Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ασφαλής Επικοινωνία και Τοπολογική γνώση

Διπλωματική Εργασία

Γεώργιος Παναγιωτάκος

Επιβλέπων: Αριστείδης Παγουρτζής
Επίκουρος Καθηγητής Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών
Αθήνα, Δεκέμβριος 2014

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ασφαλής Επικοινωνία και Τοπολογική γνώση

Διπλωματική Εργασία

Γεώργιος Παναγιωτάκος

Επιβλέπων: Αριστείδης Παγουρτζής
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την $10^{η}$ Δεκεμβρίου 2014.

.....	.....	.....
Άγγελος Κιαγιάς	Άρης Παγουρτζής	Δημήτρης Φωτάκης
Επίκουρος Καθηγητής Ε.Κ.Π.Α	Επίκουρος Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών
Αθήνα, Δεκεμβριος 2014

.....
Γεώργιος Παναγιωτάκος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η αξιόπιστη μετάδοση είναι ένα από τα βασικά προβλήματα στα δίκτυα επικοινωνίας. Μελετάμε αυτό το πρόβλημα σε γενικά δίκτυα ενάντια σε Βυζαντινό αντίπαλο και σε σχέση με την τοπολογική γνώση των παικτών. Θεωρούμαι ότι ο αντίπαλος περιγράφεται είτε από το τοπικά περιορισμένο μοντέλο του Koo (2004) είτε από το μοντέλο γενικού αντιπάλου των Hirt και Maurer (1997) και διερευνούμε την σχέση μεταξύ του επιπέδου γνώσης της τοπολογίας και της επιλυσιμότητας του προβλήματος.

Βελτιώνουμε την τεχνική του τοπικού ζεύγους διαχωριστών των Pelc και Peleg (2005) έτσι ώστε να αποκτήσουμε αποτελέσματα μη επιλυσιμότητας για κάθε επίπεδο τοπολογικής γνώσης και για κάθε τύπο αντιπάλου. Στην θετική πλευρά κατασκευάζουμε πρωτόκολλα που ταιριάζουν σε αυτά τα όρια μη επιλυσιμότητας, και έτσι χαρακτηρίζουμε πλήρως την κλάση των γραφημάτων στα οποία η αξιόπιστη μετάδοση είναι δυνατή.

Ανάμεσα στα άλλα, δείχνουμε ότι το πρωτόκολλο Certified Propagation Algorithm (CPA) του Koo είναι μοναδικό ενάντια σε τοπικά περιορισμένους αντιπάλους σε ad-hoc δίκτυα, δηλαδή μπορεί να αντέξει όσο δυνατούς τοπολογικά περιορισμένους αντιπάλους όσο οποιοσδήποτε άλλος αλγόριθμος. Αυτό το αποτέλεσμα δίνει απάντηση και σε μια ανοιχτή ερώτηση των Pelc και Peleg. Επίσης κατασκευάζουμε μια προσαρμογή του CPA ενάντια σε γενικούς αντιπάλους και δείχνουμε την μοναδικότητα του. Απ΄ όσο γνωρίζουμε αυτός είναι ο πρώτος βέλτιστος αλγόριθμος για αξιόπιστη μετάδοση σε ad-hoc δίκτυα ενάντια σε γενικούς αντιπάλους.

Λέξεις Κλειδιά

αξιόπιστη μετάδοση, βυζαντινοί στρατηγοί, τοπικά περιορισμένος αντίπαλος, γενικός αντίπαλος, τοπολογική γνώση, *ad hoc* δίκτυα

Abstract

Reliable Broadcast is a fundamental problem of communication networks. We study this problem in incomplete networks against a Byzantine adversary and with respect to player's topology knowledge. 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.

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 that match the obtained bounds and thus, exactly characterize the classes of graphs in which Reliable Broadcast is possible.

Among others, we show that Koo's Certified Propagation Algorithm (CPA) is *unique* against locally bounded adversaries in *ad hoc* networks, that is, it can tolerate as many local corruptions as any other non-faulty algorithm; this settles an open question posed by Pelc and Peleg. We also provide an adaptation of CPA against general adversaries and show its uniqueness. To the best of our knowledge this is the first optimal algorithm for Reliable Broadcast in generic topology *ad hoc* networks against general adversaries.

Keywords

reliable broadcast, byzantine generals, locally bounded adversary, topology knowledge, *ad hoc* networks, general adversary

Ευχαριστίες

Κατ΄ αρχήν θα ήθελα να ευχαριστήσω τους Δημήτρη Σακαβάλα και Άρη Παγουρτζή γιατί πρώτον, ένα κομμάτι αυτής της δουλειάς είναι και δικό τους, και δεύτερον γιατί με έφεραν σε επαφή με την θεωρητική κρυπτογραφία και την ερευνητική διαδικασία δείχνοντας ατέλειωτη υπομονή. Ακόμα κατά χρονολογική σειρά ευχαριστώ τον κ. Νίκο Παπασπύρου, ο οποίος με έφερε πρώτη φορά σε επαφή με την θεωρητική πληροφορική, και τον κ. Δημήτρη Φωτάκη, που με μύησε σε βαθύτερες έννοιες της θεωρίας αλγορίθμων. Χάρη σε αυτούς κατάφερα να πραγματοποιήσω ένα όνειρο μου, την παρουσίαση κομματιού της παρούσας εργασίας σε ενα μεγάλο επιστημονικό συνέδριο.

Ακόμα θα ήθελα να ευχαριστήσω όλα τα παιδιά από το εργαστήριο λογικής και επιστήμης υπολογισμών για την υποστήριξη τους, αλλά και για την ανθρώπινη ατμόσφαιρα που έχει διαμορφωθεί, σε αντίθεση με τον ανταγωνισμό που κυριαρχεί σε κάθε πτυχή της επαγγελματικής ζωής. Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου για την υποστήριξη σε όλα τα επίπεδα στην προσπάθεια να κυνηγήσω τα όνειρα μου.

Γιώργος Παναγιωτάκος

Contents

List of Figures

Chapter 1

Introduction

Technology has always played an important role in human history. Whole eras where named after the technological advancements that shaped them. For example tool making technologies defined three large periods of human history, the stone, bronze and iron age. It is no exaggeration to say that the last twenty years are the age of the Internet.

However, as with any technological achievement, whether its use is for better or for worse, is entirely determined by the ones who use it. Internet is not an exception to this rule. Lately, it was made clear after the Snowden leaks that through the internet one can achieve massive privacy violations with little effort. So it is in our hands to guard and support the fair use of this technology.

This work is towards this direction and tries to explore some of the fundamental problems of the Internet, and of communication networks in general.

Modern communication networks consist of millions of nodes. At any time some of them may crash or even behave maliciously. To resolve this seemingly chaotic behavior we need to explore algorithms that exhibit reliability and privacy properties, starting from the most simple actions in a network.

Some of those actions are :

- *Reliable broadcast*, where a node needs to sent the same message to the whole network while preserving some fundamental reliability properties.
- *Secure message transmission*, where a node needs to send a message to another node in the network , and no one else should learn the content of this message.
- *Byzantine agreement*, where all nodes in a network should agree collectively on the same value.

These are the building blocks for even more complicated actions, like e-voting or e-shopping.

A main design idea behind all these protocols, is that we don't want our networks to have a single point of failure. On the contrary we need our algorithms to behave in a distributed manner, utilizing all the resources in the network and distributing the reliability to the whole network.

1.1 The Communication Model

We start by explaining the way we model communication networks. The network is represented by a graph. Nodes represent the players and edges the communication channels between them. We will use the terms player, process, node interchangeably from now on. Nodes have some initial input at the start of the protocol. Information is exchanged between the players by sending messages through the communication channels.

The channels used can be classified according to their reliability and privacy properties. They can be:

- *Authenticated*, where information send cannot be tampered. That is the adversary cannot alter the messages sent between two honest players through a communication channel. However he may be able to read what is being sent.
- *Secret/Confidential*, messages cannot be read by the adversary, but they may be tampered.
- *Secure*, messages cannot be read or tampered by the adversary.

Depending on whether assumptions about process execution speeds and message delivery delays have been made we differentiate between two types of communication networks.

- *Synchronous*, where players have access to a global clock. We can think of our protocol taking place in successive rounds.
- *Asynchronous*, where no global clock exists and message delays are unbounded but finite.

In the rest of this work, if it is not specified differently, we will talk about synchronous systems.

1.2 The Adversary Model

As mentioned earlier in the real world scenarios we are trying to model, some of the players may behave maliciously or crash. Protocols that take into account this kind of behavior are called fault tolerant protocols. We are interested in the worst case assumption, so we consider an external entity, the adversary, which controls and coordinates the actions of the faulty (corrupted) players.

