NATIONAL TECHNICAL UNIVERSITY OF ATHENS

DOCTORAL THESIS

Reliable Communication Despite Limited Knowledge

Author:
Dimitris SAKAVALAS

Advisor:
Aris PAGOURTZIS

Computation and Reasoning Laboratory
School of Electrical and Computer Engineering

Athens, July 2016

Abstract

As communication networks grow in size, they become increasingly vulnerable to component failures. These networks consist of numerous interacting entities (agents). Since distributed systems have become popular and widely used in contemporary networking, the provided solutions need to cope with erroneous and malicious components in the underlying communication network. Security and reliability issues that arise have been objects of extensive research in the fields of Secure Multiparty Computations and Distributed Computing. In our work we contribute to the realization of fundamental communication primitives (Reliable Broadcast and Reliable Message Transmission) in an adversarial distributed environment, by investigating the impact of the network structure and the agents' topology knowledge level on the achievability of these tasks. We consider a worst-case (Byzantine) adversary, which makes the agents misbehave arbitrarily,

Initially, we consider the t-locally bounded adversary model, introduced in 2004 by Koo, where a fixed upper bound on the number of corruptions in each agent's neighborhood is imposed. We explore the tradeoff between the level of topology knowledge and the solvability of the problem by developing a versatile technique which allows us to obtain impossibility results for every level of topology knowledge. Checking the necessary conditions for the solvability of the problem proves to be NP-hard but along the way we obtain an efficient 2-approximation algorithm for the ad hoc case (where agents only know their local neighborhood). On the positive, we generalize the algorithmic idea behind the simple, yet powerful Certified Propagation Algorithm (CPA), also introduced by Koo in 2004, and propose algorithms, that match the obtained bounds (unique algorithms) in every case. Thus, we exactly characterize the classes of graphs in which reliable communication is possible with respect to topology knowledge. In order to achieve these we introduced the Partial Knowledge Model in which each agent knows a part of the network, namely a connected subgraph containing itself. As a part of the latter contribution, we manage to settle an open question of Pelc and Peleg (2005) in the affirmative, by showing that in ad hoc networks, CPA is unique, that is, it can tolerate as many local corruptions as any other Broadcast algorithm.

Furthermore, we manage to generalize our results in the General Adversary model of Hirt and Maurer (1997), which subsumes earlier models by adapting our techniques and algorithms from the t-locally bounded model. Thus, we devise the first optimally resilient algorithms for Reliable Broadcast/Message transmission under restricted knowledge and general adversaries. We also study the efficiency of RMT protocols by introducing an algorithmic property which implies that a protocol scheme is as efficient as any other for a certain problem with respect to polynomial time. To obtain our latter results we employ, among others, a novel notion of joining operation on adversary structures, appropriate notions of separators in unreliable networks, and a self-reducibility property of the RMT problem.

Finally, we study energy-efficient Broadcast in wireless networks, where simultaneous transmissions lead to signal interference which prevents message propagation. In particular, we examine the k-shot wireless network model, in which a bound k on the number of transmissions for every player is given. We prove a lower bound on the Broadcast time of any protocol.

Keywords: reliable broadcast, reliable message transmission, byzantine adversary, partial knowledge, general adversary, incomplete networks, ad hoc networks, distributed computing, topology knowledge, wireless networks, energy efficiency, k-shot Broadcast

Preface

The current thesis involves studies which resulted from my cooperation with Christos Litsas, Aris Pagourtzis and Giorgos Panagiotakos.

The rapid growth of communication networks and the plethora of accompanying applications constantly bring up new needs and challenges. Communication networks consist of numerous interacting entities. These entities often wish to collaborate in order to achieve certain tasks, which vary from basic ones, for example the distribution of digital content and common decision making, to more involved ones, which build on the former ones; such an example is electronic voting. While it is generally expected that the entities act decently, following some generally respected rules, ethics and agreements, faulty or malicious parties may exist in the network, wishing to violate these common rules or laws in order to serve their own goals against the interests of benign participants. Such considerations put forth the need for securing distributed computing environments.

Real-life applications involve networks of particularly complex structure. Therefore the need for strong theoretical support for reliable communication between parts of the network increases. It is often the case that research has focused on simplified network structures and strong knowledge assumptions for impossibility and feasibility results to be obtained. In our line of work, we explicitly consider the network structure and the a priori knowledge that the interacting entities possess about the network and study how this parameters affect the correctness of information-exchange procedures. We contribute in the identification of the minimal structural and knowledge-related demands, which render security and reliability issues solvable. Our results can be applied in the design of networks that can optimally support the usage of reliable communication protocols. Another practical benefit of our work is that, using our techniques, one can exactly determine the worst fault situations that can be tolerated in existing network infrastructures. The above studies can be applied in mission- critical applications, such as flight control systems, control systems in nuclear power plants and military operations where the communication infrastructure must be able to cope with failures (or even worse malicious corruption) of some devices. Naturally low-memory/computing power devices such as wireless sensors are used is these circumstances and thus minimal connectivity and knowledge requirements are imposed.

The problems of Reliable Broadcast and Reliable Message Transmission, extensively studied in our work, constitute fundamental building blocks for achieving more complex distributed tasks in weakly connected networks and restricted participant knowledge models. The motivation for partial knowledge considerations comes from large scale networks (e.g. the Internet), where topologically local estimation of corruption patterns may be possible, while global estimation may be hard to obtain due to geographical or jurisdiction constraints. Additionally, proximity in social networks is often correlated with an increased amount of available information, further justifying the relevance of the model. Our introduction of a robust model for restricted knowledge, contributes to the foundations of a more realistic modeling regarding modern communication networks and their reliability.

The emergence of social networking, electronic commerce and electronic voting can potentially have a massive impact in ensuring economic and social prosperity in an open and interconnected digital world. In the course of achieving this aim public trust should be built by guaranteeing the reliability of the procedures and the participants' protection from malicious behaviors. These issues can only be settled by rigorous theoretical analysis in the context of related fields such as Distributed Computing and Cryptography. Ideally the trustworthiness of such electronic services will be established and will lead to increased participation of people regardless of financial and social factors, since access in those procedures can be highly affordable. The latter can result in direct-democratic procedures governed by principles of equality and justice.

Finally, our research can be interpreted as a study on knowledge propagation in the framework where communication between participants increases the amount of information that each one holds. By explicitly studying structural and a priori knowledge variations, we provide bounds on knowledge propagation in different models. Relaxing the knowledge notion in a probabilistic fashion, one can use similar techniques to obtain analogous results on belief or opinion propagation. Among others, this would provide an approach to detect minimal crucial components of communication networks, the control of which would result in opinion and belief dissemination. The above could apparently be employed in advertising strategies but on the other hand it could be used for securing networks against false belief propagation.

Contents

List of Figures

Chapter 1

Introduction

1.1 Distributed systems fundamentals

Distributed computing environments model the situation where several interacting entities, hereafter referred as *players*, cooperate to achieve a common goal in the absence of a central authority. In the message passing framework, we assume that players communicate by sending messages over communication channels in order to cooperate. The player set and the pattern of connections provided by the communication channels describes the directed communication network $G = (V, E)$ where the node set $V = \{v_1, \ldots, v_n\}$ is identified with the player set the edge set E contains a directed edge (v_i, v_j) if and only if there is a communication channel between the players v_i, v_j through which v_i can send messages to v_j. In the following we will use the terms players and nodes interchangeably since the notions coincide in the present study. We denote the set of all possible graphs with $\mathcal{G}$ and the nodes of graph G with $V(G)$. Observe that through the definition of V each player can be considered to be assigned a unique identifier v_i. With $\mathcal{V}$ we denote the space of all possible node identifiers. That is, the assignment of ids is fixed and is given by the description of the network. Different assignments of ids in the same graph have been considered in the literature to address issues which depend expressly on the id assignment. However these issues can also be addressed through our approach by considering families of isomorphic graphs instead of different id assignments.

1.1.1 Initial knowledge of players

As is usual in the distributed computing literature, we assume that each player, due to its participation in the communication network, has some a priori local knowledge (e.g., about the network structure).

Specifically, we assume that each player knows its id and the ids of its neighbors in the network. This assumption is natural and realistic, because even if a player has no knowledge at all, communicating with all its neighbors can obtain some basic local information regarding the set of players with which it can communicate. Variations of a more refined model are considered in [48] and [1] where each node-player has a number of ports, i.e., external connection points and every communication channel connects two ports at adjacent nodes. In that approach, a processor sends a message to its neighbor by loading it onto the appropriate port. Although only knowledge of the distinct ports is assumed in that model it actually coincides with the knowledge of the neighbors' ids if the id assignment is fixed, thus in this case our knowledge assumptions are as weak as these of [48, 1].

Having assumed the lowest level of knowledge as described above, in the following, when further initial knowledge is assumed, e.g., additional topological knowledge, it will be clearly stated in the description of the specific knowledge model studied. We assume that the initial knowledge of every player (including its id and the ids of its neighbors) is provided to it as a part of its initial input information (see below)[1]. Whenever the distinction between of the knowledge and the input value of a player is necessary we will refer to them as *initial knowledge* and *initial input value* respectively. These two components constitute the input of a player.

1.1.2 Inputs and outputs

In order to formalize the concept of a problem in the distributed setting, one should primarily address the notions of input and output of players. Each player is assumed to provide input and output values depending on some real-world concerns. In [25] it is assumed that each player consists of two entities, a process which performs operations on the messages communicated and an agent which provides input to and receives output from the corresponding process. Intuitively one might think of processes as computers, and of agents as the humans using the computers.

Although the latter distinctively captures the intuition, for ease of exposition, in our study, given a communication graph G, we assume

[1]From an algorithmic point of view, this means that the initial knowledge of the players in the context of a specific knowledge model, can be modeled as a part of their input in any algorithm considered in the specific model.

the existence of a virtual node $v_e \notin V(G)$ which we call the *environment node* and is connected with all players in $V(G)$ through both incoming and outgoing edges. In our model the inputs of a player v are modeled as messages sent by v_e to v and the outputs of v as messages sent from v to v_e. The graph which results from G by the addition of environment node v_e and all the edges connecting v_e with $V(G)$ will be denoted by G^*, i.e., given the communication graph G we define,

$$G^* = (V^*, E^*), \text{ with } V^* = V(G) \cup \{v_e\}, E^* = E \cup \{(e, v), (v, e) \mid \forall v \in V(G)\}$$

The environment node v_e and the graph G^* will not be explicitly included in the study of the corresponding distributed system. Instead G^* is a theoretical construction which is used to model the inputs and output of players.

With X (e.g. $\{0,1\}^*$) we denote the message space which also includes the input and the output values; let the element $\epsilon \in X$ denote the empty value (indicates abscence of message, input or output). Finally with F we will denote the family of all functions $f : V(G) \to X$.

1.1.3 Player interactions and synchrony

Let $G = (V, E)$ be a communication network of players. For a player $v \in V$ denote with $\mathcal{N}_G^+(v) = \{u \in V \mid (v, u) \in E\}$ the set of outgoing neighbors, with $\mathcal{N}_G^-(v) = \{u \in V \mid (u, v) \in E\}$ the set of incoming neighbors of v and with $\mathcal{N}_G(v) = \mathcal{N}_G^-(v) \cup \mathcal{N}_G^+(v)$ the set of all neighbors. When the graph G is clearly implied by the context we will omit the subscript G from the neighborhood notation.

Interaction history

We define the *interaction history sequence* which is a full description of the messages exchanged between players, including their inputs and outputs (as messages exchanged with node v_e). The *interaction history* sequence, is a finite sequence of function pairs,

$$(h_i)_{i \in \{1,\ldots,k\}} = h_1, \ldots, h_k = (S_1, R_1), \ldots (S_k, R_k)$$

where

$$S_i : E^* \to X \text{ (Assignment of sent-messages to edges).}$$
$$R_i : E^* \to X \text{ (Assignment of received-messages to edges).}$$

We will also denote sequences $(h_i)_{i\in\{1,\dots,k\}}$ simply by h when it is clear by the context that h is a sequence. The interaction history sequence naturally describes the succession of events (sent or received messages) happening in distributed environment. Obviously, such a succession can only be defined if we assume an external source of “real time" that in general is not directly observable by the processors (cf. [24]). For the sake of convenience, we assume that real time is represented by the sequence of natural numbers. The existence of the two different assignments of sent and received messages to an edge can be justified by the delays on the delivery time of a message or even by the loss of messages due to channel malfunction as considered in the description of the synchronous model in [39].

Input/Output uniqueness

As has been explained before, an assignment of a value/message x to the edge (v_e, w) indicates that x is an input value of player w, whereas a value assignment to the edge (w, v_e) indicates an output of player w. In our model we assume single input and output values for an interaction history of the distributed system, namely, a player will receive at most one input value in the beginning of the interaction history and will output at most one value during the interaction history. Technically, we require that,

$$\text{(Uniqueness of input)}\ \ \forall v \in V(G), \forall i \in \{2,\dots,k\}, R_i((v_e, v)) = \epsilon$$
$$\text{(Uniqueness of output)}\ \ \forall v \in V(G), |\{i \in \{1,\dots,k\} : S_i((v_e, v)) \neq \epsilon\}| \leq 1.$$

Finally denote with $\mathcal{H}$ the space of all possible interaction history sequences which is obviously a function of $\mathcal{G}$ and X.

In our model single input/output values are assumed. More general models (as those used to describe the Multiparty Computation problem introduced in [56]) considering multiple input [2] and output values

[2]Differences between the single and multiple input case are clarified in [3]

can be easily expressed in our model by omitting the input/output uniqueness assumptions.

Time and synchrony

Having assumed an external source of “real time" we may also assume that each player has access to a non decreasing function of real time called *clock*, necessary for the player to observe any succession of events. The time period between two consecutive *clock pulses* $(clock(i), clock(i+1))$ is often referred to as a *clock cycle*. Important notions associated with timing are the synchrony between the players' clocks and the existence of bounds on message delivery delay. Variations of these parameters is necessary to consider, even in the level of abstraction where no computational model has been defined, in order to define some basic notions. As is common in the literature, we will focus in the following two extreme models regarding synchrony:

Synchronous model. In the *synchronous model* we assume that all players have access to the same *global clock* (or else that all clocks are completely synchronized) and the fact that messages sent through a channel in a clock cycle will arrive at the destination in the same clock cycle. An equivalent assumption is that all channels delays are bounded by a known constant time. In this model the clock cycle is called a *round* or *step*. Since all players have access to a global clock, the common knowledge of time can be used by the players to deduce some information. Considering the above, one can observe that in the synchronous model, the sequence term h_i refers to the events (sending, reception of messages, inputs and outputs) that have occurred during round i. Since all messages sent in round i arrive at their destination in the same round, it holds that $S_i = R_i, \forall i \in \{1, \ldots, k\}$.

Asynchronous model. In contrast, in the asynchronous model, no synchrony between players' clocks is assumed. More importantly, the delivery delay of messages is assumed to be finite but no known time bound is assumed on it. That is a message sent through a channel will arrive at its destination within some finite but unpredictable time. The latter also implies that players' clocks are rather useless, at least as far as communication is concerned (as analyzed in [48]). The above suggest that the only additional information that the players can deduce will be due to the events they observe and their succession; because of this, algorithms in this model are said to be *event driven*. As in [48], for a given interaction history h, we define the *asynchronous round* to be the time period equal to the maximum message delivery delay that occurs in h. This definition implies that each message incurs a delay of at most one asynchronous round and thus is compatible with the definition of round in the synchronous model.

Interaction view of a player

Given an interaction history h, in order to determine the part of h of which a player v is aware we need to define the following notions:

Let $E_v^+ = \{(v,u) \mid (v,u) \in E^*\}$ the set of outgoing from v edges and $E_v^- = \{(u,v) \mid (u,v) \in E^*\}$ the set of incoming to v edges in graph G^*. Given the interaction history $(h_i)_{i\in\{1,\dots,k\}}$ we define[3] $S_i^v = S_i|_{E_v^+}, R_i^v = R_i|_{E_v^-}$, i.e., the messages that player v sends and receives including its input and output as messages exchanged with the environment node v_e. The interaction view of a player is actually the part of the interaction history of which the player is aware of. There is a clear distinction on this notion regarding the synchronous and asynchronous model.

Interaction view in the synchronous model. Note that in the synchronous model, even if no message has been received by v in some round i, v actually knows that it has received and sent nothing (empty value ϵ) in round i. Therefore, given an interaction history $(h_i)_{i\in\{1,\dots,k\}}$, the *interaction view of player v* in the synchronous model, can be defined as:

$$(h_i^v)_{i\in\{1,\dots,k\}} = (S_1^v, R_1^v), \dots, (S_k^v, R_k^v)$$

Interaction view in the asynchronous model. In the asynchronous model, a player can only observe that some messages have been received (or sent) after others. The only (useful) notion of time in this model, can be extracted from the succession of events observed by the players. Therefore the interaction view of a player v will be a subsequence of (h_i^v) which constitutes only of the terms that v receives or

[3]Where $h|_A$ is the restriction of function h to the domain A.

sends a (non-empty) message. Namely, we define the set of indexes:

$$I_v = \Big\{ i \in \{1, \ldots, k\} : \Big(\exists (v,u) \in E \text{ s.t. } S_j((v,u)) \neq \epsilon \Big) \vee \\ \vee \Big(\exists (u,v) \in E \text{ s.t. } R_j((u,v)) \neq \epsilon \Big) \Big\}$$

The interaction view of a player in the asynchronous model is defined to be the subsequence $(h_i^v)_{i \in I_v}$ of $(h_i^v)_{i \in \{1,\ldots,k\}}$.

Observe that this sequence is defined only if $v \in V(G^*)$, where G^* is the graph implied by the interaction history sequence h. For each interaction view sequence $(a_i)_{i \in \{1,\ldots,k\}}$ of a player v, we define the set of *partial interaction views* which constitutes of all the subsequences of sequence a which contain the first term of a. For a partial interaction view subsequence p of a we will say that p is a *j-round partial interaction view* if the index of the last term of a contained in p is j; i.e., if $j = \max\{i \in \{1, \ldots, k\} \mid a_i \in p\}$. With $\mathcal{P}$ we denote the space of all possible partial interaction views. Obviously $\mathcal{P}$ is a function of $\mathcal{H}$.

1.1.4 Message function

In the distributed setting we are interested in the case where the communication between players follows some rules [4], after the receipt of their input. The rules that players use to communicate can be described by a *message function* [5]. A message function determines the messages that a player is sending to its neighbors and its output value according to its partial interaction view. These sent messages and output of a player can be described as a term of an interaction view sequence. Therefore we assume that given a j round partial interaction view a, its image through the message function f_{mes} is a $(j+1)$-round partial interaction view. To model situations where the rules only depend on specific partial interaction view subsequences

[4]After the introduction of a computational model, it will be clear that these rules actually constitute the distributed algorithm.

[5]Often, in the related literature, an assignment of message functions to all the players is assumed. However, assuming a universal message function is also realistic because it matches the case where all the communication rules are known to all the players. This is compatible with the usual case where a distributed algorithm is known to its entirety by all participating players. The definitions and results can be trivially modified to capture the case where different message functions are assumed to be assigned to players.

(e.g. rules that depend only in messages received in the last round) [6], for an arbitrary class $\mathcal{P}_I$ of partial interaction view sequences we define:

$$f_{mes} : \mathcal{P}_I \to \mathcal{P}$$

Specifically, we assume that for a j-round partial interaction view a of player v, $f_{mes}(a) = a'$ is the sequence a appended with a term which constitutes of an assignment of sent messages to the outgoing edges of v, i.e., S^v_{j+1}, and thus a' is a $(j+1)$-round partial interaction view. Note that the id of a player v with interaction view a is included in a as a part of its initial input and will trivially be included in all its partial interaction views because they all contain the first term of the corresponding interaction view.

The partial interaction views of player v and the messages sent to its neighbors can be treated as intermediate inputs and outputs of player v. Observe that given a message function and the initial input of every player, then an entire interaction history (or a set of possible interaction histories is the asynchronous model) can be uniquely determined with repeatedly applying the message function. We denote the space of all possible message functions with F_{mes}.

1.1.5 Problems in distributed computing

To define the problem notion in our model, we first have to address the notion of instance.

Instance. As in [18], we assume that an *instance* is a pair $I \in \mathcal{G} \times F$ which consists of a graph G and a function (or equivalently vector) f_{in} which assigns initial (input) values to all the players $V(G)$. We denote with $\mathcal{I}$ the space of all possible instances.

Problem and solution set. Considering the notion of a *search problem* (cf. [22]) in the sequential setting we define its analog in the distributed setting. We will simply refer to it as *problem*.

[6]This detail is useful to model systems where players have restricted memory and their rules may consider only a specific part of their interaction view

A *distributed problem* is a relation $R \subseteq \mathcal{I} \times F$. Each of these pairs contains an instance (G, f_{in}) and a corresponding output value assignment (vector) f_{out} to all the players $V(G)$. We define the *set of solutions* for the instance I as $R(I) = \{f_{out} : (I, f_{out}) \in R\}$.

Solver of a problem

Defining a solver of a problem in the distributed setting is more complicated than in the sequential setting (cf. [22]). In the latter, the solver of a problem R can simply be defined as a function that maps each instance x to a corresponding output string y s.t. $(x, y) \in R$. In the distributed setting, a meaningful solver definition should include a description of the rules that the players use to communicate, i.e., the message function. The latter allows us to determine every intermediate input and output (received and sent messages) of a player.

A *solver* of problem R in the distributed setting is a message function $S \in F_{mes}$, such that for every instance I of a problem R, the outputs of the players due to the repeated application of S can be described from the function $S_{out} \in F$ with $(I, S_{out}) \in R$. Given the message function S, the output vector S_{out} is well defined, as shown below. We handle the cases of the synchronous and asynchronous model separately to clarify the difference on the determination of the output.

Output $\mathrm{S}_{\mathrm{out}}$ of players in the synchronous setting. In the synchronous model the output of the players can be deterministically determined by the instance and the players' message function S. Observe that given an instance (G, f_{in}), which includes the initial input of players, one can define the output of each player $v \in V(G)$ by repeatedly applying the function S. We denote the output vector of all players with $S_{out}(I)$. For technical reasons we assume that $S_{out}(I)$ is the singleton which contains the corresponding output function (output vector).

Output $\mathrm{S}_{\mathrm{out}}$ of players in the asynchronous setting. As argued in [48], the asynchronous model is inherently non-deterministic. This holds true even when the protocols used are strictly deterministic and use no randomization whatsoever. The reason for this is that the model contains an inherently nondeterministic component, namely, the ordering of message deliveries, which in the fully asynchronous model is completely arbitrary. Despite the fact that the this observation complicates the behavior of the solver function, we can still obtain a well defined determination of the set of all possible outputs. Namely, given the description of possible delays (which is the fully asynchronous model can be any amount of time) and the instance (G, f_{in}), one can define the set of all possible outputs as described in the synchronous model, taking into consideration all possible orderings of message deliveries. Thus, in the asynchronous model with $S_{out}(I)$ we denote the set of all possible outputs of the players $V(G)$ (all possible output vectors).

More formally we define the function $S_{out} : \mathcal{I} \to 2^F \cup \{\perp\}$ with the property that if $S_{out}(I) \neq \perp$ then $S_{out}(I)$ is defined as above. Slightly abusing the definition we will write $S(I)$ instead of $S_{out}(I)$ in the following.

Solver of problem R. We say that S is a *solver for problem R (or solves R)* if for all $I \in \mathcal{I}$ the following holds: if $R(I) \neq \emptyset$ then $S(I) \subseteq R(I)$ and $S(I) = \perp$ otherwise.

1.2 Computational model and reductions between distributed problems

The main component of a distributed algorithm is the formal description of the rules that the players should follow regarding the messages they send to other players. More specifically, at any given time, it should be clear how a player decides what messages to send to each of its neighbors and what output it produces, with respect to information that it has already acquired since the initiation of the protocol. This actually means that the main component of a distributed algorithm is the computation of the previously defined message function by each player.

Primarily, a distributed algorithm can be described by a (sequential) state machine M, an identical copy[7] of which is assigned to each player and can be used to compute the corresponding message function (message exchanges can be regarded as intermediate inputs and outputs of players).

[7] As in the case of the message functions, assignment of different state machines to each player can be assumed, without essentially changing the model

We consider machines (algorithms) which compute a message function $f \in F_{mes}$. Specifically, given a graph G an algorithm M computes the function $f_{mes} : \mathcal{P}_I \to \mathcal{P}$ defined by $f_M(a) = a'$ if, when invoked on sequence a, algorithm M halts with output a'. We say that *algorithm M solves problem R* if f_M is a solver of R. We will use the notation $M(x)$ instead of $f_M(x)$.

Oracle Machine. Loosely speaking, an oracle machine is a machine that is augmented such that it may pose questions to the outside. We consider the case in which these questions, called queries, are answered consistently by some function f, called the oracle. That is, if the machine makes a query q then the answer it obtains is $f(q)$ and we assume that this anwser is obtained in one computation step. In such a case, we say that the oracle machine is given access to the oracle f. For an oracle machine M, a string x and a function f , we denote by $M^f(x)$ the output of M on input x when given access to the oracle f. We denote this machine with oracle f by M^f.

We next introduce a new notion of reduction in distributed systems analogous to the classic Turing reduction. This notion of reduction has been informally used the literature (e.g. in generalizing the impossibility result for the Consensus problem from the case of 3 players to the case of n, cf. [39])

Oracle Reduction in Distributed Systems. A distributed problem R reduces to a problem R' ($R \leq R'$) if there exists an oracle machine M such that for every solver S' of R' it holds that $M^{S'_{out}}$ solves R.

1.2.1 Distributed Systems

Detailed descriptions of computational models for distributed systems, are presented in [39, 48, 24]. Despite the several different computational models appearing in the literature, the main idea is roughly the same and presented below:

We can assume that through a distributed algorithm Π each player v_i is associated with a state machine Π_i, with possibly infinite state set Q such that at any given moment, Π_i is at some state $q \in Q$. The communication of players can be modeled if we assume that the machines Π_i are interactive state machines (cf. [23]) or if we assume that the links are also state machines as proposed in [48]. Observe that this model is not contradictory with the idea of a sequential machine M used in the previous section. This is because we can assume that each state machine Π_i has access to the sequential machine M or else, contains it in its description and can simulate all the intermediate inputs and outputs of M internally. Naturally, we require that Π_i provides M with input corresponding to the partial interaction view that it received and sends messages to all its outgoing neighbors (state machines) according to the output of the sequential machine M. In the following, we assume that a distributed protocol Π also includes the description of the sequential machine M as described. When we want to explicitly argue about the sequential machine M implied by the distributed algorithm Π we will also refer to the distributed algorithm as (Π, M).

Players as state machines. Through a distributed protocol Π, each player is assigned a state machine Π_i with a (possibly infinite) state set Q. As in [39], for each state machine Π_i we assume a set of *initial states* $Q_0 \in Q$ and a set of halting states $Q_H \in Q$; if Π_i reaches a halting state then no further activity (no messages generated) can occur on behalf of Π_i and the only state transition is a self-loop. With $\mathcal{Q}$ we denote the space of all states. State transitions of Π_i are determined by the function $trans_i$ which maps the vectors of sent a received messages of player v and a state q_l to a state q_{l+1}. The function $trans_v$ is actually specified by the sequential machine M. The messages that are sent by a player v are determined by the function mes_v which maps the current state of Π_i to a vector of messages sent to v's neighbors.

Inputs and outputs of players. Adopting ideas from [39], we use the simple convention of encoding the inputs and outputs in the states. In particular, inputs are placed in designated input variables in the start states and outputs appear in designated output variables which can be assigned a value only once (write-once variable). The fact that a machine Π_i can have multiple start states is important here, so that we can accommodate different possible inputs. In fact, we normally assume that the only source of multiplicity of start states is the possibility of different input values in the input variables.

Runs

The execution of a distributed algorithm Π can be described in a similar way with that of interaction history defined in Section 1.1.3 extended in a way that it also includes the states of the players-machines. We will call this description a *run* [8]. A run of Π in graph $G = (V, E)$ is a finite sequence of function triples,

$$(r_i)_{i\in\{1,\dots,k\}} = r_1, \dots, r_k = (C_1, S_1, R_1), \dots (C_k, S_k, R_k)$$

where

$C_i : V \to Q$ (Assignment of states to all players).
$S_i : E^* \to X$ (Assignment of sent-messages to edges).
$R_i : E^* \to X$ (Assignment of received-messages to edges).

Specifically we assume that in the initial term, C_1 is the assignment of initial states (and therefore inputs) to all players. Note that the environment node v_e we used in Section 1.1.3 to model the inputs does not exist in this description. The sent-messages vector S_i is a function (the mes_v function) of C_i and the state vector C_{i+1} is a function (the $trans_v$ function) of the triple (C_i, S_i, R_i).

View of a player v. The *view* of a player is actually the part of the run of which the player is aware of and can be defined as a sequence in a way completely analogous to that of the interaction view (cf. Section 1.1.3) of the player, i.e., by restricting the state, sent-messages and received messages assignments to the ones that v can observe. We denote the view sequence of a player in a run r with $view(v, r)$.

We next define the notion of indistinguishable runs which will be used repeatedly in our study.

Definition 1.1 (Indistinguishable runs). *We will say that two runs r, r' are indistinguishable with respect to player v, denoted $r \overset{v}{\sim} r'$, if $view(v, r) = view(v, r')$, i.e. if v has the same sequence of states outgoing and incoming messages in both runs r, r'. Observe that the notion is well defined even if we assume runs in different communication networks.*

[8]The terms *execution* and *scenario* have been used in the literature for the same notion.

Distributed Systems. A distributed system is usually defined as a set of players connected by a communication network G and run a distributed algorithm Π. However, Halpern and Moses [24] gave a more flexible definition of a distributed system which we are going to adopt in this thesis. They identified a distributed system (running distributed algorithm Π in G) with a set runs R_Π. Observe that, corresponding to every distributed system, given an appropriate set of assumptions about the properties of the system, there is a natural set R_Π of all possible runs of the system. Thus, for example, a distributed system R_Π is synchronous exactly if in all possible runs of the system the players and the communication medium work in synchronous phases.

As a trivial case one could consider the system R_Π which is synchronous and algorithm Π which is deterministic; the only source of multiplicity of runs in this system is the different input vectors of the players.

1.3 Efficiency of distributed protocols

Usually in the literature, the efficiency of distributed protocols is measured with respect to the number of communication rounds that are required for all players to halt and with respect to the amount of communication that is required among the players.

Definition 1.2 (Round complexity). *The round complexity of a distributed algorithm Π is the maximum number of rounds over all runs $r \in R_\Pi$ such that all players halt.*

Recall that in agreement with [48], we have defined a round of a run r in the asynchronous system to be the time period equal to the maximum message delivery delay that occurs in r. Thus the above definition also holds for the asynchronous model[9].

Definition 1.3 (Bit/Message complexity). *The bit complexity (resp. message complexity) of a distributed algorithm Π is maximum total number of bits (resp. messages) transmitted during a run over all runs $r \in R_\Pi$.*

[9] A different definition of the round complexity of asynchronous systems was assumed by Fitzi in 2002 where it is stated that the round complexity of an asynchronous protocol is its round complexity when run in a synchronous network.

Lastly, we define the *local computations complexity*, which represents the worst case of computational complexity of the sequential machine M over all runs and players. Recall that a distributed algorithm Π implies the existence of the sequential machine M which essentially can be assumed to execute all local (internal) computations of the players. As stated before, we will also refer to the distributed algorithm also as (Π, M).

Definition 1.4 (Local computations complexity)**.** *The local computations complexity of a distributed algorithm (Π, M) is the maximum number of local computational steps that the corresponding sequential machine M executes in a round, over all players and all runs $r \in R_\Pi$.*

Definition 1.5 (Fully polynomial algorithm)**.** *We will say that a distributed algorithm Π is fully polynomial if it is of polynomial round, bit and local computations complexity over all instances.*

We next give the definition of a polynomial time reduction between distributed problems which is analogous to the notion of a Cook reduction.

Definition 1.6 (Polynomial oracle reduction in distributed systems)**.** *A distributed problem P polynomially reduces to a problem P' ($P \leq_p P'$) if there exists a fully polynomial distributed algorithm (Π, M) with M being an oracle machine, such that for every solver S' of R' it holds that $M^{S'_{out}}$ solves R.*

1.4 Adversarial behavior of players

As communication networks grow in size, they become increasingly vulnerable to component failures. Since distributed computing has become popular and widely used in contemporary networking, the provided solutions need to cope with erroneous and malicious components in the underlying communication network.

The dishonesty or malfunction of players is modeled by a *central adversary* that corrupts some players hereafter referred to as the *corrupted players*; we will refer to the players that are not corrupted by the adversary as *honest*. For instance, one may think of the adversary as a hacker who attempts to break into the players computers. Different adversary models can be determined with respect to the corruption capacity of the adversary; among all corruption types, *Byzantine* (or *active*) corruption models a worst-case fault scenario, namely components can behave arbitrarily (even maliciously) as transmitters, by either stopping, rerouting, or altering transmitted messages in a way most detrimental to the communication process. The study of distributed systems in the presence of adversarial behavior involves the design of algorithms that work correctly despite the existence of corrupted players in the network and usually without knowing their exact location.

1.4.1 The Adversary Model

In the distributed systems setting an adversary model defines the sets of players that can be corrupted by the adversary (possible/admissible corruption sets) as well as the possible behavior of the corrupted players, i.e., all the possible actions that the corrupted players can execute. The adversarial behavior in an execution of a distributed protocol can be described exactly by the set and the actions of the corrupted players; more concretely, in the context of our study we can consider the adversary model as a set of pairs $T = (C, \Pi_C)$ where C is the set of corrupted players and Π_C is the protocol they execute. We consider the *byzantine adversary model* which imposes no restrictions on the behavior of the corrupted players; i.e., the corrupted players are under full control of the central adversary and misbehave in arbitrary manner.

Computational power of the adversary. The adversary model also determines the adversarys computing power. Most common assumptions are that the adversary is either *unlimited*, or is *computationally bounded* to probabilistic polynomial time computations in a security parameter κ.

