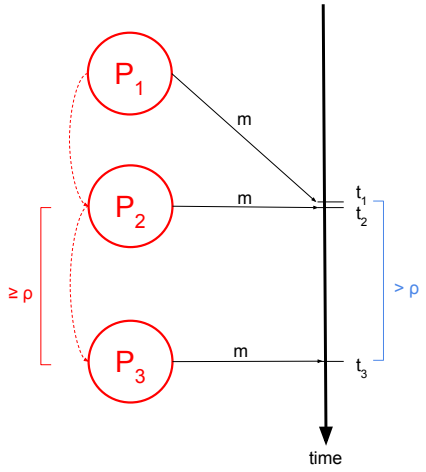


Figure 14.1: Graphical execution example of Theorem 28.

is received by p_i. Process p_i stores $\langle s, t, \hat{c} \rangle$ with the timestamp $t^1_{\langle s,t,\hat{c} \rangle}$ obtained by its local clock at the reception of the message. Subsequently, the Byzantine agent may move on a different process p_2 and start sending another copy of $\langle s, t, \hat{c} \rangle$ to p_i at time t_2^{start}, that it concludes at time t_2^{end} when the message is received by p_i. Again, process p_i stores $\langle s, t, \hat{c} \rangle$ with the timestamp $t^2_{\langle s,t,\hat{c} \rangle}$. And once more, the agent can move another time on a process p_3 and iterate again the transmission of $\langle s, t, \hat{c} \rangle$. According with the absence of link latency guarantees, it could happen that $t_i^{end} - t_i^{start} \to 0$. On the other hand, $t^{x+2}_{\langle s,t,\hat{c} \rangle} - t^{x}_{\langle s,t,\hat{c} \rangle} > \hat{\jmath}$, because a mobile agent must move twice in order to send a spurious message for three distinct processes. It follows that assuming f mobile Byzantine agents, if a process p_i receives more than $2f$ copies of a message m in a time windows shorter than $\hat{\jmath}$, then it can safely accept the contained content. For ease of explanation, the execution stated above is depicted in Figure 14.1. □

14.4 Reliable communication in synchronous systems

14.4.1 `RC-Sasaki-et-al.` protocol [71]

Sasaki et al. [71] defined a reliable communication protocol to solve mobile Byzantine agreement in multi-hop routed networks (where messages are relayed over and only fixed routes). Specifically, it allows all correct processes but $2f$ (faulty or cured at transmission time) in round r_x to reliably communicate with all correct processes in round $r_{x+\beta}$. The Sasaki et al. protocol is not aimed to enable each correct process to spread its contents to all eventually correct peers, because only a certain number of nodes need to reliably communicate to succeed in solving the agreement. We slightly adapt the protocol they defined to address our reliable communication specification, providing the pseudo-code in Algorithm 2 and referring to such a protocol with `RC-Sasaki-et-al.`. In detail, differently from the protocol defined in [71], we substitute the tamper-proof round counter assumed by

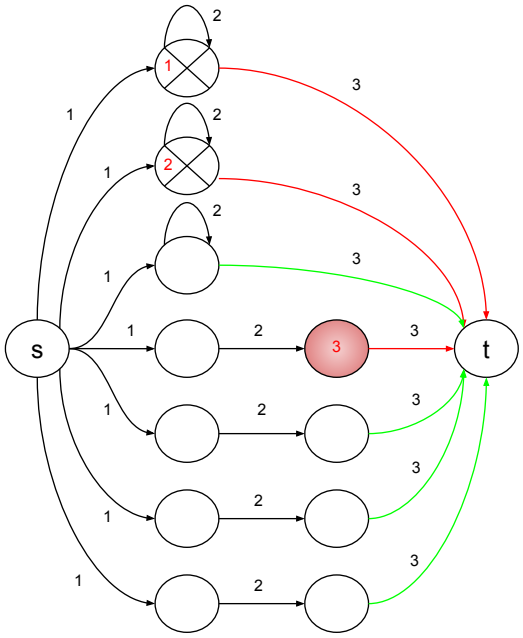


Figure 14.2: Example of execution of `RC-Sasaki-et-al.` protocol

Sasaki et al. with the safe implementation provided by Bonnet et al. [10], allowing each process to retrieve the actual round index at the beginning of the computation phase (the round counter is necessary to the proper execution of the protocol), and we extend it to diffuse not only a binary value. In `RC-Sasaki-et-al.`, between every pair of processes p_s and p_t there exist a disjoint path solution $\Pi_{s,t}$, $|\Pi_{s,t}| = a$ in which every path has length at most β, $\Pi_{s,t}$ is known by every peer. Every process relays messages between p_s and p_t only in specific rounds, namely every β ones. Furthermore, every process relays a message m at round r_{x+1} if m was received from more than βf neighbors at the round r_x: this allows to bound the number of processes that concurrently may send a spurious message to βf.

Figure 14.2 depicts an example of execution of a reliable communication instance from a source process p_s to a target process p_t with `RC-Sasaki-et-al.`. In such an example, $f = 1$, $a = 7$ and $\beta = 3$. The numbers in black represent the round when a message from p_s to p_t is relayed, the ones in red are the rounds when processes are compromised.

14.4.2 RCMB protocol

We define a new solution addressing the reliable communication problem, `RCMB`. Differently from the one proposed by Sasaki et al., it aims to keep the number of processes that concurrently send spurious contents bounded over time.

Reliable Communication Mobile Byzantines - RCMB:

- the source process p_s computes the content c addressed to a target process p_t at round r_x and saves message $\langle s, t, c \rangle$ in a set variable *delivered*.

Algorithm 7 `RC-Sasaki-et-al`.