There are many types of faults. Different types of faults limit the power of the adversary in different ways. Relevant to this work are the following:

- *Active corruption*, where the adversary has total control over the corrupted player. He can read all the information this player has and can make him behave arbitrarily. This is the strongest type of corruption.

- *Passive corruption*, where the adversary knows everything the player knows as before, namely his input and the messages he receives, but cannot make him deviate from the specified protocol. That is the passively corrupted player will follow the protocol as a fully honest player, but the adversary will know his input and the messages he received.

The adversary may also be limited by his computational power. We are mostly interested in two cases. First the computationally bounded adversary. In this case the adversary is considered to have computational power equivalent to a probabilistic polynomial time deterministic Turing machine. For this type of machines it is hypothesized that certain problems (e.g. the discrete logarithm problem) are hard. A large part of modern cryptography bases its security on this assumption.

If we don't want to base security on this kind of assumptions we can consider a computationally unbounded adversary. We don't make any assumptions for such an adversary and we consider him to have unbounded computational power.

A different limitation on the power of the adversary is related to the possible set of nodes he can corrupt. For example in the threshold model the adversary can corrupt up to some maximum number of players, in the whole communication network. This model is not always relevant; for example in this work we assume that we only have information about the adversary that are topologically limited. But the threshold model refers to the whole communication graph, and thus is not relevant. The main adversarial modes with respect to this parameter that are going to be used in this work are the following:

- *Threshold adversary*: the adversary is described by the total number of nodes he can corrupt in the whole communication graph.
- *Locally bounded adversary*: the adversary is described by the total number of nodes he can corrupt in each neighborhood in the graph.
- *General adversary*: the adversary is described by the set of possible corruption sets of nodes.

The threshold adversary is the one that was considered earlier in the fault tolerance literature. The majority of the relevant research in the 80' and the 90' was done under this adversary model. The general adversary was introduced by Fitzi and Maurer in 1998 [11] and it is a strict generalization of the threshold and the locally bounded model. The locally bounded model was introduced in [17] to describe uniform spatial distribution of the corruptions.

1.3 Security

The primary goal when designing a fault tolerant protocol is to prove that it is secure in a well defined security model. This model should specify which assumptions we make about the communication network and the adversary, as described before. Security is

expressed with respect to the possibility of wrong output at the end of the protocol and according to the security parameter λ. A negligible error probability may be allowed. We are interested in three levels of security:

- *Perfect security*, security against a computationally unbounded adversary with zero error probability.
- *Unconditional security*, security against a computationally unbounded adversary with negligible error probability.
- *Computational security*, security against a computationally bounded adversary with negligible error probability.

1.4 Measures of Efficiency

The second goal when designing distributed protocols is efficiency. Defining the efficiency of distributed protocols is not an easy task. Usually we measure efficiency by the time complexity of our algorithm. But in distributed protocols other resources except time are important. For example in mobile networks we have limited bandwidth and each bit transfered costs money. Having a protocol that completes some task with low amount of data exchanged may be equally significant to the time it takes to complete the task. A different limitation arises in sensor networks. The machines used by the players in these networks have limited energy capacity. One of our targets when we design algorithms for this setting is to minimize the energy consumed. In these work we are mainly interested in the following efficiency measures of protocols:

- *Time complexity* is defined as the worst case number of rounds it takes to finish an execution of the protocol i.e. all the players produce their output and halt.
- *Local computational complexity* is defined as the maximum over the worst time computational complexities over all players.
- *Message complexity* is defined as the worst case total number of messages sent over the network during an execution of the protocol.
- *Communication* or *Bit complexity* is defined as the worst case total number of bits sent over the network during an execution of the protocol.

Notice that all these measures in case of a faulty system are defined on the set of the honest players, because otherwise a corrupt player could not terminate, or keep sending messages for an infinite amount of time.

Chapter 2

Agreement problems

As discussed before a fundamental task of distributed computing is agreement on a common value. If processors cannot agree on a common value, there is not much hope that they can do more complicated tasks distributively. That is why this problem has been studied extensively since 1980. There are many ways to formalize this task and we are going to describe two of them.

2.1 Formal definitions

The first one is Byzantine Agreement. In this problem every player has an initial input. We want all honest players at the end of the protocol to decide on a common value, despite some of them being faulty.

Definition 1. *A protocol achieves Byzantine Agreement iff the following properties hold:*

- *Validity: If all honest players have as input the same value, then all honest players should agree on this value.*
- *Agreement: All honest players should decide on the same value.*
- *Termination: After a finite number of rounds, every honest player should decide on a value.*

A closely related problem is Reliable Broadcast. In this problem one player is the dealer and he wants to send his value to all other players.

Definition 2. *A protocol achieves Reliable Broadcast iff the following properties hold:*

- *Validity: If the dealer follows the protocol then all players should decide on the dealer's value.*
- *Agreement: All honest players should decide on the same value.*
- *Termination: After a finite number of rounds, every honest player should decide on a value.*

Byzantine Agreement and Reliable Broadcast are closely related. For a t-threshold adversary with $t < n/2$ and on a complete communication graph these two protocols are equivalent. Having a protocol for one of them implies a protocol for the other one. If we have Reliable Broadcast then we can achieve Byzantine Agreement by letting all players reliably broadcast their value, and then decide on the majority of the values they have taken. If we have Byzantine Agreement, then the dealer can send his value to all players, and using this value as input, run the Byzantine Agreement protocol.

Additionally for the rest of this work we will not prove the termination property, because for all cases that the validity and agreement properties hold, termination is trivially implied.

2.2 History

The problem of reliable broadcast was first solved at 1980 by Pease, Shostak and Lamport [23]. In this work, the protocol runs in $t + 1$ rounds and can tolerate less than $n/3$ corrupted players, but players send messages of exponential size and take exponential number of computational steps to finish. This paper initiated a long line of results, on getting an optimal protocol on the number of rounds and the resilience with polynomial communication and computation complexity. Polynomial time was achieved initially by taking more rounds [8]. Coan in 1986 [5] presented a family of broadcast protocols that tolerate up to $n/4$ corruptions and halt in less than $2t$ rounds, while using messages of polynomial size but taking exponential number of computational steps locally. Lots of other results continued this line of work towards a fully polynomial Broadcast protocol. Finally in 1998 Garay and Moses [12] presented the first fully polynomial Byzantine Agreement protocol of optimal resilience and round complexity.

2.3 An upper bound on the number of corruptions

In this section we present the classical byzantine agreement impossibility result on a complete communication graph of Dolev [6] with a slightly different proof.

Theorem 1. *Byzantine Agreement is impossible with $t >= n/3$ corrupted players, where n denotes the number of players in the network.*

Proof. We are going to prove this theorem by first analyzing the case with 3 players and then generalizing this result to an arbitrary number of players.

Let G be the complete communication network and a, b, c the players. Without loss of generality suppose the set of possible outputs for the players at the end of the agreement is $\{0, 1\}$. Suppose that a Byzantine Agreement protocol Π exists that can tolerate $n/3$ corrupted players. We are going to consider three different runs of this protocol and show that a contradiction arises.

Let $\sigma_1, \sigma_2, \sigma_3$ be the scenarios depicted in figure 2.1. For σ_1 the corrupted node is a and the honest nodes b, c have the same initial input 0. Since Π achieves Byzantine

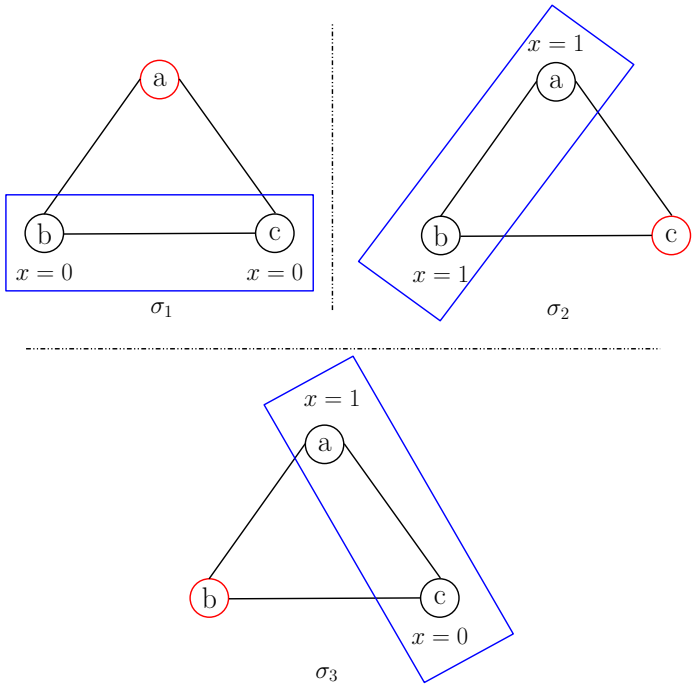

Figure 2.1: Three scenarios that Π must achieve byzantine agreement on.The honest nodes are blue. The corrupt node is red.