Reliability of Protocols. The extend to which a distributed protocol achieves a given task under the existence of an adversary, is defined with respect to a security parameter κ, allowing an error probability ϵ that is a negligible function of κ. A protocol is said to achieve a task in the *information-theoretic* setting (or is unconditionally reliable) if it achieves the task with a negligible error probability ϵ against an unlimited adversary. If it achieves the task with zero error probability against an unlimited adversary then the protocol is called *perfectly reliable*.

In the current thesis we consider the existence of an unbounded Byzantine adversary and study the solution of problems with zero error probability, thus we focus on the perfectly reliable setting. Our results also hold for the other adversary types of adversarial corruption capacity, since a byzantine adversary models a worst-case scenario on the corruption strength.

As mentioned, the adversary model can be distinguished with respect to the sets of players that can be corrupted. The extensively studied *t-threshold adversary model* corresponds to the situation where the adversary can corrupt at most t players in the network. In this work we consider the following adversary models with respect to the possible corruption sets:

t-Locally Bounded Adversary. At most a certain number t of corruptions are allowed in the neighborhood of every node. The model was introduced by Koo in [30]. The importance of this model comes, among others, from the local restrictions imposed to the adversary, which may be used to derive local criteria which can be employed in unknown topology networks. The locally bounded adversarial model is particularly meaningful in real-life applications and systems. For example, in social networks it is more likely for an agent to have a quite accurate estimation of the maximum number of malicious agents that may appear in its neighborhood, than having such information, as well as knowledge of connectivity, for the whole network. In fact, this scenario applies to all kinds of networks, where each node is assumed to be able to estimate the number of traitors in its close neighborhood. It is also natural for these traitor bounds to vary among different parts of the network. Motivated by such considerations, in this work we also introduce a generalization of the t-locally bounded model with a varying bound for each neighborhood.

Considering the possible corruption sets. the general adversary model was initiated by Hirt and Maurer in [26], subsumes all known adversary models, both the aforementioned included. A description follows.

General Adversary. The adversary can corrupt any set in a given family of sets $\mathcal{Z}$ called the *adversary structure*. A structure $\mathcal{Z}$ for the set of players V is a monotone family of subsets of V, i.e. $\mathcal{Z} \subseteq 2^V$, where all subsets of Z are in $\mathcal{Z}$ if $Z \in \mathcal{Z}$.

1.4.2 The Communication Model

We assume the message passing model as described in the distributed systems model in the beginning of the chapter. In this model the players communicate through channels which are represented by the edges of the communication network. A different case of adversary has been considered in the literature (cf.[6]), namely, an adversary who corrupts the communication channels in a way that either obtains all information communicated through a corrupted edge or even alters the messages that are send through that edge. Our study is conducted in the model where the channels of communication are reliable and once the message is sent through a channel, it will arrive intact to its destination in some finite time. Specifically, we consider the *authenticated channels model*.

Authenticated communication channels. In our study we assume *authenticated channels* of communication which means that the channels are resistant to tampering but not necessarily resistant to overhearing, i.e., messages, once sent, cannot be changed but can be read by an adversary. Moreover we adopt the usual assumption that the receiver of a message always knows the identity of the sender.

Regarding the synchrony of channels we explicitly consider the synchronous model as described in Section 1.1.3. However, our studies can be easily extended to the asynchronous model because all the protocols presented are event-driven.

1.4.3 Efficiency of distributed protocols in the presence of adversaries

The definitions of distributed protocols complexity measures as presented in Section 1.3 take in to account the actions of all players. This allows a byzantine adversary to completely determine every complexity measure of a protocol by either running for a very large number of rounds, sending large messages or performing arbitrary local computations. Thus, in the case where some players are controlled by an adversary we should define the complexity measures only with respect to the actions of honest players, which actually run the protocol correctly; namely, we modify the definitions as follows:

Round complexity (RC): The maximum number of rounds that are required by any honest player in the worst case.

Bit Complexity (BC): The total number of bits sent by all honest players during the protocol execution in the worst case.

Local Computations Complexity (LCC): The maximum over the local computational worst-case complexities of all honest players.

1.5 Agreement and information propagation

1.5.1 Reliable Broadcast Problem

A fundamental problem in distributed networks is Reliable Broadcast (or *Byzantine generals*), in which the goal is to distribute the message of a designated player, called the *dealer*, correctly despite the presence of Byzantine corruptions. In general, agreement problems have been primarily studied under the threshold adversary model, where a fixed upper bound t is set for the number of corrupted players and it has been shown that Reliable Broadcast can be achieved if and only if $t < n/3$, where n is the total number of players.

The Reliable Broadcast problem has been extensively studied in complete networks under the threshold adversary model mainly in the period from 1982, when it was introduced by Lamport, Shostak and Pease [37], to 1998, when Garay and Moses [19] presented the first fully polynomial Reliable Broadcast protocol optimal in resilience and round complexity. The difficulty of designing a solution for can be primarily summarized in a scenario where the dealer is corrupted because the dealer may send conflicting values to the players in which case we demand that the players finally agree on the same arbitrary value. Essentially, the major task is to circumvent errors without losing unanimity. The formal definition of the Reliable Broadcast problem follows,

Definition 1.7 (Reliable Broadcast). *Let $\mathcal{V} = \{p_1, p_2, \cdots, p_n\}$ be a set of n players, X be a finite domain and $D \in \mathcal{V}$ be the dealer. Then we say, that Π is a Reliable Broadcast (Byzantine Generals) protocol among players in $\mathcal{V}$ with values in X, where D has as input a value $m \in X$ and all players finally decide on (output) a value $y_i \in X$, if it satisfies the following conditions:*

1. *Validity: If the dealer is honest, then all honest players will decide on m;*

2. *Consistency: All honest players finally decide on the same value. i.e. $\forall p_i, p_j$ honest players, $y_j = y_i$ holds;*

In multi-hop networks where the dealer is not directly connected with all the players the problem becomes more difficult; even in the case where the dealer is honest the adversary may corrupt a crucial part of the network and make it impossible for some players to decide on the correct value. This situation is depicted in Figure 1.1,

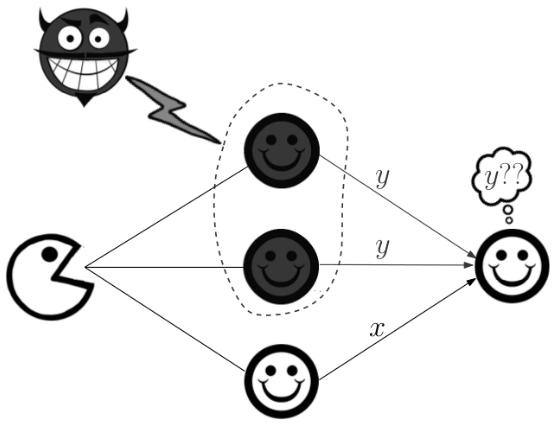


FIGURE 1.1: Broadcast in Incomplete Networks

The case of Reliable Broadcast in incomplete networks has been studied to a much lesser extent, in a study initiated in [14, 15, 36], mostly through protocols for Secure or Reliable Message Transmission which, combined with a Reliable Broadcast protocol for complete networks, yield Reliable Broadcast protocols for incomplete networks. Naturally, connectivity constraints are required to hold in addition to the $n/3$ bound. For instance, in the threshold model, at most $t < c/2$ corruptions can be tolerated, where c is network connectivity, and this bound is tight[14].

1.5.2 Reliable information propagation

In this work we address the problem of *Reliable Broadcast with an honest dealer* in generic (incomplete) networks. As we will see in Section 3.7, this case essentially captures the difficulty of the general problem, where even the dealer may be corrupted by simulating the message transmissions of a Broadcast protocol for complete networks. The problem definition follows.

Reliable Broadcast with Honest Dealer. The network is represented by a graph $G = (V, E)$, where V is the set of players, and

E represents authenticated channels between players. We assume the existence of a designated honest player, called the *dealer*, who wants to broadcast a certain value $x_D \in X$, where X is the initial input space, to all players. We say that a distributed protocol achieves Reliable Broadcast if by the end of the protocol every honest player has *decided on* x_D, i.e. if it has been able to output the value x_D originally sent by the dealer.

As we previously pointed out, the problem is trivial in complete networks; we will consider the case of incomplete networks here. For brevity we will refer to this problem as the Broadcast problem. Observe that achieving Broadcast in this way is actually equivalent with achieving correct (reliable) message transmission from the dealer D to all players. The honest dealer case is particularly meaningful in the case of wireless networks. Due to he local broadcasts that occur in this model the dealer is roughly committed to send the same value to all its neighbors.

We also consider the closely related Reliable Message Transmission problem where a player wants to correctly transmit a message to another. In fact a solution for Reliable Message Transmission for all the possible dealer-player pairs implies a solution for the Reliable Broadcast problem with an honest dealer. The definition of the problem follows.

Reliable Message Transmission. We assume the existence of a designated player D, called the *dealer*, who wants to propagate a certain value $x_D \in X$, where X is the initial message space, to a designated player R, called the receiver. We say that a distributed protocol achieves (or solves) RMT if by the end of the protocol the receiver R has *decided on* x_D, i.e. if it has been able to output the value x_D originally sent by the dealer.

1.6 Topology Knowledge

Regarding the initial knowledge that the players possess about the topology of the network, the next two cases are the ones that have been mostly studied in the literature.

- *Ad Hoc* Networks: Each player is only aware of its neghbors' ids.
- Known Topology Networks: Each player is aware of the entire network topology.

Furthermore, in Chapter 5 we consider a lower level of initial topology knowledge where the players are only aware of their own ids. In this work we also introduce the *Partial Knowledge Model*, in which each player only has knowledge over the topology of an arbitrary subgraph of the network.

As can be seen in the rest of this work, the knowledge over the possible corruption sets is naturally related with the topological knowledge that each player possesses. The motivation for partial knowledge considerations comes from large scale networks (e.g. the Internet) where topologically local estimation of the power of the adversary may be possible, while global estimation may be hard to obtain due to geographical or jurisdiction constraints. Additionally, proximity in social networks is often correlated with an increased amount of available information, further justifying the relevance of the model.

Chapter 2

Ad Hoc Broadcast in the Locally Bounded Model

In this chapter we consider the Reliable Broadcast problem in incomplete networks. We study the resilience of the Certified Propagation Algorithm (CPA) [30], which is particularly suitable for *ad hoc* networks. We address the issue of determining the maximum number of corrupted players $t_{\max}^{\mathrm{CPA}}$ that CPA can tolerate under the t-locally bounded adversary model, in which the adversary may corrupt at most t players in each player's neighborhood. For any graph G and dealer-node D we provide upper and lower bounds on $t_{\max}^{\mathrm{CPA}}$ that can be efficiently computed in terms of a graph theoretic parameter that we introduce in this work. Along the way we obtain an efficient 2-approximation algorithm for $t_{\max}^{\mathrm{CPA}}$. We further introduce two more graph parameters, one of which matches $t_{\max}^{\mathrm{CPA}}$ exactly. Our approach exactly captures the information propagation of CPA and thus allows to provide intuitive and easily manageable conditions concerning the behavior of the algorithm.

Finally, our study allows us to show that CPA is *unique*, against locally bounded adversaries in *ad hoc* networks, among all *safe algorithms*, i.e., algorithms which never cause a node to decide on an incorrect value. This means that CPA can tolerate as many local corruptions as any other safe algorithm; this settles the open question of CPA Uniqueness posed by Pelc and Peleg in [47].

2.1 Introduction

In the case of an honest dealer, particularly useful in wireless networks due to local Broadcasts, the impossibility threshold of $n/3$ does not hold; for example, in complete networks the problem becomes trivial. However, in incomplete networks the situation is different.

A small number of traitors (corrupted players) may manage to block the entire protocol if they control a critical part of the network, e.g. if they form a separator of the graph. It therefore makes sense to define criteria depending on the structure on the graph (graph parameters), in order to bound the number or restrict the distribution of traitors that can be tolerated.

An approach in this direction is to consider topological restrictions on the adversary's corruption capacity. The importance of local restrictions comes, among others, from the fact that they may be used to derive local criteria which the players can employ in order to achieve Broadcast in *ad hoc* networks. Such an example is the *t-locally bounded adversary model*, introduced in [30], in which at most t-corruptions are allowed in the neighborhood of every node.

2.1.1 Related work

Koo [30] proposed a simple, yet powerful protocol for the t-locally bounded model, the *Certified Propagation Algorithm* (CPA) (a name coined by Pelc and Peleg in [47]), and applied it to networks of specific topology, namely grid networks. In 2005 Pelc and Peleg [47] considered the t-locally bounded model in generic graphs and gave a sufficient topological condition for CPA to achieve Broadcast in such graphs. They also provided an upper bound on the number of corrupted players t that can be locally tolerated in order to achieve Broadcast by any protocol, in terms of an appropriate graph parameter; they left the deduction of tighter bounds as an open problem. To this end, Ichimura and Shigeno [27] proposed an efficiently computable graph parameter which implies a tighter, but not exact, characterization of the class of graphs on which CPA achieves Broadcast.

Two important open questions were stated in the study [47] regarding reliable broadcast: (a) to derive a tight parameter revealing the maximum number of traitors that can be locally tolerated by CPA in a graph G with dealer D, thus yielding a tight condition for CPA correctness, and (b) to check whether the *CPA Uniqueness* conjecture holds, which essentially states that whenever Broadcast is possible CPA will manage to achieve it; equivalently, that no *ad hoc* algorithm can tolerate more local corruptions than CPA on any instance (G, D). Both these open questions have been addressed and answered mainly in our work [38, 44]. In this chapter we will present the results of [38] and subsequent work regarding both these questions.

For the former question, we define a new graph parameter that captures exactly the number that can be locally tolerated by CPA, thus providing a tight condition for CPA correctness as mentioned above. Very recently Tseng *et al.* [53] independently also gave a necessary and sufficient condition for CPA correctness; a corresponding tight parameter is implicit in their work. Our condition is obviously equivalent to that of [53]; however, no result concerning CPA Uniqueness, neither any necessary condition for Broadcast in general are given in [53]. On the other hand, our approach not only gives a faster way to check the tight condition, most importantly it allows us to prove that the same condition is necessary for achieving Broadcast through *any safe*[1] algorithm in the *ad hoc* model, thus establishing that whenever Broadcast is possible, CPA can achieve it. This provides an affirmative answer to the second question of [47], namely we have proved that the *CPA Uniqueness* conjecture holds with respect to safe Broadcast algorithms. Note the restriction in the class of safe algorithms was also implied in [47].

Moreover, our approach leads to an efficient 2-approximation algorithm for the problem of determining the maximum local number of traitors that CPA (and by the uniqueness result any safe algorithm) can tolerate on a given instance (G, D). A 2-approximation scheme was also implied in the study of [27]; however, our algorithm has $O(|E| \log \delta)$ time complexity, where δ is the minimum degree over all players. This significantly improves upon the complexity bound for the approximation scheme implied in [27] which is $O(|V|(|V| + |E|))$.

The significance of this approximability result comes from the fact that this number is NP-hard to compute as shown in our work in [44] which is presented in the Chapter 3. Let us also mention that in [44] a new (equivalent) necessary and sufficient condition for Broadcast was given, which, despite being more adaptable to different adversary models, does not yield an efficient approximation algorithm in any obvious way. Hence, the condition presented in this chapter is interesting *per se*.

2.2 Outline

We study the behavior of CPA in generic (incomplete) networks, with an honest dealer. As we will see in Section 2.10, this case essentially

[1]By the term 'safe' we refer to the notion of algorithms that never make a node take a wrong decision.

captures the difficulty of the general problem, where even the dealer may be corrupted. Our first contribution is the exact determination of the maximum number of corrupted players $t^{\mathrm{CPA}}_{\max}(G, D)$ that can be locally tolerated by CPA, for any graph G and dealer D. We do this by developing three graph parameters:

- $\mathcal{K}(G, D)$ is determined via an appropriate *level-ordering* of the nodes of the graph. We show that $t < \mathcal{K}(G, D)/2$ is a sufficient condition for CPA to be t-locally resilient and that $t < \mathcal{K}(G, D)$ is a necessary condition, implying that $\lceil \mathcal{K}(G, D)/2 \rceil - 1 \leq t^{\mathrm{CPA}}_{\max} < \mathcal{K}(G, D)$. We prove that our parameter coincides with the parameter $\widetilde{\mathcal{X}}(G, D)$ of [27]. We further propose an efficient algorithm for computing $\mathcal{K}(G, D)$ which is faster than the algorithm for computing $\widetilde{\mathcal{X}}(G, D)$ proposed in [27]. Note that this immediately gives an asymptotic 2-approximation for $t^{\mathrm{CPA}}_{\max}$; we provide an example that shows that the ratio of this algorithm is tight.

- $\mathcal{M}(G, D, t)$, depending also on a value t, is a parameter that immediately reveals whether CPA is t-locally resilient for graph G and dealer D, by simply checking whether $\mathcal{M}(G, D, t) \geq t + 1$. Therefore, via this parameter, we provide a *necessary and sufficient condition* for CPA to be t-locally resilient. Such a condition was not known until very recently, when a necessary and sufficient condition was independently given in [53]. However, the way in which the condition of [53] is defined implies a super-exponential time algorithm to check it (actually no algorithm is given in [53]). On the other hand, we will see that even a naïve algorithm to compute $\mathcal{M}(G, D, t)$ would need single exponential time.

- $\mathcal{T}(G, D) = \max\{t \in \mathbb{N} \mid \mathcal{M}(G, D, t) \geq t + 1\}$, gives the maximum number of corrupted players that CPA can tolerate in every node's neighborhood, hence exactly determining $t^{\mathrm{CPA}}_{\max}(G, D)$.

In addition, using the $\mathcal{M}(G, D, t)$ parameter we prove that CPA is *unique* among the t-locally safe *ad hoc* broadcast algorithms. That is, if a t-locally safe *ad hoc* broadcast algorithm is t-resilient for a graph G with dealer D, then CPA is also t-resilient for G, D. Thus we provide and affirmative answer to the open problem of CPA Uniqueness posed in [47].

2.3 Problem and Model Definition

In this section, we formally define the problem and the model as well as some usefull notions which are going to be used throughout this thesis. As we previously mentioned, the goal of Reliable Broadcast is to have some designated player, called the dealer, consistently send an input value to all other players of the network even in the presence of a central adversary which corrupts some players and controls them in some extend. Therefore the effectiveness of a Reliable Broadcast protocol should be considered w.r.t. the capacity of the adversary, i.e. the adversary model.

Adversary model $\mathcal{T}$. An adversary model $\mathcal{T}$ defines the sets of players that can be corrupted by the $\mathcal{T}$-adversary (possible/admissible corruption sets) as well as the possible behavior of the corrupted players, i.e., all the possible actions that the corrupted players can execute. The adversarial behavior in an execution of a distributed protocol can be described exactly by the set and the actions of the corrupted players; more concretely, $\mathcal{T}$ can be regarded as a set of pairs $T = (C, \Pi_C)$ where C is the set of corrupted players and Π_C is the protocol they execute. We consider the byzantine adversary model which imposes no restrictions on the behavior of the corrupted players. Regarding the possible corruption sets, in this chapter we only consider the t-locally bounded model.

The network model that we use in this chapter is defined below.

Network model. We assume that the players V are arranged in a communication network which is represented by a graph $G = (V, E)$ where E is a set of undirected, authenticated channels of communication between pairs of players.

In this chapter we address the problem of *Reliable Broadcast with an honest dealer* in generic (possibly incomplete) networks. For brevity we will refer to it simply as the *Broadcast problem*. The problem is trivial in complete networks; we will consider the case of incomplete networks here. As we will see in Section 3.7, the case of an honest dealer in incomplete networks essentially captures the difficulty of the general problem, where even the dealer may be corrupted. A protocol for the general case can be devised by simulating the message exchange of Broadcast protocols in complete networks, which have been extensively studied. We consider deterministic protocols for the solution of the problem.

Definition 2.1 (Reliable Broadcast with honest dealer/Broadcast). *Let $V = \{v_1, \ldots, v_n\}$ be the set of n players arranged in a communication network $G = (V, E)$ as described above and X be a finite domain. Consider a distributed protocol Π among players V, where player $D \in V$ (called the dealer) holds an input value $x_D \in X$ and every player $v \in V$ finally decides on a single output value $y_v \in X$. Also assume any adversary model $\mathcal{T}$ s.t. the dealer can not be corrupted. Protocol Π achieves Broadcast (or is a Broadcast protocol) in (G, D) under the adversary model $\mathcal{T}$ if for any possible corruption set T and any adversarial behavior of this set, all honest players decide on the dealer's input value, i.e., $\forall v \in V \setminus T, y_v = x_D$.*

Usually, also “termination" is demanded by the standard definition of Broadcast, i.e., that it must be guaranteed that all correct players eventually terminate the protocol. As usual in the related literature, we omit the termination study, which is implied by the studies of the algorithms' correctness.

Our study in this chapter concerns the t-locally bounded adversary model introduced by [30]. The family of *t-local* sets (defined below) plays an important role in our study since it coincides with the family of admissible corruption sets.

Definition 2.2 (t-local set). *Given a graph $G = (V, E)$ and an integer $t \in \mathbb{N}$, a t-local set is a set $C \subseteq V$ for which $\forall u \in V$, $|\mathcal{N}(u) \cap C| \leq t$.*

t-locally bounded adversary model. In this model the adversary can only corrupt a t-local set during an execution of a protocol in the system.

In contrast with the extensively studied t-threshold model (cf. [37]), which bounds the total number of corrupted players, the t-locally bounded model was introduced to bound the corruptions in the neighborhood of honest players. This could be viewed as modeling the situation where corrupted players are distributed somewhat uniformly across the network. Besides, bounding the total number of corrupted players, may not be very interesting in incomplete networks since an adversary could simply corrupt all players in the neighborhood of a specific honest player and thus, block any message propagation to it.

2.3.1 Protocol Properties

We next define some protocol properties, some of which were introduced in [47], that facilitate our study and are used throughout this thesis.

Definition 2.3 (t-locally resilient algorithm for (G, D))**.** *An algorithm which achieves Broadcast in a given graph G with dealer D for any t-local corruption set T and any behavior of T is called t-locally resilient for (G, D).*

According to the definition of a Broadcast algorithm, a t-locally resilient algorithm for (G, D) is an algorithm which achieves Broadcast in (G, D) under the t-locally bounded adversary model.

Definition 2.4 (Safe / t-locally safe algorithm)**.** *An algorithm which never causes an honest node to decide on (output) an incorrect value, for any graph-dealer pair (G, D), is called safe.*
An algorithm which never causes an honest node to decide on an incorrect value under any t-local corruption set and any behavior of it, for any graph-dealer pair (G, D), is called t-locally safe.

Note that a safe algorithm might still fail, particularly by not correctly delivering the message to all nodes of the network. Essentially, a safe Broadcast algorithm ensures that a player will decide on a value only in the case she can undoubtedly deduce from her view (input and exchanged messages) that this is the actual value of the dealer.

Observe that an algorithm is t-locally safe if it satisfies the desired property for every instance (G, D). On the other hand, the algorithm is t-locally resilient for (G, D) if it satisfies the property for the specific instance (G, D). Therefore, it might be the case that an algorithm is t-locally resilient for (G, D) but not t-locally safe, even if the first trivially implies that the safeness property holds for (G, D).

The importance of safeness is pointed out in [47], where it is regarded as a basic requirement of a Broadcast algorithm; it guarantees that even if all players do not have sufficient information to decide on the dealer's value, no one will eventually decide on an incorrect value or accept false data. We next formally introduce the notion of the *uniqueness* of an algorithm.

Definition 2.5 (Uniqueness of Algorithm)**.** *Let $\mathcal{A}$ be a family of algorithms. An algorithm A is unique (for Broadcast) among algorithms in $\mathcal{A}$ if the existence of an algorithm of family $\mathcal{A}$ which achieves Broadcast in an instance (G, D) implies that A also achieves Broadcast in (G, D).*

A unique algorithm A among $\mathcal{A}$, naturally defines the class of instances (G, D) in which the problem is solvable by $\mathcal{A}$-algorithms, namely the ones that A achieves Broadcast in.

2.4 The Certified Propagation Algorithm

As explained in the model, we consider a network where nodes may be corrupted, but at most t-corruptions are allowed in the neighborhood of every node. A corruption set with the above property is called *t-local set*. Given a graph G and dealer D, an algorithm which achieves Broadcast for any t-local corruption set is called t-locally resilient.

An algorithm which achieves Broadcast in the t-locally bounded adversary model is called *t-locally resilient*.

The previously mentioned Certified Propagation algorithm uses only local information and thus is particularly suitable for *ad hoc* networks. CPA is probably the only Broadcast algorithm known up to now for the t-locally bounded model, not requiring knowledge of the network topology.

Protocol 1: ***Certified Propagation Algorithm (CPA) [30]***

Input (for each node v): Dealer's label D, labels of v's neighbors, corruption bound t.

Code for D: send value x_D to all neighbors, decide on x_D and terminate.

Code for $v \in \mathcal{N}(D)$: upon reception of x_D from the dealer, decide on x_D, send it to all neighbors and terminate.

(* *certified propagation rule* *)

Code for $v \notin \mathcal{N}(D) \cup D$: upon reception of $t+1$ messages with the same value x from $t+1$ distinct neighbors, decide on x, send it to all neighbors and terminate.

As shown in [30], CPA is a t-locally safe Broadcast algorithm. The proof is given for completeness.

Theorem 2.1. *CPA is t-locally safe.*

Proof. We will show that if a player decides on a value x through CPA then $x = x_D$. Assume on contrary that there is a set of players $V' \subseteq V$

that decide on values different than x_D. Let v be the player of V' that decides in the earliest round among all players in V', i.e., the first player to make an incorrect decision, and assume that v decides on $x \neq x_D$. v cannot be a neighbor of the dealer since all neighbors of the dealer only decide on x_D as can be shown in the respective decision rule of CPA. Therefore v has received $t+1$ copies of x from $t+1$ distinct neighbors. Since at most $t(v)$ neighbors can be corrupted, at least one honest player has decided in $x \neq x_D$ before v. A contradiction to the fact that v is the first player to make an incorrect decision.

□

We next define the quantity, the computation of which is the issue of this chapter, namely, the maximum number of local corruptions that CPA can tolerate.

Definition 2.6 (Max CPA Resilience). *For a graph G and dealer-node D, $t^{\text{CPA}}_{\max}(G, D)$ is the maximum t such that CPA is t-locally resilient.*

Whenever G and D are implied by the context, we will simply write $t^{\text{CPA}}_{\max}$.

Bounds vs Conditions. Let us now make a simple but useful observation: for a graph-theoretic parameter X, showing that $t < X$ is a sufficient topological condition for CPA to be t-locally resilient provides a lower bound of $\lceil X \rceil - 1$ on $t^{\text{CPA}}_{\max}$. Respectively, necessary conditions of similar form imply upper bounds on $t^{\text{CPA}}_{\max}$. We will often use this relation between bounds and conditions throughout the chapter.

2.5 Lower Bounds on Max CPA Resilience

Pelc and Peleg [47] were the first to present a graph-theoretic parameter $\mathcal{X}(G, D)$ that associates the maximum tolerable number of local corruptions with the topology of the graph. This parameter represents the maximum number b such that every node v has at least b neighbors with distance to D smaller than that of v. They give a sufficient condition for CPA resilience, namely $\mathcal{X}(G, D) \geq 2t + 1$, which implies that the nodes of graph G can be arranged in levels w.r.t. their distance from D, the first level being the neighborhood of D, and every node in level k having at least $2t+1$ neighbors in level $k-1$. The situation is depicted in Figure 2.1. This, in turn implies that every

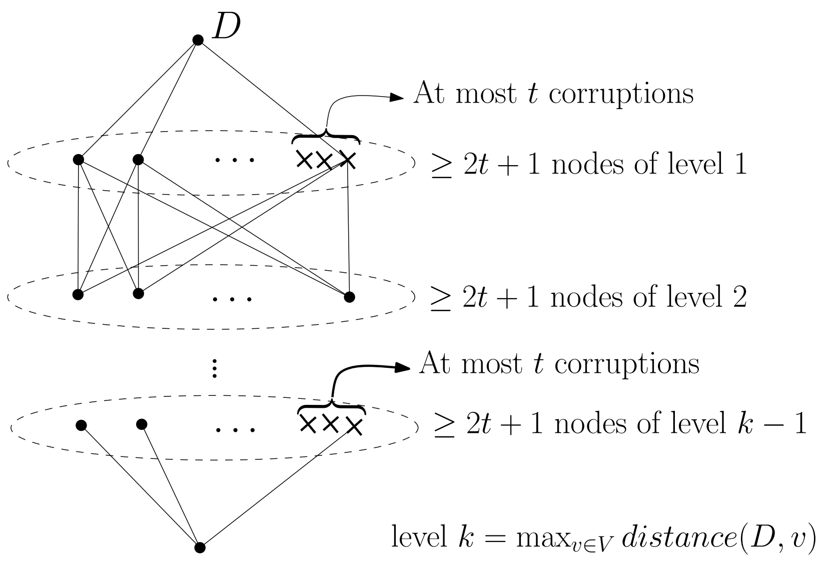

D
At most t corruptions
$\geq 2t+1$ nodes of level 1
$\geq 2t+1$ nodes of level 2
At most t corruptions
$\geq 2t+1$ nodes of level $k-1$
level $k = \max_{v \in V} distance(D, v)$

FIGURE 2.1: Level partition based on parameter $\mathcal{X}$ defined in [47]

node in distance k from D (level k) decides in the k-th round, because it will certainly receive at least $t+1$ correct values from honest nodes in level $k-1$. However, as shown in the same paper, this condition is not necessary, because a node in level k may collect correct values from neighbors in level k or $k+1$ also, thus completing the necessary number of $t+1$ identical values. In other words, $\lceil \mathcal{X}/2 \rceil - 1$ is a lower bound for Max CPA Resilience but not a tight one.

2.5.1 A new parameter for bounding Max CPA Resilience

In order to derive tighter bounds on $t_{\max}^{\text{CPA}}$ we introduce the notion of *minimum k-level ordering* of a graph which generalizes the level ordering that was implicit in [47]. Intuitively, a minimum k-level ordering is an arrangement of nodes into disjoint levels, such that every node has at least k neighbors in previous levels and belongs to the minimum level for which this property is satisfied for this node. Formally:

Definition 2.7. *A Minimum k-Level Ordering $\mathcal{L}_k(G, D)$ of a graph $G = (V, E)$ for a given dealer-node D is a partition $V \setminus \{D\} = \bigcup_{i=1}^{m} L_i$, $m \in \mathbb{N}$*

s.t.

$$\begin{aligned} L_1 =& \mathcal{N}(D), \\ L_2 =& \{v \in V \setminus \ L_1 : |\mathcal{N}(v) \cap L_1| \geq k\} \\ &\vdots \\ L_m =& \{v \in V \setminus \bigcup_{j=1}^{m-1} L_j : |\mathcal{N}(v) \cap \bigcup_{j=1}^{m-1} L_j| \geq k\} \end{aligned}$$

We next define the relaxed k-level ordering notion which will be useful for our proofs, by dropping the level minimality requirement for nodes.

Definition 2.8. *A Relaxed k-Level Ordering of a graph $G = (V, E)$ for a given dealer-node D is a partition $V \setminus \{D\} = \bigcup_{i=1}^{m} L_i$, $m \in \mathbb{N}$ s.t.*

$$L_1 = \mathcal{N}(D), \qquad \forall v \in L_i : |\mathcal{N}(v) \cap \bigcup_{j=1}^{i-1} L_j| \geq k$$

Properties of k-level orderings.

Note that while there may exist several relaxed k-level orderings of a graph, the minimum k-level ordering is unique, as can be shown in the proof of Theorem 2.2. Let us also observe that a relaxed k-level ordering may be easily transformed to the unique minimum k-level ordering; to show this we will use the notion of a *delayed node*:

Definition 2.9 (Delayed node)**.** *Given a relaxed k-level ordering $\mathcal{L}$: $V = \bigcup_{i=1}^{m} L_i$, $m \in \mathbb{N}$ we will refer to a player $u \in L_h \in \mathcal{L}$ as delayed node in $\mathcal{L}$ if $\exists$ d with $1 < d < h \leq m$ s.t. $|\mathcal{N}(u) \cap \bigcup_{j=1}^{d-1} L_j| \geq k$.*

The following is immediate from the previous definitions,

Fact. A relaxed k-level ordering with no delayed nodes is a minimum k-level ordering.

Now, given any relaxed k-level ordering $\mathcal{L}$ we can construct a minimum k-level ordering $\mathcal{L}_k$ simply by repeatedly moving every delayed node to the lowest level such that the partition remains a relaxed k-level ordering. It is not hard to see that a relaxed k-level ordering with no delayed nodes is actually a minimum k-level ordering. Therefore, the following holds,

Theorem 2.2. *Given a graph G and dealer D, for every $k \in \mathbb{N}$, if there exists a Relaxed k-Level Ordering for G, D then there exists a unique Minimum k-Level Ordering for G, D.*

Proof. We prove that if we change the partitions set in a certain manner the partition remains a relaxed k-level ordering and in the end we obtain the Minimum k-Level Ordering $\mathcal{L}_k(G, D)$. We will use the following Claim,

We first easily observe that if there exists a relaxed k-level ordering $\mathcal{L}$: $V = \bigcup_{i=1}^{m} L_i$, $m \in \mathbb{N}$ with $1 < d < h \leq m$, then for arbitrary $u \in L_h$ (delayed node) with $|\mathcal{N}(u) \cap \bigcup_{j=1}^{d-1} L_j| \geq k$ the partition $\mathcal{L}'$:

$$V = L_1 \cup L_2 \cup \ldots \cup \{L_d \cup \{u\}\} \cup \cdots \cup \{L_h \setminus \{u\}\} \cup \cdots \cup L_m = \bigcup_{i=1}^{m} L'_i$$

is also a relaxed k-level ordering.

Based on the above observation, given any relaxed k-level ordering $\mathcal{L}$ we can construct a minimum k-level ordering $\mathcal{L}_k$ simply, by moving all the delayed nodes in the lowest level for which the partition remains a relaxed k-level ordering. Namely,

Given relaxed k-level ordering $\mathcal{L} : V = \bigcup_{i=1}^{m} L_i$, for every delayed node v, move v to the set L_i s.t.

$$i = \min \left\{ d \in \{1, \ldots, m\} \;\middle|\; |\mathcal{N}(v) \cap \bigcup_{j=1}^{d-1} L_j| \geq k \right\}$$

Furthermore, whenever we move a delayed node we should check all other nodes that possibly became delayed due to this move. The process terminates after at most polynomial number of moves without delayed nodes left.