Send phase

1: **if** *cured* = `TRUE` **then**
2: $\quad$ *to_send* $\leftarrow \emptyset$
3: $\quad$ $C \leftarrow \emptyset$
4: **for** $\langle *, receiver \rangle \in$ *to_send* **do**
5: $\quad$ `SEND`(*, *receiver*)
6: *to_send* $\leftarrow \emptyset$

Receive phase

7: $received_{msg} \leftarrow \emptyset$
8: $received_r \leftarrow \emptyset$
9: **for** `RECEIVE`($\langle CNT, s, t, c \rangle$, q) **do**
10: $\quad$ $received_{msg} \leftarrow received_{msg} \cup \langle \langle CNT, s, t, c \rangle, q \rangle$
11: **for** `RECEIVE`($\langle$`RN`, $\bar{r}\rangle$, q) **do**
12: $\quad$ $received_r \leftarrow received_r \cup \langle \bar{r}, q \rangle$

Computation phase

13: **function** INIT
14: $\quad$ *to_send* $\leftarrow \emptyset$
15: $\quad$ $r \leftarrow 0$
16: $\quad$ $C \leftarrow \emptyset$

17: **function** RC-SEND(c,t)
18: $\quad$ $C \leftarrow C \cup \langle c, t \rangle$

19: **function** RC-DELIVER(c,s)

20: $r \leftarrow$ `getMajority`($received_r$, σ)

21: **if** $r \bmod \beta = \beta - 1$ **then**
22: $\quad$ **for** $\langle c, t \rangle \in C$ **do**
23: $\quad\quad$ **for** $\pi^k_{s,t} \in \Pi_{s,t}$ **do**
24: $\quad\quad\quad$ *to_send* $\leftarrow$ *to_send* $\cup \langle \langle$`CNT`$, s, t, c \rangle, \pi^k_{s,t}[1] \rangle$
25: $\quad$ $C \leftarrow \emptyset$

26: **for** $\langle \langle$`CNT`$, s, t, c \rangle, q \rangle \in received_{msg}$ **do**
27: $\quad$ **if** $\pi^k_{s,t}[r \bmod \beta] \neq s$ **or** $\pi^k_{s,t}[(r \bmod \beta) + 1] \neq t$ **then**
28: $\quad\quad$ $received_{msg} \leftarrow received_{msg} \setminus \{\langle \langle$`CNT`$, s, t, c \rangle, q \rangle\}$
29: $\quad$ **else if** $r \bmod \beta \neq \beta - 1$ **then**
30: $\quad\quad$ *to_send* $\leftarrow$ *to_send* $\cup \langle \langle$`CNT`$, s, t, c \rangle, \pi^k_{s,t}[(r \bmod \beta) + 2] \rangle$

31: **for** $M_{\langle s,t,c \rangle} \leftarrow \bigcup \langle \langle CNT, s, t, c \rangle, * \rangle \in received_{msg}$ **do**
32: $\quad$ **if** $|M_{\langle s,t,c \rangle}| > \sigma$ **then**
33: $\quad\quad$ *to_send* $\leftarrow$ *to_send* $\cup \langle \langle$`CNT`$, s, t, c \rangle, q \rangle$
34: $\quad\quad$ **if** $j = t$ **then**
35: $\quad\quad\quad$ `RC-DELIVER`(c, s)

36: **for** $q \in \Gamma(j)$ **do**
37: $\quad$ *to_send* $\leftarrow$ *to_send* $\cup \langle \langle$`RN`$, r + 1 \rangle, q \rangle$

- any message $\langle s, t, c \rangle$ stored in *delivered* is removed after τ rounds.
- every process p_i queues every message stored in *delivered* at round r_x to be sent in round r_{x+1} to itself and to all of its neighbors;
- if a correct process p_i receives a message $\langle s, t, c \rangle$ from p_s at round r_x, then p_i saves $\langle s, t, c \rangle$ in a set variable *delivered*, and it delivers c from p_s if $i = t$;
- if a correct process p_i receives more than σ copies of a message $\langle s, t, c \rangle$ from distinct neighbors at round r_x, then p_i saves $\langle s, t, c \rangle$ in a set variable *delivered*, and it delivers c from p_s if $i = t$;

The parameter σ is a safety threshold: at least $\sigma + 1$ copies of the same content that must concurrently be received to deliver it. The parameter τ allows processes that were faulty in the unaware failure model to remove the spurious contents that may have been injected by malicious agents. It can be ignored in the aware failure model because cured processes directly wipe their local variables. Notice that, in case of $\tau = 1$, every message stored in *delivered* at round r_x is queued to be sent at round r_{x+1} and then dropped.

14.4.3 Reliable communication correctness conditions

We provide several correctness conditions enabling reliable communication with one of the two protocols presented in the previous subsection. We investigate the problem solvability in two scenarios: a correct source and a permanently correct source (that is, a source that is correct in every round r_x). The latter case is motivated by the fact that such an additional assumption enables solving the reliable communication problem in further topologies.

Unaware failure model

Theorem 29. Reliable communication cannot be solved in the unaware mobile Byzantine failure model with $n \leq 4f$.

Proof. The result can be deduced from the lower bound implementing the safe register abstraction in the unaware mobile Byzantine failure model [18]. Let us consider a set of $4f$ processes connected through a complete communication network. Let us assume a correct source p_s that computes a content c at round r_0, that p_s sends it to all other processes at round r_1 and that p_t and other $f - 1$ processes are faulty at r_1. Thus, p_t is faulty while the reliable communication protocol is diffusing c according to a distributed protocol $\mathcal{P}$. Subsequently, the mobile Byzantine agents move on process p_s and on $f - 1$ other processes between rounds r_1 and r_2. It follows that at round r_2 there are $2f$ processes that share a state that contains c and $2f$ processes (f Byzantine faulty at r_2 and f that were faulty in r_1) that may share a state injected by the adversary, thus it is not possible to distinguish which set of processes is storing the content sent by the correct source. □

Theorem 30. The `RCMB` protocol with $\sigma = (\tau + 1)f$ guarantees safety of reliable communication in the unaware mobile Byzantine failure model.

Proof. Let us consider a set of n process connected through a complete network. Let us assume, for the ease of contradiction, that a target process p_t delivers a content $\hat{c}$ at round r_x from p_s but $\hat{c}$ has not been sent by its source (i.e. $\hat{c}$ is a spurious content).

The delivery of a content $\hat{c}$ in the RCMB protocol is independent from the process local variables and it is only determined by the messages that are currently received in a round. On the other hand, the messages that are diffused in a round depends on the content of the *delivered* variable.