Agreement with $n/3$ corrupted nodes, it should work on this scenario. From validity we have that nodes b, c will output 0 for any possible behavior of the corrupted node a. Thus $y_b^{\sigma_1} = y_c^{\sigma_1} = 0$, where y_v^{σ} denotes the output of node v in scenario σ. For σ_2 the corrupted node is c and the honest nodes a, b have the same initial input 1. Repeating the same argument we can show that for any possible behavior of the corrupted node c, $y_a^{\sigma_2} = y_b^{\sigma_2} = 1$.

For σ_3 the corrupted node is b and the honest nodes a, c have different initial input. Node a has input 1, and node c has input 0. Let b play in the following way in σ_3:

- b sends to a in σ_3 exactly the same messages that b sends to a in σ_2, when c in σ_2 sends the same messages as c in σ_3.
- b sends to c in σ_3 exactly the same messages that b sends to c in σ_1, when a in σ_1 sends the same messages as a in σ_3.

Node a has the same view in σ_2 and σ_3, since he has the same initial value and exchanges the same messages in both scenarios and has to give the same output. So $y_a^{\sigma_3} = y_a^{\sigma_2} = 1$. But node c also has the same view in σ_1 and σ_3, so $y_b^{\sigma_3} = y_b^{\sigma_1} = 0$. But this is a contradiction since it breaks agreement, two honest nodes output different values at the end of the protocol. Notice that b can launch his attack by simulating the system shown in figure 2.2. This also shows that our system is well defined. So no protocol exists that achieves Byzantine Agreement with 3 nodes where 1 is corrupted.

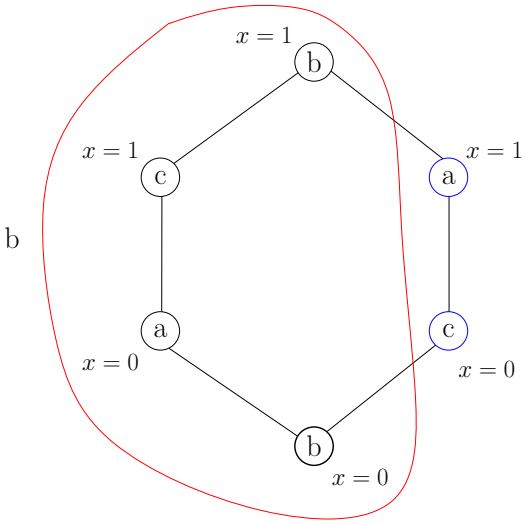

Figure 2.2: The simulation player b does to attack protocol Π. Blue nodes represent the honest nodes. The nodes on the red area are being simulated by b.

We can now do a reduction from the general case with $n <= 3 * t$ players and at least t corrupt nodes to the one with 3 players an 1 corrupted node. We split the players into three non-empty groups A, B, C, where each group has at most t participants.Then any two groups have at least $n - t$ participants.

We can consider each node group as one super-node from the $n = 3$ case and show that every protocol that achieves Reliable Broadcast in the n node system also achieves Reliable Broadcast in the super-node system. All internal communication and computation inside a group can be simulated by a super-node and all messages exchanged between nodes of different groups are send between two super-nodes. As before all 3 super-nodes can be corrupted. Additionally we can simulate the input of some super-node, as all nodes in this group having the same input. If validity holds for the n node system then validity should hold for the super-node system. And if agreement holds for the n node system then agreement should hold for the super-node system.

If we had a protocol that achieved Reliable Broadcast for the n node system then we could use it to construct a protocol that solves Reliable Broadcast for the super node case. But the super node case is equivalent to the $n = 3$ case, where Reliable Broadcast is impossible. Thus a contradiction and the theorem is proved. □

2.4 A lower bound on message complexity

As we saw in the previous section, to achieve Reliable Broadcast each node should be able to distinguish between scenarios where a different output should be given. But the way it differentiates between different scenarios is by his view e.g. his initial input and the messages he exchanges with other nodes. So is there a minimum number of

messages a node should get to be able to differentiate between different scenarios? The next theorem from [9] gives us a first answer to this question.

Theorem 2. *Each protocol that achieves Reliable Broadcast with up to t corruptions requires at least $n(t+1)/4$ messages to be exchanged.*

Proof. Let σ_0 and σ_1 be the scenarios where all players are honest and the dealer transmits 0 and 1 respectively. Also C_i^u is the set of players that exchange messages with u in σ_i.

Suppose there exists some u s.t. $|C_0^u \cup C_1^u| <= t$. Then let σ_2 be the scenario where $T = C_0^u \cup C_1^u$ are corrupt and nodes in T behaves towards u like they are in σ_0 and towards the other players like they are in σ_1. For u, σ_0 and σ_2 seem indistinguishable, he has exactly the same view in both scenarios. Also for players in $V \setminus \{u\} \cup T$ scenarios σ_1 and σ_2 are indistinguishable. Notice that if u could interact with nodes in σ_0 and σ_1 in σ_2 the two scenarios would not be indistinguishable. Now since in indistinguishable scenarios a node should give the same output, u outputs 0 and nodes in $V \setminus \{u\} \cup T$ output 1. But this is a contradiction since agreement should be preserved. So $|C_0^u \cup C_1^u| > t$

But then in σ_0 and σ_1 a total of at least $n(t+1)/2$ messages should be exchanged. So in one of the two scenarios at least $n(t+1)/4$ messages are exchanged establishing a lower bound on the message complexity of every protocol that achieves Reliable Broadcast. □

Chapter 3

Protocols for Reliable Broadcast

In this chapter, a number of reliable broadcast protocols are presented. At first, a protocol for the threshold setting, the Phase King protocol. It has been chosen because of its good performance, although not optimal, and its very simple modular nature. Next, a protocol for the general adversary setting is presented. This protocol is based on the same idea as the first fully polynomial protocol of optimal resiliency and round complexity [12], presented earlier by Bar-noy et al. [1]. Notable in this work is the shift of protocol design towards active corruption detection from the players. That is, players throughout the protocol find and update information about corrupted players and use them accordingly. Finally, two protocols are briefly summarized that employ different perspectives to Reliable Broadcast; first from a knowledge theoretic and then from the randomized algorithm point of view.

3.1 The Phase King Agreement protocol

In this section, we focus on the Phase King agreement protocol [3]. We are going to build our protocol through three weaker agreement primitives: weak, graded and king consensus.

3.1.1 Weak consensus

In weak consensus a first agreement is build between a subgroup of the honest players on a common value. All other honest players do not decide on any value. So two properties should hold for the weak consensus protocol:

- Weak Consistency: The output value of all honest players is in $\{x, \perp\}$ for some x in $\{0, 1\}$.
- Correctness: If all honest players at the beginning of the protocol have the same input x, then they all output x.

A protocol realizing these properties is the following:

---

function WeakConsensus(P,t,$x = (x_1, ..., x_n)$)

1. Every player $p_i \in P$ sends x_i to all players.
2. Player p_j outputs $y_i = \begin{cases} x, & \text{if}(|\{p_j | x_j^i = x\}| >= n - t) \\ \perp, & \text{otherwise} \end{cases}$

end function

Proof. If all honest players have the same input x, there will be at least $n - t$ messages with this value sent to all players, since there are at most t corrupted players. So correctness is preserved.

Let's suppose that there exist two honest nodes p_i and p_j that output x and x' respectively. Since p_i outputs x, it got $n-t$ messages with this value. At most t messages were from corrupted players, so there are at least $n - 2t$ honest players with this value. These players also send the same value to p_j. So for p_j to output x', $2t >= n - t$ should hold. But then $n <= 3t$ which is a contradiction. So the output value of all honest players is in $\{x, \perp\}$, for some x, and this protocol preserves Weak Consistency. □

3.1.2 Graded consensus

In graded consensus each player outputs two values (y, g). y_i refers to the output value of p_i and g_i to his confidence level on whether all other honest players have output this value. We want the following properties to be preserved:

- Graded Consistency: If an honest players outputs $(y, 1)$, then all honest player should output $(y, 1)$ or $(y, 0)$.
- Graded Correctness: If all honest players at the beginning of the protocol have the same input x, then they all output $(x, 1)$.