According to the fact that a relaxed k-level ordering with no delayed nodes is a minimum k-level ordering, the resulting partition is a minimum k-level ordering.

Regarding the uniqueness of the minimum k-level ordering $\mathcal{L}_k$ we can assume that for graph G and dealer D there exist two different minimum k-level orderings $\mathcal{L} = \{L_1, \cdots, L_m\}, \mathcal{L}' = \{L'_1, \cdots, L'_h\}$. From the definition of minimum k-level ordering $L_1 = L'_1$ holds. Let i be the lowest integer for which $L_i \neq L'_i$ and assume wlog that $\exists v$, s.t. $v \in L_i$ and $v \notin L'_i$. It is clear that v is a delayed node in L'_i, thus L'_i is not a minimum k-level ordering.

□

Definition 2.10 (Parameter $\mathcal{K}$)**.** *For a graph G and dealer D,*

$$\mathcal{K}(G,D) \overset{def.}{=} \max\{k \in \mathbb{N} \mid \exists \text{ a Minimum } k\text{-Level Ordering } \mathcal{L}_k(G,D)\}$$

Theorem 2.3 (Sufficient Condition)**.** *For every graph G, dealer D and $t \in \mathbb{N}$, if $t < \mathcal{K}(G,D)/2$ then CPA is t-locally resilient.*

Proof. Observe that $2t < \mathcal{K}(G,D)$ implies the existence of a minimum $(2t+1)$-level ordering $\mathcal{L}_{2t+1}(G,D)$. Let $\mathcal{L}_{2t+1}(G,D)$ be the partition $\{L_1,\ldots,L_m\}$ of V, i.e. $V = \bigcup_{i=1}^m L_i$. It suffices to show that for $1 \le i \le m$, every honest player $v \in L_i$ decides on the dealer's value x_D. By strong induction on i:

Every honest player $v \in L_1 = \mathcal{N}(D)$ decides on the dealer's value x_D due to the CPA steps 1 and 2. If all honest players $u \in L_i, 1 \le i \le h$, decide on x_D at some round, then every honest player $v \in L_{h+1}$ receives $|\bigcup_{j=1}^h L_j \cap \mathcal{N}(v)| \ge 2t+1$ messages from its decided neighbors in previous levels and at least $t+1$ of them are honest. Thus v decides on x_D. A graphical representation of the condition can be seen in Figure 2.2.

□

Corollary 2.4 (Lower Bound)**.** *For any graph G and dealer D it holds that $t_{\max}^{\mathrm{CPA}} \ge \lceil \mathcal{K}(G,D)/2 \rceil - 1$*

2.5.2 Non-tightness of the lower bound

In Theorem 2.3 we proved that $t < \mathcal{K}(G,D)/2$ is sufficient for CPA to be t-locally resilient; we next prove that it is not a necessary condition. Intuitively, the reason is that the topology of the graph may prevent the adversary from corrupting t players in *each* player's neighborhood, hence some players will correctly decide by executing CPA even if they have only $t+1$ neighbors in previous levels.

Proposition 2.5. *There exists a family of instances (G,D), such that CPA is $(\mathcal{K}(G,D)-1)$-locally resilient for (G,D).*

Proof. Figure 2.3 provides such an instance for each value of t. In this instance the neighborhood of D consists of $2t^2+2t$ nodes, nodes $v_1,\ldots,v_{2t}$ form a clique of size $2t$ and are connected with $\mathcal{N}(D)$ as

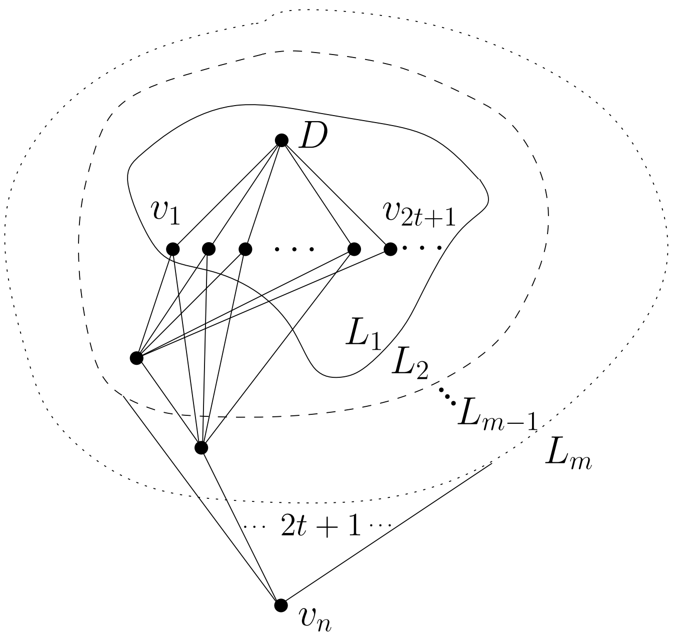


FIGURE 2.2: Level partition based on parameter $\mathcal{K}$

shown in the figure. We can easily check that $t = \mathcal{K}(G, D) - 1$. If we run CPA on G then any player $v_i \in \{v_1, \ldots, v_{2t}\}$ receives M correct messages, with

$$M = M_A + M_B \tag{1}$$

where, M_A = number of messages received from $\mathcal{N}(D)$ and
M_B = number of messages received from $B = \{v_1, \ldots, v_{2t}\} \setminus \{v_i\}$.

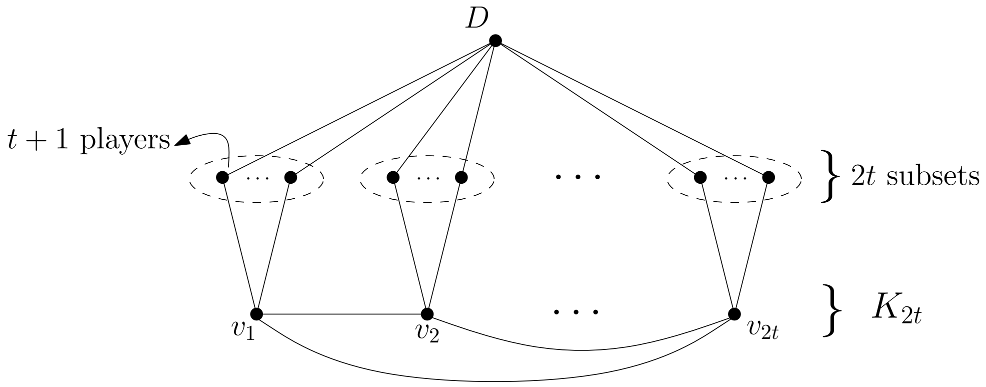


FIGURE 2.3: Graph with $\mathcal{K}(G, D) = t + 1$, for which CPA is t-locally resilient.

Let $T_i = T \cap \mathcal{N}(D) \cap \mathcal{N}(v_i)$ be the set of traitors that are common neighbors of D and v_i. Then

$$M_A = |\mathcal{N}(D) \cap \mathcal{N}(v_i) \setminus T_i| = t + 1 - |T_i| \tag{2}$$

In order to compute the number of correct messages that v_i receives from players in B, we define the sets:

$$\begin{aligned} C_{B_1} &= \{v \in B \mid v \text{ receives at most } t \text{ messages from } \mathcal{N}(D)\ \} \\ C_{B_2} &= \{v \in B \mid v \text{ is corrupted } \} \\ C_B &= C_{B_1} \cup C_{B_2} \end{aligned}$$

We observe that C_{B_1} becomes maximum in cardinality if the adversary corrupts exactly one player in every set $\mathcal{N}(v_j) \cap \mathcal{N}(D), \forall v_j \in B$. Therefore $\max\limits_{T:t\text{-local set}} |C_{B_1}| = \max\limits_{T:t\text{-local set}} |T \cap (\mathcal{N}(D) \setminus \mathcal{N}(v_i))| = t - |T_i|$. Also $|C_{B_2}| \leq t - |T_i|$ because B and $\mathcal{N}(v_i) \cap \mathcal{N}(D)$ form the neighborhood of v_i where the corruptions can be at most t. Next we compute an upper bound on C_B.

$$|C_B| = |C_{B_1} \cup C_{B_2}| \leq |C_{B_1}| + |C_{B_2}| \leq (t - |T_i|) + (t - |T_i|) = 2t - 2|T_i|$$

and thus,

$$M_B = 2t - 1 - |C_B| = 2t - 1 - 2t + 2|T_i| = 2|T_i| - 1 \tag{3}$$

Finally we can compute the total number of messages M,

$$\begin{aligned} (1), (2), (3) \Rightarrow M = M_A + M_B \geq t + 1 - |T_i| + 2|T_i| - 1 = \\ = t + |T_i| \end{aligned}$$

For any v_i, if $|T_i| > 0$ then $M \geq t+1$. Otherwise $|T_i| = 0$ and v_i receives $t+1$ correct messages from $\mathcal{N}(D)$. Thus CPA successfully achieves Broadcast on (G, D).

$\square$

2.6 An Upper Bound on Max CPA Resilience

In the previous section we have shown that $t_{\max}^{\text{CPA}} \geq \lceil \mathcal{K}(G, D)/2 \rceil - 1$; we have also demonstrated cases in which $\mathcal{K}(G, D) - 1$ traitors are locally tolerated by CPA. In this section we will show that the latter is the best possible: $\mathcal{K}(G,D)-1$ is an upper bound on the number of local traitors for any G and D. We do this by proving a necessary condition for CPA to be t-locally resilient.

Theorem 2.6 (Necessary Condition)**.** *For any graph G, dealer D and $t \geq \mathcal{K}(G,D)$, CPA is not t-locally resilient.*

Proof. Assume that CPA is t-locally resilient, with $t \geq \mathcal{K}(G,D)$. Since, by assumption, CPA is t-locally resilient there must be a positive integer, let s, so that the algorithm terminates after s steps in G. Consider now the operation of CPA on graph G in terms of sets. Let L_i denote the set of nodes that decide in the i-th round. Since every node in L_i decides at the i-th round we get that it has at least $t+1$ neighbors in sets $L_1,\ldots,L_{i-1}$. That is,

$$\forall v \in L_i \Rightarrow |\mathcal{N}(v) \cap \bigcup_{j=1}^{i-1} L_j| \geq t+1.$$

Observe that the above sequence is a relaxed $(t+1)$-level ordering for G, D. From the above observation and according to the Theorem 2.2 we get that there must be a minimum $(t+1)$-level ordering for G, D. But this is a contradiction since we assumed that $t \geq \mathcal{K}(G,D)$.

□

Corollary 2.7 (Upper bound on $t^{\text{CPA}}_{\max}$)**.** *For any graph G and dealer D it holds that $t^{\text{CPA}}_{\max} < \mathcal{K}(G,D)$*

2.6.1 Comparison with the Ichimura-Shigeno parameter

In [27], Ichimura and Shigeno introduce a graph theoretic parameter $\widetilde{\mathcal{X}}(G,D)$ which can be used to obtain a sufficient condition for CPA resilience. For a graph $G=(V,E)$ and dealer D, they consider a total ordering $\sigma=(v_1,v_2,\ldots)$ of the set $V\setminus(\mathcal{N}(D)\cup D)$, and use $\delta(W_i,v)$ to denote the number of neighbors that v has in the set $\mathcal{N}(D)\cup\{v_1,\ldots,v_{i-1}\}$. The total ordering σ has the property that $\forall i,j,$ *with* $1\leq i<j\leq|V\setminus(\mathcal{N}(D)\cup D)|$ it holds that $\delta(W_{i-1},v_i)\geq\delta(W_{i-1},v_j)$. This ordering is also referred to as *max-back ordering*. They define parameter $\widetilde{\mathcal{X}}(G,D)=\min\{\delta(W_{i-1},v_i)\mid i=1,2,\ldots\}$. and

prove that it is unique, i.e., is the same for all max-back orderings. They essentially prove that,[2]

$$\lceil \widetilde{\mathcal{X}}(G,D)/2 \rceil - 1 \leq t_{\max}^{\mathrm{CPA}} < \widetilde{\mathcal{X}}(G,D). \qquad (1)$$

Hence, their parameter gives similar bounds as ours. We next show that there is a good reason for this coincidence: despite the different way of defining the parameters $\mathcal{K}(G,D)$ and $\widetilde{\mathcal{X}}(G,D)$, they prove to be equal.

Theorem 2.8. $\mathcal{K}(G,D) = \widetilde{\mathcal{X}}(G,D)$

Proof. Consider the max-back ordering $\sigma = (v_1, v_2, \ldots)$. Then the sequence $\{L_1 = \mathcal{N}(D), L_2 = \{v_1\}, L_3 = \{v_2\}, \ldots\}$ is trivially a relaxed $\widetilde{\mathcal{X}}(G,D)$-level ordering, because the minimum connectivity between a level and its predecessors is $\widetilde{\mathcal{X}}(G,D)$. Thus, due to Proposition 2.2, there exists a minimum $\widetilde{\mathcal{X}}(G,D)$-level ordering, therefore $\mathcal{K}(G,D) \geq \widetilde{\mathcal{X}}(G,D)$. Thus, combining the last inequality with inequality (1) we get the following:

$$t_{\max}^{\mathrm{CPA}} < \widetilde{\mathcal{X}}(G,D) \leq \mathcal{K}(G,D)$$

Since Proposition 2.5 implies that there is a graph for which CPA is $(\mathcal{K}(G,D)-1)$-locally resilient the above relation yields the equality of $\mathcal{K}(G,D)$ and $\widetilde{\mathcal{X}}(G,D)$, since $\widetilde{\mathcal{X}}(G,D) < \mathcal{K}(G,D)$ would lead to $t_{\max}^{\mathrm{CPA}} < \mathcal{K}(G,D) - 1$, a contradiction.

□

Although the two parameters $\mathcal{K}(G,D)$ and $\widetilde{\mathcal{X}}(G,D)$ are equal, the fact that $\mathcal{K}(G,D)$ is defined in a completely different way leads to an improved complexity of computing it, as we will see in the next section.

2.7 Approximation of Max CPA Resilience

Let us now consider the approximability of computing the *Max CPA Resilience*; we will give an efficient 2-approximation algorithm. We first show how to check if there exists a minimum m-level ordering, for a graph G and dealer D, using a slight variation of the standard

[2]Note that the condition $t \leq \widetilde{\mathcal{X}}(G,D)$ was given as necessary in [27]; however their proof can be easily modified to show the tighter bound $t < \widetilde{\mathcal{X}}(G,D)$, implying the right part of (1).

BFS algorithm. Subsequently, we obtain the approximation by simply computing $\mathcal{K}(G, D)$, using the above check. The ratio follows immediately, by combining Corollaries 2.4 and 2.7.

Protocol 2: ***Existence check of a minimum*** m***-level ordering***

On input (G, D, m) do the following:

1. Assign a zero counter to each node.
2. Enqueue the dealer and every one of its neighbors.
3. Dequeue a node and increase the counters of all its neighbors. Enqueue a neighbor only if its counter is at least m.
4. Repeat Step 3 until the queue is empty.
5. If all nodes have been enqueued then output '*True*' (a minimum m-level ordering exists); otherwise, output '*False*'.

Note that the above algorithm can be modified to compute the minimum m-level ordering $\mathcal{L}_m(G, D)$.

Protocol 3: ***2-Approximation of*** $t_{\max}^{\mathrm{CPA}}$

---

1. Compute $\mathcal{K}(G, D)$: since $\mathcal{K}(G, D) < \min\limits_{v \in V \setminus (\mathcal{N}(D) \cup D)} \deg(v) = \delta$, the exact value of $\mathcal{K}(G, D)$ is computed by $\log \delta$ repetitions of the existence check, by simple binary search.
2. Return $\lceil \mathcal{K}(G, D)/2 \rceil - 1$

Since $t \geq \mathcal{K}(G, D) \Rightarrow$ CPA is not t-locally resilient, it holds that $t_{\max}^{\mathrm{CPA}} < \mathcal{K}(G, D)$, consequently, the returned value is at least $\lceil t_{\max}^{\mathrm{CPA}}/2 \rceil - 1$.

A tight example for the approximation ratio of the algorithm is in fact given by the instance in Figure 2.3 in which we present a graph for which $\mathcal{K}(G, D) = t + 1$ and CPA is t-locally resilient.

The complexity of the above approximation algorithm is obviously given by the complexity of the computation of $\mathcal{K}(G, D)$. As explained above the algorithm requires at most $\log \delta$ executions of the existence check. The latter requires $O(|E|)$ time (same complexity as BFS). Altogether, we get that the time complexity of the algorithm is $O(|E| \log \delta)$, which significantly improves upon the complexity bound for the equivalent parameter $\widetilde{\mathcal{X}}(G, D)$ given in [27]; the complexity stated there is $O(|V|(|V| + |E|))$.

2.8 Determining $t_{\max}^{\mathrm{CPA}}$ Exactly

In this section we present a procedure to compute the exact value of $t_{\max}^{\mathrm{CPA}}$. To this end, we introduce two new graph parameters.

For a corruption set (t-local set) T and graph $G = (V, E)$ we will denote with $G_{\bar{T}} = (V \setminus T, E')$ the node induced subgraph of G on the node set $V \setminus T$.

Definition 2.11 (t-safety threshold). *For any graph G, dealer D and positive integer t, the t-safety threshold is the quantity*

$$\mathcal{M}(G, D, t) = \min_{T:\ t\text{-local set}} \mathcal{K}(G_{\bar{T}}, D)$$

.

Theorem 2.9 (Necessary and Sufficient Condition). *For a graph $G = (V, E)$ and dealer D, CPA is t-locally resilient if and only if $\mathcal{M}(G, D, t) \geq t + 1$.*

Proof. ($\Leftarrow$) Assume $\mathcal{M}(G, D, t) \geq t + 1$ and let $T \subseteq V \setminus D$ be any t-local corruption set. It must hold that $\mathcal{K}(G_{\bar{T}}, D) \geq t + 1$. Hence, there exists a minimum $(t + 1)$-level ordering $\mathcal{L}_{t+1}(G_{\bar{T}}, D) = \{L_1, \ldots, L_m\}$. Therefore every honest player v has at least $t + 1$ honest neighbors in previous levels of $\mathcal{L}_{t+1}(G_{\bar{T}}, D)$; by a simple induction we can show that v will decide on the dealer's value x_D.

($\Rightarrow$) If CPA is t-locally resilient then for any t-local corruption set, T, we have that every honest player in $G_{\bar{T}}$ decides on x_D and let the total number of rounds for the termination of the protocol is $m \in \mathbb{N}$. Define the sequence of sets $L_i = \{v \in V \setminus T \mid v \text{ decides in round } i \text{ of CPA }\}, i \in \{1, \ldots, m\}$. Then we will show by induction that the sequence $(L_i)_{i=1}^{m}$ is the (unique) minimum $(t+1)$-level ordering on graph $G_{\bar{T}}$ with dealer D. Note first that $L_1 = \mathcal{N}(D) \setminus T$ because the players that decide in round 1 are exactly the neighborhood of the dealer. For the induction basis, we observe that $L_2 = \{v \in V \setminus T \mid \mathcal{N}(v) \cap L_1 \geq t+1\}$ because the players that decide in round 2 are exactly those who will receive $t+1$ identical messages from decided players in round 1. Assuming now that $L_k = \{v \in V \setminus \{T \cup \bigcup_{j=1}^{k-1} L_j\} : |\mathcal{N}(v) \cap \bigcup_{j=1}^{k-1} L_j| \geq t+1\}$ it turns out that $L_{k+1} = \{v \in V \setminus \{T \cup \bigcup_{j=1}^{k} L_j\} : |\mathcal{N}(v) \cap \bigcup_{j=1}^{k} L_j| \geq t+1\}$ due to the fact that the players that decide in round $k+1$ are exactly the players who receive at least $t+1$ messages from previously decided players. Since the above hold for any T, the claim follows.

□

For exactly determining the maximum CPA resilience $t_{\max}^{\mathrm{CPA}}$ we need the parameter,

$$\mathcal{T}(G, D) = \max\{t \in \mathbb{N} \mid \mathcal{M}(G, D, t) \geq t+1\}$$

It should be clear by the above discussion that $\mathcal{T}(G, D)$ is exactly the maximum CPA resilience:

Corollary 2.10. $t_{\max}^{\mathrm{CPA}}(G, D) = \mathcal{T}(G, D)$

A simple algorithm to compute the t-safety threshold requires exponential time (consider all the t-local corruption sets and compute $\mathcal{K}(G_{\bar{T}}, D)$ as in Section 2.7). Note that a different necessary and sufficient condition for CPA to be t-locally resilient was independently given in [53]. However, a superexponential time to check that condition is implicit (no algorithm is given in [53]).

Moreover, for computing $t_{\max}^{\mathrm{CPA}} = \mathcal{T}(G, D)$ it suffices to perform at most $\log \delta\, \mathcal{M}(G, D, t)$ computations, where δ is the minimum degree of any node in $V \setminus (\mathcal{N}(D) \cup D)$.

2.9 CPA Uniqueness in *Ad Hoc* Networks

Based on the necessary and sufficient condition for CPA to be t-locally resilient in a graph G with dealer D we can now prove the *CPA uniqueness conjecture* for *ad hoc* networks, which was posed as an open problem in [47]. The conjecture states that no algorithm can locally tolerate more traitors than CPA in networks of unknown topology.

We consider only the class of *t-locally safe* Broadcast algorithms which never cause a node to decide on an incorrect message under any t-local corruption set, cf.[47]

We consider the *ad hoc* network model, in which the nodes know only their own labels and the labels of their neighbors. Often, knowledge of the dealer's label is assumed; this assumption can be omitted in our setting if we assume that the dealer sends its id along with the message x_D.. We call a distributed Broadcast algorithm that operates under these assumptions an *ad hoc Broadcast algorithm.*

Theorem 2.11. *Let $\mathcal{A}$ be a t-locally safe ad hoc Broadcast algorithm. If A is t-locally resilient for a graph G with dealer D then CPA is t-locally resilient for G, D.*

Proof. From Theorem 2.9 we have that, if CPA is not t-locally resilient in (G, D) then, $\mathcal{M}(G, D, t) = \min_{T:\ t\text{-local set}} \mathcal{K}(G_{\bar{T}}, D) \leq t$ which implies that there exists a t-local corruption set T s.t. in the remaining graph $G_{\bar{T}}$ a minimum $(t+1)$-level ordering does not exist. From the definition of the $(t+1)$-level ordering we have that given the sequence of subsets of the nodes $V_{\bar{T}} = V \setminus (T \cup \{D\})$,

$$L_1 = \mathcal{N}_{G_{\bar{T}}}(D),\ \ L_i = \{v \in V_{\bar{T}} \setminus \bigcup_{j=1}^{i-1} L_j : |\mathcal{N}_{G_{\bar{T}}}(v) \cap \bigcup_{j=1}^{i-1} L_j| \geq t+1\}, 2 \leq i \leq m$$

there exists $h \in \mathbb{N}$ s.t. $\forall j \geq h,\ L_j = \emptyset$ and $\bigcup_{i=1}^{h} L_i \subsetneq V_{\bar{T}}$. We denote with $h_{\min}$ the minimum $h \in \mathbb{N}$ with the above property. We can assume wlog that $h_{\min} \geq 2$, because $h = 1$ implies that in the graph $G_{\bar{T}}$ the dealer D is disconnected from the rest of the graph which in turn, trivially implies that no algorithm will achieve Broadcast under the corruption of set T.

Let $A = \bigcup_{i=1}^{h_{\min}} L_i$ and $B = V_{\bar{T}} \setminus A$. It is now obvious from the definition of the minimum $(t+1)$-level ordering that $\forall w \in B,\ |\mathcal{N}_{G_{\bar{T}}}(w) \cap A| \leq t$. Moreover $\bigcup_{i=1}^{h_{\min}} L_i \subsetneq V_{\bar{T}}$ implies that $B \neq \emptyset$. Finally let $H = \bigcup_{w \in B} (\mathcal{N}_{G_{\bar{T}}}(w) \cap A)$ and observe that H constitutes a node-cut in graph $G_{\bar{T}}$ separating the dealer D from the subgraph B. The partition of graph G in the three subgraphs A, B, T is depicted in Figure 3.1.

Let G' be a graph that results from G if we remove edges (u, v) from the set $E' = \{(u, v) | u, v \in A \cup T\}$ s.t. the set H becomes t-local in G' (e.g. we can remove all edges that connect nodes in the set $A \cup T$). The existence of a set of edges that guarantees such a property is implied by the fact that $\forall w \in B,\ |\mathcal{N}_{G_{\bar{T}}}(w) \cap H| \leq t$.

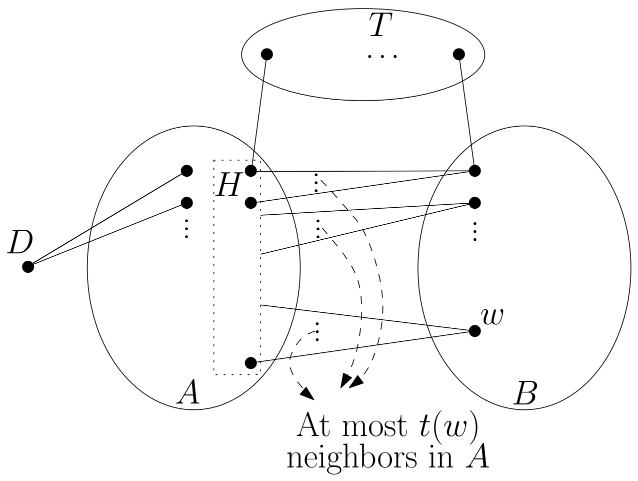


FIGURE 2.4: Partition of G in the subgraphs A, B, T

The proof is by contradiction. Suppose that there exists a t-locally safe Broadcast algorithm $\mathcal{A}$ which is t-locally resilient in graph G with dealer D. We consider the following executions σ and σ' of $\mathcal{A}$,

- Execution σ is on the graph G with dealer D, for the dealer's value we have that $x_D = 0$, the corruption set is the set T and in each round, all the players in this set perform the actions that are instructed to perform in the respective round of execution σ' where T is a set of honest players.
- Execution σ' is on the graph G' with dealer D, for the dealer's value we have that $x_D = 1$, the corruption set is the set H and in each round, all the players in this set perform the actions that are instructed to perform in the respective round of execution σ where H is a set of honest players.

Note that the corruption sets T, H are admissible corruption sets in G, G' respectively due to their t-locality. It is easy to see that the set $H \cup T$ is a node-cut which separates D from B in both G and G' and actions of all nodes of this cut are identical in both executions σ, σ'. Consequently the actions of any honest node $w \in B$ must be identical in both executions. Since by our assumption algorithm $\mathcal{A}$ is t-locally resilient on G with dealer D, w must decide on the dealer's message 0 in execution σ on G with dealer D. It must perform the same action in execution σ' on G' with dealer D. However, in this execution the dealer's message is 1. This contradicts the assumption that $\mathcal{A}$ is t-locally safe. $\square$

According to the definition of uniqueness, the theorem actually states that that, CPA is *unique among t-locally safe algorithms*.

Beyond safe algorithms. We can show that if we drop the requirement for t-local safety, then the theorem does not hold. Intuitively, the reason is that an *ad hoc* protocol that assumes certain topological properties for the network may be t-locally resilient in a family of graphs that have the assumed topological properties.

Indeed, Pelc and Peleg [47] introduced another algorithm for the uniform model, the *Relaxed Propagation Algorithm* (RPA) which uses knowledge of the topology of the network and they proved that there exists a graph G'' with dealer D for which RPA is 1-locally resilient and CPA is not. So if we use RPA in an *ad hoc* setting assuming that the network is G'' then this algorithm will be t-locally resilient for (G'', D) while CPA will not. Non-t-local safety of RPA follows from the fact that the decisions depend on the assumed topology and therefore they could be incorrect if the topological assumptions do not hold. More specifically, a player, running RPA, could decide on a message which she receives from $2t+1$ disjoint paths and for which she can verify, from the assumed topology, that at most t may contain corrupted nodes. However, if the topology is actually not as assumed, then it could even be the case that all $2t+1$ paths contain corrupted nodes and thus the decision value is incorrect. The fact that the non-safe algorithm RPA is resilient in instances where CPA is not, shows that there exist non-safe algorithms of higher resilience than CPA.

2.10 Conclusions

Since the existence of a t-locally resilient Broadcast algorithm in a graph G with dealer D obviously depends on the topology of G, for a given local number of corruptions t we may define and compare the classes of graphs (with a designated dealer-node) determined by the properties and topological conditions that have appeared in the literature so far, including the ones defined in this chapter. An overview of the corresponding classes and their relation is depicted in Figure 3.4. Parameters $LPC(G, D)$ and $\mathcal{X}(G, D)$ are defined in [47] and $\widetilde{\mathcal{X}}(G, D)$ is from [27]. Note that the condition $t < LPC(G, D)$, studied in [47], was proved necessary for every algorithm to achieve Broadcast in the t-locally bounded model.

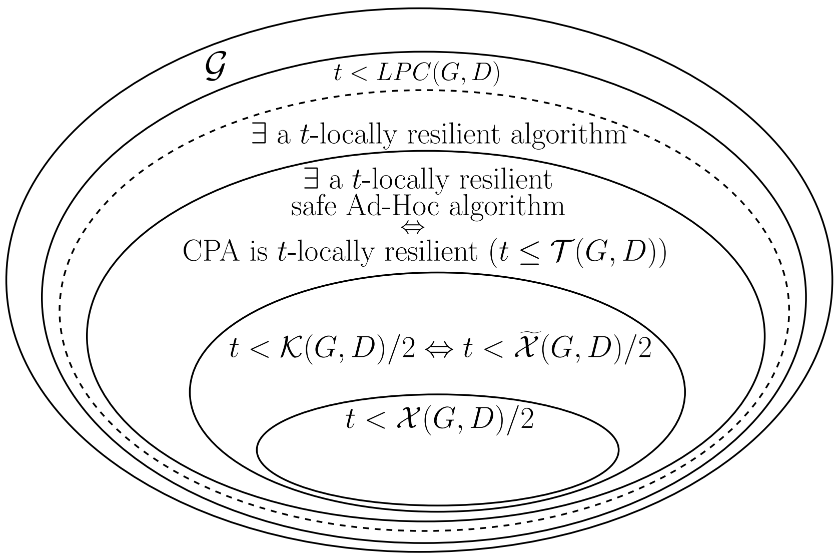

$\mathcal{G}$
$t < LPC(G, D)$
$\exists$ a t-locally resilient algorithm
$\exists$ a t-locally resilient safe Ad-Hoc algorithm $\Leftrightarrow$ CPA is t-locally resilient ($t \leq \mathcal{T}(G, D)$)
$t < \mathcal{K}(G, D)/2 \Leftrightarrow t < \widetilde{\mathcal{X}}(G, D)/2$
$t < \mathcal{X}(G, D)/2$

FIGURE 2.5: Overview of conditions related to the existence of t-locally resilient algorithms. Continuous lines show strict inclusions.

The General Case (Corrupted Dealer). As has been explained, CPA achieves Broadcast under the assumption that the dealer is honest. In order to address the case in which the dealer is corrupted one may observe that if the total number of traitors is strictly less than $n/3, n = |V|$, and the number of traitors in each node's neighborhood is bounded by $\min_{D \in V} \mathcal{T}(G, D)$ then we can achieve Broadcast by simulating any protocol for complete graphs as follows: each one-to-many (or even one-to-one) transmission is simulated by an execution of CPA. We observe that $\min_{D \in V} \mathcal{T}(G, D)$ may not be tight in this case. We can obtain a better bound if we define $\mathcal{M}(G, D, t)$ by considering only corruption sets of size strictly less than $n/3$. Subsequently, we derive an upper bound for Broadcast with corrupted dealer, namely $t \leq \min\left(\lceil n/3 \rceil - 1, \min_{D \in V} \mathcal{T}(G, D)\right)$. The deduction of a tight bound on the number of corrupted players as well as the study of more efficient algorithms for this problem are interesting open questions.

An other open question that arises from the studies of this chapter is to determine another efficiently computable parameter yielding more tight bounds than $\mathcal{K}(G, D)$ in order to obtain an efficient approximation algorithm for $t_{\max}^{\text{CPA}}$ of ratio smaller than 2. Finally, the t-locally bounded adversary model has been mainly considered in the wireless network setting where one also has to deal with collisions (cf. [30, 35, 31]). In this line of work however, the authors only have considered networks of specific topology (grid networks) and it may be interesting to extend these results to the generic topology case considering the results of this chapter.

Chapter 3

Partial Knowledge and Propagation Cuts

In this chapter we further our study on the Reliable Broadcast problem in incomplete networks against a Byzantine adversary. We examine the problem under the *locally bounded adversary model* of Koo (2004) and the *general adversary model* of Hirt and Maurer (1997) and explore the tradeoff between the level of topology knowledge and the solvability of the problem. In order to explore this tradeoff we introduce the *partial knowledge model* which captures the situation where each player has arbitrary topology knowledge.

We refine the local pair-cut technique of Pelc and Peleg (2005) in order to obtain impossibility results for every level of topology knowledge and any type of corruption distribution. On the positive side we devise protocols for every case of topology knowledge; the protocols match the obtained bounds for the extreme cases of *ad hoc* networks and full topology knowledge and thus, exactly characterize the classes of graphs in which Reliable Broadcast is possible in these cases. Concluding the chapter, we generalize our results, for the extreme cases of knowledge level, in the *general adversary model* of Hirt and Maurer (1997) which subsumes earlier models by adapting our techniques and algorithms.

Among others, our generalized pair-cut notion allows for an alternative, to that of Chapter 2, and more intuitive proof of CPA *uniqueness conjecture* [47], which states that CPA can tolerate as many local corruptions as any other safe algorithm. As has been stated previously, safe algorithms are algorithms which never cause a node to decide on an incorrect value. Note that the proof on CPA uniqueness presented in this chapter is the first published proof on the problem and first appeared in [44]. Lastly, we provide an adaptation of CPA achieving reliable broadcast against general adversaries and prove that this algorithm too is unique under this model. To the best of our knowledge this is the first optimal algorithm for Reliable Broadcast in generic topology *ad hoc* networks against general adversaries.