The content $\hat{c}$ has not been received by a process through a link with the source process p_s according to our hypothesis. It follows that there have been more than $\sigma = (\tau + 1)f$ processes that sent $\langle s, t, \hat{c}\rangle$ to p_t at round r_x. The mobile Byzantine agents can force f processes to send $\langle s, t, \hat{c}\rangle$ at round r_x and they can inject $\langle s, t, \hat{c}\rangle$ in the *delivered* sent of the processes that were faulty at r_{x-k}, $k \in [1, \tau]$ if $\tau \geq 1$. Thus, at most τf correct processes may potentially relay $\langle s, t, \hat{c}\rangle$ in a round because they were previously faulty, since after τ rounds $\langle s, t, \hat{c}\rangle$ is dropped from the *delivered* set. Every other correct processes process $p_j \neq p_t$ sends $\langle s, t, \hat{c}\rangle$ at r_i only if either p_j received such a content through a link with process p_s, or from more than $(\tau+1)f$ neighbors. It follows that at most $(\tau+1)f$ processes in the system may concurrently send $\langle s, t, \hat{c}\rangle$. Thus content $\hat{c}$ has been sent by its source. This leads to a contradiction and the claim follows. □

Theorem 31. The RCMB protocol with $\tau = 1$ and $\sigma = (\tau+1)f$ achieves reliable communication in complete networks of size $n > 4f$ in the unaware mobile Byzantine failure model.

Proof. We verified the safety property of the RCMB protocol with $\sigma = (\tau + 1)f$ in the unaware mobile Byzantine failure model in Theorem 30. We need to prove the liveness property of reliable communication employing the protocol in a complete network of size $n > 4f$ considering $\tau = 1$.

Let us assume a correct source p_s that computes a content c at r_0 and sends it at r_1 to itself and to all of its neighbors according to the RCMB algorithm. It follows that more than $3f$ processes queue $\langle s, t, c\rangle$ to be sent at r_2, because it has been received through a link from its source. At any round r_x there are at most f processes that get faulty and at most f ones that were faulty in r_{x-1}. Thus, all correct processes receive at least $2f+1$ copies of $\langle s, t, c\rangle$ from distinct nodes at every round $r_y \geq r_2$ and they relay it at the subsequent round. It follows that message $\langle s, t, c\rangle$ is relayed by at least $2f + 1 > \sigma$ processes on avery round $r_y \geq r_2$, and that process p_t delivers c in a round $r_z \geq r_y$ it is correct. □

Theorem 32. The RCMB protocol with $\tau = 1$ and $\sigma = (\tau+1)f$ achieves reliable communication in the unaware mobile Byzantine failure model in a k-clique community network topology with $k > 4f + 1$.

Proof. We verified the safety property of the RCMB protocol with $\sigma = (\tau + 1)f$ in the unaware mobile Byzantine failure model in Theorem 30. We need to prove the liveness property of reliable communication in a k-clique community network topology with $k > 4f + 1$ considering $\tau = 1$.

Let us assume a correct source p_s that computes a content c at round r_0 and sends it at round r_1. Given a k-clique community network, two processes p_s and p_t are either both parts of a k-clique, or they are included in two distinct k-cliques that are connected through a sequence of adjacent ones.

Let us assume that p_s and p_t are both parts of a k-clique $\mathcal{K}_0$. We showed in Theorem 31 that all correct processes in a complete network of at least $4f+1$ nodes continuously relay a message $\langle s, t, c\rangle$ sent by a correct sender. It follows that p_t delivers c in a round $r_x \geq r_1$ it is correct.

Let us assume that p_t is part of a k-clique $\mathcal{K}_1$ adjacent to $\mathcal{K}_0$. All correct processes but $2f$ in $\mathcal{K}_0$ sends $\langle s, t, c\rangle$ at every round $r_x \geq r_2$ to all of their neighbors. It follows that p_t receives at least $2f+1 > \sigma$ copies of $\langle s, t, c\rangle$ on every round $r_x \geq r_2$ because it is connected to at least $4f+1$ nodes in $\mathcal{K}_0$. Thus, it delivers c in a round $r_y \geq r_2$ it is correct.

Such an argumentation extends to any process in a k-clique reachable through a sequence of adjacent k-cliques. □

Theorem 33. The RCMB protocol with $\tau = 2$ and $\sigma = (\tau+1)f$ achieves reliable communication in the unaware mobile Byzantine failure model in a network topology G where $n > 6f$ and $x(G) > 6f$.

Proof. The condition $x(G) > 6f$ allows to arrange nodes of a graph G in a *DKLO* with $k > 6f$ of two or more levels $[L_0, \cdots L_m]$ with respect to any vertex. Let us consider a correct source p_s that computes a content c at r_0 and sends it at round r_1.

Let us assume that the *DKLO* with respect to p_s is composed of 2 levels. It follows that all processes have a link with the source and that all the correct ones receive $\langle s, t, c\rangle$ at round r_1 directly from the source, thus they save it into their *delivered* set and relay it at r_2. Subsequently, the mobile Byzantine agents can move between r_1 and r_2. At round r_2 all correct processes are connected to at least $4f + 1 > \sigma$ processes that relays m. It follows they relay $\langle s, t, c\rangle$ at round r_3 and at all the subsequent rounds.

Let us assume that the *DKLO* with respect to p_s is composed by 3 or more levels. At round r_1 all correct processes in L_1 receive $\langle s, t, c\rangle$ directly from the source, thus they save it into their *delivered* set and they relay it at r_2. Subsequently, the mobile Byzantine agents can move between r_1 and r_2, and at round r_2 all correct processes in L_1 relay m to all nodes in L_2. Every process in L_2 has at least $6f + 1$ neighbors in L_1 and at least $4f + 1 > \sigma$ of them relay $\langle s, t, c\rangle$. It follows they all save and relay $\langle s, t, c\rangle$ at round r_3. Between rounds r_2 and r_3 the mobile Byzantine agents move and compromise further f processes. It follows that at round r_3 every process in levels L_1, L_2 and L_3 receives $\langle s, t, c\rangle$ from at least $3f+1 > \sigma$ processes, because each of them has at least $6f + 1$ neighbors inside the first three levels and at most $3f$ processes may have been compromised from the beginning of the transmission. It follows that all correct processes in the first three levels relay $\langle s, t, c\rangle$ at every round $r_i \geq r_4$. This reasoning extends considering more levels. □

Theorem 34. The `RC-Sasaki-et-al.` protocol with $\sigma = \beta f$ achieves reliable communication in the unaware mobile Byzantine failure model in networks $G(a, \beta)$ where the inequality $a > 2\beta f$ is satisfied [71].