A protocol realizing these properties is the following:

---

function GradedConsensus(P,t,$x = (x_1, ..., x_n)$)

1. Run WeakConsensus(P,t,$x = (x_1, ..., x_n)$) and let $x' = (x'_1, ..., x'_n)$ be its output.
2. Every player $p_i \in P$ sends x'_i to all players, except if $x'_i = \perp$, p_i sends nothing.
3. Player p_i calculates y_i, g_i as follows:

$$y_i = \begin{cases} x, & \text{if}(|\{p_j | x'^i_j = x\}| > t) \\ 0, & \text{otherwise} \end{cases}$$

$$g_i = \begin{cases} 1, & \text{if}(|\{p_j | x'^i_j = x\}| >= n - t) \\ 0, & \text{otherwise} \end{cases}$$

end function

Proof. Firstly let's note that honest players that have output $\perp$ value from the weak consistency phase, do not send anything. Let p_i output $(y, 1)$. Then at least one honest player has sent y to p_i. From weak consistency this implies that all honest players sent y or did not send anything. Also $n - t$ players have sent x. Since there are at most t corrupted players, $n-2t$ players have sent x and $n-2t > t$. So all honest players decide on $y_i = y$. If some other player p_j could decide on some $y' \neq y$, then at least $t+1$ players should have sent him y'. This is a contradiction since at most t players can send a value different than y and the protocol preserves Graded Consistency.

If all honest players have the same input x initially then weak consensus will preserve these values. Obviously all honest players will output $(x, 1)$ since they get this value from at least $n - t$ players. So graded correctness is preserved. □

3.1.3 King consensus

In the king consensus one player takes the role of the king. If he is honest, then consensus is guaranteed. Additionally we want correctness to be preserved.

- King Consistency: If the king is an honest player, then all honest players output the same value.
- Correctness: If all honest players at the beginning of the protocol have the same input x, then they all output x.

A protocol realizing these properties is the following:

---

function KingConsensus(P,t,$x = (x_1, ..., x_n)$,p_k)

1. Run GradedConsensus(P,t,$x = (x_1, ..., x_n)$) and let (y_i, g_i) be the output of player p_i.
2. The king (p_k) sends its value to all players.
3. Player p_i calculates its output as follows:

$$z_i = \begin{cases} y_i, & \text{if}(g_i = 1) \vee (y_k = \perp) \\ y_k, & \text{otherwise} \end{cases}$$

end function

Proof. Let's suppose the king is honest. If from graded consistency exists some honest player with output $(y, 1)$, then all honest players have output either $(y, 1)$ or $(y, 0)$ and for player p_i, it holds that $y_i = y_k = y = z_i$. If no honest player outputs $(y, 1)$, then $z_i = y_k$ for all players. So king consistency is preserved.

If all honest players have the same input x initially, then the output of graded consensus will be in the form of $(y, 1)$ for all honest players and $z_i = y_i = y$. So king correctness is preserved. □

3.1.4 Putting it all together

If we run King Consensus with $t+1$ different kings, and while using as input each time the input of the previous run, at least one of them is honest and we achieve consensus. Consensus will be preserved, from the correctness property until the end. Additionally if all honest players have the same input at the beginning, they will decide on this value from round 1.

3.2 Byzantine Agreement Against General Adversaries

In this section, we present an efficient protocol for Byzantine Agreement against a General Adversary by Fitzi and Maurer [11]. This setting generalizes the one in the previous section, in the sense that a threshold adversary can be described by a general adversary structure as we explain in the next section.

3.2.1 General Adversary

A general adversary is described by the set of possible corruption sets. More formally:

Definition 3. *An adversary structure $\mathcal{A}$ over the player set $\mathcal{P}$ is a monotone set of subsets of P i.e. $\mathcal{A} \subseteq 2^{\mathcal{P}}$ and $(B \in \mathcal{A} \wedge C \subset B) \Rightarrow C \in \mathcal{A}$.*

Moreover, the elements of $\mathcal{A}$ are called adversary sets. A short way of representing an adversary structure is by its maximal elements. We denote the basis of $\mathcal{A}$ with $\bar{\mathcal{A}} = \{B \in \mathcal{A} : \nexists B' \in \mathcal{A} \text{ s.t. } B \subset B'\}$. We also denote the restriction of an adversary structure $\mathcal{A}$ to the player set $S \subseteq P$ by $\mathcal{A}_S = \{B \cap S | B \in \mathcal{A}\}$.

In an analogous manner to the $n/3$ condition for reliable broadcast against a threshold adversary, in the general adversary case we need to define when k sets of the adversary structure cover the whole player set.

Definition 4. *An adversary structure $\mathcal{A}$ satisfies $Q^k(\mathcal{P}, \mathcal{A})$ if no k sets in $\mathcal{A}$ cover $\mathcal{P}$.*

Observation. *The adversary structure $\mathcal{A} = \{C | C \subseteq \mathcal{P} \wedge |C| <= t\}$ is equivalent to having a t-threshold adversary.*

3.2.2 A simple protocol

A simple but inefficient Broadcast protocol is presented for the case where the adversary structure satisfies $Q^3(\mathcal{P}, \mathcal{A})$, that is, no 3 sets in $\mathcal{A}$ cover the whole player set. This condition is tight for the general adversary case. If we consider the adversary structure of a threshold adversary, the condition is equal of having $t < n/3$, which is also tight. The simple protocol can be divided in two phases, the information gathering (IG) and data conversion (DC) phase. We denote the dealer by the letter d.

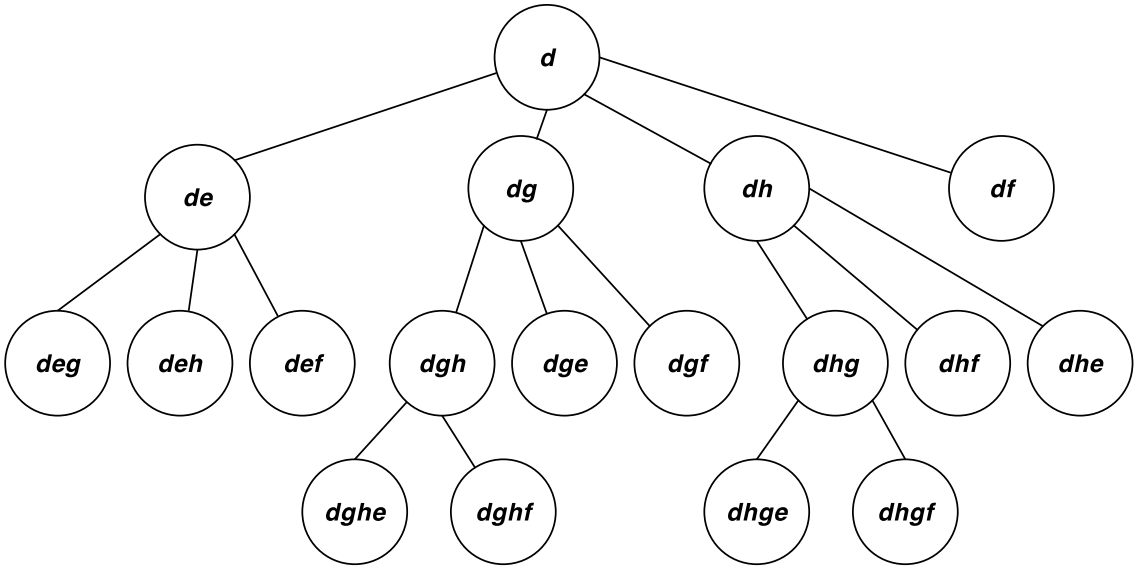

Figure 3.1: An example of an IG-tree.

3.2.3 Information Gathering

Every player u maintains a local information gathering tree that contains all the data he got from the other players. For player v, the root node of the tree corresponds to the dealer d and has the value d sent to v. An internal node is added to the tree for any sequence of nodes that forms a possible corruption set. Finally, the leaves of the tree correspond to minimal sequences of nodes that do not belong to the adversary structure. We denote the local tree of player v by $tree_v$. The value stored in the node defined by the sequence $v_1v_2...v_k$ is denoted by $tree_v(v_1v_2...v_k)$. The sequence $v_1v_2..v_k$ as we said before corresponds to the path from the root to the node. The meaning of the value x of the local tree of some node v, corresponding to the string $v_1v_2...v_k$ is that "v_k said to v, that v_{k-1} said to v_k,...,that v_1 said to v_2 that the broadcast value is x. We give an example of the structure of such a tree in figure 3.1 for $\mathcal{P} = \{d, e, f, g, h\}$ and $\bar{\mathcal{A}} = \{\{d, e\}, \{d, g, h\}, \{e, h\}, \{f, g\}\}$.