3.1 Outline

In this chapter we study the tradeoff between the level of topology knowledge and the solvability of the problem, under various adversary models. In the course of this study we consider the natural class of *safe* Broadcast algorithms, i.e., algorithms that never cause a player to decide on an incorrect value. The importance of this class of algorithms has been argued in [47] and further analyzed in the previous Chapter.

We first consider a natural generalization of the t-locally bounded model, namely the *non-uniform t-locally bounded model* which subsumes the (uniform) model studied so far. The new model allows for a varying bound on the number of corruptions in each player's neighborhood. In Section 3.3, we address the issue of locally resilient *ad hoc* Broadcast in the non-uniform model. We present a new necessary and sufficient condition for CPA to be t-locally resilient by extending the notion of *local pair cut* of Pelc and Peleg [47] to the notion of *partial local pair cut*. Note that although equivalent conditions exist [55, 38], the simplicity of the new condition makes it highly adaptable to different adversary and topology knowledge models. Although the equivalent condition presented in Chapter 2 easily yields some interesting approximation results, the condition we present in this chapter allows for extending the solvability results in different models. We also present an alternative and more intuitive proof for the open question of CPA Uniqueness [47] which states that if any safe algorithm achieves Broadcast in an *ad hoc* network then so does CPA. Moreover we show that computing the validity of the condition is NP-hard and observe that the latter negative result also has a positive aspect, namely that a polynomially bounded adversary is unable to design an optimal attack unless $\mathrm{P} = \mathrm{NP}$.

In Section 3.4, we shift focus on networks of known topology and devise an optimal resilience protocol, which we call *Path Propagation Algorithm* (PPA). Using PPA we prove that a topological condition which was shown in [47] to be necessary for the existence of a Broadcast algorithm is also sufficient. Thus, we manage to exactly characterize the class of networks for which there exists a solution to the Broadcast problem. On the downside, in [44] we have proven that it is NP-hard to compute an essential decision rule of PPA, rendering the algorithm impractical. However, in the same work we also have provided an indication that probably no efficient protocol of optimal resilience exists, by showing that efficient algorithms through which players always take the same decisions as they would if they ran PPA do not exist if $\mathrm{P} \neq \mathrm{NP}$.

We then take one step further in Section 3.5, by considering a hybrid between *ad hoc* and known topology networks: each node knows a part of the network, namely a connected subgraph containing itself. We propose a protocol for this setting as well, namely the *Generalized Path Propagation Algorithm* (GPPA). We use GPPA to show that this *partial knowledge* model allows for Broadcast algorithms of increased resilience.

Finally, in Section 3.6, we study the general adversary model and show that an appropriate adaptation of CPA is unique against general adversaries in *ad hoc* networks. To the best of our knowledge this is the first algorithm for Reliable Broadcast in generic topology *ad hoc* networks against a general adversary. We show an analogous result for known topology networks, which however can be obtained implicitly from [36] as mentioned above. We conclude by discussing how to extend our results to the case of a corrupted dealer by simulating Broadcast protocols for complete networks.

A central tool in our work is a refinement of the local pair-cut technique of Pelc and Peleg [47] which proves to be adequate for the exact (in most cases) characterization of the class of graphs for which Broadcast is possible for any level of topology knowledge and type of corruption distribution. A useful by-product of practical interest is that the refined cuts can be used to determine the exact subgraph in which Broadcast is possible under any corruption set.

For clarity we have chosen to present our results for the t-local model first (Sections 3.3,3.4, 3.5), for which proofs and protocols are somewhat simpler and more intuitive, and then for the more involved general adversary model (Section 3.6). This chapter includes results presented in [44, 45].

3.2 Model and Definitions

We will now formally define the adversary model by generalizing the notions originally developed in [30, 47] and presented in Chapter 2. We will also define basic notions and terminology that we will use throughout the chapter. We refer to the participants of the protocol by using the terms *node* and *player* interchangeably.

Corruption function. Taking into account that each player might be able to estimate her own upper bound on the corruptions of its neighborhood, as discussed earlier, we introduce a model in which the maximum number of corruptions in each player's neighborhood may vary from player to player. We thus generalize the standard t-locally bounded model [30] in which a uniform upper bound on the number of local corruptions was assumed. Here we consider $t : V \to \mathbb{N}$ to be a *corruption function* over the set of players V.

Non-Uniform t-Locally Bounded Adversary Model. The network is represented by a graph $G = (V, E)$ and one player $D \in V$ is the dealer (sender) as explained before. A corruption function $t : V \to \mathbb{N}$ is also given, implying that an adversary may corrupt at most $t(u)$ nodes in the neighborhood $\mathcal{N}(u)$ of each node $u \in V$. The family of *t-local* sets (defined below) plays an important role in our study since it coincides with the family of admissible corruption sets.

Definition 3.1 (t-local set)**.** *Given a graph $G = (V, E)$ and a function $t : V \to \mathbb{N}$ a t-local set is a set $C \subseteq V$ for which $\forall u \in V$, $|\mathcal{N}(u) \cap C| \leq t(u)$. For $V' \subseteq V$ a t-local w.r.t. V' set is a set $C \subseteq V$ for which $\forall u \in V'$, $|\mathcal{N}(u) \cap C| \leq t(u)$.*

Uniform vs Non-Uniform Model. Obviously the original t-locally bounded model corresponds to the special case of t being a constant function. Hereafter we will refer to the original t-locally bounded model as the *Uniform Model* as opposed to the *Non-Uniform Model* which we introduce here. Hereafter we will also refer to the Non-Uniform Model simply as the t-locally bounded model.

In our study we will often make use of node-cuts which separate some players from the dealer, i.e., node-cuts that do not include the dealer. From here on we will simply use the term *cut* to denote such a node-cut. The notion of *t-local pair cut* was introduced in [47] and is crucial in defining the bounds for which correct dissemination of information in a network is possible.

Definition 3.2 (t-local pair cut). *Given a graph $G = (V, E)$ and a function $t : V \to \mathbb{N}$, a pair of t-local sets C_1, C_2 s.t. $C_1 \cup C_2$ is a cut of G is called a t-local pair cut.*

The next definition extends the notion of t-local pair cut and is particularly useful in describing capability of achieving Broadcast in networks of unknown topology (*ad hoc* networks) where each player's knowledge of the topology is limited in its own neighborhood.

Definition 3.3 (t-partial local pair cut). *Let C be a cut of G, partitioning $V \setminus C$ into sets $A, B \neq \emptyset$ s.t. $D \in A$. C is a t-partial local pair cut (t-plp cut) if there exists a partition $C = C_1 \cup C_2$ where C_1 is t-local and C_2 is t-local w.r.t. B.*

In the uniform model the *Local Pair Connectivity* (LPC(G, D)) [47] parameter of a graph G with dealer D, was defined to be the minimum integer t s.t. G has a t-local pair cut. To define the corresponding notion in the non-uniform model we need to define a (partial) order among corruption functions. Nevertheless, as implied by Theorems 3.1 and 3.2, for reasoning about the feasibility of Broadcast it suffices to consider the following decision problem:

Definition 3.4 (*pLPC*). *Given a graph G, a dealer D and a corruption function t determine whether there exists a t-plp cut in G.*

3.2.1 The Partial Knowledge Model

We next introduce the *Partial Knowledge Model* where each player has initial knowledge over an arbitrary subgraph of the actual network G. The Partial Knowledge model was first presented in [44].

The motivation for partial knowledge considerations comes from large scale networks (e.g. the Internet) where topologically local estimation of the power of the adversary may be possible, while global estimation may be hard to obtain due to geographical or jurisdiction constraints. Additionally, proximity in social networks is often correlated with an increased amount of available information, further justifying the relevance of the model.

In this setting each player v only has knowledge of the topology of a certain subgraph G_v of G which includes v. Namely if we consider the family $\mathcal{G}$ of subgraphs of G we use the *view function* $\gamma : V(G) \to \mathcal{G}$, where $\gamma(v)$ represents the subgraph over which player v has knowledge of the topology. We extend the domain of γ by allowing as input a set $S \subseteq V(G)$. The output will correspond to the *joint view* of nodes in S. More specifically, if $\gamma(v) = G_v = (V_v, E_v)$ then $\gamma(S) = G_S = (\bigcup_{v \in S} V_v, \bigcup_{v \in S} E_v)$. The extensively studied *ad hoc* model can be seen as a special case of the Partial Knowledge Model, where we assume that the topology knowledge of each player is limited to its own neighborhood, i.e., $\forall v \in V(G),\ \gamma(v) = \mathcal{N}(v)$.

3.3 Ad Hoc Networks

3.3.1 Certified Propagation Algorithm (CPA)

The Certified Propagation algorithm [30] uses only local information and thus is particularly suitable for *ad hoc* networks. CPA is probably the only safe Broadcast algorithm known up to now for the t-locally bounded model, which does not require knowledge of the network topology or use topology discovery subroutines.

Probably another, more complex, algorithm for this setting could be devised by employing a topology discovery algorithm (e.g. variation of [42]), and then use the topology knowledge obtained to execute some known Broadcast algorithm which requires topology knowledge (e.g. RPA presented in [47]). CPA does not use any topology discovery subroutine; despite its simplicity and minimal propagation (a player only propagates the value she decides to all her neighbors) it proves to be of optimal resilience (unique). The latter means that one cannot achieve better solvability of the problem by employing more complex schemes. Moreover the combination of the results of the current section with those of Sections 3.4, 3.5 imply that there are instances in which the problem is not solvable under the *Ad Hoc* model but is solvable assuming higher level of topology knowledge. This suggests that employing any topology discovery topology algorithm in the *ad hoc* model does not provide any useful information which will affect the solvability of the problem.

Protocol 4, presented here, is a modification of the original CPA that can be employed under the generalized corruption model introduced here. Namely a node v, upon reception of $t(v) + 1$ messages with the same value x from $t(v) + 1$ distinct neighbors, decides on x, sends it to all neighbors and terminates. The description of the protocol follows:

Protocol 4: ***Certified Propagation Algorithm (CPA) for the Non-Uniform Model***

Input (for each node v): Dealer's label D, labels of v's neighbors, corruption bound $t(v)$.
Message format: A single value $x \in X$.

Code for D: send value $x_D \in X$ to all neighbors, decide on x_D and terminate.

Code for $v \in \mathcal{N}(D)$: upon reception of x_D from the dealer, decide on x_D, send it to all neighbors and terminate.

(* *certified propagation rule* *)

Code for $v \notin \mathcal{N}(D) \cup D$: upon reception of $t(v)+1$ messages with the same value x from $t(v)+1$ distinct neighbors, decide on x, send it to all neighbors and terminate.

As has been argued in Section 2.4, CPA is a t-locally safe Broadcast algorithm. The proof for the non-uniform case is identical to that presented for the uniform case.

3.3.2 CPA Uniqueness in *Ad Hoc* Networks

Based on the above definitions we can now give an alternative and more intuitive proof for the *CPA uniqueness conjecture* in *ad hoc* networks, which was posed as an open problem in [47]. The conjecture states that no algorithm can locally tolerate more corrupted nodes than CPA in networks of unknown topology.

We consider only the class of *t-locally safe* Broadcast algorithms. We assume the *ad hoc* network model, in which nodes are assumed to know only their own labels and the labels of their neighbors. We call a distributed Broadcast algorithm that operates under these assumptions an *ad hoc Broadcast algorithm.*

Theorem 3.1 (Sufficient Condition). *Given a graph G, a corruption function t and a dealer D, if no t-plp cut exists, then CPA is t-locally resilient for (G, D).*

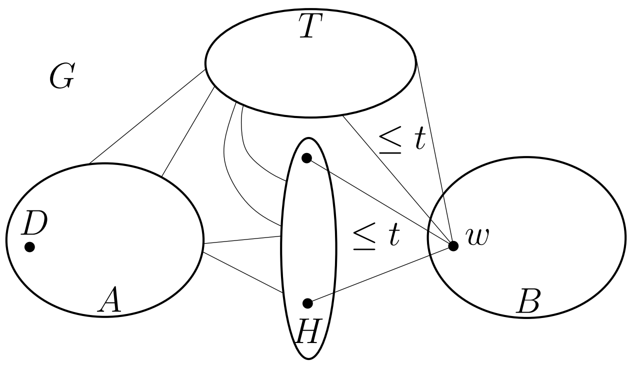
Proof. Suppose that no t-plp cut exists in G. Assume an execution of CPA where the actual corruption set is T. By definition, T is t-local, since we are in the t-locally bounded adversary model; clearly $T \cup \mathcal{N}(D)$ is a cut on G as defined before (i.e. not including node D). Since T is t-local and $T \cup \mathcal{N}(D)$ is not a t-plp cut there must exist $u_1 \in V \setminus (T \cup \mathcal{N}(D) \cup D)$ s.t. $|\mathcal{N}(u_1) \cap (\mathcal{N}(D) \setminus T)| \geq t(u_1) + 1$. Since u_1 is honest and all players in $\mathcal{N}(D) \setminus T$ will trivially decide on the correct value x_D through CPA as direct neighbors of the dealer, u_1 will receive $t(u_1)$ copies of x_D and decide on the *correct* dealer's value x_D. Let us now use the same argument inductively to show that every honest node will eventually decide on the *correct* value x_D through CPA. Let $C_k = (\mathcal{N}(D) \setminus T) \cup \{u_1, u_2, ..., u_{k-1}\}$ be the set of the honest nodes that have decided until a certain round of the protocol, and assume that they decided on the correct value x_D. Then $C_k \cup T$ is a cut. Since T is t-local, by the same argument as before there exists a node u_k s.t. $|C_k \cap \mathcal{N}(u_k)| \geq t(u_k) + 1$ and u_k will decide *correctly* on x_D. Eventually all honest players will *correctly* decide on x_D. Thus CPA is t-locally resilient in G. □

Observe that the latter proof does not explicitly use the fact that CPA is t-locally safe. Instead, we inductively show that in every step (before all terminate), there are some nodes which decide and that all of them decide correctly. A slight modification of the proof can be used as an alternative proof for CPA's t-local safety since in the induction hypothesis we assume that all decided nodes have decided on the correct value.

Theorem 3.2 (Necessary Condition). *Let $\mathcal{A}$ be a t-locally safe ad hoc Broadcast algorithm. Given a graph G, a corruption function t and a dealer D, if a t-plp cut exists, then $\mathcal{A}$ is not t-locally resilient in (G, D).*

Proof. Assume the partition of set V in the sets A, B, T, H such that $C = T \cup H$ is a t-plp cut in graph G with dealer D which disconnects the node sets A, B. Let T be the t-local set of the cut partition and H the t-local w.r.t. to B set (Figure 3.1). Let G' be a graph that results from G if we remove some edges that connect nodes in $A \cup T \cup H$ with nodes in H so that the set H becomes t-local in G' (e.g. we can remove all edges that connect nodes in $A \cup T \cup H$ with nodes in H). Note that the existence of a set of edges that guarantees such a property is implied by the fact that H is t-local w.r.t. B.

The proof is by contradiction. Suppose that there exists a t-locally safe Broadcast algorithm $\mathcal{A}$ which is t-locally resilient in graph G with dealer D. We consider the following executions σ and σ' of $\mathcal{A}$:



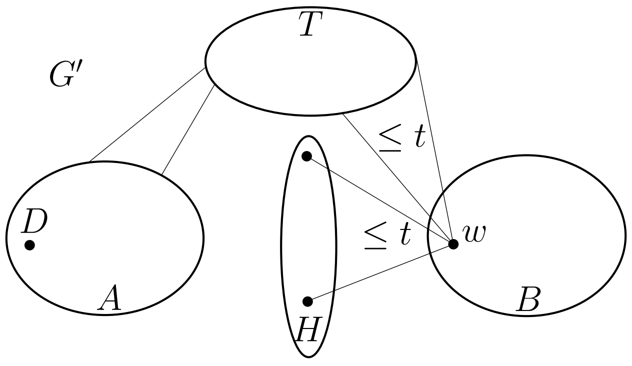


FIGURE 3.1: Graphs G and G'

- Execution σ is on the graph G with dealer D, with dealer's value $x_D = 0$, and corruption set T; in each round, each corrupted player in T performs the actions that its corresponding player performs in the respective round of execution σ' (where T is a set of honest players).
- Execution σ' is on the graph G' with dealer D, with dealer's value $x_D = 1$, and corruption set H; in each round, each corrupted player in H performs the actions that its corresponding player performs in the respective round of execution σ (where H is a set of honest players).

Note that T, H are admissible corruption sets in G, G' respectively due to their t-locality. It is easy to see that $H \cup T$ is a cut which separates D from B in both G and G' and that actions of every node of this cut are identical in both executions σ, σ'. Consequently, the actions of any honest node $w \in B$ must be identical in both executions. Since, by assumption, algorithm $\mathcal{A}$ is t-locally resilient on G with dealer D,

w must decide on the dealer's message 0 in execution σ on G with dealer D, and must do the same in execution σ' on G' with dealer D. However, in execution σ' the dealer's message is 1. Therefore $\mathcal{A}$ makes w decide on an incorrect message in (G', D). This contradicts the assumption that $\mathcal{A}$ is locally safe. □

Note on the proof of Theorem 3.2. Although the argument of the two simultaneous executions σ, σ' is standard in the literature (e.g. [14, 36, 30, 47]), it may seem that the definition of the actions of the corrupted players is circular and thus are not well defined. For ease of presentation we denote with T, H the sets of the execution σ and with T', H' their respective sets in the execution σ'. The circularity of the definition may (falsely) appear in the following example; the actions of T depend on the actions of T' which may in turn depend on the messages they receive from H' which depend on the actions of H in σ which may lastly depend on the actions of T in the same execution. To overcome this obstacle we observe that the actions of all players are uniquely defined in an inductive manner, i.e., in the first round of both executions the actions of honest players in the sets H, T' are uniquely defined by the deterministic protocol $\mathcal{A}$ and their initial values due to the fact that no messages have been received. Therefore, the actions of the first round that the respective corruption sets H', T take are uniquely defined by the actions of H, T'. Assuming that the actions (exchanged messages) of all players are uniquely defined until the end of round k, one can observe that the actions of all players are uniquely defined in round $k+1$ due to the fact that the exchanged messages of round $k+1$ are completely determined by actions taken until round k.

Corollary 3.3 (CPA Uniqueness). *Given a graph G and dealer D, if there exists an ad hoc Broadcast algorithm which is t-locally resilient in (G, D) and t-locally safe, then CPA is t-locally resilient in (G, D).*

Proof. Immediate from Theorems 3.1,3.2. □

This, according to the definition of uniqueness means that, CPA is unique among t-locally safe algorithms.

3.3.3 Hardness of *pLPC*

Ichimura and Shigeno in [27] prove that the *set splitting* problem, known as NP-hard [20], can be reduced to the problem of computing the minimum integer t such that a t-local pair cut exists in a graph G. By generalizing the notion of the t-local pair cut to that of t-plp cut and defining the *pLPC* problem analogously one can use a nearly identical proof to that of [27] and show that the *pLPC* problem is NP-hard.

Theorem 3.4. *pLPC is* NP*-hard.*

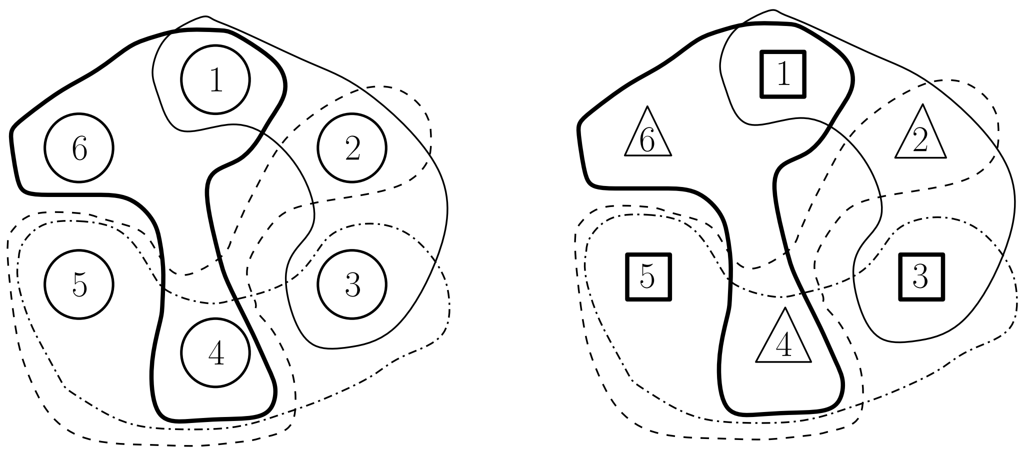


FIGURE 3.2: An instance and the solution of a set splitting problem with $X = \{1, 2, 3, 4, 5, 6\}$ and $A = \{\{1, 2, 3\}, \{3, 4, 5\}, \{1, 4, 6\}, \{2, 4, 5\}\}$. The solution is depicted by the the two sets $X_1 = \{1, 3, 5\}$ and $X_2 = \{2, 4, 6\}$ the elements of which are marked with squares and triangles respectively. Notice that all sets in A have at least one node of both colors.

Proof. We first consider a different (general) version of the pLPC problem which asks if there is a t-plp cut in the graph where no dealer is specified, i.e., if there exists a t-plp cut for any possible dealer-node in the node set. Concluding the proof we will show that if the general pLPC problem is NP-hard then so is our original pLPC problem (with specified dealer).

We first show that the *set splitting* problem known as NP-hard [20] can be reduced to the general pLPC problem. Given a collection S of 3-element subsets of a finite set X, the set splitting problem asks whether there is a partition of X into two subsets X_1 and X_2 such that no subset in S is entirely contained in either X_1 or X_2. An instance of this problem is shown in Figure 3.2.

We propose the following reduction. Let $S+$ be a multiple collection adding dummy subsets $\{v\}$ to S such that the cardinality of $\{s \in$

$S+ : v \in s\}$ is at least six for each $v \in X$. A complete graph with node set $S+$ and a copy of it are denoted by K_{S+} and K'_{S+}, respectively. We denote with $s' \in V(K'_{S+})$ the copy of node $s \in S+$. We construct a graph G_{SSP} (Figure 3.3) with vertex set $V(G_{SSP}) = V(K_{S+}) \cup V(K'_{S+}) \cup X$ and edge set

$$E(G_{SSP}) = E(K_{S+}) \cup E(K'_{S+}) \cup \{(v, s), (v, s') : v \in X, s \in S+, v \in s\}$$

where s' is a copy of s as mentioned above.

We next prove that there is a set splitting of X if and only if there is a 2-plp cut C in G_{SSP}.

For the "only-if" direction it suffices to observe that a partition $X = X_1 \cup X_2$ for which no subset in S is entirely contained in either X_1 or X_2, implies that each of the sets X_1, X_2 will contain at most 2 nodes (elements) that appear in the neighborhood of every node (set) in K_{S+} and K'_{S+} and thus $X = X_1 \cup X_2$ is a 2-plp cut.

For the "if" direction we argue as follows. Considering a 2-plp cut C on G_{SSP} we distinguish between two cases, the case $X \setminus C \neq \emptyset$ and the case $X \setminus C = \emptyset$. In the first case we observe that if a subgraph of G_{SSP} obtained by removing C from G_{SSP} consists of at least two connected components, then C must contain $\mathcal{N}(v) \cap V(K_{S+})$ or $\mathcal{N}(v) \cap V(K'_{S+})$ for each $v \in X \setminus C$. Since each $v \in X$ has at least six neighbors in both $V(K_{S+})$ and $V(K'_{S+})$, for any possible partition of C, either each node in $V(K_{S+}) \setminus C$ or each node in $V(K'_{S+}) \setminus C$ has at least 3 neighbors in some set of the partition. Therefore, since C is a 2-plp cut the case $X \setminus C \neq \emptyset$ cannot hold.

It remains to consider the case of a 2-plp cut $C = C_1 \cup C_2$ where $X \setminus C = \emptyset$, which implies that $X \subseteq C$; note that C_1, C_2 are in fact 2-local due to symmetry. Observe that X also constitutes a cut in G_{SSP}; moreover, in this case both sets $X_i = C_i \cap X, i = 1, 2$, are 2-local (being subsets of the 2-local sets $C_i, i = 1, 2$), hence $X = X_1 \cup X_2$ is a 2-plp cut. Therefore, no set in $s \in S$ can be entirely contained in some $X_i, i = 1, 2$, because $|s| = 3$, hence the corresponding vertex s in K_{S+} (and s' in K'_{S+}) would have three neighbors in X_i contradicting the fact that X_i is a 2-local set. Thus the set splitting instance (S, X) has a solution $X = X_1 \cup X_2$.

We conclude the proof by showing that NP-hardness for *pLPC*(G, t) without a dealer (general case) implies NP-hardness for the case with a dealer D, i.e., problem *pLPC*(G, t, D). Indeed, if *pLPC*(G, t, D) could be solved with a polynomial-time algorithm then solving *pLPC*(G, t, v) for every node $v \in V$ would suffice to build a polynomial algorithm for *pLPC*(G, t). Therefore to compute *pLPC*(G, t, D) is NP-hard. □

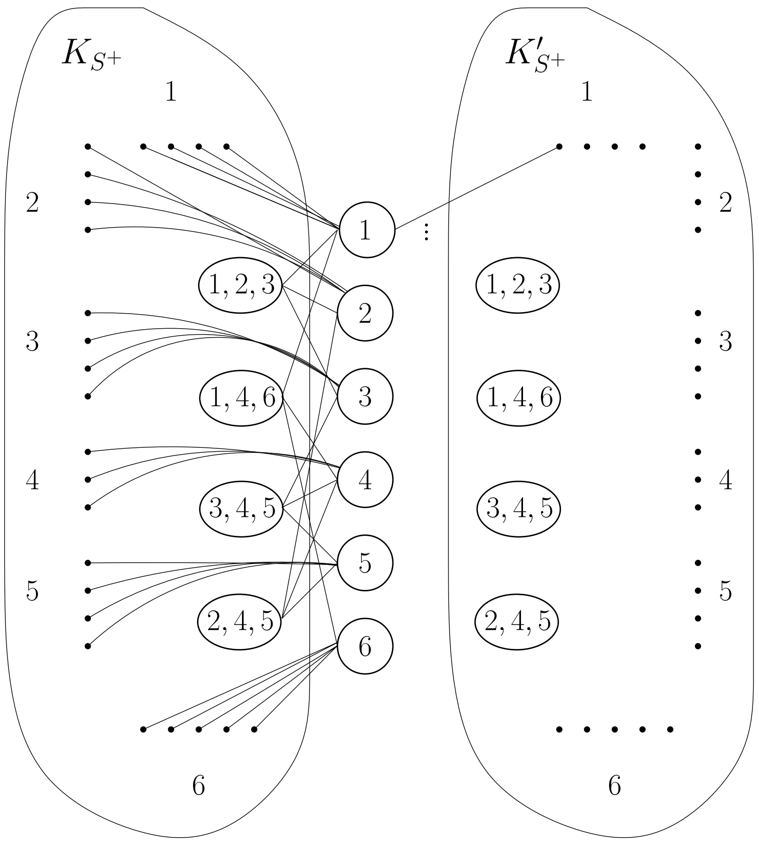


FIGURE 3.3: The graph G_{SSP} for the set splitting problem in Figure 3.2. Edges on the right side are formed symmetrically to those on the left side and are omitted for simplicity.

We have thus established that computing the necessary and sufficient condition for CPA to work is NP-hard. Observe that this negative result also has a positive aspect, namely that a polynomially bounded adversary is unable to always compute an optimal attack unless P = NP.

3.4 Known topology Networks

3.4.1 The Path Propagation Algorithm

Considering only safe Broadcast algorithms, the uniqueness of CPA in the *ad hoc* model implies that an algorithm that achieves Broadcast in cases where CPA does not, must operate under a weaker model e.g., assuming additional information on the topology of the network. It thus makes sense to consider the setting where players have full knowledge of the topology of the network. In this section we propose the *Path Propagation Algorithm* (PPA) and show that is of optimal resilience in the full-knowledge model. For convenience we will use the following notions:

Definition 3.5 (Cover of paths). *A set $S \subseteq V \setminus D$ is called a cover of a set of paths $\mathcal{P}$ if and only if $\forall p \in \mathcal{P}$, $\exists s \in S$ s.t. $s \in p$ (s is a node of p).*

As one can see in the algorithm, each path which is propagated, is transmitted along with a value which it carries. This value corresponds to the value initially sent by the first node of the path (*source* of the path). The other endpoint of the path, i.e., the last node of path p will be denoted with $tail(p)$. When a node v acts as a relay of a value which has reached to it through path p, it appends its id v to the last node of p and thus it creates a new path p' with $tail(p') = v$, whereas the source of p and p' remains the same. The description of PPA follows.

Protocol 5: ***Path Propagation Algorithm (PPA)***

Input (for each node v): dealer's label D, graph G, $t(v)$ = max #corruptions in $\mathcal{N}(v)$.
Message format: pair (x, p), where $x \in X$ (message space), and p is a path of G (message's propagation trail).

Code for D: send message (x_D, D) to all neighbors, decide on x_D and terminate.

```
Code for $v \neq D$: Initialize $decision_v := \perp$.
upon reception of $(x, p)$ from node $u$ do:
    if $(v \in p) \vee (tail(p) \neq u)$ then discard the message else send $(x, p||v)$ [1] to all neighbors.
    if $(decision_v = \perp) \wedge (\mathrm{decision}(v) \neq \perp)$ then
        $decision_v := \mathrm{decision}(v)$;
        send message $(decision_v, v)$ to all neighbors;
        decide on $decision_v$.
```

function decision(v)

[1]By $p||v$ we denote the path consisting of path p and node v, with the last node of p connected to v.

```
    (* dealer propagation rule *)
    if $v \in \mathcal{N}(D)$ and $v$ receives $(x_D, D)$ then return $x_D$.
    (* honest path propagation rule *)
    if $v$ receives messages $(x, p_1), \ldots, (x, p_m)$ and $\nexists$ $t$-local cover of
a path set $\mathcal{P} \subseteq \{p_1, \ldots, p_m\}$
        then return $x$ else return $\perp$.
```

Concerning the path propagation rule of PPA, note that $\mathcal{P}$ is not the whole set of collected paths received by v but rather any subset of the paths through which v receives a value x. Observe that each player can check the validity of the honest path propagation rule only if it has knowledge of the corruption function t and the network's topology. Next, we argue about the safeness of PPA.

Theorem 3.5. *PPA is t-locally safe.*

Proof. We will show that if a player decides on a value x through PPA then $x = x_D$. Assume on the contrary that there is a set of players $V' \subseteq V$ that decide on values different than x_D. Let v be the player of V' that decides in the earliest round among all players in V', i.e., the first player to make an incorrect decision, and assume that v decides on $x \neq x_D$. Player v cannot be a neighbor of the dealer since all neighbors of the dealer only decide on x_D as can be seen in the respective decision rule of PPA. Therefore v has decided on x through the honest path propagation rule. This means that v received value x from a set of paths $\mathcal{P}$ such that there does not exist a t-local cover of $\mathcal{P}$. Moreover, through the check $tail(p) \neq u$, we ensure that at least one corrupted node will be included in a path which contains faulty nodes. Due to the latter, we avoid the case where all the corrupted nodes hide their identity in a path by changing the actual propagation trail; this is a commonly used idea which was first presented in [14].

Since there does not exist a t-local cover for $\mathcal{P}$, it is now obvious that at least one path p of $\mathcal{P}$ is entirely corruption free and if p is entirely corruption free, then value x, which is relayed through p, is the actual value that the source-node w of p has decided on. Thus, at least one honest player has decided in $x \neq x_D$ before v. A contradiction to the fact that v is the first player to make an incorrect decision.

□

3.4.2 A necessary and sufficient condition

We will now show that the non-existence of a t-local pair cut is a sufficient condition for PPA to achieve Broadcast in the t-locally bounded model in networks of known topology.

Theorem 3.6 (Sufficiency). *Given a graph G with dealer D and corruption function t, if no t-local pair cut exists in (G, D) then all honest players will decide through PPA on x_D.*

Proof. All players in $\mathcal{N}(D)$ decide on x_D due to the *dealer propagation rule*, since the dealer is honest. We next show the rest of the players will decide on x_D due to the *honest path propagation rule*. Observe that since PPA is t-locally safe, it suffices to show that, at some step, every player will receive the correct value x_D through a set of paths $\mathcal{P}$ which will allow her to decide on x_D through the honest path propagation rule (if she has not aldready decided on it in a previous step).

Let v be any player in $V \setminus \mathcal{N}(D)$ and assume that no t-local pair cut exist in (G, D). Let T be a t-local set and consider an execution σ_T of PPA where T is the corruption set. Let $\mathcal{P}_{D,v}$ be the set of all paths connecting D with v that are composed entirely by nodes in $V \setminus T$ (honest nodes). Observe that $\mathcal{P}_{D,v} \neq \emptyset$, otherwise T is a cut separating D from v and T is trivially a t-local pair cut, a contradiction. Since paths in $\mathcal{P}_{D,v}$ are entirely composed by honest nodes it is easy to see that v will receive the correct value x_D through all paths in $\mathcal{P}_{D,v}$. Since $\mathcal{P}_{D,v}$ is a set of paths that propagate the same value to v, player v will check if there is a t-local cover for it, as is dictated by the honest path propagation rule, i.e. $\mathcal{P}_{D,v}$ will coincide with $\mathcal{P}$ as is shown in the algorithm.

We next prove that there does not exist a t-local cover of $\mathcal{P}_{D,v}$. Assume that $\exists T'$: t-local cover of $\mathcal{P}_{D,v}$. Then obviously $T \cup T'$ is a cut separating D from v, since every path that connects D with v contains at least a node in $T \cup T'$. Moreover the cut $T \cup T'$ can be partitioned in the sets $T \setminus T', T'$ which are trivially t-local and thus, $T \cup T'$ is a t-local pair cut, a contradiction. Hence, there does not exist exist a t-local cover of $\mathcal{P}_{D,v}$.

Consequently, in execution σ_T, node v will receive the correct value, in some step k, through every path in $\mathcal{P}_{D,v}$ along with the corresponding propagation trail. If player v has already decided before step k then her decision will certainly be on x_D due to the t-local safety of PPA. In any other case, v will also decide on the correct value x_D by the end of step k due to the honest path propagation rule, because $\mathcal{P}_{D,v}$ is not covered by any t-local set.