Proof. Every reliable communication instance between a source process p_s and a destination process p_t lasts exactly β rounds in the `RC-Sasaki-et-al.` protocol. The inequality $a > 2\beta f$ guarantees that between every pair of processes there exist at least $2a + 1$ disjoint paths of length at most β. Any process can relay messages between peers p_s and p_t at only one defined round every β ones. It follows that the mobile Byzantine agents can compromise at most βf processes (and thus disjoint paths) in β rounds, and thus no correct process receives more than σ copies of a spurious content in a round. The assumption $a > 2\beta f$ guarantees instead that there always exist $\beta f + 1$ disjoint paths that are not compromised by Byzantine agents in every communication instance. □

Theorem 35. The `RCMB` protocol with $\tau = 1$ and $\sigma = (\tau + 1)f$ achieves reliable communication from a permanently correct source in the unaware mobile Byzantine failure model in networks where $h(G) > 4f$.

Proof. We verified the safety property of the `RCMB` algorithm with $\sigma = (\tau + 1)f$ in the unaware mobile Byzantine failure model in Theorem 30. We need to prove the liveness property of reliable communication employing the protocol in networks where $h(G) > 4f$ in case of a permanently correct source and $\tau = 1$.

The condition $h(G) > 4f$ allows to arrange the nodes of a network G in a *MKLO* with $k = 4f + 1$ with respect to every vertex of G.

Let us assume that process p_s sends a content c employing `RCMB` to process p_t at round r_0. Process p_t can either be in L_1 or in $L_{z>1}$. In the former case it receives $\langle s, t, c \rangle$ through a link from p_s starting from round $r_x \geq r_1$ it is correct, and thus it eventually delivers the content c. In the latter case, all correct processes in L_1 receive $\langle s, t, c \rangle$ at every round $r_x \geq r_1$. Thus, they queue $\langle s, t, c \rangle$ to be sent at every round $r_y \geq r_2$. At every round, there are at most f processes that can be faulty among all levels. It follows that at least $2f + 1$ processes in L_1 relay $\langle s, t, c \rangle$ to processes in L_2 at every round $r_y \geq r_2$, because f nodes may have been faulty at r_1 and f ones are faulty at r_2. Therefore, all correct processes in L_2 relays $\langle s, t, c \rangle$ to all of their neighbors at every round $r_z \geq r_3$, and if process p_t is in L_2 then it delivers m at $r_z \geq r_3$ when it is correct. The reasoning extends to any other level given the assumption of $h(G) > 4f$, and the claim follows. □

Aware failure model

Theorem 36. Reliable communication cannot be solved in the aware mobile Byzantine failure model with $n \leq 3f$.

Proof. The result can be deduced from the lower bound implementing the safe register abstraction in the aware mobile Byzantine failure model [18]. Let us consider a set of $3f$ processes connected through a complete communication network. Let us assume a correct source p_s that computes a content c at round r_0, that p_s sends it to all other processes at round r_1 and that p_t and other $f - 1$ processes are faulty

at r_1. Thus, p_t is faulty while the reliable communication protocol is diffusing $\langle s, t, c\rangle$ according to a distributed protocol $\mathcal{P}$. Subsequently, the mobile Byzantine agents moves on process p_s and on $f-1$ other processes between rounds r_1 and r_2. It follows that at round r_2 there are f processes that share a state that contains $\langle s, t, c\rangle$, f cured processes (i.e. with wiped local variables) and f faulty processes. Thus, it is not possible to distinguish which set of processes (the f faulty or the f not cured ones) is storing the content sent by the correct source. □

Theorem 37. The RCMB protocol with $\sigma = f$ guarantees safety of reliable communication in the aware mobile Byzantine failure model.

Proof. Let us consider a set of n process connected through a complete network. Let us assume, for the ease of contradiction, that a target process p_t has delivered a content $\hat{c}$ at round r_x from p_s but $\hat{c}$ has not been sent by its source (i.e. $\hat{c}$ is a spurious content).

The delivery of a content in RCMB is independent from the process local variables and it is only determined by the messages that are received in a round. The message $\langle s, t, \hat{c}\rangle$ has been received by no process through a link with the source process p_s according to our hypothesis. It follows there have been more than $\sigma = f$ processes that sent $\langle s, t, \hat{c}\rangle$ to p_t at round r_x. The mobile Byzantine agents can force f processes to send $\langle s, t, \hat{c}\rangle$ at round r_x. The correct processes at r_x that were faulty at r_{x-1} turn to the cured state, thus they wipe their local variables (and thus their *delivered* set) and remove any message previously queued for the submission. Any correct process $p_j \neq p_t$ sends $\langle s, t, \hat{c}\rangle$ at r_x only if either p_j has received such a message through a link with process p_s, or from more than f neighbors in a round. It follows that at most f processes in the system may concurrently send $\langle s, t, \hat{c}\rangle$. Thus content $\hat{c}$ has been sent by its source. This leads to a contradiction and the claim follows. □

Theorem 38. The RC-Sasaki-et-al. protocol with $\sigma = (\beta-1)f$ achieves reliable communication in the aware mobile Byzantine failure model in networks $G(a, \beta)$ where the inequality $a > (2\beta - 1)f$ is satisfied.

Proof. The cured processes remain silent because they drop every message previously queued for the submission. In the first round of a reliable communication instance, only the source is allowed to transmit. It follows that no process can diffuse spurious messages in such a round. Therefore, spurious contents can only traverse $(\beta - 1)f$ disjoint paths in a reliable communication instance. On the other hand, Byzantine agents can still compromise f processes per round, preventing peers from receiving and relaying messages, and thus up to βf ones may be compromised in every communication instance. The inequality follows considering that $(\beta - 1)f + 1$ copies of a content received in a single round are sufficient to ensure safety and that at most βf process can be compromised during a reliable communication instance. □

Theorem 39. The RCMB protocol with $\sigma = f$ achieves reliable communication in complete networks of size $n > 3f$ in the aware mobile Byzantine failure model.