Notice that the tree has at most height n, since any sequence has at most n players. We say that the root node is in level 1, its children on level 2, etc. The protocol for the information gathering phase is presented below:

---

function INFORMATION GATHERING($\mathcal{P}$, $\mathcal{A}$, d, x)

1. The dealer d sends its value x to all the players, decides on this value and terminates.
2. On round k (> 1), each player v_i, sends to everyone else the value of all the nodes of level $k-1$ of the tree ($tree_{v_i}(v_1..v_r)$), that have a child corresponding to the sequence $v_1..v_r v_i$.
3. Each player $v \in \mathcal{P}$, stores the value y send by v' about the value of $tree_{v'}(v_1..v_k)$ to the node with label $v_1..v_k v'$. So $tree_v(v_1..v_k v') = y$.

end function

With the information gathering phase of the protocol each player receives the broadcast value from different paths in the communication graph. Obviously, if some path contains corrupted nodes this value may have been altered.

3.2.4 Data Conversion

For the data conversion we use the function $resolve_p$, that computes the output of node p using $tree_p$. This step is entirely computed locally. No interaction is needed between the players during this phase.

With $\perp$ we denote that a decision cannot be taken from the resolve function. By $C(v_1..v_k)$ we denote the set of the children of the node defined by $v_1..v_k$. By $C_a(v_1..v_k)$ we denote the set of the honest children of the node defined by $v_1..v_k$. The way the resolve function works is bottom-up. It starts by resolving the values of the leaves of the tree, and uses these intermediate values to build its way up to the root. Each node decides on the the resolved value of the root.

$$resolve_p(v_1..v_k) = \begin{cases} tree_p(v_1..v_k) & \text{if } v_1..v_k \text{ is a leaf} \\ x & \exists! x \neq \perp : Q^1(\{c \in C(v_1..v_k) | resolve_p(v_1..v_k c) = x\}) \\ 0 & \text{if } k = 1 \text{ and } v_1 = d \\ \perp & \text{otherwise} \end{cases}$$

The second rule corresponds to the notion that for node $v_1..v_k$ the function resolves to x if the set of the children of this node that decide on x is not a possible corruption set. The third rule is needed because players should output some value from output domain and not $\perp$.

Definition 5. *A node $v_1..v_k$ of the tree is common, if every correct player resolves the same value for this node. The subtree rooted on node $v_1..v_k$ has a common frontier if every path from this node to a leaf has at least one common node.*

Lemma 3. *If the adversary structure $\mathcal{A}$ satisfies $Q^k(\mathcal{P}, \mathcal{A})$ then for every internal node $v_1..v_k$, $\mathcal{A}_{C(v_1..v_k)}$ satisfies $Q^{k-1}(C(v_1..v_k), \mathcal{A}_{C(v_1..v_k)})$.*

Proof. By the way we construct our tree, a node corresponds to a possible corruption set. So if $Q^{k-1}(C(v_1..v_k), \mathcal{A}_{C(v_1..v_k)})$ is not satisfied, there exists a covering with $k-1$ sets from $\mathcal{A}$. But if we also add the set corresponding to this node, we get a covering of the whole player set with k sets, which is a contradiction. □

Lemma 4. *All nodes $v_1..v_k r$, with r being a correct player are common with the value $tree_r(v_1..v_k) = x$.*

Proof. Let p be an honest player. Since p receives $tree_r(v_1..v_k)$ from r, who is also honest, then $tree_p(v_1..v_k r) = tree_r(v_1..v_k) = x$. We will prove that $resolve_p(v_1..v_k r) = tree_p(v_1..v_k r)$ by induction. If $v_1..v_k r$ is a leaf, then $resolve_p(v_1..v_k r) = tree_p(v_1..v_k r)$ by definition. Let the lemma hold for all nodes of level $k+1$ and $v_1..v_k r$ be a node of level k. By lemma 4 and since $Q^3(p)$ holds, then $Q^2(C(v_1..v_k r), \mathcal{A}_{C(v_1..v_k r)})$ holds. That means no 2 sets from $\mathcal{A}_{C(v_1..v_k r)}$ cover $C(v_1..v_k r)$. If $C_a(v_1..v_k r) \in \mathcal{A}_{C(v_1..v_k r)}$, then since by definition $C(v_1..v_k r) \setminus C_a(v_1..v_k r)$ can be corrupted, $\{C_a(v_1..v_k r), C(v_1..v_k r) \setminus C_a(v_1..v_k r)\}$ would cover $C(v_1..v_k r)$, which is a contradiction. So $C_a(v_1..v_k r) \notin \mathcal{A}_{C(v_1..v_k r)}$ which implies $Q^1(C_a(v_1..v_k r), \mathcal{A}_{C_a(v_1..v_k r)})$. Thus, since for each node $u \in C_a(v_1..v_k r)$ by the induction hypothesis $resolve_p(v_1..v_k r u) = x$ holds, $resolve_p(v_1..v_k r) = x$ and the proof is complete. □

Lemma 5. *Let $v_1..v_k$ be a node of the tree. If there is a common frontier on the subtree of this node, then this node is common.*

Proof. If $v_1..v_k$ is a leaf, then by definition $v_1..v_k$ has a common frontier when $v_1..v_k$ is common. Let $v_1..v_k$ be an internal node that has a common frontier and is not common. Since $v_1..v_k$ is not common, at least one of its honest child nodes should not be common, or else all honest child nodes will resolve to the same value and from $Q^3(\mathcal{P})$ will not form a possible corruption set, thus $v_1..v_k$ will decide on the common value of its honest children, a contradiction. Recursively, the child of $v_1..v_k$ which is not common, should have a child that is not common. Hence, a path from $v_1..v_k$ to a leaf is formed from nodes that are not common, which contradicts the assumption that $v_1..v_k$ has a common frontier. So $v_1..v_k$ must be common. □

Theorem 6. *For any player set $\mathcal{P}$ and adversary structure $\mathcal{A}$ satisfying $Q^3(\mathcal{P})$ the simple protocol achieves reliable broadcast.*

The proof of this theorem follows from the fact that all nodes in any root-leaf path do not belong in a single adversary set. So in any path at least one node is honest. From lemma 4 the root has a common frontier, because in any path from root to leaf at least one node is common since it is in the form of $v_1..v_k r$, where r is an honest player. But from lemma 5 the root node is common. So all players resolve the root node on the same value and they decide on the same value (agreement). If the dealer is also honest, then again from lemma 4 the root node is common and correctness is preserved.

3.2.5 Towards an efficient protocol

The simple protocol described in the previous section is not efficient. For example, for any adversary structure with exponential size the tree is also of exponential size. We are going to present a protocol that is efficient (fully polynomial with respect to the number of players) assuming only that there exists an algorithm polynomial in n for deciding whether a given subset of the players is an element of $\mathcal{A}$.

The main idea used is that of [1]. We fix some parameter $4 \leq b < n$ and the protocol is run repeatedly on the tree where levels $l > b$ are pruned. In every run we collect only information on existing nodes in the tree. The data conversion phase proceeds as before. What we gain by this process is that at the end of each run we can detect and permanently blacklist corrupt nodes and do not consider them in the subsequent runs.

The information gathering phase in each run is the same as before. Nodes broadcast level by level their information on specific nodes on the tree. The only difference is that the dealer does this only for the first run. On the following runs the dealer node of each new local tree takes the value it resolved in the previous run i.e. $tree_p^m(d) = resolve_p^{m-1}(d)$. After the information gathering phase each player processes the information of his local tree as before by the resolve function.

As we said before, we are going to try to detect corrupt nodes in each run by the information they send. We can do this by examining what other players say about some value they got in a previous run. So if some honest player r send the same value to all players concerning some node $v_1..v_k$, then, honest children of this node should send the same value concerning node $v_1..v_k r$. So for each possible corruption set, honest player r, and using the information on the corrupted players we have already, it must hold that

$$\exists u \exists C \in \mathcal{A} : (L_p \cup \{c \in C(v_1..v_k r) | tree_p(v_1..v_k rc) \neq u\}) \subseteq C$$

where L_p is the list of already detected players. Additionally, from lemma 4 this condition can be applied on the DC phase also. That's because r is honest, so all nodes $v_1..v_k r$ are common with $tree_r(v_1..v_k)$. After the resolve function has been applied, the honest children of $v_1..v_k r$ will compute the same value that r sent them, given that r is honest.

Formally, we have 2 fault detection rules:

$$\text{If } \nexists u : \neg Q^1(\{c \in C(v_1..v_k r) | tree_p(v_1..v_k rc) \neq u\} \cup L_p) \text{ then } r \in L_p$$

Similarly for the DC phase:

$$\text{If } \nexists u : \neg Q^1(\{c \in C(v_1..v_k r)) | resolve_p(v_1..v_k rc) \neq u\} \cup L_p) \text{ then } r \in L_p$$

After a corrupted player has been detected by some honest player, all subsequent messages of the corrupted player are considered to have the value 0. So when all honest players detect him, all nodes that their value is determined by this node will be common.