$\square$

Using the same arguments as in the proof of the necessity of condition $t < LPC(G, D)$ [47] it can be seen that the non-existence of a t-local pair cut is a necessary condition for any algorithm to achieve Broadcast under the non-uniform model. The proof uses similar arguments with that of Theorem 3.2 but is much simpler; the different executions are considered in the same graph. One cannot consider executions in two different graphs since the topology is known to all the players and the players would be able to distinguish the two scenaria. For completeness the proof is presented below.

Theorem 3.7 (Necessity). *Given a graph G with dealer D and corruption function t, if there exists a t-local pair cut in (G, D) then there is no t-locally resilient algorithm for (G, D).*

Proof. Assume that there exists a t-local pair cut $C = C_1 \cup C_2$ in (G, D) partitioning $V \setminus C$ into sets $A, B \neq \emptyset$ such that $D \in A$. Also let $\mathcal{A}$ be a t-locally resilient algorithm for (G, D). We will show that $\mathcal{A}$ does not allow any $v \in B$ to correctly decide on the value of the dealer x_D in all possible executions, which contradicts its t-local resilience. Consider the following two executions σ and σ' of $\mathcal{A}$ on the instance (G, D).

- In execution σ the dealer's value is $x_D = 0$ and the corrupted players are precisely those in C_1. In each round $t \geq 1$ of the execution σ, every player in C_1 performs the action that she is instructed to perform in round t of execution σ' (where she is honest).
- In execution σ' the dealer's value is $x_D = 1$ and the corrupted players are precisely those in C_2. In each round $t \geq 1$ of the execution σ', every player in C_2 performs the action that she is instructed to perform in round t of execution σ (where she is honest).

The same standard argument of the two simultaneous executions is used here. Its correctness regarding the unambiguous definition of the players' actions is proved in Section 3.3.2 in the paragraph "*Note on the proof of Theorem 3.2*".

Similarly with the proof of Theorem 3.2, it follows that any player $v \in B$ performs identical actions in executions σ and σ' of $\mathcal{A}$. Hence

v decides on the same value in σ and σ', which cannot be correct in both executions, since D has a different initial value in each of them.

□

Thus the non-existence of a t-local pair cut proves to be a necessary and sufficient condition for the existence of a t-locally resilient algorithm in both the uniform and the non-uniform model. Therefore PPA is of optimal resilience.

3.5 Partial knowledge

Until now we have presented optimal resilience algorithms for Broadcast in two extreme cases, with respect to the knowledge over the network topology: the *ad hoc* model and the full-knowledge model. A natural question arises: is there any algorithm that works well in settings where nodes have partial knowledge of the topology?

To address this question we introduce the *partial knowledge model*, where each player has restricted knowledge over the topology of the network and devise a new, generalized version of PPA that can run with partial knowledge of the topology of the network. More specifically, as explained in the Section 3.2.1 we assume that each player v only has knowledge of the topology of a certain subgraph G_v of G which includes v. As has been defined, we represent the assignment of initial topology knowledge to all players by the *topology view function* γ where $\gamma(v)$ represents the subgraph over which player v has knowledge of the topology. We will also use the *joint view* $\gamma(S)$ of a set S, which has also been defined in Section 3.2.1 and represents the subgraph which yields if all nodes in S combine their topology knowledge. We will call an algorithm which achieves Broadcast for any t-local corruption set in graph G with dealer D and view function γ, (γ, t)*-locally resilient* for (G, D).

GPPA algorithm. Now given a corruption function t and a view function γ we define the Generalized Path Propagation Algorithm (GPPA) to work exactly as PPA apart from a natural modification of the honest path propagation rule. The modified decision rule will be denoted as *generalized path propagation rule* an is explained in the following.

Generalized path propagation rule. Player v receives the same value x from a set $\mathcal{P}$ of paths that are entirely contained in the subgraph $\gamma(v)$ and is able to deduce (from the topology) that no t-local cover of $\mathcal{P}$ exists.

Remark. Note that GPPA generalizes both CPA and PPA. Indeed, if $\forall v \in V,\ \gamma(v) = \mathcal{N}(v)$, then GPPA$(G, D, t, \gamma)$ coincides with CPA(G, D, t). If, on the other hand, $\forall v \in V,\ \gamma(v) = G$ then GPPA(G, D, t, γ) coincides with PPA(G, D, t).

We also notice that, quite naturally, as γ provides more information for the topology of the graph, resilience increases, with CPA being of minimal resilience in this family of algorithms, and PPA achieving maximal resilience.

To prove necessary and sufficient conditions for GPPA being t-locally resilient we need to generalize the notion of t-plp cut as follows:

Definition 3.6 (type 1 (γ, t)-partial local pair cut). *Let C be a cut of G, partitioning $V \setminus C$ into sets $A, B \neq \emptyset$ s.t. $D \in A$. C will be called a type 1 (γ, t)-partial local pair cut (plp1 cut) if there exists a partition $C = C_1 \cup C_2$ s.t. C_1 is t-local and $C_2 \cap \gamma(B)$ is t-local in the graph $\gamma(B)$.*

Definition 3.7 (type 2 (γ, t)-partial local pair cut). *Let C be a cut of G, partitioning $V \setminus C$ into sets $A, B \neq \emptyset$ s.t. $D \in A$. C will be called a type 2 (γ, t)-partial local pair cut (plp2 cut) if there exists a partition $C = C_1 \cup C_2$ s.t. C_1 is t-local and $\forall u \in B$, $C_2 \cap \mathcal{N}(u)$ is t-local in the graph $\gamma(u)$.*

We can now show the following two theorems. The proofs build on the techniques presented for CPA and PPA.

Theorem 3.8 (sufficient condition). *Let t be corruption function and γ be a view function, if no (γ, t)-plp2 cut exists in G with dealer D then GPPA(G, D, t, γ) is (γ, t)-locally resilient for G, D.*

Proof. Suppose no (γ, t)-plp2 cut exists. Assume an execution of GPPA where the actual corruption set is T. By definition, T is t-local, since we are in the t-locally bounded adversary model; clearly $T \cup \mathcal{N}(D)$ is a cut on G as defined before (i.e. not including node D). Since T is t-local and $T \cup \mathcal{N}(D)$ is not a (γ, t)-plp2 cut there must exist $u_1 \in V \setminus (T \cup \mathcal{N}(D) \cup D)$ s.t. $\mathcal{N}(D) \cap \mathcal{N}(u_1)$ is not t-local on $\gamma(u_1)$. But since all the honest nodes in $\mathcal{N}(D) \cap \mathcal{N}(u_1)$ have decided correctly as neighbors of the dealer, u_1 will receive the value x_D from paths of length 1, starting from these nodes. Finding a t-local corruption set covering these paths is impossible since it would have to include all these nodes, and from the above, it would not be t-local. So u_1 will decide on the dealer's value x_D. We can use the same argument inductively to show that every honest node will eventually decide on the correct value x_D through GPPA. Let $C_k = (\mathcal{N}(D) \setminus T) \cup \{u_1, u_2, ..., u_{k-1}\}$ be the set of the nodes that have decided until a certain round of the protocol and assume that they have decided correctly on x_D. Then $C_k \cup T$ is a cut. Since T is t-local by the same argument as before there exists an undecided node u_k s.t. $C_k \cap \mathcal{N}(u_k)$ is not t-local on $\gamma(u_k)$. Using the same argument as before u_k will decide on the correct value. Eventually all honest players will decide on x_D. Thus *GPPA* is t-locally resilient in G. □

Again, as in the proof of Theorem 3.1, observe that in this proof we do not use the fact that GPPA is safe but rather prove inductively that in the case discussed all nodes will decide correctly.

Theorem 3.9 (necessary condition). *Let t be a corruption function, γ be a view function and $\mathcal{A}$ be a t-locally safe ad hoc Broadcast algorithm. If a (γ, t)-plp1 cut exists in graph G with dealer D, then $\mathcal{A}$ is not (γ, t)-locally resilient for G, D.*

Proof. Assume that there exists a (γ, t)-plp1 cut $C = T \cup H$ in graph G with dealer D and with T being the t-local set of the partition (Figure 3.1). $\gamma(B)$ is the joint view of the nodes in B. G' is the graph that results from G if we remove edges from $A \setminus \gamma(B)$ s.t. the set H becomes t-local in G'. The existence of a set of edges that guarantees such a property is implied by the second property of the (γ, t)-plp1 cut. Suppose that there exists a t-locally safe Broadcast algorithm $\mathcal{A}$ which is t-locally resilient in graph G with dealer D. We can argue the same way we did on Theorem 3.2 which leads to a contradiction. □

One can argue that increased topology knowledge implies increased resilience for GPPA compared to CPA; for example, the sufficient condition of GPPA holds in settings where the sufficient condition of CPA does not hold. An overview of our results concerning the t-local model with respect to the level of topology knowledge appears in Figure 3.4.

Notice that the reason for which GPPA is not optimal is that nodes in $\gamma(v)$ do not share their knowledge of topology. An optimal resilience protocol should also include exchange of topological knowledge among players. Such a protocol is presented in the next chapter and is indeed proved unique for the partial knowledge model.

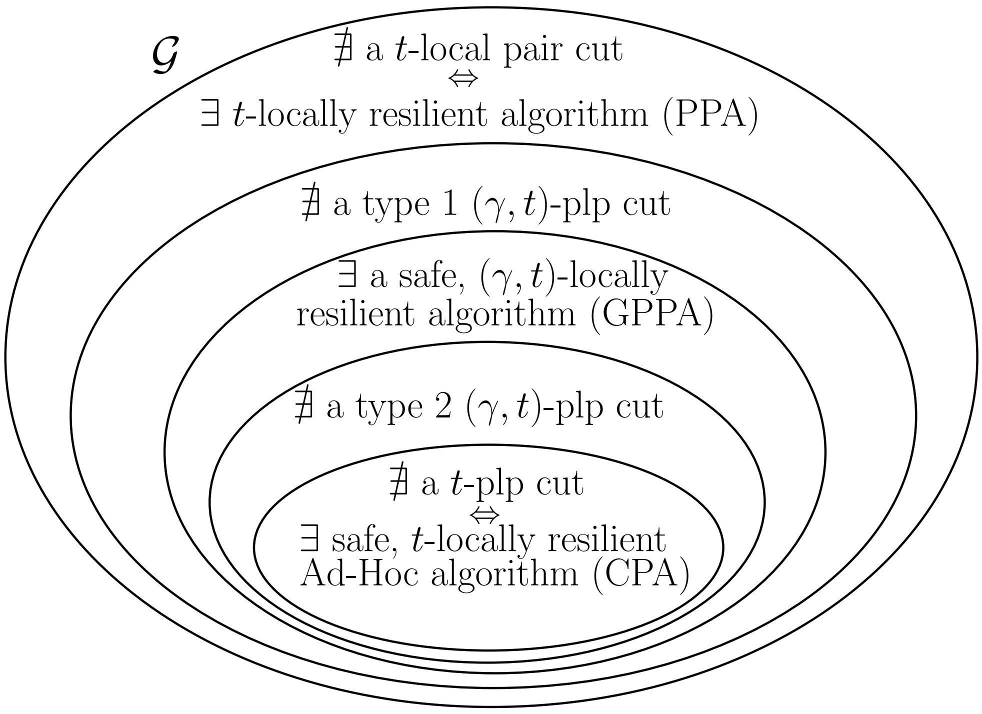

$\mathcal{G}$
$\nexists$ a t-local pair cut
$\Leftrightarrow$
$\exists$ t-locally resilient algorithm (PPA)
$\nexists$ a type 1 (γ, t)-plp cut
$\exists$ a safe, (γ, t)-locally resilient algorithm (GPPA)
$\nexists$ a type 2 (γ, t)-plp cut
$\nexists$ a t-plp cut
$\Leftrightarrow$
$\exists$ safe, t-locally resilient Ad-Hoc algorithm (CPA)

FIGURE 3.4: Overview of conditions concerning the existence of t-locally resilient algorithms with respect to the level of topology knowledge. Note that $\mathcal{G}$ refers to the family of pairs (G, D).

3.6 General Adversary

We next shift focus to the *general adversary model* of Hirt and Maurer [26] in order to generalize our results in this model.

General adversary model. Hirt and Maurer in [26] study the security of multiparty computation protocols with respect to an *adversary structure*, that is, a family of subsets of the players; the adversary is able to corrupt one of these subsets. More formally,

A structure $\mathcal{Z}$ for the set of players V is a monotone family of subsets of V, i.e. $\mathcal{Z} \subseteq 2^V$, where all subsets of a set $Z \in \mathcal{Z}$ are in $\mathcal{Z}$ too, (alternatively, $\forall Z \in \mathcal{Z}$, if $Z' \subseteq Z$ then it holds that $Z' \in \mathcal{Z}$).

Let us now redefine some notions that we have introduced in this chapter in order to extend our results to the case of a general adversary. We will call an algorithm that achieves Broadcast for any corruption set $T \in \mathcal{Z}$ in graph G with dealer D, *$\mathcal{Z}$-resilient*. A cover

$S \in \mathcal{Z}$ of a set of paths $\mathcal{P}$ will be called a $\mathcal{Z}$-cover. We next generalize the notion of a t-local pair cut.

Definition 3.8 ($\mathcal{Z}$-pair cut). *A cut C of G for which there exists a partition $C = C_1 \cup C_2$ and $C_1, C_2 \in \mathcal{Z}$ is called a $\mathcal{Z}$-pair cut of G.*

3.6.1 Known Topology Networks

We adapt PPA in order to address the Broadcast problem under a general adversary. The Generalized $\mathcal{Z}$-PPA algorithm can be obtained by a modification of the path propagation rule of PPA (Protocol 5).

$\mathcal{Z}$-PPA Honest Path Propagation Rule. Player v receives the same value x from a set $\mathcal{P}$ of paths and is able to deduce that for any $T \in \mathcal{Z}$, T is not a cover of $\mathcal{P}$.

Moreover, the following theorems hold and their proofs are essentially the same as the proofs of Theorems 3.6, and 3.7. The only technical modification in the proofs is that one should replace the notions of t-local pair cut, t-local set, t-local cover, with that of $\mathcal{Z}$-pair cut, admissible corruption set (or set which belongs to $\mathcal{Z}$) and $\mathcal{Z}$-cover respectively.

Theorem 3.10 (Sufficiency). *Given a graph G, dealer D, and an adversary structure $\mathcal{Z}$, if no $\mathcal{Z}$-pair cut exists, then all honest players will decide on x_D through $\mathcal{Z}$-PPA.*

Theorem 3.11 (Necessity). *Given a graph G, dealer D, and an adversary structure $\mathcal{Z}$, if there exists a $\mathcal{Z}$-pair cut then there is no $\mathcal{Z}$-resilient Broadcast algorithm for (G, D).*

3.6.2 *Ad Hoc* Networks

Since in the *ad hoc* model the players know only their own labels, the labels of their neighbors and the label of the dealer it is reasonable to assume that a player has only local knowledge on the actual adversary structure $\mathcal{Z}$. Specifically, given the actual adversary structure $\mathcal{Z}$ we assume that each player v knows only the *local adversary structure* $\mathcal{Z}_v = \{A \cap \mathcal{N}(v) : A \in \mathcal{Z}\}$. A similar assumption was used in [54].

As in known topology networks, we can describe a generalized version $\mathcal{Z}$-CPA of CPA, which is an *ad hoc* Broadcast algorithm for the general adversary model. In particular, we modify step (3) of CPA (Protocol 4) in the following way.

$\mathcal{Z}$-CPA certified propagation rule. If a node v is not a neighbor of the dealer, then upon receiving the same value x from all its neighbors in a set $\mathcal{N} \subseteq \mathcal{N}(v)$ s.t. $N \notin \mathcal{Z}_v$, it decides on value x.

The proof of $\mathcal{Z}$-CPA safety is essentially the same as the proof of safety of the original CPA, presented in Theorem 2.1. Observe that the $\mathcal{Z}$-CPA certified propagation rule a very clear generalization of the CPA certified propagation rule. It actually states that if a player receives the same value x from a set of neighbors which cannot all be corrupted, the player can safely decide on this message, because at least one of them is definitely honest and sends the correct value.

In order to argue about the topological conditions which determine the effectiveness of $\mathcal{Z}$-CPA we generalize the notion of partial t-local pair cut.

Definition 3.9 ($\mathcal{Z}$-partial pair cut)**.** *Let C be a cut of G partitioning $V \setminus C$ into sets $A, B \neq \emptyset$ s.t. $D \in A$. C is a $\mathcal{Z}$-partial pair cut ($\mathcal{Z}$-pp cut) if there exists a partition $C = C_1 \cup C_2$ with $C_1 \in \mathcal{Z}$ and $\forall u \in B$, $\mathcal{N}(u) \cap C_2 \in \mathcal{Z}_u$.*

Analogously to CPA Uniqueness, we can now prove $\mathcal{Z}$-CPA Uniqueness in the general adversary model. We present an alternative proof, a modification of which can also be used for the proof of Theorem 3.1.

Theorem 3.12 (Sufficient Condition)**.** *Given a graph G, dealer D, and an adversary structure $\mathcal{Z}$, if no $\mathcal{Z}$-pp cut exists, then $\mathcal{Z}$-CPA is $\mathcal{Z}$-resilient.*

Proof. Suppose that $\mathcal{Z}$-CPA is not $\mathcal{Z}$-resilient. Then there exists a scenario where C are the corrupted nodes, A are the honest and decided nodes, and B are the honest undecided nodes. All nodes in A have decided on the correct value because $\mathcal{Z}$-CPA is safe. Since every node in B is undecided we have that $\forall u \in B : \mathcal{N}(u) \cap A \in \mathcal{Z}_u$, otherwise u would have decided because a set of nodes that are not in Z_u would have sent him the same broadcast value. But then $C \cup A$ is a $\mathcal{Z}$-pp cut which is a contradiction. Hence, $\mathcal{Z}$-CPA is $\mathcal{Z}$-resilient. □

Theorem 3.13 (Necessary Condition)**.** *Let $\mathcal{A}$ be a safe ad hoc Broadcast algorithm. Given a graph G, dealer D, and an adversary structure $\mathcal{Z}$, if a $\mathcal{Z}$-pp cut exists then $\mathcal{A}$ is not $\mathcal{Z}$-resilient for G, D.*

Proof. Let $C = C_1 \cup C_2$ be the $\mathcal{Z}$-pp cut which partitions $V \setminus C$ in sets $A, B \neq \emptyset$ s.t. $D \in A$. Let $\mathcal{Z}' = \{\bigcup_{u \in B} Z \cap \mathcal{N}(u) : Z \in \mathcal{Z}\} \cup \{C_2\}$.

For every node u in B we have:

$$\begin{aligned}
\mathcal{Z}'_u &= \{Z \cap \mathcal{N}(u) : Z \in \mathcal{Z}'\} \cup \{C_2 \cap \mathcal{N}(u)\} \\
&= \left\{ \left(\bigcup_{v \in B} Z \cap \mathcal{N}(v) \right) \cap \mathcal{N}(u) : Z \in \mathcal{Z} \right\} \cup \{C_2 \cap \mathcal{N}(u)\} \\
&= \{Z \cap \mathcal{N}(u) : Z \in \mathcal{Z}\} \cup \{C_2 \cap \mathcal{N}(u)\} \\
&= \mathcal{Z}_u
\end{aligned}$$

since $\forall u \in B : \mathcal{N}(u) \cap C_2 \in \mathcal{Z}_u$.

So far we have established that (a) nodes in B cannot tell whether $\mathcal{Z}$ or $\mathcal{Z}'$ is the adversary structure since $\forall u \in B : \mathcal{Z}_u = \mathcal{Z}'_u$ and (b) C_2 is an admissible corruption set in $\mathcal{Z}'$.

Suppose a node in B could decide on some value in the scenario where $\mathcal{Z}$ is the adversary structure. Then using the standard argument employed in Theorem 3.2, an attack on the safeness of the algorithm would be possible in a different scenario where $\mathcal{Z}'$ is the adversary structure. The details of the proof are similar and are based on the difficulty of the honest players in B to distinguish which scenario they participate in, with respect to the adversary structure: the one with $\mathcal{Z}$ or the one with $\mathcal{Z}'$. □

Complexity of $\mathcal{Z}$-CPA.

We will now make a simple but practical observation on the complexity of $\mathcal{Z}$-CPA. We measure the complexity of $\mathcal{Z}$-CPA with respect to the size of the graph $|G|$ only, because it is interesting to consider if CPA is *fully-polynomial* (of polynomial round, communication and local computations complexity) regardless of the size of the adversary structure description. We consider its complexity on the instances where Broadcast is solvable, i.e., there does not exist a $\mathcal{Z}$-pp cut.

Since $\mathcal{Z}$-CPA is trivially of polynomial round and communication complexity it holds that $\mathcal{Z}$-CPA is fully polynomial if its local computations complexity is polynomial. Observe that the local computations of every node essentially are comprised of membership checks dictated by the $\mathcal{Z}$-CPA Certified Propagation rule. Thus given any instance $(G, D, \mathcal{Z})$ of a family of instances $\mathcal{I}$, if there exists a polynomial algorithm $\mathcal{B}$ which given a set $S \subseteq \mathcal{N}(v)$ decides whether

$S \in \mathcal{Z}_v$, for every player v, then $\mathcal{Z}$-CPA, with subroutine $\mathcal{B}$ for membership checks, is fully polynomial in $\mathcal{I}$. Practically, if the description of the adversary structure allows polynomial membership checks on the local adversary structures of all players then $\mathcal{Z}$-CPA is fully polynomial. Such an example is the t-locally bounded adversary model described is the first sections. In that model the description of the adversary structure is $\mathcal{Z} = \{S \in V : \forall v \in V, |S \cap \mathcal{N}(v)| \leq t\}$, which allows efficient local membership checks which essentially constitute of comparing the cardinality of a set with t. A more detailed study on the complexity of $\mathcal{Z}$-CPA is presented in Section 4.4.

3.7 Dealer Corruption

We have studied the problem of Broadcast in the case where the dealer is honest. In order to address the general case in which the dealer may also be corrupted one may observe that for a given adversary structure $\mathcal{Z}$ and graph G, $\mathcal{Z}$-resilient Broadcast in *ad hoc* networks can be achieved if the following conditions both hold:

1. $\nexists Z_1, Z_2, Z_3 \in \mathcal{Z}$ *s.t.* $Z_1 \cup Z_2 \cup Z_3 = V$.
2. $\forall v \in V$ there does not exist a $\mathcal{Z}$-pp cut for G with dealer v.

Condition 1 was proved by Hirt and Maurer [26] sufficient and necessary for the existence of secure multiparty protocols in complete networks. $\mathcal{Z}$-resilient Broadcast in the general case where the network is incomplete can be achieved by simulating any protocol for complete graphs (e.g. the protocol presented in [17]) as follows: each one-to-many transmission is replaced by an execution of $\mathcal{Z}$-CPA. It is not hard to see that the conjunction of the above two conditions is necessary and sufficient for Broadcast in incomplete networks in the case of corrupted dealer. Similarly in networks of known topology, there exists a $\mathcal{Z}$-resilient Broadcast algorithm if condition 1 holds and for every $v \in V$ a $\mathcal{Z}$-pair cut does not exist for graph G with dealer v. Naturally, the above observations hold also in the special case of a locally bounded adversary.

3.8 Conclusions

As we have shown, in both the t-locally bounded adversary and general adversary models the idea of CPA, despite its simplicity and minimal propagation (a player only propagates the value she decides to all her neighbors) yields algorithms which prove to be of optimal resilience (unique). The latter means that one cannot achieve better solvability of the problem by employing more complex propagation schemes. Moreover the results presented in this chapter imply that there are instances in which the problem is not solvable under the *ad hoc* model but is solvable assuming higher level of topology knowledge. This suggests that employing any topology discovery topology algorithm in the *ad hoc* model does not provide any useful information which will affect the solvability of the problem. Since CPA is of optimal resilience, it is also natural to ask whether it provides the more efficient solution for the problem. We deal with the latter question in the next chapter.

An interesting open problem would be to determine the largest class of algorithms among which CPA (respectively $\mathcal{Z}$-CPA) is unique. Previous results (cf. RPA algorithm [47]) combined with our work in this chapter suggest that in order to achieve better solvability than CPA one has to assume additional topological knowledge.

We have shown that necessary and sufficient criteria for Broadcast on known topology and ad-hoc networks are NP-hard to compute. So what is the best attack a polynomially bounded adversary could deploy? Similar issues may be raised from the point of view of system designers. Defining an appropriate meaningful optimization objective on the network resilience is essential in answering such questions.

Chapter 4

Partial Knowledge and Reliable Message Transmission

A fundamental primitive in distributed computing is *Reliable Message Transmission* (RMT), which refers to the task of correctly sending a message from a party to another, despite the presence of byzantine corruptions. In this chapter we address the problem in the general adversary model of Hirt and Maurer, which subsumes earlier models such as the global or local threshold adversaries. Regarding the topology knowledge, we employ the *Partial Knowledge Model*, introduced in the previous chapter, which encompasses both the full knowledge and the *ad hoc* model.

The following contributions are presented in this chapter: (a) A necessary and sufficient condition for achieving RMT in the partial knowledge model with a general adversary; in order to show sufficiency, we propose the RMT-Partial Knowledge Algorithm (RMT-PKA), an algorithm that solves RMT whenever this is possible, therefore it is a *unique* algorithm. To the best of our knowledge, this is the first unique protocol for RMT against general adversaries in the partial knowledge model. (b) A study of efficiency in the case of the *ad hoc* network model: we show that either the $\mathcal{Z}$-CPA protocol, introduced in the previous chapter, is fully polynomial or no unique fully polynomial protocol for RMT exists, thus introducing a new notion of uniqueness with respect to efficiency that we call *poly-time uniqueness*.

To obtain our results we introduce, among others, a *joint view* operation on adversary structures, a new notion of separator (RMT-cut), appropriate for RMT in unreliable networks, and a self-reducibility property of the RMT problem, which we show by means of a protocol composition. The latter plays a crucial role in proving the poly-time uniqueness of $\mathcal{Z}$-CPA.

4.1 Introduction

Achieving reliable communication in unreliable networks is fundamental in distributed computing. Of course, if there is an authenticated channel between two parties then reliable communication between them is guaranteed. However, it is often the case that certain parties are only indirectly connected, and need to use intermediate parties as relays to propagate their message to the actual receiver. The *Reliable Message Transmission* problem (RMT) is the problem of achieving correct delivery of a message m from a *dealer* (sender) D to a receiver R even if some of the intermediate nodes are corrupted and do not relay the message as agreed. In this chapter we consider the Reliable Message Transmission under the existence of a general Byzantine Adversary. The RMT problem has been initially considered by Dolev [14] in the context of the closely related Broadcast problem.

RMT under Byzantine adversaries has been studied extensively in various settings: secure or reliable transmission, general or threshold adversary, perfect or unconditional security. Here we focus on perfectly reliable transmission under a general adversary and the partial knowledge model. More specifically, RMT under a threshold Byzantine adversary, was addressed in [15, 13], where additional secrecy restrictions were posed and in [50] where a probability of failure was allowed. Results for RMT in the general adversary model [26], where given in [36, 51, 52]. In general, very few studies have addressed RMT or related problems in the partial knowledge setting despite the fact that this direction was already proposed in 2002 by Kumar *et al.* [36].

We consider the RMT problem under the General Adversary and the Partial Knowledge model, introduced in the previous chapter.

The strength of the results of this chapter lies in the combination of these two quite general models (general adversary and partial knowledge), forming the most general setting we have encountered so far within the synchronous deterministic model.

Trivially all the aforementioned results for Broadcast with an honest dealer, presented in previous chapters can be adapted for the RMT problem. However, determining a necessary and sufficient condition (tight) for the most general case of the partial knowledge model hasn't been achieved up to now. Moreover these previous studies have focused on feasibility and not efficiency and no complexity studies have been conducted in this context. The latter two issues appeared to be most challenging and are both considered and answered in this chapter.

4.1.1 Outline

We study the RMT problem under general adversaries. Our contribution is twofold:

(a) Feasibility of RMT in the Partial Knowledge model. We prove a necessary and sufficient condition for achieving RMT in this setting, and present an algorithm that achieves RMT whenever this condition is met. In terminology used in previous chapters, this is a *unique* algorithm for the problem, in the sense that whenever any algorithm achieves RMT in a certain instance so does our algorithm. This completes the feasibility study we initiated in Chapter 3. To the best of our knowledge, the first unique algorithm for this general setting.

A key notion that we define and use is the *joint adversary structure* of (a set of) players which corresponds to the worst case adversary structure that conforms to each player's initial knowledge; this notion is crucial in obtaining the tight condition mentioned above since it provides a way to safely utilize the maximal valid information from all the messages exchanged. We also make use of the concept of local pair-cut technique, introduced by Pelc and Peleg [47] in the context of Broadcast. This technique was extended in Chapter 3 in order to obtain characterizations of classes of graphs for which Broadcast is possible for various levels of topology knowledge and type of corruption distribution. However, an exact characterization of teh solvable instances for the partial knowledge setting was left unanswered. Here we answer this question by proposing an adequate pair-cut for the partial knowledge model together with a unique algorithm for RMT, the first unique algorithm for this specific model proposed. This new algorithm is quite general and encompasses earlier algorithms such as CPA [30], PPA and $\mathcal{Z}$-CPA [44], presented in previous chapters, as special cases. A useful by-product of practical interest is that the new cut notion can be used to determine the exact subgraph in which RMT is possible in a network design phase. A remarkable property of our algorithm is its *safety*: even when RMT is not possible the receiver will never make an incorrect decision despite the increased adversary's attack capabilities, which include reporting fictitious topology and false local knowledge among others.

(b) Efficiency of RMT in the *Ad Hoc* network model. We study the *ad hoc* case in terms of efficiency because even in this simple case of partial knowledge, it is not clear whether an efficient (fully polynomial[1]) protocol exists

We propose an adaptation of $\mathcal{Z}$-CPA [44] appropriate for RMT. We prove that this protocol is unique for RMT in the *Ad Hoc* model and is the first such algorithm that we know of. We examine whether and when this algorithm is fully polynomial. We show that no unique fully polynomial protocol for RMT exists if $\mathcal{Z}$-CPA is not fully polynomial, thus introducing a new meaningful notion of *poly-time uniqueness*. In particular, we prove that if $\mathcal{Z}$-CPA is not fully polynomial in any class of instances where RMT is solvable, then there exists a corresponding class of (solvable) simpler instances in which any protocol that achieves RMT cannot be fully polynomial. We obtain this result by showing that $\mathcal{Z}$-CPA yields a polynomial time self-reduction for the RMT problem. Therefore $\mathcal{Z}$-CPA, despite its simplicity and minimal propagation, proves to be at least as efficient (in the sense described above) as any other RMT protocol.

More intuitively, we enhance the uniqueness property of $\mathcal{Z}$-CPA by implicitly stating that, not only one cannot achieve better solvabililty by employing more complex propagation schemes but one cannot even achieve significantly lower complexity in this way. This restriction seems to be inherent in the *ad hoc* network setting where players' knowledge strictly relies on the information received by their neighbors.

This chapter includes results presented in [46, 43].

4.1.2 Model and definitions

In this chapter we address the problem of Perfectly Reliable Message Transmission, hereafter simply referred as Reliable Message Transmission (RMT) under the influence of a general Byzantine adversary. In our model the players have partial knowledge of the network topology and of the adversary structure.

We assume a synchronous network represented by a graph G consisting of the player (node) set $V(G)$ and edge set $E(G)$ which represents authenticated channels between players. The set of neighbors of a player v is denoted with $\mathcal{N}(v)$. The problem definition follows.

[1]A fully polynomial protocol is a protocol of polynomial round, bit and local computations complexity.

Reliable Message Transmission. We assume the existence of a designated player D, called the *dealer*, who wants to propagate a certain value $x_D \in X$, where X is the initial message space, to a designated player R, called the receiver. We say that a distributed protocol achieves (or solves) RMT if by the end of the protocol the receiver R has *decided on* x_D, i.e. if it has been able to output the value x_D originally sent by the dealer.

As in Section 3.6.2, we make the natural assumption that the knowledge of a player over the adversary structure is restricted by its topological knowledge; namely, the combination of the partial knowledge and the general adversary model that we propose regarding the players' knowledge is as follows:

Partial Knowledge with General Adversaries. Considering the partial knowledge model under the existence of a general adversary, we extend the model of Section 3.6.2 and assume that given the actual adversary structure $\mathcal{Z}$ each player v only knows the possible corruption sets in his view $\mathcal{Z}_v = \{A \cap V(\gamma(v)) \mid A \in \mathcal{Z}\}$ (*local adversary structure*).

We denote an instance of the problem by the tuple $\mathcal{I} = (G, \mathcal{Z}, \gamma, D, R)$. Analogously with the protocol properties previously defined we will use the following notions:

Protocol properties. We will say that an RMT protocol is *resilient* for an instance $\mathcal{I}$ if it achieves RMT on instance $\mathcal{I}$ for any possible corruption set and any admissible behavior of the corrupted players. We say that an RMT protocol is *safe* if it never causes the receiver R to decide on an incorrect value in any instance. An algorithm A is unique (for RMT) among algorithms in family $\mathcal{A}$, if the existence of an algorithm of family $\mathcal{A}$ which achieves RMT in an instance $\mathcal{I}$ implies that A also achieves RMT in $\mathcal{I}$.

Similarly with the previous chapter, we make use of node-cuts (separators) which separate the receiver R from the dealer, hence, node-cuts that do not include the dealer. From here on we will simply use the term *cut* to denote such a separator.

4.2 Partial knowledge and general adversaries

Considering two players who have partial knowledge of the adversary, it would be useful to define an operation to calculate their joint knowledge about the adversary. For an adversary structure $\mathcal{E}$ and a node set A let $\mathcal{E}^A = \{Z \cap A \mid Z \in \mathcal{E}\}$ denote the restriction of $\mathcal{E}$ to the set A. The joint adversary structure from two restricted adversary structures can be obtained through the $\oplus$ operator. We define the operation on two possibly different structures $\mathcal{E}, \mathcal{F}$ so that the operation is well defined even if a corrupted player provides a different structure than the real one to an honest player.