Proof. We verified the safety property of the RCMB algorithm with $\sigma = f$ in the aware mobile Byzantine failure model in Theorem 37. We need to prove the liveness property of reliable communication in complete networks of size $n > 3f$.

Let us assume a correct source p_s that computes a content c at r_0 and sends it at r_1 to itself and all of its neighbors according to the RCMB algorithm. It follows that p_s and at least $2f$ processes queue $\langle s, t, c\rangle$ to be sent at r_2, because $\langle s, t, c\rangle$ has been received through a link from its source. At any round r_x there are at most f processes that are faulty and at most f ones that were faulty in r_{x-1}. Thus, all correct processes receive at least $f + 1 > \sigma$ copies of c from distinct nodes at any round $r_y \geq r_2$ and they relay it in the subsequent round. It follows that message $\langle s, t, c\rangle$ is relayed by at least $f + 1$ processes at any round $r_y \geq r_2$, and that process p_t delivers it in a round $r_z \geq r_2$ it is correct. □

Theorem 40. The RCMB protocol with $\sigma = f$ achieves reliable communication in the aware mobile Byzantine failure model in *i)* k-clique community networks with $k > 3f + 1$ and *ii)* in graphs where $x(G) > 5f$.

Proof. We verified the safety property of the RCMB algorithm with $\sigma = f$ in the unaware mobile Byzantine failure model in Theorem 37.

The liveness property in case of k-clique community networks with $k > 3f + 1$ or networks where $x(G) > 5f$ follows from the same argumentation provided respectivelly in Theorems 32 and 33 considering that σ is reduced to f. □

Theorem 41. The RCMB protocol with $\sigma = f$ achieves reliable communication from a permanent correct source in the aware mobile Byzantine failure model in networks where $h(G) > 3f$.

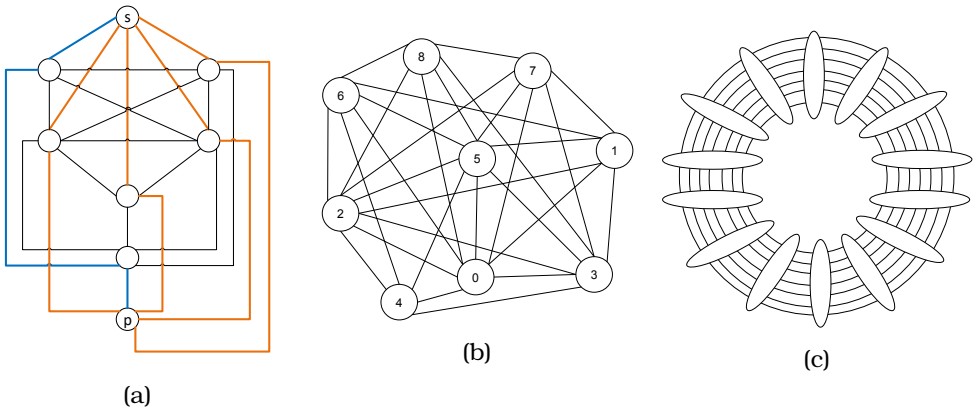
Proof. We verified the safety property of the RCMB algorithm with $\sigma = f$ in Theorem 37.

The liveness property in networks where $h(G) > 3f$ follows from the same argumentation provided in Theorem 35 considering that σ is reduced to f. □

Graph parameters comparison

We provide in this section some examples of the topology where the condition $a > 2\beta f$ by Sasaki et al. [71] is not satisfied, but the reliable communication problem remains solvable.

Theorems 32 and 40 identify the k-clique community as a topology where the reliable communication problem is solvable. There exist k-clique community graphs where $a \leq 2\beta f$ but $k > 4f + 1$, an example is depicted in Figure 14.3a: a 6-clique community graph. According with Theorem 32, it is possible to provide reliable communication tolerating one mobile Byzantine agents ($f = 1$) in such a topology (indeed, $k > 4f + 1 = 5$) considering the unaware failure model with RCMB. On the other hand, in such a graph $\beta = 3$ and $a = 5$, thus the inequality $a > 2\beta f$ is not satisfied for $f \geq 1$ and the algorithm by Sasaki et al. [71] does not guarantee reliable communication in such a network.



(a) (b) (c)

Figure 14.3: (a) 6-clique community example. (b) $\Psi(G) = 5$, $a > 2\beta f$ not satisfied with $f > 1$. (c) $\langle 7, 14 \rangle$-multipartite cycle.

Theorems 35 and 41 identify graphs where the parameter $h(G)$ is greater than certain values as topologies where the reliable communication problem is solvable from a permanent correct source. There exist topologies where $a \leq 2\beta f$ but $h(G) > 4f$, and an example is depicted in Figure 14.3b. According with Theorem 34, one mobile Byzantine agent ($f = 1$) cannot be tolerated by the algorithm by Sasaki et al [71], indeed $a = 5$ and $\beta = 3$. Instead, $\Psi(G) > 4f$ in such an example, allowing to achieve reliable communication against one mobile Byzantine agent with algorithm `RCMB`.

The conditions defined in Theorems 33 and 40 identify new topologies where it is possible to solve the reliable communication problem. Specifically, there exist topologies where $a \leq 2\beta f$ but $x(G) > 6f$. An example is depicted in Figure 14.3c: a $\langle 7, 14 \rangle$-multipartite cycle. In such a network, $X(G) = 7$, $a = 14$ and $\beta = 7$. According with Theorem 33 it is possible to achieve reliable communication against one mobile Byzantine agents (indeed, $X(G) > 6f$) with Algorithm `RCMB`. On the other hand the inequality $a > 2\beta f$ is not satisfied in such a topology, so the algorithm by Sasaki et al. [71] cannot guarantee reliable communication in such a setting.