We can now give the complete protocol:

---

function EFFICIENT PROTOCOL($\mathcal{P}$, $\mathcal{A}$, d, x)
The dealer d sends its value x to all the players, decides on this value and terminates.
  for $i = 1 \rightarrow \lceil \frac{n-3}{b-3} \rceil + 1$ **do**
    Information Gathering phase with fault detection for $b-1$ rounds.
    Data conversion with fault detection on the local tree.
    For every player p, $tree_p(d) = resolve_p(d)$.
  end for
For every player p, decide on $tree_p(d)$ and halt.
end function

It can be shown that if $Q^3(\mathcal{P}, \mathcal{A})$ holds then this protocol achieves Byzantine Agreement. If additionally there exists a polynomial algorithm on n for deciding whether a given player set is in $\mathcal{A}$, then this protocol is fully polynomial.

3.2.6 Analysis of the efficient protocol

Notice that if agreement has been achieved after the data conversion phase of some run, then it will persist on the rest of the runs. This follows immediately from lemma 4 since the value of the root node of $tree_p^{m-1}$ after some data conversion phase becomes the value of the root node of $tree_p^m$ on the next information gathering phase. Without proof we state the following lemma: [11]

Lemma 7. *Let $v_1..v_k r$ be an internal node of the IG-tree but not the parent of a leaf. If all players corresponding to the path of $v_1..v_k r$ are faulty and there is a correct player p who does not detect r to be faulty by either of the fault detection rules, then $v_1..v_k r$ is common.*

Let's see what happens at the end of the first run of the loop for some value of b. If the root node is common, then as we argued before, agreement will persist and byzantine agreement will be achieved. So suppose that the root node is not common. Then by lemma 5 the root node does not have a common frontier. That means that there exists a path from the root to some leaf that does not contain any common node. From lemma 4 this path consists only of faulty nodes, because if some node was honest, then a node of the path would be common. But from lemma 7 all faulty nodes on the path, except the leaf and its parent, are detected by all honest players. So $b-2$ faulty nodes are detected.

The situation is the same on the subsequent runs. Either agreement is achieved or $b-3$ faulty players are detected. Because every player detected is masked, and so these nodes are common on the runs following, a new path of faulty nodes is detected every time. But as we said the root is not masked, and that's why only $b-3$ nodes are detected.

Theorem 8. *For any player set P and adversary structure $\mathcal{A}$, if $Q^3(\mathcal{P}, \mathcal{A})$ holds, then the efficient protocol achieves Byzantine Agreement. The message complexity of the algorithm is polynomial and the number of rounds less than $3n$. If additionally there*

exists a polynomial algorithm on n for deciding whether a given player set is in $\mathcal{A}$, then this protocol is fully polynomial.

Proof. We will first show why this protocol achieves Byzantine Agreement. Let's suppose it does not. Then in every run all honest nodes detect at least $b-3$ (or $b-2$ if they are on the first run) faulty players. Otherwise they would have achieved agreement in some run and it would have persisted until the final run. So at the end of the protocol the honest players have detected k corrupt players:

$$\begin{aligned} k &= b-2+\lceil\frac{n-3}{b-3}\rceil(b-3) \\ &\geq b-2+n-3 \\ &\geq n-1 \end{aligned}$$

Since at least $n-1$ players are detected, at most one is honest. But since only one player is honest, all nodes are common by definition, and so is the root, which is a contradiction. Thus, Byzantine Agreement has been achieved in an earlier run, and persists until the end. The message complexity of the protocol is polynomial if we for example set $b=4$. The round complexity of the protocol is $\#r = b+\lceil\frac{n-3}{b-3}\rceil(b-1) < 3n$. □

3.3 Different Perspectives

In this section we briefly look over some different ideas on consensus. Firstly, we present the idea of unbeatable consensus in a setting with crash failures. On the second part, we study a recent development on randomized consensus.

3.3.1 Unbeatable Consensus

Unbeatable Consensus is studied in [4] and the basic idea is that we try to find a protocol so that each player stops as early as possible. We proceed with some definitions that we will need to state the results of this work.

A run r describes the possibly infinite behavior of a system. A deterministic protocol Π and a deterministic adversary uniquely determine a run, denoted by $r = \Pi[a]$. Our attention is restricted on those here.

As usual we consider a synchronous communication model but with crash failures, e.g. faulty players follow the protocol until some round m that they crash. Messages sent at round m may or may not succeed in arriving. All this information define a failure pattern.

A failure pattern and the input define the context γ. Given context γ, a protocol Q dominates a protocol P if for each adversary a, every process in $Q[a]$ decides earlier or on the same round as the respective process in $P[a]$. Moreover, a protocol Q strictly dominates a protocol P if P is dominated by Q and additionally on some run a process in Q decides strictly before a process in P.

If some protocol solves a particular task, say Consensus, and it also dominates every other protocol that solves this task, it is called an all case optimal protocol. Unfortunately for this setting it was proved [20] that no all case optimal protocol exists. So a different notion of optimality was defined and studied, unbeatibility. A protocol Π is unbeatable if it solves some task and no other protocol that solves this task strictly dominates Π on some context.

A full information protocol is roughly a protocol that in every round each node shares all the messages it has received with every other node in the graph. What distinguishes different full information protocols is the decision function of each node. A process j at time l is seen from process i at time m if there exists a message chain from j at time l to i at time m. A process j at time l is hidden from process i at time m if i (a)does not know that j has failed before time l and (b) j at time l is not seen from i at time m. A hidden path with respect to process i and time m on some run r exists if there exists a sequence of processes $j_0, .., j_m$ s.t. j_l at time l is hidden from i at time m. Notice that no hidden path exists for (i, m) if $\exists l < m$ s.t. every process (j, l) either is seen by i at time m or i at time m knows that j has crashed before time l.

To argue about these notions, a knowledge based analysis is used. Informally, node i knows fact A, denoted by $K_i(A)$, on a well defined system R on time m in run r iff for all possible runs in this system that are indistinguishable to i, A holds. A fact A is said to be a precondition of some action σ on a specific system and point [1] (r, m) if A holds whenever this action is performed. It holds that:

Theorem 9. *If A is a precondition for i performing σ in the set of runs of some deterministic protocol Π then $K_i(A)$ is also a precondition for i to perform σ.*

The intuition behind this very simple but strong principle is that if A needs to hold for i to perform some action, then i will perform this action only if he actually knows A.

For the rest of this section we consider a threshold adversarial model, parametrized by some value t. A fact we will use is $\exists u$ which means that some player has input u. From the validity property of consensus and theorem 9 we easily get that:

Lemma 10. *$K_i(\exists u)$ is a precondition for i to decide on any value u in any protocol that achieves consensus.*

If we use the rule "if $K_i(\exists 0)$ then i decides 0" in the unbeatable protocol we are trying to construct, then by the agreement property we have the following: "no honest process ever decides on 0" is a precondition to decide on 1. This is equivalent to knowing that no active process currently knows $\exists 0$. Furthermore, it is equivalent to $\neg K_i(\exists 0)$ and no hidden path exists for process i at time m. So we can now construct an unbeatable protocol:

[1] A point is a specific time m on a run r.

```
function UP(for process i at time m)
    if K_i(0) then
        decide 0
    else if no hidden path w.r.t process i at time m exists then
        decide 1
    end if
end function
```

Theorem 11. *Protocol UP is unbeatable.*

The proof of this theorem follows from the two observations we made concerning preconditions needed in order to decide either on 0 or 1. Any protocol that beats UP should have also its first decision rule. The second decision rule ensures that decision on 1 is taken as soon as possible, without process i risking being wrong.

Concluding, one can observe that there exists a symmetric protocol for the value 1 that is also unbeatable. These two protocols are biased towards value 0 and 1 respectively. In the same work, a majority based unbeatable protocol is also presented. Extending these results against a byzantine adversary is an interesting open question.

3.3.2 Byzantine Agreement in Polynomial Expected Time

This section presents a recent work on Byzantine Agreement using a randomized algorithm [16]. Randomized algorithms provide a more realistic option in the sense that they do not achieve agreement in some negligible number of scenarios, but they are more efficient compared to deterministic algorithms. The setting of this work has the following differences from what we have seen before:

- The adversary is adaptive, he can determine which process he will corrupt as the algorithm proceeds. He also knows everything except future coin flips of the private coins other players have.
- The communication is asynchronous.
- The adversary is byzantine and corrupts a fraction of the players.

Theorem 12. *There exists an algorithm [16] that solves Byzantine Agreement in expected time $O(n^{2.5})$ and expected polynomial number of messages, where n is the number of the players, when the adversary controls at most up to 1/500 players.*

The algorithm builds up from concepts coming from past works. First, the idea of a global common coin simulation. By flipping their private coins repeatedly and broadcasting the result of this flip, players try to efficiently simulate a global coin flip. However, the adversary can disrupt this process by sending arbitrary values in order to steer the outcome the way he wants.