Definition 4.1. *Let $\mathbb{T}^A = 2^{2^A}$ denote the space of adversary structures on a set of nodes A. For any node sets A, B and adversary structures $\mathcal{E}, \mathcal{F}$, the operation $\oplus : \mathbb{T}^A \times \mathbb{T}^B \to \mathbb{T}^{(A \cup B)}$, is defined as follows:*

$$\mathcal{E}^A \oplus \mathcal{F}^B = \{Z_1 \cup Z_2 | (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{F}^B) \wedge (Z_1 \cap B = Z_2 \cap A)\}$$

Informally, the $\mathcal{E}^A \oplus \mathcal{F}^B$ operation unites possible corruption sets from $\mathcal{E}^A$ and $\mathcal{F}^B$ that 'agree' on $A \cap B$. In the following, we show that the $\oplus$ operation is commutative, associative and idempotent. A simple example of the $\oplus$ operation is depicted in Figure 4.1.

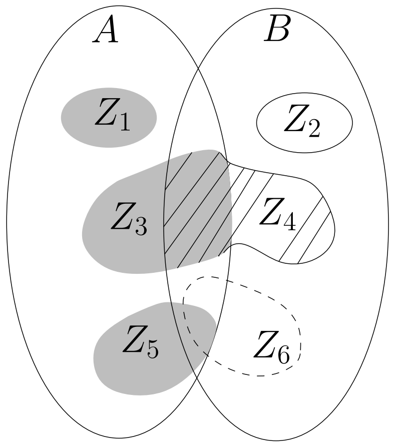


FIGURE 4.1: Assuming that $Z_1, Z_3, Z_5 \in \mathcal{E}^A$ and $Z_2, Z_4, Z_6 \in \mathcal{F}^B$, we observe that $\mathcal{E}^A \oplus \mathcal{F}^B$ should include $Z_1 \cup Z_2, Z_3 \cup Z_4$ but not $Z_5 \cup Z_6$.

Fact. An equivalent definition of the $\oplus$ operation is

$$\mathcal{E}^A \oplus \mathcal{F}^B = \{Z_1 \cup Z_2 \mid (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{F}^B) \wedge (Z_1 \cap B \subseteq Z_2) \wedge (Z_2 \cap A \subseteq Z_1)\}$$

We next show some algebraic properties of this operation.

Theorem 4.1. *Operator $\oplus$ is commutative.*

Proof. For any adversary structures $\mathcal{E}, \mathcal{F}$ and node sets A, B:

$$\begin{aligned}\mathcal{E}^A \oplus \mathcal{F}^B &= \{Z_1 \cup Z_2 \mid (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{F}^B) \wedge (Z_1 \cap B = Z_2 \cap A)\}\\ &= \{Z_2 \cup Z_1 \mid (Z_2 \in \mathcal{F}^B) \wedge (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \cap A = Z_1 \cap B)\}\\ &= \mathcal{F}^B \oplus \mathcal{E}^A\end{aligned}$$

So operator $\oplus$ is commutative. □

Theorem 4.2. *Operation $\oplus$ is idempotent.*

Proof.

$$\begin{aligned}\mathcal{E}^A \oplus \mathcal{E}^A &= \{Z_1 \cup Z_2 \mid (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{E}^A) \wedge (Z_1 \cap A = Z_2 \cap A)\}\\ &= \{Z_1 \cup Z_2 \mid (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{E}^A) \wedge (Z_1 = Z_2)\}\\ &= \{Z_1 | (Z_1 \in \mathcal{E}^A)\}\\ &= \mathcal{E}^A\end{aligned}$$

So operation $\oplus$ is idempotent. □

Theorem 4.3. *Operation $\oplus$ is associative.*

The associativity proof is deferred to the Appendix 6.

The next theorem shows the importance of the $\oplus$ operation in this work.

Theorem 4.4. *For any adversary structures $\mathcal{E}, \mathcal{F}$, node sets A, B and $\mathcal{H} = \mathcal{E}^A \oplus \mathcal{F}^B$, it holds that $\forall \mathcal{H}' \in \mathbb{T}^{A\cup B}$: if $\mathcal{H}'^A = \mathcal{E}^A$ and $\mathcal{H}'^B = \mathcal{F}^B$ then $\mathcal{H}' \subseteq \mathcal{H}$.*

Proof. Suppose that there existed some $\mathcal{H}'$ s.t. $\exists Z \in \mathcal{H}' : Z \notin \mathcal{H}$. For Z we have $Z_1 = Z \cap A \in \mathcal{E}^A$ and $Z_2 = Z \cap B \in \mathcal{F}^B$. Also $Z_1 \cap B = Z \cap A \cap B = Z_2 \cap A$. But then, definition 4.1 implies $Z \in \mathcal{H}$, a contradiction. □

Corollary 4.5. *For any adversary structure $\mathcal{Z}$ and node sets A, B: $\mathcal{Z}^{(A\cup B)} \subseteq \mathcal{Z}^A \oplus \mathcal{Z}^B$.*

What Corollary 4.5 tells us is that the $\oplus$ operation gives the maximal (w.r.t inclusion) possible adversary structure that is indistinguishable by two agents that know $\mathcal{Z}^A$ and $\mathcal{Z}^B$ respectively, i.e., it coincides with their knowledge of the adversary structures on sets A and B respectively. Recall that $\mathcal{Z}_u = \mathcal{Z}^{V(\gamma(u))}$. We will prefer to use $\mathcal{Z}_u$ to denote the local adversary structure of player u and $\mathcal{Z}^{V(\gamma(u))}$ to denote the corresponding restriction of the adversary structure. This allows us to define the combined knowledge of a set of nodes B about the adversary structure $\mathcal{Z}$ as follows. For a given adversary structure $\mathcal{Z}$, a view function γ and a node set B let

$$\mathcal{Z}_B = \bigoplus_{v \in B} \mathcal{Z}^{V(\gamma(v))}$$

Note that $\mathcal{Z}_B$ exactly captures the maximal adversary structure possible, restricted in $\gamma(B)$, relative to the initial knowledge of players in B. Also notice that using Corollary 4.5 we get $\mathcal{Z}^{V(\gamma(B))} \subseteq \mathcal{Z}_B$. The interpretation of this inequality in our setting, is that what nodes in B conceive as the worst case adversary structure indistinguishable to them, it always contains the actual adversary structure in their scenario.

4.3 Reliable message transmission in the partial knowledge model

In RMT we want the dealer D to send a message to some player R (the receiver) in the network. We assume that the dealer knows the id of player R. We denote an instance of the problem by the tuple $(G, \mathcal{Z}, \gamma, D, R)$. To analyze feasibility of RMT we introduce the notion of RMT-cut.

Definition 4.2 (RMT-cut). *Let $(G, \mathcal{Z}, \gamma, D, R)$ be an RMT instance and $C = C_1 \cup C_2$ be a cut in G, partitioning $V \setminus C$ in two sets $A, B' \neq \emptyset$ where $D \in A$ and $R \in B'$. Let $B \subseteq B'$ be the node set of the connected component that R lies in. Then C is a RMT-cut iff $C_1 \in \mathcal{Z}$ and $C_2 \cap V(\gamma(B)) \in \mathcal{Z}_B$.*

We next prove that the non existence of an RMT-cut is a necessary condition for the existence of safe RMT algorithms. The proof combines ideas from from previous impossibility proofs with the $\oplus$ operation and follows in brief.

Theorem 4.6 (Necessity). *Let $(G, \mathcal{Z}, \gamma, D, R)$ be an RMT instance. If there exists a RMT-cut in G then no safe and resilient RMT algorithm exists for $(G, \mathcal{Z}, \gamma, D, R)$.*

Proof. Let $C = C_1 \cup C_2$ be the RMT-cut which partitions $V \setminus C$ in sets $A, B \neq \emptyset$ s.t. $D \in A$ and $R \in B$. Without loss of generality assume that B is connected. If it is not, then by adding all nodes, that do not belong to the connect component of R, to A, a different RMT-cut is built with the desired properties. Consider the instance where $\mathcal{Z}' = \mathcal{Z}_B$ and all other parameters are the same as the previous scenario. Then, all nodes in B have the same initial knowledge in both instances, since $\mathcal{Z}_B = \mathcal{Z}'_B$.

Suppose R could decide correctly with $\mathcal{Z}$ being the actual adversary structure. Then using a standard argument of the two runs and presented in the previous chapter, an attack on the safety of the algorithm would be possible in the same setting with $\mathcal{Z}'$ being the actual adversary structure. The basic idea is that we can construct two indistinguishable scenarios, where the value that the dealer broadcasts is different. The details of the proof are similar and are based on the difficulty of the honest players in B to distinguish which scenario they participate in, with respect to the actual adversary structure: the one with $\mathcal{Z}$ or the one with $\mathcal{Z}'$. □

4.3.1 The RMT-Partial Knowledge Algorithm (RMT-PKA)

We next present the *RMT Partial Knowledge Algorithm* (RMT-PKA), an RMT protocol which succeeds whenever the condition of Theorem 4.6 (in fact, its negation) is met, rendering it a tight condition on when RMT is possible. To prove this we provide some supplementary notions.

Messages comminicated in Protocol 6. In the RMT-PKA protocol there are two type of messages exchanged:

- *Type 1 messages* are used to propagate the dealer's value and are of the form (x, p) where $x \in X$ and p is a path.
- *Type 2 messages* of the form $((v, \gamma(v), \mathcal{Z}_v), p)$ are used for every node v to propagate its initial information $\gamma(v), \mathcal{Z}_v$ throughout the graph.

Let M denote a subset of the messages of type 1 and 2 that the receiver node R receives at some round of the protocol on $(G, \mathcal{Z}, \gamma, D, R)$. We will say that $value(M) = x$ if and only if all the type 1 messages of M report the same dealer value x, i.e., for every such message (y, p), it holds that $y = x$, for some $x \in X$. Observe that M may consist of messages which contain contradictory information. We next define the form of a message set M which contains no contradictory information in our setting (a valid set M).

Definition 4.3 (Valid set M)**.** *A set M of both type 1 and type 2 messages corresponds to a valid scenario, or more simply is valid, if*

- *$\exists x \in X$ s.t. $value(M) = x$. That is, all type 1 messages relay the same x as dealer's value.*
- *$\forall m_1, m_2 \in M$ of type 2, their first component is the same when they refer to the same node. That is, if $m_1 = ((v, \gamma(v), \mathcal{Z}_v), p)$ and $m_2 = (((v', \gamma'(v), \mathcal{Z}'_v), p')$, then $v = v'$ implies that $\gamma(v) = \gamma'(v)$ and $\mathcal{Z}_v = \mathcal{Z}'_v$.*

For every valid M we can define the pair (G_M, x_M) where $x_M = value(M)$. To define G_M let V_M be the set of nodes u for which the information $\gamma(u), \mathcal{Z}_u$ is included in M, namely $V_M = \{v \mid ((v, \gamma(v), \mathcal{Z}_v), p) \in M$ for some path $p\}$. Then, G_M is the node induced subgraph of graph $\gamma(V_M)$ on node set V_M. Therefore, a valid message set M uniquely determines the pair (G_M, x_M). We next propose two notions that we use to check if a valid set M contains correct information.

Definition 4.4 (full message set)**.** *A full message set M is a valid set M that contains all the $D-R$ paths which appear in G_M as part of type 1 messages.*

Definition 4.5 (Adversary cover of set M)**.** *A set $C \subseteq V_M$ is an adversary cover of message set M if C has the following property: C is a cut between D, R on G_M and if B is the node set of the connected component that R lies in, it holds that $(C \cap V(\gamma(B))) \in \mathcal{Z}_B$.*

Protocol 6: *RMT-PKA*

Input (for each node v): dealer's label D, $\gamma(v)$, $\mathcal{Z}_v$.
Message format: type 1 : pair (x, p) or type 2 : pair $((u, \gamma(u), \mathcal{Z}_u), p)$, where $x \in X$ (message space), u the id of some node, $\gamma(u)$ is the view of node u, $\mathcal{Z}_u$ is the adversary structure of node u, and p is a path of G (message's propagation trail).

```
Code for D: send messages ($(value : x_D, \{D\})$) and $((D, \gamma(D), \mathcal{Z}_D), \{D\})$ to all neighbors and terminate.
Code for $v \notin \{D, R\}$: send message $((v, \gamma(v), \mathcal{Z}_v), \{v\})$ to all neighbors.
    upon reception of type 1 or type 2 message $(a, p)$ from node $u$ do:
        if $(v \in p) \vee (tail(p) \neq u)$[2] then discard $(a, p)$ else send $(a, p||v)$ [3] to all neighbours.
Code for R: upon reception of $(x, p)$ from node $u$ do:
    if decision$\neq \perp$ then decide on decision and terminate.

Subroutine decision
    (* dealer propagation rule *)
    if $R \in \mathcal{N}(D)$ and $R$ receives $(x_D, \{D\})$ then return $x_D$.
    (* full message set propagation rule *)
    if $R$ receives a full set $M$ with $value(M) = x$ and $\nexists$ an adversary-cover for $M$
        then return $x$ else return $\perp$.
```

We next show the somewhat counterintuitive safety property of RMT-PKA, i.e., that the receiver will never decide on an incorrect value despite the increased adversary's attack capabilities, which includes reporting fictitious nodes and false local knowledge.

Theorem 4.7 (RMT-PKA Safety). *RMT-PKA is safe.*

Proof. It is trivial to see that the receiver R will not decide on an incorrect dealer value by using the dealer propagation rule (case $R \in \mathcal{N}(D)$) due to the dealer's presumed honesty.

The hard part is to prove that R will not decide on any value $x \neq x_D$ by using the full message set propagation rule (case $R \notin \mathcal{N}(D)$). Let $T \in \mathcal{Z}$ be any admissible corruption set and consider the run e_T of RMT-PKA where T is the actual corruption set. Assume that at some round of e_T, R receives a full message set M' with $value(M') = x \neq x_D$. Since all $D - R$ paths of $G_{M'}$ propagate an incorrect value x it means

[2]We use $tail(p)$ to denote the last node of path p. Checking whether $tail(p) \neq u$ we ensure that at least one corrupted node will be included in a faulty propagation path.

[3]By $p||v$ we denote the appending of path p with node v.

that $C = T \cap V_{M'}$ forms a $D - R$ cut in graph $G_{M'}$, otherwise there would be a $D - R$ path in $G_{M'}$ consisting only of honest nodes and propagating x_D, a contradiction because $value(M') = x$. Since $C \in \mathcal{Z}$, it holds by definition that $C \cap V(\gamma(S)) \in \mathcal{Z}_S$, $\forall S \subseteq V(G)$. Therefore if B is the connected component that R lies in under the partition that C imposes in $G_{M'}$, it holds that $C \cap V(\gamma(B)) \in \mathcal{Z}_B$ due to the fact that B only contains honest nodes; more specifically, B does not contain any corrupted nodes due to the definition of C. Moreover, the adversary cannot introduce any fictitious nodes in B because T has to be a cut between R and every nonexistent node claimed by the adversary. The latter observations about B imply that R can correctly compute $\mathcal{Z}_B$. Thus M' has an adversary cover and R will not decide in value $x \neq x_D$ due to the full message set propagation rule. □

Theorem 4.8 (Sufficiency)**.** *Let $(G, \mathcal{Z}, \gamma, D, R)$ be an RMT instance. If no RMT-cut exists, then RMT-PKA achieves reliable message transmission.*

Proof. Observe that if $R \in \mathcal{N}(D)$ then R trivially decides on x_D due to the *dealer propagation rule*, since the dealer is honest. Assuming that no RMT-cut exists, we will show that if $R \notin \mathcal{N}(D)$ then R will decide on x_D due to the *full message set propagation rule.*

Let $T \in \mathcal{Z}$ be any admissible corruption set and consider the run e_T of RMT-PKA where T is the actual corruption set. Let P be the set of all paths connecting D with R and are composed entirely by nodes in $V(G) \setminus T$ (honest nodes). Observe that $P \neq \emptyset$, otherwise T is a cut separating D from R which is trivially a RMT-cut, a contradiction.

Since paths in P are entirely composed by honest nodes, it should be clear by the protocol that by round $|V(G)|$, R will have obtained x_D through all paths in P by receiving the corresponding type 1 messages M_1. Furthermore, by round $|V(G)|$, R will have received type 2 messages set M_2 which includes information for all the nodes connected with R via paths that do not pass through nodes in T. This includes all nodes of paths in P. Consequently, R will have received the full message set $M = M_1 \cup M_2$ with $value(M) = x_D$.

We next show that there is no adversary cover for M and thus R will decide on x_D through the full message set propagation rule on M. Assume that there exists an adversary cover C for M. This, by definition means that C is a cut between D, R on G_M and if B is the node set of the the connected component that R lies in, it holds that and $(C \cap V(\gamma(B))) \in \mathcal{Z}_B$ (observe that R can compute $\mathcal{Z}_B$ using the information contained in M_2 as defined in the previous paragraph).

Then obviously $T \cup C$ is a cut in G separating D from R, since every path of G that connects D with R contains at least a node in $T \cup C$. Let the cut $T \cup C$ partition $V(G) \setminus \{T \cup C\}$ in the sets A, B s.t. $D \in A$. Then clearly $T \cup C$ is an RMT cut by definition, a contradiction. Thus there is no adversary cover for M and R will decide on x_D.

Moreover, since RMT-PKA is safe, the receiver will not decide on any other value different from x_D.

□

Corollary 4.9 (Uniqueness). *RMT-PKA is unique among safe algorithms, i.e., given an RMT instance $(G, \mathcal{Z}, \gamma, D, R)$, if there exists any safe RMT algorithm which is resilient for this instance, then RMT-PKA also achieves reliable message transmission on this instance.*

4.4 RMT in *ad hoc* networks

In this section we consider the Reliable Message Transmission problem (*RMT*) in *ad hoc* networks. In the closely related problem of Reliable Broadcast the receiver is not a single node but instead the whole set $V(G)$. Reliable Broadcast in *ad hoc* networks under the influence of a general Byzantine adversary was initially studied in Section 3.6.2 where an algorithm for this model was presented and proven unique. The results can trivially be adapted to the case of the RMT problem.

4.4.1 *Ad hoc* RMT

An instance of the RMT problem in the *ad hoc setting* consists of a tuple $(G, \mathcal{Z}, D, R)$ as explained in previous sections. In the closely related problem of Reliable Broadcast with an honest dealer studied in previous sections, the notion of *$\mathcal{Z}$-pp cut* was given and it was proved that a necessary and sufficient condition for the solvability of the problem is that a $\mathcal{Z}$-pp cut does not exist in the instance. Furthermore, the protocol $\mathcal{Z}$-CPA (Certified Propagation Algorithm) was given and proved that it achieves Broadcast in every instance where Broadcast is possible, i.e., it is *unique*.

Since in the RMT problem we are only concerned about the decision of the receiver R, we slightly modify the definition of the $\mathcal{Z}$-pp cut in order to capture an analogous cut (*RMT $\mathcal{Z}$-pp cut*) between the dealer D and the receiver R,

Definition 4.6 (*RMT $\mathcal{Z}$-*pp cut)**.** *Let C be a cut of G partitioning $V \setminus C$ into sets $A, B \neq \emptyset$ s.t. $D \in A$ and $R \in B$. C is an RMT $\mathcal{Z}$-pp cut if there exists a partition $C = C_1 \cup C_2$ with $C_1 \in \mathcal{Z}$ and $\forall u \in B$, $\mathcal{N}(u) \cap C_2 \in \mathcal{Z}_u$.*

The $\mathcal{Z}$-CPA algorithm can be trivially adapted for solving the RMT problem. In this algorithm the dealer first sends its initial value x_D to all its neighbors and terminates. After that the actions of any player v are defined as follows.

RMT $\mathcal{Z}$-CPA code for v

1. If $v \in \mathcal{N}(D)$ then upon reception of x_D from the dealer, decide on x_D.
2. If $v \notin \mathcal{N}(D)$ then upon receiving the same value x from all neighbors in a set $N \subseteq \mathcal{N}(v)$ s.t. $N \notin \mathcal{Z}_v$, decide on value x.
3. If $v = R$ and decided on x then output decision x and terminate, else if $v \neq R$ and decided on x, send x to all neighbors $\mathcal{N}(v)$ and terminate.

Note that $\mathcal{Z}$-CPA is safe, in a sense that, never causes any honest player to decide on an incorrect value. Following an analysis identical to that of [44], where $\mathcal{Z}$-CPA is proven unique among safe Broadcast algorithms, we prove the uniqueness of $\mathcal{Z}$-CPA (modified as explained) among safe *RMT* algorithms. The following theorems are completely analogous with those proving the uniqueness of $\mathcal{Z} - CPA$ for Reliable Broadcast.

Theorem 4.10 (Sufficient Condition)**.** *Given an RMT instance $(G, \mathcal{Z}, D, R)$, if no RMT $\mathcal{Z}$-pp cut exists on G, then $\mathcal{Z}$-CPA achieves RMT in $(G, \mathcal{Z}, D, R)$.*

Proof. Suppose that $\mathcal{Z}$-CPA does not achieve RMT in $(G, \mathcal{Z}, D, r)$. Then we can split the graph in 3 parts: A being the honest decided nodes, B being the honest undecided nodes with $R \in B$ and C being the corrupted nodes. Now since every node in B is undecided we have that $\forall u \in B : N(u) \cap A \in \mathcal{Z}_u$ (otherwise u would have decided). But then $C \cup A$ is an *RMT* $\mathcal{Z}$-pp cut which is a contradiction. Hence, $\mathcal{Z}$-CPA achieves RMT in $(G, \mathcal{Z}, D, R)$. □

Theorem 4.11 (Necessary Condition)**.** *Given an RMT instance $(G, \mathcal{Z}, D, R)$, if an RMT $\mathcal{Z}$-pp cut exists on G then no safe RMT algorithm exists for $(G, \mathcal{Z}, D, R)$.*

The proof is a trivial adaptation of the impossibility proof for Reliable Broadcast and is deferred to the Appendix.

Thus, $\mathcal{Z}$-CPA, the first algorithm we have encountered for RMT in generic topology *ad hoc* networks against general adversaries, proves to be unique.

4.5 Protocol uniqueness with respect to efficiency

Up to now we have seen that $\mathcal{Z}$-CPA is unique among the safe *ad hoc* RMT algorithms. In terms of efficiency it is interesting to study whether $\mathcal{Z}$-CPA is also the more efficient one among unique RMT algorithms w.r.t. polynomial time. We will measure protocol complexity with respect to the size of the graph $|G| = n$ only, because we are mainly interested in protocols that are *fully polynomial* (of polynomial round, bit and local computations complexity) regardless of the size of the adversary structure description. Observe that, if the adversary structure is given explicitly, $\mathcal{Z}$-CPA is trivially *fully polynomial* w.r.t. the total size of the input. However $\mathcal{Z}$ can be of exponential size w.r.t. $|G|$. Measuring the complexity of $\mathcal{Z}$-CPA is not straightforward since the computations included in the protocol are not explicitly defined. In fact, $\mathcal{Z}$-CPA is actually a *protocol scheme* that refers to a “functionally specified" subroutine rather than to an actual (implementation of such a) subroutine. We will use the following notions to facilitate our study on distributed protocol schemes.

First we define the notion of a protocol scheme for a problem Q; essentially, if protocol scheme $\mathcal{A}$ solves problem Q, then $\mathcal{A}$ is in fact a reduction from Q to S in the distributed setting.

Definition 4.7 (Protocol scheme). *A protocol scheme $\mathcal{A}$ is a family of protocols which contains calls to a subroutine X for solving a problem S. The computation of X is not specified, that is, X is used as a black box. Therefore, for every algorithm B which solves problem S a different member (protocol) $\mathcal{A}_B$ of $\mathcal{A}$ is defined; that is, $\mathcal{A}_B$ implements subroutine X through algorithm B.*

Fully polynomial protocol scheme. We will say that a *protocol scheme $\mathcal{A}$ is fully polynomial* if there exists an algorithm B, solving S, for which $\mathcal{A}_B$ is fully polynomial.

Observe that $\mathcal{Z}$-CPA is a protocol scheme which contains the membership check subroutine ($N \notin \mathcal{Z}_v$) appearing in its second rule. Regarding the efficiency of $\mathcal{Z}$-CPA, one can easily observe that it is of polynomial round and bit complexity (details appear in the proof of Theorem 4.12). To argue about the local computations complexity of the scheme we need to take into account the complexity of the membership check subroutine. Indeed, the $\mathcal{Z}$-CPA scheme is fully polynomial if there exists an algorithm B through which the membership check can be performed in polynomial time w.r.t. the size of the input graph $|G|$. We next introduce the property of *poly-time uniqueness*; a protocol scheme that is poly-time unique for a problem is, in a sense, optimally efficient w.r.t. a polynomial factor.

Definition 4.8 (Poly-time Uniqueness). *We call a protocol scheme $\mathcal{A}$ poly-time unique for problem $\mathcal{P}$ if it is unique (with respect to feasibility) and the existence of a unique fully polynomial protocol for $\mathcal{P}$ implies that $\mathcal{A}$ is also fully polynomial for $\mathcal{P}$.*

In other words, either $\mathcal{A}$ is fully polynomial (on all solvable instances) or no fully polynomial protocol that solves Π on all solvable instances exists. In terms of reducibility, the concept of poly-time uniqueness of a protocol scheme $\mathcal{A}$ implies that $\mathcal{A}$ can be used as a self reduction for the given problem, as will be clear in the following.

We believe that this concept could be of more general interest, since it can be used to argue about optimality of protocol schemes and identify subproblems that are crucial for solving the original problem.

In the main theorem of this section we prove that the $\mathcal{Z}$-CPA scheme is poly-time unique for the RMT problem, and thus show that the $\mathcal{Z}$-CPA scheme is at least as efficient, up to a polynomial factor, as any other RMT protocol scheme. To show that, we build a self reduction for RMT based on $\mathcal{Z}$-CPA. We essentially show that if a unique fully polynomial RMT algorithm exists, it must be able to answer the membership check in polynomial time w.r.t. $|G|$ and therefore can be used as a subroutine to make $\mathcal{Z}$-CPA fully polynomial.

4.6 Self-reducibility of RMT

Consider the family of instances $\mathcal{G}$ where achieving RMT is possible. By Theorems 4.10,4.11:

$$\mathcal{G} = \{(G, \mathcal{Z}, D, R) \mid \nexists RMT\ \mathcal{Z}\text{-pp cut in } G\ \}$$

Also consider the family of *basic* instances $\mathcal{G}' \subseteq \mathcal{G}$ which contains the tuples $(G, \mathcal{Z}, D, R)$ where G is of the form shown in Figure 4.2 and RMT is solvable. More specifically, G contains the two distinguished nodes D, R and a "middle set" which we call $A(G)$. The only edges appearing are those which connect each player in the set $A(G)$ with the dealer D and the receiver-node R and in the resulting graph there does not exist a *RMT* $\mathcal{Z}$-pp cut. Finally for any $\mathcal{G}_1 \subseteq \mathcal{G}$ we define

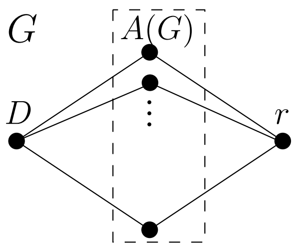


FIGURE 4.2: Family of instances $\mathcal{G}'$. No *RMT* $\mathcal{Z}$-pp cut exists.

the family of instances $\mathcal{I}(\mathcal{G}_1) \subseteq \mathcal{G}'$ which consists of all the instances $(G', \mathcal{Z}', D', R') \in \mathcal{G}'$ such that graph's G' middle-set $A(G')$ is a subset of a neighborhood of a node v in a graph contained in family $\mathcal{G}_1$, as a part of the instance tuple $(G, \mathcal{Z}, D, R)$, and $\mathcal{Z}' = \mathcal{Z}_v$ [4]. More precisely,

$$\mathcal{I}(\mathcal{G}_1) = \{(G', \mathcal{Z}', D', R') \in \mathcal{G}' \mid \exists (G, \mathcal{Z}, D, R) \in \mathcal{G}_1,\ \exists v \in V(G) \setminus \{D\}, A(G') \subseteq \mathcal{N}(v), \mathcal{Z}' = \mathcal{Z}_v\}$$

Intuitively the above family consists of the decomposition of every graph G in the $\mathcal{G}_1$ family into "small" graphs of the family $\mathcal{G}'$ whose middle sets appear in G as (partial) neighborhoods of nodes, the adversary structures are subsets of the original structure, and the RMT problem is solvable.

We next show that the RMT problem in any family of instances $\mathcal{G}_1 \subseteq \mathcal{G}$ (denoted $RMT|_{\mathcal{G}_1}$), also referred to as the RMT problem with *promise set* $\mathcal{G}_1$ (cf. [22]), reduces in polynomial time w.r.t. the size of the graph n to the $RMT|_{\mathcal{I}(\mathcal{G}_1)}$ problem. That is, if there exists an algorithm for solving RMT in $\mathcal{I}(\mathcal{G}_1)$ in fully polynomial time it can be used, as a subroutine of $\mathcal{Z}$-CPA, to solve RMT in $\mathcal{G}_1$ in fully polynomial time.

[4]In this point we slightly abuse the terminology, for ease of exposition, and use $\mathcal{Z}' = \mathcal{Z}_v$ instead of $\mathcal{Z}' = \{S \cap A(G') \mid S \in \mathcal{Z}_v\}$. The second statement is more accurate in the case where $A(G') \subsetneq \mathcal{N}(v)$ because we defined $\mathcal{Z}$ as a subset of the powerset of the nodes in the instance. This however does not affect our study because we can add the extra nodes $\mathcal{N}(v) \setminus A(G')$ in our instance $(G', \mathcal{Z}', D', R')$ as isolated nodes.

For convenience, we will use the following notation regarding the executions (runs) of the algorithms and the views of the players.

Runs and Views. Given a run (execution) e of a distributed protocol, the $view(v, e, k)$ of player v consists of the messages exchanged by v and its neighbors until round k. For simplification we will write $view(v, e)$ to refer to all the messages exchanged by v and its neighbors until the end of the run e. With $view(v, e, k)|_A$ (and $view(v, e)|_A$) we will denote the corresponding messages exchanged by v and the set $A \subseteq \mathcal{N}(v)$. The decision of a player v in run e will be denoted by $decision_e(v)$; for deterministic protocols, considered in this work, the $decision_e(v)$ function is in fact completely determined by player's v view on run e. We will simply write $decision(v)$ whenever the run is implied by the context.

Theorem 4.12. *If there exists a fully polynomial (in n) algorithm* Π *for solving* $RMT|_{\mathcal{I}(\mathcal{G}_1)}$ *then there exist a fully polynomial algorithm (in n) that solves* $RMT|_{\mathcal{G}_1}$.

Proof. We will use $\mathcal{Z}$-CPA to solve $RMT|_{\mathcal{G}_1}$. $\mathcal{Z}$-CPA has been proven unique, i.e., solves *RMT* in all instances where it is solvable, hence also for the family of instances $\mathcal{G}_1$ that we consider in this theorem.

We will show that $\mathcal{Z}$-CPA with protocol Π as a subroutine yields a fully polynomial algorithm for $RMT|_{\mathcal{G}_1}$. Namely, the decision rule of $\mathcal{Z}$-CPA which consists of a membership check for $\mathcal{Z}_v$ will be answered through simulations of protocol Π in time $poly(n)$. Since the subroutine protocol Π will only be used in the local computations phase of $\mathcal{Z}$-CPA, the round and bit complexity of $\mathcal{Z}$-CPA will be maintained in the resulting algorithm.

First, from the description of $\mathcal{Z}$-CPA observe that the *round complexity* is linear in n because at least one new player decides in every round and each player terminates after decision. Thus the receiver R will decide in at most n rounds. Second, one can see that the *bit complexity* of $\mathcal{Z}$-CPA is also of order $poly(n)$ due to the fact that each player sends one message to all of its neighbors. For deducing the latter we can reasonably assume that the messages sent by honest players are of size $poly(n)$ or, to drop any such assumption, consider the space X of the messages exchanged as a part of the input of size n. It thus remains to show that in $\mathcal{Z}$-CPA, the *local computations complexity*, can be of order $poly(n)$ if we use Π as subroutine.

For an arbitrary run e of $\mathcal{Z}$-CPA in some instance of $\mathcal{G}_1$, we can define $\mathcal{D}(i)$ to be the set of players that decide in round i of $\mathcal{Z}$-CPA. Moreover since run e is on an instance in the family $\mathcal{G}_1 \subseteq \mathcal{G}$, i.e., the RMT problem is solvable, it should be the case that $\exists i \in \{1, \dots, n\}, R \in \mathcal{D}(i)$. Observe that the function $\mathcal{D}$ is well defined as we can assume that we use an arbitrary algorithm, e.g. exhaustive search, to answer the membership check for $\mathcal{Z}_v$ (possibly in exponential time).

We next show that if we use Π as a subroutine for the local computations of the run e of $\mathcal{Z}$-CPA, we can achieve RMT in time $poly(n)$. Namely, we show by induction that for every round i, each player $v \in \mathcal{D}(i)$ will decide in $poly(n)$. Since $\exists i \in \{1, \dots, n\}, R \in \mathcal{D}(i)$ RMT will be achieved.

For round $i = 1$ all $v \in \mathcal{N}(D)$ receive the dealer's value x_D from the dealer and trivially decide on it in $poly(n)$ time.