14.5 Conclusion

We analyzed in this chapter the reliable communication problem in static distributed systems affected by Mobile Byzantine Faults. We highlighted the specific difficulties that arise when considering mobile malicious agents able to move in the system and to continuously compromise nodes. We showed that the reliable communication problem arises even in complete communication networks and that it is not possible to address it in an asynchronous system. Then, starting from the only solution available in the literature, the one proposed by Sasaki et al. [71], we identified additional settings where the problem is solvable. Subsequently, we defined a new solution to the reliable communication protocol, `RCMB`, and we identified new multi-hop topologies where reliable communication primitives remain

feasible.

These contributions pave the way toward deeper analyzes of reliable communication and other related distributed system problems with mobile Byzantine faults in multi-hop networks. A particularly interesting question is the feasibility of tolerating both mobile Byzantine failures and self-stabilization (as in the register construction of Bonomi et al. [20]) for the purpose of reliable communication. To our knowledge, this problem was only shown solvable by Maurer et al. [57] for the static Byzantine case.

Chapter 15

Conclusion

In this thesis we investigated the Byzantine-tolerant reliable communication problem. Namely, we sought for distributed solutions enabling correct message exchanges between all processes, guaranteeing authorship, integrity, and delivery of the informations exchanged.

Although several solutions to the problem are available in the literature, some of them cannot realistically be employed due to the prohibitive amount of messages they need to spread, others require specific topological condition that are computationally hard to evaluate.

The contributions of this thesis are threefold. In the static network scenario, we proposed modifications and improvements to the state-of-art protocol for unknown networks, `DolevU`, reducing the amount of messages to exchange solving the reliable communication problem, in the globally bounded Byzantine failure model, from factorial to nearly quadratic in several synchronous systems.
Nevertheless, the achieved performances are partially due to the considered synchrony assumption, guaranteeing progress in the computation. Moreover, the computational complexity of the procedure that verifies the validity of the contents exchanged is unmodified with respect to the base protocol `DolevU`, and requires the resolution of a NP-Complete problem.
We therefore proposed an alternative approach: leveraging a limited number of computational expensive reliable communication instances (like `BFT`) to setup an efficient primitive. Specifically, we employed a solution to the Byzantine tolerant topology reconstruction problem to partial infer the structure of the communication network and then setup an optimal primitive. Also, we defined a cryptographic prototype of an efficient authenticated reliable communication protocol, that initially enables all processes to generate and distribute cryptographic keys supporting a digital signature primitive, and then guarantees reliable and efficient content exchanges.
In the dynamic network scenario, there was only one contribution in the literature to the Byzantine tolerant reliable communication problem in dynamic distributed systems. Despite it precisely characterizes the conditions enabling reliable communication in the globally bounded failure model, the conditions provided are not "general" and need to be evaluated for every content exchange between two pro-

cesses. We therefore defined an alternative network condition through the extension of a known class of dynamic networks, the 1-interval connected. The defined condition enables reliable communication between all processes at any time (i.e., it guarantees solvability of the any-to-any reliable communication problem) in a synchronous dynamic distributed system.
Moreover, we extended the `CPA` protocol to dynamic distributed systems. In more details, we considered the locally bounded Byzantine failure model and we defined `DynCPA`, solving the reliable communication problem in dynamic distributed systems. We then characterized the necessary and sufficient conditions enabling one-to-all reliable communication in the assumed failure model, and we provided alternative network conditions, on 1-interval connected topologies, guaranteeing the solvability of the any-to-any reliable communication problem with `DynCPA`.

Finally, the reliable communication problem was analyzed considering the mobile Byzantine failure model in static distributed systems. We provided an alternative problem specification that considers the potential variable state of all processes (between correct and faulty), and after demonstrating the impossibility of solving the problem in an asynchronous system, we characterized the necessary conditions enabling reliable communication in synchronous systems, and we identified several network topologies where the reliable communication primitive can be implemented.

Following from this work, there are still several open problems that could be worth investigating in the short term and long term future. Among the former, we showed the optimality of `DolevR` in the number of message exchanges solving the one-to-one reliable communication problem, precisely that is linear in the size n of the system. Nevertheless, it is still not clear whether the one-to-all reliable communication problem (where a source processes aims to exchange a content with all others) is only solvable with a protocol having $O(n^2)$ message complexity or more efficient solutions are definable. Moreover, in our analysis of the reliable communication problem in dynamic distributed systems, we consider synchronous systems in order to guarantee that some evolutions of the communication network are not lost by the latency of the processes. It would be interesting to identify network conditions (if possible) supporting reliable communication even in asynchronous dynamic systems (further the ones that considers repeated availability of all links). Finally in the long term, it could be worth addressing the reliable communication problem even in dynamic distributed system with mobile Byzantine faults. Despite the problem may have interesting margin of investigation, given that the both dynamic networks and dynamic failures require more complex problems to be addressed, and even stronger network assumptions, it could be worth investigating the possibility to move to weaker or probabilistic specifications that may better address the problem despite the high dynamicity of the system.

Bibliography

[1] Communication, https://www.merriam-webster.com/dictionary/communication.

[2] Jiří Adámek and Václav Koubek. Remarks on flows in network with short paths. *Commentationes Mathematicae Universitatis Carolinae*, 12(4):661–667, 1971.

[3] Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondrej Pangrác, Heiko Schilling, and Martin Skutella. Length-bounded cuts and flows. *ACM Trans. Algorithms*, 7(1):4:1–4:27, 2010.

[4] Roberto Baldoni, Silvia Bonomi, Leonardo Querzoni, and Sara Tucci Piergiovanni. Investigating the existence and the regularity of logarithmic harary graphs. *Theor. Comput. Sci.*, 410(21-23):2110–2121, 2009.

[5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[6] Albert-László Barabási et al. *Network science*. Cambridge university press, 2016.

[7] Vartika Bhandari and Nitin H Vaidya. Implementing a reliable local broadcast primitive in wireless ad hoc networks. https://disc.georgetown.domains/publications/rbcast-tech.pdf.

[8] Vartika Bhandari and Nitin H. Vaidya. On reliable broadcast in a radio network. In Marcos Kawazoe Aguilera and James Aspnes, editors, *Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing, PODC 2005, Las Vegas, NV, USA, July 17-20, 2005*, pages 138–147. ACM, 2005.