As we have seen before in section 3.2, to efficiently overcome the misbehavior of the adversary, detection mechanisms can be used. Misbehavior of players controlled by the adversary is detected and these players are blacklisted. Because we are talking here about randomized behavior, an honest player may also be punished. It is ensured that the number of honest players blacklisted is sufficiently small and does not ruin the performance of the protocol.

So by combining these two ideas, if a player's broadcast coin flips do not "look" from a statistical point of view like he is actually flipping a coin for the global coin protocol, then with high probability he is malicious and he is blacklisted. This way the global common coin is built.

The global coin is used on a protocol proposed by Ben-Or [2]. In this protocol the players calculate the majority of votes of the other players. If a large number of the players agree on the same value, then the adversary cannot change the value of the majority and players decide on this value. Alternatively some of the players flip their coins in order to simulate a global coin flip.

The proposed protocol repeats this process enough times, while honest players blacklist other suspicious players to eventually reach agreement. The algorithm produced is Las Vegas, it always achieves byzantine agreement.

At first what is evident from this work, and the work on section 3.2, is that if we want to be efficient, we should carefully analyze and detect malicious behavior, and even force it. In older works, the treatment of adversarial behavior was "passive", in the sense that protocols were not build with the idea of actively exposing misbehavior and blacklisting misbehaved players for the rest of the protocol. A large chunk of information was overlooked or only used at a much later time than it was obtained. Secondly, as in almost every field of modern computing, randomization is a very efficient tool against worst case behavior and should not be overlooked.

Chapter 4

Reliable broadcast and topology knowledge

Most of the results described on this chapter were published on the 28th International Symposium of Distributed Computing (DISC 2014) [22].

4.1 Introduction

The case of Reliable Broadcast under a threshold adversary in incomplete networks has been studied to a much lesser extent, in a study initiated in [6, 7, 18], mostly through protocols for Secure Message Transmission which, combined with a Broadcast protocol for complete networks, yield Broadcast protocols for incomplete networks. Naturally, connectivity constraints are required to hold in addition to the $n/3$ bound. Namely, at most $t < c/2$ corruptions can be tolerated, where c is network connectivity, and this bound is tight[6].

In the case of an honest dealer, particularly meaningful in wireless networks, the impossibility threshold of $n/3$ does not hold; for example, in complete networks with an honest dealer the problem becomes trivial regardless of the number of corrupted players. 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 (or parameters) depending on the structure of the graph, 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. We will first focus on local restrictions, the importance of which comes, among others, from the fact that they may be used to derive criteria which can be employed in *ad hoc* networks. Such a paradigm is the *t-locally bounded adversary model*, introduced in [17], in which at most a certain number t of corruptions are allowed in the neighborhood of every node.

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 will introduce a generalization of the t-locally bounded model.

4.1.1 Related Work

Considering t-locally bounded adversaries, Koo [17] proposed a simple, yet powerful protocol, namely the *Certified Propagation Algorithm* (CPA) (a name coined by Pelc and Peleg in [24]), and applied it to networks of specific topology. CPA is based on the idea that a set of $t + 1$ neighbors of a node always contain an honest one. Pelc and Peleg [24] considered the t-locally bounded model in generic graphs and gave a sufficient topological condition for CPA to achieve Broadcast. 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 [15] proposed an efficiently computable graph parameter which implies a more tight, but not exact, characterization of the class of graphs on which CPA achieves Broadcast. It had remained open until very recently 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. Such a parameter is implicit in the work of Tseng *et al.* [26], who gave a necessary and sufficient condition for CPA Broadcast. Finally, in [19] such a graph parameter was presented explicitly, together with an efficient 2-approximation algorithm for computing its value.

A more general approach regarding the adversary structure was initiated by Hirt and Maurer in [14] where they studied the security of multiparty computation protocols with respect to an *adversary structure*, i.e. a family of sets of players, such that the adversary may entirely corrupt any set in the family. This line of work has yielded results on Broadcast against a general adversary in complete networks [11] but, to the best of our knowledge, the case of Broadcast against general adversaries in incomplete networks has not been studied as such.[1] A study on the related problem of Iterative Approximate Byzantine Consensus against general adversaries can be seen in [25] where a similar model for the *ad hoc* case is considered.

4.1.2 Our Results

In this work we study the tradeoff between the level of topology knowledge and the solvability of the problem, under various adversary models.

[1] Some related results are implicit in [18], but in the problem studied there, namely Secure Message Transmission, additional secrecy requirements are set which are out of the scope of our study.

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. We address the issue of locally resilient 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 [24] to the notion of *partial local pair cut*. Note that although equivalent conditions exist [26, 19], the simplicity of the new condition allows to settle the open question of CPA Uniqueness [24] in the affirmative: we show that if any *safe* (non-faulty) algorithm achieves Broadcast in an *ad hoc* network then so does CPA. We next prove 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 P = NP.

We next 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 [24] 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, we prove that it is NP-hard to compute an essential decision rule of PPA, rendering the algorithm inefficient. However, we are able to provide an indication that probably no efficient protocol of optimal resilience exists, by showing that efficient algorithms which behave exactly as PPA w.r.t. decision do not exist if P $\neq$ NP.

We then take one step further, 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, 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 [18] 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 [24] 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.

For clarity we have chosen to present our results for the t-local model first (Sections 3,4,5), for which proofs and protocols are somewhat simpler and more intuitive, and then for the more involved general adversary model (Section 6).

4.2 Problem and Model Definition

In this paper we address the problem of *Reliable Broadcast with an honest dealer* in generic (incomplete) networks. As we will see in Section 4.6.1, this case essentially captures the difficulty of the general problem, where even the dealer may be corrupted. 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 deduce that x_D is the value originally sent by the dealer and output it as its own decision.

The problem is trivial in complete networks; we will consider the case of incomplete networks here. For brevity we will refer to the problem as the Broadcast problem.

We will now formally define the adversary model by generalizing the notions originally developed in [17, 24]. We will also define basic notions and terminology that we will use throughout the paper. 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 [17] 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 $G = (V, E)$. One player $D \in V$ is the dealer (sender). 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 plays an important role in our study since it coincides with the family of admissible corruption sets.

Definition 6 (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.

In our study we will often make use of node-cuts which separate some players 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 node-cut. The notion of *t-local pair cut* was

introduced in [24] and is crucial in defining the bounds for which correct dissemination of information in a network is possible.

Definition 7 (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 8 (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)) [24] 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, for reasoning about our results it suffices to consider the following decision problem:

Definition 9 (pLPC)**.** *Given a graph G, a dealer D and a corruption function t determine whether there exists a t-plp cut in G.*

Definition 10 (t-locally resilient algorithm)**.** *An algorithm which achieves Broadcast for any t-local corruption set in graph G with dealer D is called* t-locally resilient *for (G, D).*

Definition 11 (safe / t-locally safe algorithm)**.** *A Broadcast algorithm which never causes an honest node to decide on an incorrect value, is called* safe*.*
A Broadcast algorithm which never causes an honest node to decide on an incorrect value under any t-local corruption set, is called t-locally safe*.*

4.3 Ad Hoc Networks

4.3.1 Certified Propagation Algorithm (CPA)

The Certified Propagation algorithm [17] 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, which does not require knowledge of the network topology. We use a modification of the original CPA that can be employed under the non-uniform t-locally bounded adversary model. 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.
Fact. CPA is a t-locally safe Broadcast algorithm. By induction an honest node decides only when at least one honest neighbor has decided on the same value.

Protocol 1: ***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.