Assume that, for every round $i \leq k$ every $v \in \mathcal{D}(i)$ decides in $poly(n)$-time. Considering any $v \in \mathcal{D}(k+1)$ and the $\mathcal{Z}$-CPA message propagation, the latter means that by the end of round k, v will have received sufficient information $view(v, e, k)$ to decide, from players in $\bigcup_{i=1,\dots,k} \mathcal{D}(i)$, in $poly(n)$-time, i.e., v will have received the same value x from all its neighbors in a set $N \subseteq \mathcal{N}(v)$ s.t. $N \notin \mathcal{Z}_v$. All valid messages exchanged in $\mathcal{Z}$-CPA consist of a single value $x \in X$ which corresponds to a possible dealer's value, and each player transmits only once to all its neighbors. Messages of different form, which we call *erroneous*, can be recognized by the recipient in $poly(n)$ time since $|X| = poly(n)$. Given $view(v, e, k)$, player v, in $poly(n)$-time, can create a partition of its neighborhood $\mathcal{N}(v) = \bigcup_{i=0}^{m+1} A_i$ such that

$$\begin{aligned} A_0 &= \{u \in \mathcal{N}(v) \mid \text{u sent nothing}\} \\ A_i &= \{u \in \mathcal{N}(v) \mid \text{u sent value } a_i \in X\},\ i = \{1, \dots m\} \\ A_{m+1} &= \{u \in \mathcal{N}(v) \mid \text{u sent erroneous messages}\} \end{aligned}$$

Since sets A_0, A_{m+1} do not affect our study we let $A = \bigcup_{i=\{1,\dots m\}} A_i$. Denote with $H, Z \subseteq V$ the sets of actual honest and corrupted players of run e. Also consider the sets of honest and corrupted neighbors of v, $H_v = H \cap \mathcal{N}(v)$ and $Z_v = Z \cap \mathcal{N}(v)$ respectively. Given $view(v, e, k)$, observe that

$$\exists!\ h \in \{1, \dots, m\} \text{ s.t. } H_v \setminus A_0 \subseteq A_h$$

else there exists an honest player which sends an incorrect value, a contradiction because $\mathcal{Z}$-CPA is safe. Subsequently $Z_v \supseteq A \setminus A_h$. Note that all $u \in A_h$ transmit the correct value a_h (regardless of whether they are honest or not) and all $u \in A \setminus A_h$ transmit false values. Since, by assumption, $view(v, e, k)$ is sufficient for v to decide through $\mathcal{Z}$-CPA, it holds that $A_h \notin \mathcal{Z}_v$ due to the decision rule of $\mathcal{Z}$-CPA. Moreover $\forall i \in \{1, \dots, m\} \setminus \{h\}$ it holds that $A_i \in \mathcal{Z}_v$ since $A \setminus A_h \subseteq Z_v$. Consequently

$$\exists!\ h \in \{1, \dots, m\} \text{ s.t. } A_h \notin \mathcal{Z}_v \text{ and } A \setminus A_h \in \mathcal{Z}_v \tag{4.1}$$

We next show how player v can decide which is the actual value of h in $poly(n)$ time using the protocol Π, and thus decide on the correct value a_h.

For $l = 1, \dots m$, we define the following runs of Π that can be simulated by v.

- Run e_0^l is on the instance $(G, \mathcal{Z}_v, D, v) \in \mathcal{G}'$ with $V(G) = A \cup \{D\} \cup \{v\}$, dealer's value $x_D = 0$, and corruption set $Z_v = A \setminus A_l$; in each round, all players in Z_v send the messages that send in the respective round of run e_1^l (where $A \setminus A_l$ is a set of honest players which runs Π). The latter means that v exchanges with Z_v messages that consist the $view(e_1^l, v)|_{A \setminus A_l}$.

- Run e_1^l, is on the same graph G, with dealer's value $x_D = 1$, and corruption set $Z_v = A_l$; Analogously with e_0 player v exchanges with Z_v the messages $view(e_0^l, v)|_{A_l}$.

Player v simulates run e_1^l in order to determine the behavior of the corrupted players in e_0^l. Observe that for every l exactly one of e_0^l, e_1^l is not in the family of instances $\mathcal{I}(\mathcal{G}_1)$ (due to the selection of the corruption set) and thus the local computations complexity might not be polynomial. Since protocol Π is fully polynomial in $\mathcal{I}(\mathcal{G}_1)$, it means that there is an explicit bound B on the local computations complexity of Π in the family $\mathcal{I}(\mathcal{G}_1)$. Assuming that arbitrary player v knows such a bound [5] we modify the above runs such that if the local computations complexity of a player w in a round i of e_0^l or e_1^l exceeds the bound B then v halts the simulation of the round i local computations of w and sends nothing on behalf of player w in round i. Such a modification of the run is necessary to obtain the desired result.

[5]Although this assumption is natural and often used, it is possible to avoid it if we consider family $\mathcal{I}(\mathcal{G}_1)$ consisting of directed graphs (with edges from dealer to $A(G)$ and from $A(G)$ to v). In this case the view of all players in $A_l, A \setminus A_l$ would be the same as that of some run in $\mathcal{I}(G_1)$ and thus their local computations complexity would be polynomial.

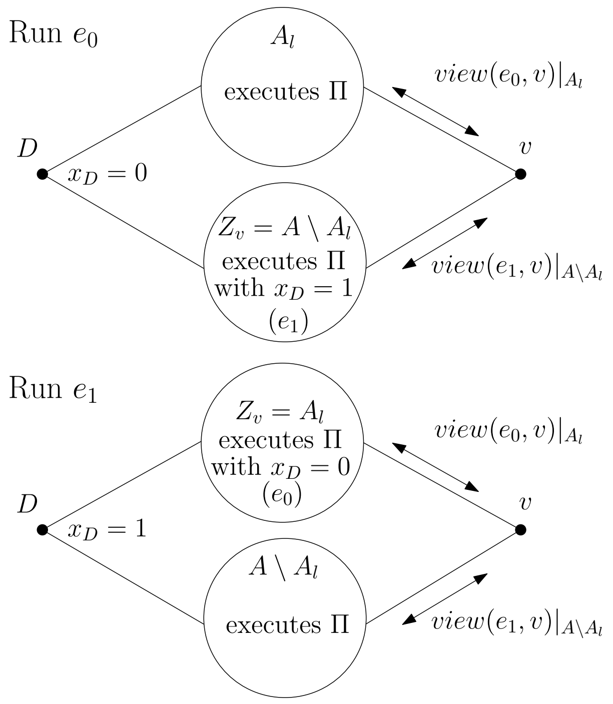


FIGURE 4.3: Runs e_0 and e_1.

Player v runs the following protocol in order to decide on the value of the dealer of run e.

Decision Protocol. Player v simulates, in parallel, $2m = poly(n)$ runs $(e_0^l, e_1^l)_{l \in \{1,\ldots,m\}}$ and halts all parallel simulations with decision a_l if run e_0^l terminates with $decision(v) = 0$.

We next show that v terminates run e_0^l with $decision(v) = 0$ if and only if $A_l \notin \mathcal{Z}_v$. More concretely

$$A_l \notin \mathcal{Z}_v \Leftrightarrow decision_{e_0^l}(v) = 0$$

"$\Rightarrow$": $A_l \notin \mathcal{Z}_v \Rightarrow Z_v = A \setminus A_l \in \mathcal{Z}_v$. Since by assumption Π solves $RMT|_{\mathcal{I}(\mathcal{G}_1)}$, for any adversarial behavior, that of Z_v in e_0^l included, v will decide on the correct value $x_D = 0$, i.e., $decision_{e_0^l}(v) = 0$.

"$\Leftarrow$": Let $A_l \in \mathcal{Z}_v$ and $decision_{e_0^l}(v) = 0$. This by equation (4.1) means that $Z_v = A \setminus A_l \notin \mathcal{Z}_v$. Observe now that the run e_0^l is not a valid run for the instance $(G, \mathcal{Z}_v, D, v)$ because the adversarial behavior of $Z_v \notin \mathcal{Z}_v$ is not valid for the adversary structure $\mathcal{Z}_v$. But the view of v is the same as the valid run e_1^l in which $x_D = 1$ and $Z_v = A_l \in \mathcal{Z}_v$. Since Π solves $RMT|_{\mathcal{I}(\mathcal{G}_1)}$, for any adversarial behavior, that of Z_v in e_1^l included, v will decide on the correct value $x_D = 1$ in the run e_1^l i.e., $decision_{e_1^l}(v) = 1$. But since the decision is a function of the view and player v receives exactly the same messages in runs e_0^l, e_1^l, it holds that $decision_{e_0^l}(v) = 1$, a contradiction.

The latter shows that the decision of player v in run e, which is acquired through the *Decision Protocol*, is correct and uniquely defined. Moreover all parallel simulations halt when the simulated run $e_0^l, l = h$ of Π terminates. Thus we have to show that run e_0^h can be be simulated in polynomial time.

The problem is that run e_1^h, which is simulated to determine the behavior of the corrupted players in e_0^l, is not a run of *RMT* in the family $\mathcal{I}(\mathcal{G}_1)$ due to the selection of the corruption set. Therefore we lose guarantee of full polynomiality in that run. Non-polynomiality of the round complexity is not an obstacle since the simulations are done in parallel. Local computations' polynomial complexity of e_1^h is ensured by the fact that we halt any local computations that exceed the explicit bound B previously mentioned. Finally it is easy to see that the bit complexity of the simulated runs is polynomial if the round and local computations complexity is polynomial. Thus the simulated run e_0^l remains fully polynomial.

Therefore it follows that v will decide in run e in polynomial time because the simulation of a fully polynomial protocol can be done in polynomial time. □

Based on the definition of poly-time uniqueness and Theorem 4.12, we obtain the following corollary on $\mathcal{Z}$-CPA.

Corollary 4.13. *Protocol scheme $\mathcal{Z}$-CPA is poly-time unique for RMT.*

Observe that in terms of reductions between distributed problems as presented in Section 1.3, Theorem 4.12 states that problem $RMT|_{\mathcal{G}_1}$ polynomially reduces to problem $RMT|_{\mathcal{I}(\mathcal{G}_1)}$, i.e., $RMT|_{\mathcal{G}_1} \leq_p RMT|_{\mathcal{I}(\mathcal{G}_1)}$.

4.7 Conclusions

Regarding the partial knowledge model, RMT-PKA makes players exchange information about the topology. Although topology discovery was not our motive, techniques used here (e.g. the $\oplus$ operation) may be applicable to that problem under a Byzantine adversary ([42],[16]). A comparison with the techniques used in this field might give further insight on how to efficiently extract information from maliciously crafted topological data.

The unique protocol proposed for the partial knowledge model only answers the feasibility question. A natural question is whether and when we can devise a unique and also efficient algorithm for this setting. The techniques used so far in the bibliography to reduce the communication complexity [36] do not seem to be directly applicable to this model. So exploring this direction might give new insights on message delivery in partially known graphs.

It would also be interesting to argue about uniqueness with respect to efficiency for RMT in the partial knowledge model by extending our analysis of the *ad hoc* case.

Finally, it is possible to define a stronger type of poly-time uniqueness: we call a protocol scheme $\mathcal{A}$ *strongly poly-time unique* for problem Π if the existence of any fully-polynomial protocol for a class of instances $\mathcal{I}$ implies that $\mathcal{A}$ is also fully polynomial for all instances in $\mathcal{I}$. We conjecture that $\mathcal{Z}$-CPA is in fact strongly poly-time unique for RMT in the *ad hoc* model.

Chapter 5

k-shot Broadcast in wireless networks

Even when no malicious or faulty behavior of players is observed in a distributed system, many different obstacles may appear during information propagation procedures according to real-world concerns and the nature of the communication network. We consider the case of wireless networks in which players possess radio transceiver devices in order to communicate through local broadcasts. The nature of wireless networks adds one more consideration regarding the correctness of message propagation; namely, it is generally required that the signal interference should be low for a message to be communicated in the network. This fact is usually abstracted by assuming that if two neighbors of a player v transmit simultaneously then a *collision* occurs and v receives no message.

We study the feasibility of Broadcast with few transmissions ('shots') in wireless *ad hoc* networks in the case where all players execute the protocol honestly. In particular, we examine the k-shot wireless network model, in which a bound k is given and a player may transmit at most k times during the execution of the protocol. The motivation for this model comes from energy efficiency considerations which are important in the widely used wireless sensor networks in which each player actually associates with a limited energy (battery supported) device.

We mainly consider the general case of *adaptive* Broadcast algorithms, which are algorithms where the actions of a player are defined considering its entire transmission history. On the contrary in an *oblivious* Broadcast algorithm the actions of a player only depend on the id of the player. We prove a lower bound of $\Omega\left(n^{\frac{1+k}{k}}\right)$ on the Broadcast time (rounds) of any protocol by introducing the *transmission tree* construction which generalizes previous approaches in the k-shot case. To obtain the lower bound we consider the round complexity of any adaptive Broadcast algorithm, which prove to be the more powerful algorithms for this model. Finally, we devise the Coordinated Transmission Algorithm (CTA), an oblivious Broadcast algorithm for the specific family of graphs on which the lower bound is proved. The round complexity of the algorithm is $\Omega\left(n^{\frac{1+k}{k}}\right)$ in the specific family and thus, only differs from the lower bound by a factor of k.

5.1 Introduction

Energy efficiency has become a central issue in wireless networks, due to the constantly increasing use of autonomous devices with limited power resources. A lot of recent research focuses on how to accomplish communication tasks in an energy-efficient manner without compromising the system performance too much. Much of the work so far has been devoted to the problem of adjusting the transmission ranges of nodes so that the energy cost is minimized.

However, if nodes transmit at a fixed power level it makes sense to consider the number of transmissions as an energy consumption measure. Such a study was initiated by Gasieniec *et al.* in [21], where Broadcast protocols with few transmissions ('shots') per node were considered for wireless networks with known topology. Here, we study the problem in *ad hoc* wireless networks, that is, networks in which nodes have no knowledge of the topology of the network.

We assume that a bound k is given and a node may transmit at most k times during the Broadcast protocol (*k-shot Broadcast*); note that the bound k may well represent the number of transmissions that the power supply of a node can handle. We also assume that the communication is *synchronous*, that is, nodes may transmit or receive simultaneously, i.e., in the same step. At each step a node may decide to act either as a *transmitter* or a *receiver*. Whenever a node transmits all its neighbors receive the message. If, however, two neighbors of a node v transmit simultaneously then a *collision* occurs and v receives no message.

We consider two types of protocols: *adaptive* and *oblivious* protocols; the former refers to protocols where a node may decide whether to transmit or not by taking into account any information it has received during the previous steps, while the latter term refers to protocols where a node makes transmission decisions with no consideration of the transmission history. Even though adaptive protocols are more powerful, oblivious algorithms are much easier to implement and demand minimal processing time for each node.

By mainly considering the adaptive model, we study the way in which the limitation on the number of transmissions affects the round complexity of Broadcast protocols.

5.1.1 Related work

Broadcast in wireless networks of unknown topology with no limitation in the number of shots has been extensively studied in the literature. The problem was first introduced by Chlamtac and Kutten [7]. Bar-Yehuda, Goldreich and Itai [2] gave the first randomized protocol, which completes Broadcast in $O(D \log n + \log^2 n)$ expected time when applied to graphs with n nodes and diameter D. Several papers followed [12, 33] that led to a tight upper bound of $O(D \log(n/D) + \log^2 n)$.

As for the deterministic case, a lower bound of $\Omega(n \log n)$ for general networks was given by Brusci and Del Pinto in [5], improved (for small D) to $\Omega(n \log D)$ by Clementi *et al.* [11]. Chlebus *et al.* [8] gave the first Broadcast protocol of sub-quadratic time complexity $O(n^{11/6})$. This bound was later improved to $O(n^{5/3} \log^3 n)$ by De Marco and Pelc [41] and then by Chlebus *et al.* [9], who gave an algorithm with time complexity $O(n^{3/2})$ based on finite geometries. Chrobak, Gąsieniec and Rytter [10] further improved the bound to $O(n \log^2 n)$. Finally, De Marco [40] gave the best currently known upper bound of $O(n \log n \log \log n)$, thus leaving a sub-logarithmic gap between the upper and lower bound.

As mentioned above, Broadcast with a limited number of shots was first proposed in [21]. It has also been considered in [28], where randomized algorithms were proposed; in both cases, only Broadcast in known networks was studied. Another approach to limiting the number of shots was presented in [4], where the authors construct algorithms which use few shots for each node and achieve nearly optimal Broadcast time.

Our approach in this chapter stems from the work of Koutris and Pagourtzis [32]. In this work the authors take a first step towards studying the behavior of adaptive Broadcast protocols by showing an $\Omega(n^2)$ lower bound for any adaptive 1-shot Broadcast protocol and pose the generalization of the lower bound for any value k as an open problem. Regarding the oblivious case, in the same work a lower bound of $\Omega(n^2/k)$ on the Broadcast time of any oblivious k-shot algorithm is given. On the positive they present an oblivious Broadcast protocol that achieves a matching upper bound, namely $O(n^2/k)$, for every $k \leq \sqrt{n}$ and an upper bound of $O(n^{3/2})$ for every $k > \sqrt{n}$.

5.2 Model and Preliminaries

We consider the case of wireless networks, that is, networks in which the players possess radio transmitting/receiving devices are spread out on some physical surface (terrain), and two nodes can communicate if there are within transmission range of each other and signal interference is low. A common abstraction is to consider the network as a graph, and assume (*collision assumption*) that communication is possible if a node receives a message from only one neighbor in a certain time-slot.

Unknown wireless networks. We model a wireless network as a directed graph. Namely, if a player v is in the transmission range of another player w, we model the situation with the existence of the directed edge (w, v). Furthermore, we assume that the nodes have unique labels from the set $V = \{1, 2, \ldots, n\}$, where n is the number of nodes in the network. Regarding the initial knowledge of the players, we assume that initially, a node is aware only of its own label and whether it is the dealer node or not. This means that it has no knowledge, full or partial, about the topology of the underlying graph. Observe that we require somewhat less initial knowledge from the *ad hoc* model we studied so far, since a node does not know the labels of its neighbors. This assumption is natural in the wireless network model, since in reality a node locally broadcasts a message and is not aware of any communication channels which actually correspond to specific neighbors.

Moreover, we assume that the nodes are not capable of *detecting collisions*, that is, if an attempt to transmit to a node v was unsuccessful, then v is not able to sense it.

Since no adversary is assumed to exist, we say that a Broadcast algorithm completes Broadcast when all nodes of the network have received the dealer message. As before, the round complexity, or *running time* of a Broadcast algorithm is defined as the worst-case number of steps needed to complete Broadcast over all possible network configurations with n nodes.

Oblivious Broadcast protocols We now define the notion of *oblivious k-shot protocols*. As mentioned earlier, a protocol is oblivious if nodes do not take into account any information that may be gained during the execution of the protocol. Formally, an oblivious protocol can be succinctly described as a sequence of *transmission sets*, which are subsets of the node set V. Since no additional information, other than the dealer's message, is utilized by the players, it is natural to assume that once a node receives the message at step t, it wakes up and transmits at the first k steps after t in which it appears in the transmission set. This model captures an important class of Broadcast algorithms, since most known algorithms for deterministic Broadcast in networks with unknown topology fall into this class.

5.2.1 Adaptive Broadcast protocols

For a formal definition of an adaptive Broadcast protocol we will use a slight generalization of the model proposed by Kowalski and Pelc [34]. In our model for adaptive protocols, we allow a node to transmit a message to its neighborhood even before it receives the dealer message. An algorithm may use this kind of transmission for topology knowledge exchange which may influence actions in later rounds. We denote by $H_t(v)$ the (interaction) view of node v until the end of step t, i.e., a complete description of all the messages received (along with the corresponding sender's id) and send by v during each round $1, \ldots, t$. We will use the notion *incoming view* for the description of the incoming messages.

A Broadcast protocol can now be defined by a function $\pi(v, t, H_{t-1}(v))$, which takes values in the set {`receive`, `transmit`}. The function decides whether node v with view $H_{t-1}(v)$ acts as a receiver (`receive`) or as a transmitter (`transmit`) at step t. If v acts as a transmitter in step t, it sends its entire view until step $t-1$ along with its id, i.e., the message $(v, H_{t-1}(v))$. Note that the maximum information exchange occurs when transmitters send their entire view, since the receiver can always deduce any information from the received view. For completeness we assume that in the initialization phase of the protocol (*step 0*) each player $v \in V \setminus \{s\}$ has the view $H_0(v) = (\emptyset, \perp)$ which actually represents the lack of an initial input value. The dealer node D has the initial view $H_0(s) = (\emptyset, m)$, where m is the initial value which will be propagated by D.

5.2.2 Outline

In this chapter we primarily address the open question of [32], regarding the generalization of the lower bound and manage to obtain a bound for any value of k. In fact, we prove this lower bound on the round complexity of any adaptive Broadcast algorithm; since the adaptive algorithms allow the use of any available information they prove to be the strongest algorithms for the model and thus the bound holds for any Broadcast algorithm.

To prove the bound, we introduce the *transmission tree* construction which provides a way to represent the actions that the players take through any Broadcast algorithm given a certain view. The construction can be conveniently used to determine the id assignment that yields the maximum delay of the message propagation in intermediate levels, from the initiation of the protocol until its termination. We believe that the transmission tree notion could be of more general interest, since it can be used to argue about the round complexity of algorithms in this model. We study the graph family $\mathcal{G}$ of a certain topology to obtain the lower bound.

Finally, considering the optimality of the lower bound for the specific family $\mathcal{G}$, in Section 5.4, we devise an oblivious protocol which achieves Broadcast in the specific family with round complexity $\Omega\left(n^{\frac{1+k}{k}}\right)$, which only differs from the lower bound by a factor of k.

This chapter includes results presented in [29].

5.3 Broadcast protocols and transmission Trees

Generalizing the lower bound approach of [5] for any number of shots we introduce a construction, which we call the *transmission tree*, that allows us to obtain a lower bound for the k-shot case. We consider the most general case of adaptive protocols, and for any such protocol π we construct a network in which the delay of completing Broadcast with π is significantly increased. We use the transmission tree tool to maximize the delay in the intermediate stages of achieving Broadcast.

5.3.1 Family of networks

The class of radio networks $\mathcal{G}$ we will use for our lower bound argument are graphs of a certain topology, namely the n nodes of each such graph can be partitioned in l layers; The first layer contains only the dealer node D and the next $l-2$ contain 2 nodes each and the last layer contains the rest of the nodes (1 or 2) to complete the partition. Moreover each node v in layer i is connected, with a directed edge (v, w), to each node w of layer $i+1$ and no other connections exist.

More concretely, For an n-node graph $G = (V, E) \in \mathcal{G}$, V can be partitioned in $l = \lfloor n/2 \rfloor + 1$ layers $L_1, \ldots, L_l$ s.t. $L_1 = \{s\}$, $|L_2| = \ldots = |L_{l-1}| = 2$ and $L_l = V \setminus \cup_{i=1}^{l-1} L_i$. Moreover, $E = \{(w, v) \in L_i \times L_j \mid i, j \in \{1, \ldots l\}, j - i = 1\}$. Having specified the topology of the family $\mathcal{G}$ the different members of the family differ in the number of nodes and the assignment of the id's. The general topology structure of family $\mathcal{G}$ is depicted in Figure 5.1.

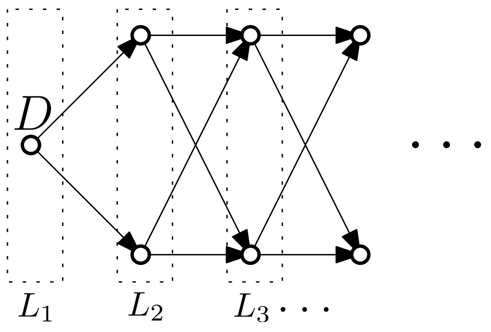


FIGURE 5.1: Family of graphs $\mathcal{G}$.

5.3.2 Designing a “bad” graph

Let us consider any deterministic k-shot Broadcast protocol π which completes Broadcast in any graph with n nodes. We will construct a graph $G_\pi \in \mathcal{G}$, by assigning ids to nodes, such that Broadcast is significantly slowed down. We will gradually construct graph G_π by using the graph families $\mathcal{G}_i$ as described in the following.

Partial id assignment. For an arbitrary assignment ID_i of ids in the first i ($i \in \{1, \ldots, l-1\}$) layers of family $\mathcal{G}$ graphs, we define family $\mathcal{G}_i \subseteq \mathcal{G}$ of graphs with n nodes that have the assignment ID_i in their first i layers and thus differ only in the next $l - i$ layers. Let S be the set of the assigned ids and $A = V \setminus S$.

Incoming view of a layer. Consider the execution of protocol π in any $G \in \mathcal{G}_i$. Observe that in all graphs $G \in \mathcal{G}_i$ nodes in layer L_{i+1} receive from L_i the same incoming view H_j, for every round j of the execution. Incoming view sequence $(H_j) = (H_1, H_2, \ldots)$ can be determined through protocol π given that the topology and the id's of the first i layers are known. Observe that according to the view definition the term H_j contains all the information contained in terms $H_1, \ldots, H_{j-1}$. We make a distinction between the incoming and the full view because in the following we will consider the actions that different nodes take given that they receive the same view. Since L_i contains all the incoming neighbors of nodes in L_{i+1}, and there is no directed path from nodes in L_{i+1} to nodes in L_i, (H_j) is guaranteed to capture the whole (incoming) view of nodes in L_{i+1} and thus their actions (`transmit, receive`) can be determined for every round of the execution.

Transmitting and receiving under a certain incoming view. To determine if node $v \in A$ transmits in round t under incoming view H_{t-1} (being in level $i+1$) one should simulate the execution of protocol π, where v receives the view H_{t-1} and construct v's view $H_{t-1}(v)$ which may additionally contain messages sent by v [1]. Working this way we can define the round t transmitting/sending nodes which belong to a set A with

$$S_t^A = \{v \in A \mid \pi(v, t, H_{t-1}(v)) = \text{“}\texttt{transmit}\text{”}\}$$

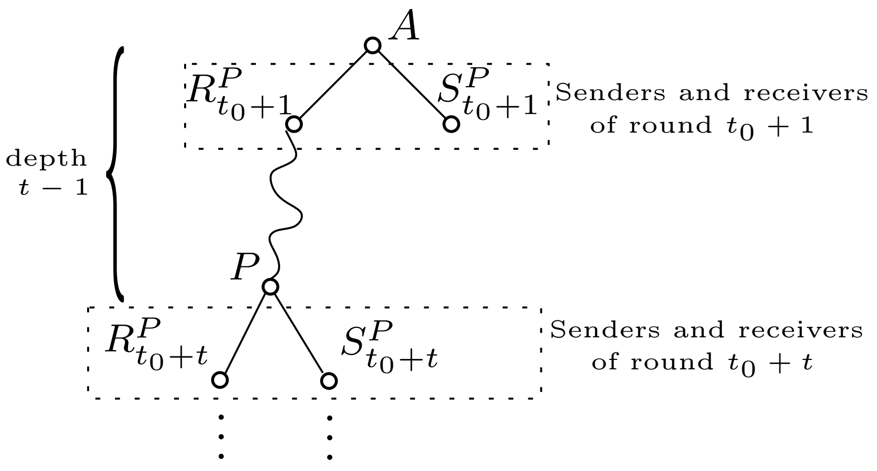
and the round t receiving nodes with

$$R_t^A = \{v \in V \mid \pi(v, t, H_{t-1}(v)) = \text{“}\texttt{receive}\text{”}\}$$

.

Given a family of graphs $\mathcal{G}_i$ and a protocol π we will construct the family of graphs $\mathcal{G}_{i+1} \subseteq \mathcal{G}_i$ in which the delay of transmitting the message from L_{i+1} to L_{i+2} maximizes. Our approach is summarized to the “worst" choice of the two ids of layer L_{i+1}, such that either by collision or non-transmission the message fails to transmit to L_{i+2} for the maximum number of steps.

[1]Obviously the messages sent by v don't have any effect on the incoming view due to the topology (no directed path exists from nodes in L_{i+1} to nodes in L_i).



A
$R^P_{t_0+1}$
$S^P_{t_0+1}$
Senders and receivers of round $t_0 + 1$
depth $t - 1$
P
$R^P_{t_0+t}$
$S^P_{t_0+t}$
Senders and receivers of round $t_0 + t$

FIGURE 5.2: Transmission Tree $T(\pi, ID_i, t_0)$

5.3.3 Transmission trees

We next introduce the notion of a *transmission tree* which can be used to argue about the id assignment that yields the maximum delay of the message propagation in intermediate layers. Recall that for the family $\mathcal{G}_i$ (defined by the assignment ID_i) we denote with A the set of unassigned ids. Also given a protocol π and a family $\mathcal{G}_i$ the incoming view (H_j) of layer L_{i+1} is well defined as explained in Section 5.3.2. The transmission tree $T(\pi, \mathcal{G}_i, t_0)$ defined above, represents the transmission sets after round t_0 given that their incoming view is (H_j), i.e., they rounds greater than t_0 that any id in A transmits as a possible candidate for layer L_{i+1}. The construction is depicted in Figure 5.2

Definition 5.1. *A transmission tree $T(\pi, \mathcal{G}_i, t_0)$ corresponding to Broadcast protocol π, family $\mathcal{G}_i$ and a round t_0 is a binary tree. The id of the root is the node set A and every child's id is a subset of its father's id. The ids of the children form a partition of the father's id and all the leaves are singletons. For a node P at depth $t - 1$, its left child's id is $R^P_{t+t_0}$ whereas the right child of P is the set $S^P_{t+t_0}$, i.e., the nodes in P which are receiving (don't transmit), respectively transmitting, in round $t + t_0$ given that their incoming view is (H_j).*

Observe that, given a family $\mathcal{G}_i$ (or equivalently an id assignment ID_i) and a start time t_i, any Broadcast protocol π defines a transmission tree $T(\pi, \mathcal{G}_i, t_i)$ which describes the actions that players in A take, under the reception of the corresponding view (H_j), executing protocol π after round t_i.

The only non-trivial point in this correspondence is why the leaves, of the corresponding tree of any protocol π, have to be singletons. The reason for this is that if there was a leaf P with $|P| \geq 2$ then this protocol would never achieve Broadcast in the family $\mathcal{G}_{i+1}$ with $L_{i+1} \subseteq P$.

Definition 5.2. *A k-shot transmission tree is a transmission tree in which each branch contains at most k right children.*

This in fact ensures that each node will transmit in at most k steps as desired for a k-shot protocol.

We next define the notion of *maximum pair depth* (mpd) of a tree $T(\pi, \mathcal{G}_i, t)$ which relates to the maximum delay time between layers. Observe that the existence of an internal node with id which consists of exactly 2 nodes, is trivially guaranteed by the structure of the transmission tree, and specifically from the fact that all leaves' ids are singletons.

Definition 5.3 (maximum pair depth)**.** *Given a transmission tree T, the maximum pair depth (mpd) of the tree is defined as the maximum depth of a node P where $|P| = 2$. We will denote it with $mpd(T)$.*

Given a family $\mathcal{G}_i$ and a protocol π, with t_i, we denote the first round where the dealer's message is propagated from L_i to L_{i+1} due to π. Observe that t_i is well defined by $\mathcal{G}_i$ and π.

Theorem 5.1. *Given a Broadcast protocol π and a family $\mathcal{G}_i$, there exists $\mathcal{G}_{i+1} \subseteq \mathcal{G}_i$ s.t. the relay of the message from layer L_{i+1} to layer L_{i+2} in all graphs of $\mathcal{G}_{i+1}$ will be delayed for more than $mpd(T(\pi, \mathcal{G}_i, t_i)) + 1$ rounds, where t_i is the round in which the dealer's message reaches layer L_{i+1}.*

Proof. To create the family $\mathcal{G}_{i+1}$ we only have to choose the two ids v_1, v_2 that will be assigned to layer L_{i+1}. The relay of the message from L_{i+1} to L_{i+2} will happen in the first round that only one of v_1, v_2 will transmit, in a different case either a collision or a non-transmission phase will occur. As argued before, there is a node P with $|P| = 2$ and $depth(P) = mpd(T(\pi, \mathcal{G}_i, t_i))$. Choosing nodes $L_{i+1} = \{v_1, v_2\} = P$, the first round where one of them transmits alone will be round $t \geq depth(P) + 1 = mpd(T(\pi, \mathcal{G}_i, t_i)) + 1$, therefore transmission to L_{i+2} will be delayed until round $mpd(T(\pi, \mathcal{G}_i, t_i)) + 1$.

□

Given a family $\mathcal{G}_i$, we would like to determine the minimum delay for the propagation of messages from layer L_{i+1} to layer L_{i+2} over all Broadcast protocols.

Maximum pair depth and tree height. Without loss of generality, for our study, we may consider only Broadcast protocols for which the corresponding transmission trees T have the property that $mpd(T) + 1 = height(T)$. We can make this simplifying assumption because for every such tree (or protocol) which does not have this property there exists another tree (protocol) due to which the propagation from L_{i+1} to L_{i+2} will be delayed for the same amount of rounds and the property holds. The latter tree results from additionally requiring that every singleton node of the tree is also a leaf, and thus, trivially, it is of smaller height than the former. Thus for determining the minimum delay, it suffices to obtain a lower bound for the minimum height of the k-shot transmission tree.

Theorem 5.2. *The minimum height of a k-shot transmission tree for family $\mathcal{G}_i$ with unassigned ids A with $|A| = a$ over all k-shot Broadcast protocols $\mathcal{B}$ is*

$$\min_{\pi \in \mathcal{B}} height\left(T(\pi, \mathcal{G}_i, t_i)\right) = \Omega(a^{\frac{1}{k}})$$

Proof. Wlog we can assume only protocols corresponding to transmission trees where every internal node has a right child. The reason for that is that if a protocol π corresponds to a tree in which a node v only has a left (non-transmitting) child w, then deleting this edge along with the left child w and connecting the children of w to v will result to a transmission tree of non-greater height and thus a protocol which achieves a non-slower relay.

For the case of $k = 1$ one can observe that each right child P will contain only one node ($|P| = 1$) and will be a leaf. This is obvious from the definition of the k-shot transmission tree; since each branch contains at most $k = 1$ right child and all the leafs are singletons, each right child must be a singleton-leaf. Therefore the minimum height tree will result if the root and every left child has a right child leaf. Subsequently for the case of $k = 1$,

$$\min_{\pi \in \mathcal{B}} T(\pi, \mathcal{G}_i, t) = a - 1 = \Omega(a)$$

Therefore the theorem holds for $k = 1$. The corresponding minimum height tree for this case is depicted in Figure 5.3

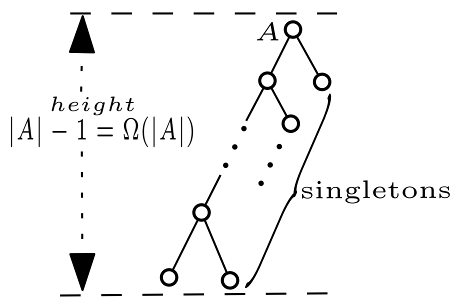


FIGURE 5.3: Minimum height 1-shot transmission tree

Assume that the claim holds for $k = i - 1$, then we will prove that it holds for $k = i$. First consider an i-shot transmission tree T and its leftmost branch *LB* including the root. Observe that each right child P of a node in *LB* is actually a root of an $(i-1)$-shot transmission tree since all nodes in P have only $i - 1$ shots left. By the induction hypothesis we know that the the minimum height of every such tree is $\Omega\left(|P|^{\frac{1}{i-1}}\right)$.