[9] Béla Bollobás. *Random Graphs, Second Edition*, volume 73 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 2011.

[10] François Bonnet, Xavier Défago, Thanh Dang Nguyen, and Maria Potop-Butucaru. Tight bound on mobile byzantine agreement. *Theor. Comput. Sci.*, 609:361–373, 2016.

[11] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast made practical. In *8th Latin-American Symposium on Dependable Computing, LADC 2018, Foz do Iguaçu, Brazil, October 8-10, 2018*, pages 155–160. IEEE, 2018.

[12] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Reliable broadcast in dynamic networks with locally bounded byzantine failures. In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, volume 11201 of *Lecture Notes in Computer Science*, pages 170–185. Springer, 2018.

[13] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast made practical simulation code `https://github.com/giovannifarina/BFT-ReliableCommunication`, July 2019.

[14] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast with honest dealer made practical. *J. Braz. Comput. Soc.*, 25(1):9:1–9:23, 2019.

[15] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Boosting the efficiency of byzantine-tolerant reliable communication. In Stéphane Devismes and Neeraj Mittal, editors, *Stabilization, Safety, and Security of Distributed Systems - 22nd International Symposium, SSS 2020, Austin, TX, USA, November 18-21, 2020, Proceedings*, volume 12514 of *Lecture Notes in Computer Science*, pages 29–44. Springer, 2020.

[16] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Broadcasting information in multi-hop networks prone to mobile byzantine faults. In *Networked Systems - 8th International Conference, NETYS 2020, Marrakech, Morocco, June 03-05, 2020, Revised Selected Papers*, Lecture Notes in Computer Science, 2020.

[17] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Une méthode efficace pour éviter la propagation des fake news. In *ALGOTEL 2020-22èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, 2020.

[18] Silvia Bonomi, Antonella Del Pozzo, and Maria Potop-Butucaru. Optimal self-stabilizing synchronous mobile byzantine-tolerant atomic register. *Theor. Comput. Sci.*, 709:64–79, 2018.

[19] Silvia Bonomi, Antonella Del Pozzo, Maria Potop-Butucaru, and Sébastien Tixeuil. Optimal storage under unsynchronized mobile byzantine faults. In *36th IEEE Symposium on Reliable Distributed Systems, SRDS 2017, Hong Kong, Hong Kong, September 26-29, 2017*, pages 154–163. IEEE Computer Society, 2017.

[20] Silvia Bonomi, Antonella Del Pozzo, Maria Potop-Butucaru, and Sébastien Tixeuil. Brief announcement: Optimal self-stabilizing mobile byzantine-tolerant regular register with bounded timestamps. In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, volume 11201 of *Lecture Notes in Computer Science*, pages 398–403. Springer, 2018.

[21] Silvia Bonomi, Antonella Del Pozzo, Maria Potop-Butucaru, and Sébastien Tixeuil. Approximate agreement under mobile byzantine faults. *Theor. Comput. Sci.*, 758:17–29, 2019.

[22] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, 1987.

[23] Harry Buhrman, Juan A. Garay, and Jaap-Henk Hoepman. Optimal resiliency against mobile faults. In *Digest of Papers: FTCS-25, The Twenty-Fifth International Symposium on Fault-Tolerant Computing, Pasadena, California, USA, June 27-30, 1995*, pages 83–88. IEEE Computer Society, 1995.

[24] Christian Cachin, Rachid Guerraoui, and Luís E. T. Rodrigues. *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer, 2011.

[25] Arnaud Casteigts. *A Journey through Dynamic Networks (with Excursions)*. 2018.

[26] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distributed Syst.*, 27(5):387–408, 2012.

[27] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[28] Reinhard Diestel. *Graph Theory*. Springer Berlin Heidelberg, 2017.

[29] Efim A Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Math. Doklady*, volume 11, pages 1277–1280, 1970.

[30] Danny Dolev. Unanimity in an unknown and unreliable environment. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 159–168. IEEE Computer Society, 1981. DOI: 10.1109/SFCS.1981.53.

[31] Danny Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982. DOI : 10.1016/0196-6774(82)90004-9.

[32] Shlomi Dolev, Omri Liba, and Elad Michael Schiller. Self-stabilizing byzantine resilient topology discovery and message delivery. *CoRR*, abs/1208.5620, 2012.

[33] Shlomi Dolev, Omri Liba, and Elad Michael Schiller. Self-stabilizing byzantine resilient topology discovery and message delivery - (extended abstract). In *Networked Systems - First International Conference, NETYS 2013, Marrakech, Morocco, May 2-4, 2013, Revised Selected Papers*, pages 42–57, 2013.

[34] John R. Douceur. The sybil attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, pages 251–260, 2002.

[35] Vadim Drabkin, Roy Friedman, and Marc Segal. Efficient byzantine broadcast in wireless ad-hoc networks. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pages 160–169. IEEE, 2005.

[36] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[37] Afonso Ferreira. Building a reference combinatorial model for manets. *IEEE Netw.*, 18(5):24–29, 2004.

[38] Andrew Gainer-Dewar and Paola Vera-Licona. The minimal hitting set generation problem: Algorithms and computation. *SIAM J. Discrete Math.*, 31(1):63–100, 2017.

[39] Juan A. Garay. Reaching (and maintaining) agreement in the presence of mobile faults (extended abstract). In Gerard Tel and Paul M. B. Vitányi, editors, *Distributed Algorithms, 8th International Workshop, WDAG '94, Terschelling, The Netherlands, September 29 - October 1, 1994, Proceedings*, volume 857 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 1994.

[40] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. ISBN : 0716710455.

[41] Petr A. Golovach and Dimitrios M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discret. Optim.*, 8(1):72–86, 2011.

[42] D Frank Hsu. On container width and length in graphs, groups, and networks–dedicated to professor paul erdös on the occasion of his 80th birthday–. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 77(4):668–680, 1994.

[43] Akira Ichimura and Maiko Shigeno. A new parameter for a broadcast algorithm with locally bounded byzantine faults. *Inf. Process. Lett.*, 110(12-13):514–517, 2010.

[44] Alon Itai, Yehoshua Perl, and Yossi Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3):277–286, 1982.

[45] Thomas Jech. *Set theory*. Springer Science & Business Media, 2013.

[46] Kate Jenkins and Alan J. Demers. Logarithmic harary graphs. In *21st International Conference on Distributed Computing Systems Workshops (ICDCS 2001 Workshops), 16-19 April 2001, Phoenix, AZ, USA, Proceedings*, pages 43–50. IEEE Computer Society, 2001.

[47] David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *J. Comput. Syst. Sci.*, 64(4):820–842, 2002.

[48] Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H. Vaidya. Exact byzantine consensus on undirected graphs under local broadcast model. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 327–336. ACM, 2019.

[49] Jeong Han Kim and Van H. Vu. Generating random regular graphs. *Comb.*, 26(6):683–708, 2006.

[50] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In Soma Chaudhuri and Shay Kutten, editors, *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 275–282. ACM, 2004.

[51] Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 513–522. ACM, 2010.

[52] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[53] Omri Liba. Errata corrige of [63]. `http://antares.cs.kent.edu/~mikhail/Research/topology.errata.html`.

[54] Chris Litsas, Aris Pagourtzis, and Dimitris Sakavalas. A graph parameter that matches the resilience of the certified propagation algorithm. In Jacek Cichon, Maciej Gebala, and Marek Klonowski, editors, *Ad-hoc, Mobile, and Wireless Network - 12th International Conference, ADHOC-NOW 2013, Wrocław, Poland, July 8-10, 2013. Proceedings*, volume 7960 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2013.

[55] L. Lovász, V. Neumann-Lara, and M. Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9(4):269–276, 1978.

[56] Alexandre Maurer and Sébastien Tixeuil. Byzantine broadcast with fixed disjoint paths. *J. Parallel Distributed Comput.*, 74(11):3153–3160, 2014.

[57] Alexandre Maurer and Sébastien Tixeuil. Self-stabilizing byzantine broadcast. In *33rd IEEE International Symposium on Reliable Distributed Systems, SRDS 2014, Nara, Japan, October 6-9, 2014*, pages 152–160. IEEE Computer Society, 2014.

[58] Alexandre Maurer and Sébastien Tixeuil. Containing byzantine failures with control zones. *IEEE Trans. Parallel Distributed Syst.*, 26(2):362–370, 2015.

[59] Alexandre Maurer and Sébastien Tixeuil. Tolerating random byzantine failures in an unbounded network. *Parallel Process. Lett.*, 26(1):1650003:1–1650003:12, 2016.

[60] Alexandre Maurer, Sébastien Tixeuil, and Xavier Défago. Communicating reliably in multihop dynamic networks despite byzantine failures. In *34th IEEE Symposium on Reliable Distributed Systems, SRDS 2015, Montreal, QC, Canada, September 28 - October 1, 2015*, pages 238–245. IEEE Computer Society, 2015.

[61] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[62] Keisuke Murakami and Takeaki Uno. Efficient algorithms for dualizing large-scale hypergraphs. *Discrete Applied Mathematics*, 170:83–94, 2014.

[63] Mikhail Nesterenko and Sébastien Tixeuil. Discovering network topology in the presence of byzantine faults. *IEEE Trans. Parallel Distrib. Syst.*, 20(12):1777–1789, 2009.

[64] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In Luigi Logrippo, editor, *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 19-21, 1991*, pages 51–59. ACM, 1991.

[65] Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. Reliable broadcast with respect to topology knowledge. *Distributed Comput.*, 30(2):87–102, 2017.

[66] Andrzej Pelc. Reliable communication in networks with byzantine link failures. *Networks*, 22(5):441–459, 1992.

[67] Andrzej Pelc and David Peleg. Broadcasting with locally bounded byzantine faults. *Inf. Process. Lett.*, 93(3):109–115, 2005.

[68] Rüdiger Reischuk. A new solution for the byzantine generals problem. *Information and Control*, 64(1-3):23–42, 1985.

[69] Ben Roberts and Dirk P. Kroese. Estimating the number of s-t paths in a graph. *J. Graph Algorithms Appl.*, 11(1):195–214, 2007.

[70] Dimitris Sakavalas and Lewis Tseng. Delivery delay and mobile faults. In *17th IEEE International Symposium on Network Computing and Applications, NCA 2018, Cambridge, MA, USA, November 1-3, 2018*, pages 1–8. IEEE, 2018.

[71] Toru Sasaki, Yukiko Yamauchi, Shuji Kijima, and Masafumi Yamashita. Mobile byzantine agreement on arbitrary network. In *Principles of Distributed Systems - 17th International Conference, OPODIS 2013, Nice, France, December 16-18, 2013. Proceedings*, pages 236–250, 2013.

[72] William Stallings. *Cryptography and network security - principles and practice (3. ed.)*. Prentice Hall, 2003.

[73] Angelika Steger and Nicholas C. Wormald. Generating random regular graphs quickly. *Comb. Probab. Comput.*, 8(4):377–396, 1999.

[74] Lewis Tseng. Towards reliable broadcast in practical sensor networks. In Aris Gkoulalas-Divanis, Miguel P. Correia, and Dimiter R. Avresky, editors, *16th IEEE International Symposium on Network Computing and Applications, NCA 2017, Cambridge, MA, USA, October 30 - November 1, 2017*, pages 45–52. IEEE Computer Society, 2017.

[75] Lewis Tseng, Nitin H. Vaidya, and Vartika Bhandari. Broadcast using certified propagation algorithm in presence of byzantine faults. *Inf. Process. Lett.*, 115(4):512–514, 2015.

[76] Lewis Tseng, Yingjian Wu, Haochen Pan, Moayad Aloqaily, and Azzedine Boukerche. Reliable broadcast with trusted nodes: Energy reduction, resilience, and speed. *Computer Networks*, 182:107486, 2020.

[77] Junming Xu. *Topological structure and analysis of interconnection networks*, volume 7. Springer Science & Business Media, 2013.

[78] Kai Zeng, Kannan Govindan, and Prasant Mohapatra. Non-cryptographic authentication and identification in wireless networks. *IEEE Wireless Commun.*, 17(5):56–62, 2010.