For any i-shot transmission tree T there are two cases (two types of transmission trees):

1. Every right child P of nodes in *LB* has cardinality $|P| = O\left(a^{\frac{i-1}{i}}\right)$. In this case, the length of *LB* in this tree will be of order

$$|LB| = \frac{a}{O\left(a^{\frac{i-1}{i}}\right)} = \Omega\left(\frac{a}{a^{\frac{i-1}{i}}}\right) = \Omega(a^{1-\frac{i-1}{i}}) = \Omega(a^{1/i})$$

   since for every pair of *LB* nodes P_r, P_{r+1} of depth r and $r + 1$ respectively, it holds that $|P_{r+1}| = |P_r| - O(a^{\frac{i-1}{i}})$.

   Moreover it holds that $height(T) \geq |LB| = \Omega(a^{\frac{1}{i}})$ and therefore

$$height(T) = \Omega(a^{\frac{1}{i}})$$

   .

2. There exists a right child P of nodes in *LB* with cardinality

$$|P| \neq O(a^{\frac{i-1}{i}}) \Leftrightarrow |P| = \omega(a^{\frac{i-1}{i}})$$

   By the induction hypothesis every such tree T_P with root P will have a minimum height of order,

$$height(T_P) = \Omega\left(\left(\omega(a^{(i-1)/i})\right)^{\frac{1}{i-1}}\right) = \Omega\left(a^{\frac{i-1}{i}\cdot\frac{1}{i-1}}\right) = \Omega\left(a^{\frac{1}{i}}\right)$$

Moreover it holds that $height(T) \geq height(T_P) = \Omega(a^{\frac{1}{i}})$ and therefore, in this case also it holds that,

$$height(T) = \Omega(a^{\frac{1}{i}})$$

.

Therefore the minimum height of any k-shot transmission tree is

$$\min_{\pi\in\mathcal{B}} height\left(T(\pi, \mathcal{G}_i, t)\right) = \Omega(a^{\frac{1}{k}})$$

□

An example of a structure of a minimum height k shot transmission tree is depicted in Figure 5.5. All the right children subtrees with root of cardinality roughly $|A|^{\frac{k-1}{k}}$.

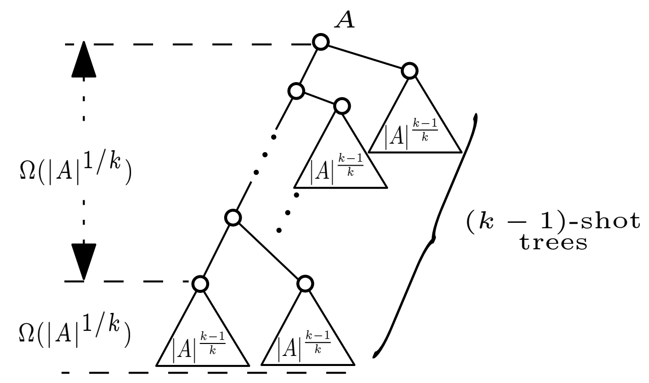


FIGURE 5.4: Example of minimum height k shot transmission tree

Theorem 5.3. *For any k-shot adaptive Broadcast protocol π, there is a n-node graph $G \in \mathcal{G}$ where π needs $\Omega(n^{\frac{1+k}{k}})$ rounds to achieve Broadcast.*

Proof. Repeatedly applying theorems 5.1,5.2 for $i = 1, \ldots \lfloor n/2 \rfloor$ we construct a graph in which π will achieve Broadcast in time asymptotically greater or equal than

$$S_1 = (n-1)^{1/k} + (n-3)^{1/k} + \cdots + 2^{1/k}$$

in the case where n is odd and

$$S_2 = (n-1)^{1/k} + (n-3)^{1/k} + \cdots + 3^{1/k}$$

when n is even.

Observe that in the case where n is odd the sum S_1 is comprised by $\lfloor n/2 \rfloor$ terms and the half of these terms are lower than the $(n/2)^{\frac{1}{k}}$. Hence it holds that,

$$S_1 \geq \left\lfloor \frac{n}{4} \right\rfloor \cdot \left(\frac{n}{2}\right)^{\frac{1}{k}} > \left(\frac{n}{4} - 1\right) \cdot \left(\frac{n}{2}\right)^{\frac{1}{k}} \geq \frac{1}{8}(n-4)n^{\frac{1}{k}} \Rightarrow S_1 = \Omega\left(n^{\frac{1+k}{k}}\right)$$

Where the last inequality holds because $2^{\frac{1}{k}} \geq 2, k \in \mathbb{N}$. Using similar arguments we can show that $S_2 = \Omega\left(n^{\frac{1+k}{k}}\right)$.

$\square$

The lower bound presented coincides with the bound of [32] for the 1-shot case.

5.4 An oblivious algorithm for family $\mathcal{G}$

Considering the optimality of the lower bound for the specific family $\mathcal{G}$, it is meaningful to study algorithms for the specific case. We next present a simple oblivious Broadcast algorithm for family $\mathcal{G}$, the round complexity of which differs from the lower bound by factor k. Observe that we only require that the specific algorithm achieves Broadcast in family $\mathcal{G}$.

Recall that an oblivious algorithm $\mathcal{A}$ can be described as a sequence of transmission sets, i.e., sequence s of sets of players ids. If a player has already received the message in a round i, its id is contained in the set s_j with $j > i$ and it has transmitted less than k times until round j, then it acts as a transmitter in round j.

Coordinated Transmission Algorithm (CTA). We assume the k-dimensional cube $\left[0, \lceil n^{1/k} \rceil - 1\right]^k$, and represent each player as an integer point of this cube. Namely we can assume that each player v is assigned a unique coordinate vector $(x_1^v, \ldots, x_k^v)$ with $0 \leq x_1^v, \ldots, x_k^v \leq \lceil n^{1/k} \rceil - 1$, and $x_1^v, \ldots, x_k^v \in \mathbb{Z}$. The oblivious algorithm CTA can now be described by the sequence $(s_i)_{i \in \mathbb{N}}$ of transmission sets as described in the following:

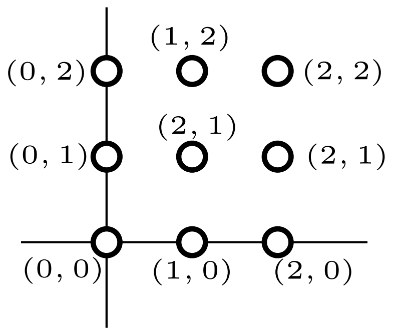


FIGURE 5.5: CTA execution example

$$\forall i \in \mathbb{N}, \text{ with } i \mod k\lceil n^{1/k}\rceil = j,$$
$$s_i = \{v \in V \mid x_{j \text{ div}\lceil n^{1/k}\rceil+1} = j \mod \lceil n^{1/k}\rceil\}$$

Where the $a \mod b$ is the remainder of the division of a with b and a div b denotes the integer division of a with b.

In words, in each round, the only players which act as transmitters are those which have the same value for a specific coordinate. In more details, in every round $i = 0, \ldots, \lceil n^{1/k}\rceil - 1$,the transmitting players are exactly the players v with $x_1^v = i$, in every round $i = \lceil n^{1/k}\rceil, \ldots, 2\lceil n^{1/k}\rceil - 1$,the transmitting players are exactly the players v with $x_2^v = i \mod \lceil n^{1/k}\rceil$ etc. The schedule repeats after $k\lceil n^{1/k}\rceil$ rounds where all possible values for all coordinates have been considered. A trivial example for the case of $k = 2$ and $n = 9$ is given in Figure 5.5. In this example we have the following transmission schedule every 6 rounds:

- *Round 0*: Players $(0, 0), (0, 1), (0, 2)$
- *Round 1*: Players $(1, 0), (1, 1), (1, 2)$
- *Round 2*: Players $(2, 0), (2, 1), (2, 2)$
- *Round 3*: Players $(0, 0), (1, 0), (2, 0)$
- *Round 4*: Players $(0, 1), (1, 1), (2, 1)$
- *Round 5*: Players $(0, 2), (1, 2), (2, 2)$

Theorem 5.4. *CTA achieves Broadcast in family $\mathcal{G}$ in $O\left(k \cdot n^{\frac{1+k}{k}}\right)$ rounds.*

Proof. Since each player is assigned a unique coordinate vector, it means that for any pair of players (v, w) there exists a coordinate in which they differ, i.e., $\exists i \in \{1, \dots, k\}$ s.t. $x_i^v \neq x_i^w$. Thus for every consecutive $k\lceil n^{1/k} \rceil$ it holds that for a pair of players (v, w), exactly one of them will transmit in at least one round. Moreover, every player will act as a transmitter in a time period of $k\lceil n^{1/k} \rceil$; specifically, each player which has already received the dealer's message will transmit exactly k times in $k\lceil n^{1/k} \rceil$ time period after the receipt. The previous arguments hold holds due to the fact that in a $k\lceil n^{1/k} \rceil$ time period, we will have considered all values for all coordinates.

For any graph $G \in \mathcal{G}$ assume that nodes in a layer L_i receive the dealer's message in round T_i (recall that both nodes of level L_i will receive the message simultaneously). Due to the arguments of the previous paragraph, it should be clear that by round $T_i + k\lceil n^{1/k} \rceil$ there exists at least one round in which exactly one node of layer L_i will transmit the message to L_{i+1} and thus the message will be propagated to the next layer. Assuming that the dealer player transmits its message to level L_2 in the initial round of the protocol it holds that the CTA will achieve Broadcast in $O\left(k \cdot n^{\frac{1+k}{k}}\right)$ rounds.

□

5.5 Conclusions

In this chapter, we consider the most general case of adaptive k-shot Broadcast protocols and manage to obtain a lower bound on the round complexity of achieving Broadcast; this partially answers the relative open question of [32], since we generalize the lower bound from the case of $k = 1$ to any value of k. We believe that the introduced transmission tree notion, used for the proof of the bound, could be of more general interest, since it relates the round complexity of algorithms with a graph parameter of this tree. We take one step further in the study of the difference between the adaptive and oblivious algorithms by providing an oblivious algorithm, the round complexity if which differs from the lower bound by a factor of k.

It would be interesting to further study the tightness of the lower bound by considering different classes of graphs or even a different way to obtain a lower bound in the specific family. We take a first step in studying the relation between the Broadcast time of adaptive and oblivious algorithms; however, the main question of determining whether adaptive protocols can provide more efficient ways for achieving Broadcast (or other distributed tasks) than oblivious algorithms still remains an intriguing open question. Essentially, the matter in question is whether using topology knowledge exchange can actually yield more efficient algorithms in the context of unknown networks. The latter point is also of practical importance since oblivious algorithms are generally simpler in concept and have very small memory requirements from the nodes.

Chapter 6

Appendix

6.1 The $\oplus$ operation

Observe that an equivalent definition for the The $\oplus$ operation is the following,

$$\mathcal{E}^A \oplus \mathcal{F}^B = \{Z_1 \cup Z_2 \mid (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{F}^B) \wedge (Z_1 \cap B \subseteq Z_2) \wedge (Z_2 \cap A \subseteq Z_1)\}$$

6.1.1 Proof of Theorem 4.3

To prove that $\oplus$ is also associative we will need the following lemma.

Lemma 6.1. *For any node sets A, B, C it holds that*

$$(Z_1 \cap B \subseteq Z_2) \wedge (Z_2 \cap A \subseteq Z_1) \wedge (Z_1 \cup Z_2 \cap C \subseteq Z_3) \wedge (Z_3 \cap A \cup B \subseteq Z_1 \cup Z_2)$$
$$\Leftrightarrow$$
$$(Z_2 \cap C \subseteq Z_3) \wedge (Z_3 \cap B \subseteq Z_2) \wedge (Z_2 \cup Z_3 \cap A \subseteq Z_1) \wedge (Z_1 \cap B \cup C \subseteq Z_2 \cup Z_3)$$

Proof. First we prove the $\Rightarrow$ direction. From $(Z_1 \cup Z_2 \cap C \subseteq Z_3)$ it follows that:

$$\begin{aligned}(Z_1 \cup Z_2) \cap C \subseteq Z_3 &\Rightarrow (Z_1 \cap C) \cup (Z_2 \cap C) \subseteq Z_3 \\ &\Rightarrow (Z_1 \cap C) \subseteq Z_3 \wedge (Z_2 \cap C) \subseteq Z_3\end{aligned}$$

From $(Z_3 \cap (A \cup B) \subseteq Z_1 \cup Z_2)$ it follows that:

$$\begin{aligned}
Z_3 \cap (A \cup B) \subseteq Z_1 \cup Z_2 &\Rightarrow (Z_3 \cap A) \cup (Z_3 \cap B) \subseteq Z_1 \cup Z_2 \\
&\Rightarrow (Z_3 \cap B) \subseteq Z_1 \cup Z_2 \\
&\Rightarrow (Z_3 \cap B) \cap B \subseteq (Z_1 \cup Z_2) \cap B \\
&\Rightarrow (Z_3 \cap B) \subseteq (Z_1 \cap B) \cup (Z_2 \cap B) \\
&\Rightarrow (Z_3 \cap B) \subseteq (Z_2 \cap B) \\
&\Rightarrow (Z_3 \cap B) \subseteq Z_2
\end{aligned}$$

$$\begin{aligned}
Z_3 \cap (A \cup B) \subseteq Z_1 \cup Z_2 &\Rightarrow (Z_3 \cap A) \cup (Z_3 \cap B) \subseteq Z_1 \cup Z_2 \\
&\Rightarrow (Z_3 \cap A) \subseteq Z_1 \cup Z_2 \\
&\Rightarrow (Z_3 \cap A) \subseteq Z_1 \cup Z_2 \\
&\Rightarrow (Z_3 \cap A) \cap A \subseteq (Z_1 \cup Z_2) \cap A \\
&\Rightarrow (Z_3 \cap A) \subseteq (Z_1 \cap A) \cup (Z_2 \cap A) \\
&\Rightarrow (Z_3 \cap A) \subseteq (Z_2 \cap A) \\
&\Rightarrow (Z_3 \cap A) \subseteq Z_2
\end{aligned}$$

Also :

$$\begin{aligned}
(Z_2 \cup Z_3) \cap A &\subseteq (Z_2 \cap A) \cup (Z_3 \cap A) \\
&\subseteq Z_1 \cup Z_1 \\
&\subseteq Z_1
\end{aligned}$$

And

$$\begin{aligned}
(Z_1 \cap (B \cup C)) &\subseteq (Z_1 \cap B) \cup (Z_1 \cap C) \\
&\subseteq Z_2 \cup Z_3
\end{aligned}$$

The proof for the $\Rightarrow$ direction is complete. The other direction follows from symmetry.

$\square$

Theorem 6.1. *Operator $\oplus$ is associative.*

Proof. For any adversary structures $\mathcal{E}, \mathcal{F}, \mathcal{H}$ and node sets A, B, C:

$$\begin{aligned}(\mathcal{E}^A \oplus \mathcal{F}^B) \oplus \mathcal{H}^C &= \{Z_1 \cup Z_2 | (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{F}^B) \wedge (Z_1 \cap B \subseteq Z_2) \wedge (Z_2 \cap A \subseteq Z_1)\} \oplus \mathcal{H}^C \\ &= \{Z_1 \cup Z_2 \cup Z_3 | (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{F}^B) \wedge (Z_3 \in \mathcal{H}^C) \wedge (Z_1 \cap B \subseteq Z_2) \\ &\wedge (Z_2 \cap A \subseteq Z_1) \wedge (Z_1 \cup Z_2 \cap C \subseteq Z_3) \wedge (Z_3 \cap A \cup B \subseteq Z_1 \cup Z_2)\}\end{aligned}$$

$$\begin{aligned}\mathcal{E}^A \oplus (\mathcal{F}^B \oplus \mathcal{H}^C) &= \mathcal{E}^A \oplus \{Z_2 \cup Z_3 | (Z_2 \in \mathcal{F}^B) \wedge (Z_3 \in \mathcal{H}^C) \wedge (Z_2 \cap C \subseteq Z_3) \wedge (Z_3 \cap B \subseteq Z_2)\} \\ &= \{Z_1 \cup Z_2 \cup Z_3 | (Z_1 \in \mathcal{E}^A) \wedge (Z_2 \in \mathcal{F}^B) \wedge (Z_3 \in \mathcal{H}^C) \wedge (Z_2 \cap C \subseteq Z_3) \\ &\wedge (Z_3 \cap B \subseteq Z_2) \wedge (Z_2 \cup Z_3 \cap A \subseteq Z_1) \wedge (Z_1 \cap B \cup C \subseteq Z_2 \cup Z_3)\}\end{aligned}$$

But from lemma 6.1 it follows that:

$$\mathcal{E}^A \oplus (\mathcal{F}^B \oplus \mathcal{H}^C) = (\mathcal{E}^A \oplus \mathcal{F}^B) \oplus \mathcal{H}^C$$

So operator $\oplus$ is associative. □

6.2 RMT $\mathcal{Z}$-pp cut

6.2.1 Proof of Theorem 4.11

Proof. Let $C = C_1 \cup C_2$ be the *RMT $\mathcal{Z}$*-pp cut which partitions $V \setminus C$ in sets $A, B \neq \emptyset$ s.t. $D \in A$ and $R \in B$. Let $\mathcal{Z}' = \{\bigcup_{u \in B} Z \cap N(u) : Z \in \mathcal{Z}\} \cup \{C_2\}$. We have that $\mathcal{Z}'_u = \{Z \cap N(u) : Z \in \mathcal{Z}'\} \cup \{C_2 \cap N(u)\} = \{(\bigcup_{v \in B} Z \cap N(v)) \cap N(u) : Z \in \mathcal{Z}\} \cup \{C_2 \cap N(u)\} = \{Z \cap N(u) : Z \in \mathcal{Z}\} \cup \{C_2 \cap N(u)\}$ but since $\forall u \in B : N(u) \cap C_2 \in \mathcal{Z}_u$, for every node u in B: $\mathcal{Z}_u = \mathcal{Z}'_u$. So far we have established that (a) nodes in B cannot tell whether $\mathcal{Z}$ or $\mathcal{Z}'$ is the adversary structure since $\forall u \in B : \mathcal{Z}_u = \mathcal{Z}'_u$ and (b) C_2 is an admissible corruption set in $\mathcal{Z}'$.

Suppose that there exists a safe algorithm $\mathcal{A}$ which achieves RMT in instance $(G, \mathcal{Z}, D, r)$. We consider the following runs e and e' of $\mathcal{A}$:

- Run e is on the instance $(G, \mathcal{Z}, D, R)$, with dealer's value $x_D = 0$, and corruption set C_1; in each round, all players in C_1 perform the actions that perform in the respective round of run e' (where C_1 is a set of honest players).
- Run e' is on the the instance $(G, \mathcal{Z}', D, R)$, with dealer's value $x_D = 1$, and corruption set C_2; in each round, all players in C_2

perform the actions that perform in the respective round of run e (where C_2 is a set of honest players).

Note that C_1, C_2 are admissible corruption sets in instances $(G, \mathcal{Z}, D, R)$, $(G, \mathcal{Z}', D, R)$ respectively. Since $C_1 \cup C_2$ is a cut which separates D from R in both $(G, \mathcal{Z}, D, R)$, $(G, \mathcal{Z}', D, R)$ and the actions of every node of this cut are identical in both runs e, e', the messages that R receives are the same in both runs, i.e., $view(v, e) = view(v, e')$. Therefore the decision of $R \in B$ must be identical in both runs. Since, by assumption, algorithm $\mathcal{A}$ achieves RMT in instance $(G, \mathcal{Z}, D, R)$, R must decide on the dealer's message 0 in run e on $(G, \mathcal{Z}, D, R)$, and must do the same in run e' on $(G, \mathcal{Z}', D, R)$. However, in run e' the dealer's message is 1. Therefore $\mathcal{A}$ makes R decide on an incorrect message in $(G, \mathcal{Z}', D, R)$. This contradicts the assumption that $\mathcal{A}$ is safe. $\square$

Bibliography

[1] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition)*. John Wiley Interscience, 2004. ISBN: 0-471-453242.

[2] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. "On the Time-Complexity of Broadcast in Radio Networks: An Exponential Gap Between Determinism and Randomization". In: *PODC*. 1987, pp. 98–108.

[3] Zuzana Beerliová-Trubíniová et al. "MPC vs. SFE: Perfect Security in a Unified Corruption Model". In: *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*. Ed. by Ran Canetti. Vol. 4948. Lecture Notes in Computer Science. Springer, 2008, pp. 231–250. ISBN: 978-3-540-78523-1. DOI: `10.1007/978-3-540-78524-8_14`. URL: `http://dx.doi.org/10.1007/978-3-540-78524-8_14`.

[4] Petra Berenbrink, Colin Cooper, and Zengjian Hu. "Energy efficient randomised communication in unknown AdHoc networks". In: *Theor. Comput. Sci.* 410.27-29 (2009), pp. 2549–2561.

[5] Danilo Bruschi and Massimiliano Del Pinto. "Lower Bounds for the Broadcast Problem in Mobile Radio Networks". In: *Distributed Computing* 10.3 (1997), pp. 129–135.

[6] Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. "Almost-Everywhere Secure Computation with Edge Corruptions". In: *J. Cryptology* 28.4 (2015), pp. 745–768. DOI: `10.1007/s00145-013-9176-3`. URL: `http://dx.doi.org/10.1007/s00145-013-9176-3`.

[7] I. Chlamtac and S. Kutten. "On Broadcasting in Radio Networks–Problem Analysis and Protocol Design". In: *Communications, IEEE Transactions on* 33.12 (1985), pp. 1240–1246. ISSN: 0090-6778.

[8] Bogdan S. Chlebus et al. “Deterministic broadcasting in unknown radio networks”. In: *SODA ’00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. San Francisco, California, United States: Society for Industrial and Applied Mathematics, 2000, pp. 861–870. ISBN: 0-89871-453-2.

[9] Bogdan S. Chlebus et al. “Deterministic Radio Broadcasting”. In: *ICALP*. Ed. by Ugo Montanari, José D. P. Rolim, and Emo Welzl. Vol. 1853. Lecture Notes in Computer Science. Springer, 2000, pp. 717–728. ISBN: 3-540-67715-1.

[10] Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. “Fast Broadcasting and Gossiping in Radio Networks”. In: *FOCS*. 2000, pp. 575–581.

[11] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. “Distributed broadcast in radio networks of unknown topology”. In: *Theor. Comput. Sci.* 302.1-3 (2003), pp. 337–364.

[12] Artur Czumaj and Wojciech Rytter. “Broadcasting Algorithms in Radio Networks with Unknown Topology”. In: *FOCS*. IEEE Computer Society, 2003, pp. 492–501. ISBN: 0-7695-2040-5.

[13] Yvo Desmedt and Yongge Wang. “Perfectly Secure Message Transmission Revisited”. English. In: *Advances in Cryptology EUROCRYPT 2002*. Ed. by LarsR. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, pp. 502–517. ISBN: 978-3-540-43553-2. DOI: 10.1007/3-540-46035-7_33. URL: http://dx.doi.org/10.1007/3-540-46035-7_33.

[14] Danny Dolev. “The Byzantine Generals Strike Again”. In: *J. Algorithms* 3.1 (1982), pp. 14–30.

[15] Danny Dolev et al. “Perfectly secure message transmission”. In: *J. ACM* 40.1 (Jan. 1993), pp. 17–47. ISSN: 0004-5411. DOI: 10.1145/138027.138036. URL: http://doi.acm.org/10.1145/138027.138036.

[16] Shlomi Dolev, Omri Liba, and Elad Michael Schiller. “Self-stabilizing Byzantine Resilient Topology Discovery and Message Delivery - (Extended Abstract)”. In: *NETYS*. Ed. by Vincent Gramoli and Rachid Guerraoui. Vol. 7853. Lecture Notes in Computer Science. Springer, 2013, pp. 42–57. ISBN: 978-3-642-40147-3.

[17] Matthias Fitzi and Ueli M. Maurer. “Efficient Byzantine Agreement Secure Against General Adversaries”. In: *DISC*. Ed. by Shay Kutten. Vol. 1499. Lecture Notes in Computer Science. Springer, 1998, pp. 134–148. ISBN: 3-540-65066-0.

[18] Pierre Fraigniaud, Amos Korman, and David Peleg. "Towards a Complexity Theory for Local Distributed Computing". In: *J. ACM* 60.5 (Oct. 2013), 35:1–35:26. ISSN: 0004-5411. DOI: `10.1145/2499228`. URL: `http://doi.acm.org/10.1145/2499228`.

[19] Juan A. Garay and Yoram Moses. "Fully Polynomial Byzantine Agreement for $n > 3t$ Processors in $t + 1$ Rounds". In: *SIAM J. Comput.* 27.1 (1998), pp. 247–290.

[20] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN: 0-7167-1044-7.

[21] Leszek Gasieniec et al. "Time efficient k-shot broadcasting in known topology radio networks". In: *Distributed Computing* 21.2 (2008), pp. 117–127.

[22] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. 1st ed. New York, NY, USA: Cambridge University Press, 2008. ISBN: 052188473X.

[23] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM J. Comput.* 18.1 (1989), pp. 186–208. DOI: `10.1137/0218012`. URL: `http://dx.doi.org/10.1137/0218012`.

[24] Joseph Y. Halpern and Yoram Moses. "Knowledge and Common Knowledge in a Distributed Environment". In: *J. ACM* 37.3 (July 1990), pp. 549–587. ISSN: 0004-5411. DOI: `10.1145/79147.79161`. URL: `http://doi.acm.org/10.1145/79147.79161`.

[25] Martin Hirt. "Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting". Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001. PhD thesis. ETH Zurich, Sept. 2001.

[26] Martin Hirt and Ueli M. Maurer. "Complete Characterization of Adversaries Tolerable in Secure Multi-Party Computation (Extended Abstract)". In: *PODC*. Ed. by James E. Burns and Hagit Attiya. ACM, 1997, pp. 25–34. ISBN: 0-89791-952-1.

[27] Akira Ichimura and Maiko Shigeno. "A new parameter for a broadcast algorithm with locally bounded Byzantine faults". In: *Inf. Process. Lett.* 110.12-13 (2010), pp. 514–517.

[28] Erez Kantor and David Peleg. "Efficient k-Shot Broadcasting in Radio Networks". In: *DISC*. Ed. by Idit Keidar. Vol. 5805. Lecture Notes in Computer Science. Springer, 2009, pp. 481–495. ISBN: 978-3-642-04354-3.

[29] Sushanta Karmakar et al. "k-shot Broadcasting in Ad Hoc Radio Networks". In: *CoRR* abs/1603.08393 (2016). URL: `http://arxiv.org/abs/1603.08393`.

[30] Chiu-Yuen Koo. "Broadcast in radio networks tolerating byzantine adversarial behavior". In: *PODC*. Ed. by Soma Chaudhuri and Shay Kutten. ACM, 2004, pp. 275–282. ISBN: 1-58113-802-4.

[31] Chiu-Yuen Koo et al. "Reliable broadcast in radio networks: the bounded collision case". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*. Ed. by Eric Ruppert and Dahlia Malkhi. ACM, 2006, pp. 258–264. ISBN: 1-59593-384-0. DOI: `10.1145/1146381.1146420`. URL: `http://doi.acm.org/10.1145/1146381.1146420`.

[32] Paraschos Koutris and Aris Pagourtzis. "Oblivious k-shot Broadcasting in Ad Hoc Radio Networks". In: *Seventeenth Computing: The Australasian Theory Symposium, CATS 2011, Perth, Australia, January 2011*. Ed. by Alex Potanin and Taso Viglas. Vol. 119. CRPIT. Australian Computer Society, 2011, pp. 161–168. ISBN: 978-1-920682-98-9. URL: `http://crpit.com/abstracts/CRPITV119Koutris.html`.

[33] Dariusz R. Kowalski and Andrzej Pelc. "Broadcasting in undirected ad hoc radio networks". In: *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*. Boston, Massachusetts: ACM, 2003, pp. 73–82. ISBN: 1-58113-708-7. DOI: `http://doi.acm.org/10.1145/872035.872045`.

[34] Dariusz R. Kowalski and Andrzej Pelc. "Time of Deterministic Broadcasting in Radio Networks with Local Knowledge". In: *SIAM J. Comput.* 33.4 (2004), pp. 870–891. DOI: `10.1137/S0097539702419339`. URL: `http://dx.doi.org/10.1137/S0097539702419339`.

[35] Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. "Fault-Tolerant Broadcasting in Radio Networks". In: *Journal of Algorithms* 39.1 (2001), pp. 47 –67. ISSN: 0196-6774. DOI: `10.1006/`

jagm.2000.1147. URL: http://www.sciencedirect.com/science/article/pii/S0196677400911477.

[36] M V N Ashwin Kumar et al. "On perfectly secure communication over arbitrary networks". In: *Proceedings of the twenty-first annual symposium on Principles of distributed computing*. PODC '02. Monterey, California: ACM, 2002, pp. 193–202. ISBN: 1-58113-485-1. DOI: 10.1145/571825.571858. URL: http://doi.acm.org/10.1145/571825.571858.

[37] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (1982), pp. 382–401.

[38] Chris Litsas, Aris Pagourtzis, and Dimitris Sakavalas. "A Graph Parameter That Matches the Resilience of the Certified Propagation Algorithm". In: *ADHOC-NOW*. Ed. by Jacek Cichon, Maciej Gebala, and Marek Klonowski. Vol. 7960. Lecture Notes in Computer Science. Springer, 2013, pp. 269–280. ISBN: 978-3-642-39246-7.

[39] Nancy A. Lynch. *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. ISBN: 1558603484.

[40] Gianluca De Marco. "Distributed broadcast in unknown radio networks". In: *SODA*. Ed. by Shang-Hua Teng. SIAM, 2008, pp. 208–217.

[41] Gianluca De Marco and Andrzej Pelc. "Faster broadcasting in unknown radio networks". In: *Inf. Process. Lett.* 79.2 (2001), pp. 53–56.

[42] Mikhail Nesterenko and Sébastien Tixeuil. "Discovering Network Topology in the Presence of Byzantine Faults". In: *IEEE Trans. Parallel Distrib. Syst.* 20.12 (2009), pp. 1777–1789.

[43] Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. "Brief Announcement: Reliable Message Transmission under Partial Knowledge and General Adversaries". In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*. 2016, pp. 203–205. DOI: 10.1145/2933057.2933080. URL: http://doi.acm.org/10.1145/2933057.2933080.

[44] Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. “Reliable Broadcast with Respect to Topology Knowledge”. In: *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*. Ed. by Fabian Kuhn. Vol. 8784. Lecture Notes in Computer Science. Springer, 2014, pp. 107–121. ISBN: 978-3-662-45173-1. DOI: 10.1007/978-3-662-45174-8_8. URL: http://dx.doi.org/10.1007/978-3-662-45174-8_8.

[45] Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. “Reliable broadcast with respect to topology knowledge”. In: *Distributed Computing* (2016), pp. 1–16.

[46] Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. “Reliable Message Transmission under Partial Knowledge”. In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 243. URL: http://eprint.iacr.org/2015/243.

[47] Andrzej Pelc and David Peleg. “Broadcasting with locally bounded Byzantine faults”. In: *Inf. Process. Lett.* 93.3 (2005), pp. 109–115.

[48] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. DOI: 10.1137/1.9780898719772. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9780898719772. URL: http://epubs.siam.org/doi/abs/10.1137/1.9780898719772.

[49] Eric Ruppert and Dahlia Malkhi, eds. *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*. ACM, 2006. ISBN: 1-59593-384-0. URL: http://dl.acm.org/citation.cfm?id=1146381.

[50] Bhavani Shankar et al. “Unconditionally Reliable Message Transmission in Directed Networks”. In: *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’08. San Francisco, California: Society for Industrial and Applied Mathematics, 2008, pp. 1048–1055. URL: http://dl.acm.org/citation.cfm?id=1347082.1347197.

[51] Kannan Srinathan and C. Pandu Rangan. “Possibility and complexity of probabilistic reliable communication in directed networks”. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*. Ed. by Eric Ruppert and Dahlia Malkhi. ACM, 2006, pp. 265–274. ISBN: 1-59593-384-0. DOI:

10.1145/1146381.1146421. URL: http://doi.acm.org/10.1145/1146381.1146421.

[52] Kannan Srinathan et al. “Unconditionally Secure Message Transmission in Arbitrary Directed Synchronous Networks Tolerating Generalized Mixed Adversary”. In: *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*. ASIACCS ’09. Sydney, Australia: ACM, 2009, pp. 171–182. ISBN: 978-1-60558-394-5. DOI: 10.1145/1533057.1533083. URL: http://doi.acm.org/10.1145/1533057.1533083.

[53] Lewis Tseng, Nitin Vaidya, and Vartika Bhandari. “Broadcast using certified propagation algorithm in presence of Byzantine faults”. In: *Information Processing Letters* 115.4 (2015), pp. 512 –514. ISSN: 0020-0190. DOI: http://dx.doi.org/10.1016/j.ipl.2014.11.010. URL: http://www.sciencedirect.com/science/article/pii/S0020019014002609.

[54] Lewis Tseng and Nitin H. Vaidya. “Iterative Approximate Byzantine Consensus under a Generalized Fault Model”. In: *Distributed Computing and Networking, 14th International Conference, ICDCN 2013, Mumbai, India, January 3-6, 2013. Proceedings*. Ed. by Davide Frey et al. Vol. 7730. Lecture Notes in Computer Science. Springer, 2013, pp. 72–86. ISBN: 978-3-642-35667-4. DOI: 10.1007/978-3-642-35668-1_6. URL: http://dx.doi.org/10.1007/978-3-642-35668-1_6.

[55] Lewis Tseng, Nitin H. Vaidya, and Vartika Bhandari. “Broadcast Using Certified Propagation Algorithm in Presence of Byzantine Faults”. In: *CoRR* abs/1209.4620 (2012).

[56] Andrew Chi-Chih Yao. “Protocols for Secure Computations (Extended Abstract)”. In: *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.38. URL: http://dx.doi.org/10.1109/SFCS.1982.38.