4.3.2 CPA Uniqueness in *Ad Hoc* Networks

Based on the above definitions we can now prove the *CPA uniqueness conjecture* for *ad hoc* networks, which was posed as an open problem in [24]. 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, as described e.g. in [24]. In particular we assume that nodes know only their own labels, the labels of their neighbors and the label of the dealer. We call a distributed Broadcast algorithm that operates under these assumptions an *ad hoc Broadcast algorithm.*

Theorem 13 (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. Let T be the corruption set; clearly $T \cup N(D)$ is a cut on G as defined before (i.e. not including node D). Since T is t-local and $T \cup N(D)$ is not a t-plp cut there must exist $u_1 \in V \setminus (T \cup \mathcal{N}(D) \cup D)$ s.t. $|N(u_1) \cap (N(D) \setminus T)| \geq t(u_1)+1$. Since u_1 is honest it will decide on the 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 $C_k = (N(D) \setminus T) \cup \{u_1, u_2, ..., u_{k-1}\}$ be the set of the honest nodes that have decided until a certain round of the protocol. 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. $|C_k \cap N(u_k)| \geq t(u_k)+1$ and u_k will decide on x_D. Eventually all honest players will decide on x_D. Thus CPA is t-locally resilient in G. □

Theorem 14 (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).*

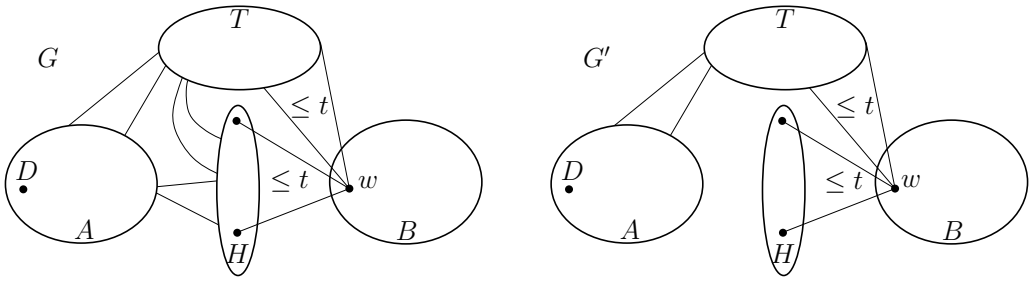

Figure 4.1: Graphs G and G'

Proof. Assume that there exists a t-plp cut $C = T \cup H$ in graph G with dealer D with T being the t-local set of the partition and H the t-local w.r.t. to B set (Figure 4.1). 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 $\mathcal{A}$:

Execution σ is on the graph G with dealer D, with dealer's value $x_D = 0$, and corruption set T; in each round, all players in T perform the actions that 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, with dealer's value $x_D = 1$, and corruption set H; in each round, all players in H perform the actions that perform 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. □

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 [24] 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 can easily be shown. This shows that there exists non-safe algorithms of higher resilience than CPA. The next corollary is immediate from Theorems 13,14.

Corollary 15 (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).*

4.3.3 Hardness of pLPC

Ichimura and Shigeno in [15] prove that the *set splitting* problem, known as NP-hard [13], 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 [15] and show that the pLPC problem is NP-hard.

Theorem 16. pLPC *is* NP*-hard.*

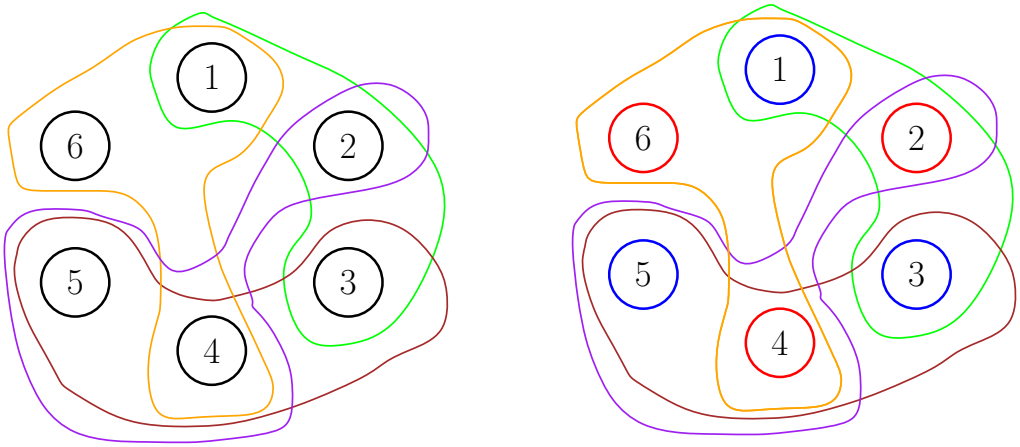

Figure 4.2: An instance and and the solution of a set splitting problem with $X = \{1, 2, 3, 4, 5, 6\}$ and $A = \{\{1, 2, 3\}, \{3, 4, 5\}, \{1, 4, 6\}, \{2, 4, 5\}\}$. The solution is depicted by the the two sets $X_1 = \{1, 3, 5\}$ and $X_2 = \{2, 4, 6\}$ in blue and red respectively. Notice that all sets in A have at least one node of both colors.

Proof. We show that the *set splitting* problem known as NP-hard [13] can be reduced to the *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 example of this problem is shown in figure 4.2. 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 vertex set $S+$ and a copy of it are denoted by K_{S+} and K'_{S+}, respectively. We construct a graph G_{SSP} (figure 4.3) 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 node in $V(K'_{S+})$ which is a copy of $s \in S+$. If a subgraph of G_{SSP} deleting $C(\subseteq V(G_{SSP}))$ has at least two connected components and $X \setminus C \neq \emptyset$, C contains $\mathcal{N}(v) \cap V(K_{S+})$ or $\mathcal{N}(v) \cap V(K'_{S+})$ for some $v \in X$. Since each $v \in X$ has at least six neighbor in both $V(K_{S+})$ and $V(K'_{S+})$, C is a t-local pair side cut with $t >= 3$. We next consider the case of $C = X$. We can partition X into two 2-local sets in G_{SSP}, if and only if the set splitting problem has a desired partition $X1$ and $X2$. Therefore, we have $pLPC(G_{SSP}, 2) = true$, if and only if the set splitting problem has a desired partition. Now we can easily show that NP-hardness for $pLPC(G,t)$ without a dealer implies NP-hardness for the case with a dealer. If $pLPC(G,t,D)$ could be solved with a polynomial-time algorithm then solving $pLPC(G,t,v)$ for every node in V would suffice to build a polynomial algorithm for $pLPC(G,t)$ which is a contradiction. Therefore to compute $pLPC(G,t,D)$ is NP-hard. □

Therefore, 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.

4.4 Known topology Networks

4.4.1 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: 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). With $tail(p)$ we will denote the last node of path p. The description of PPA follows.

Protocol 2: ***Path Propagation Algorithm (PPA)***

Input (for each node v): graph G, dealer D, $t(v)$ = max #corruptions in $N(v)$.
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 $v \neq D$: upon reception of (x,p) from node u `do:`

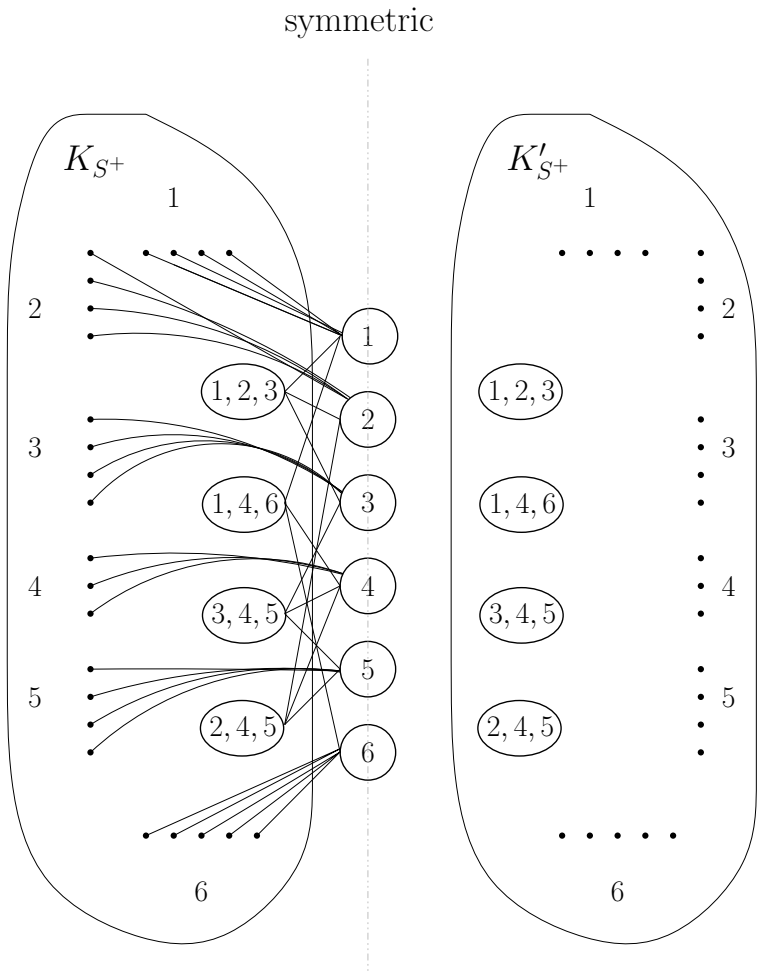

Figure 4.3: The graph G_{SSP} for the set splitting problem in figure 4.2.

`if` $(v \in p) \vee (tail(p) \neq u)$ `then discard` the message
`else send` $(x, p||v)$ [2] to all neighbors.

`if decision`$(v) \neq \perp$ `then send` message (`decision`$(v), v)$ to all neighbors.

function `decision`(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 $(x, p_1), \ldots, (x, p_n) \wedge \nexists$ t-local cover of $\{p_1, \ldots, p_n\}$
`then return` x `else return` $\perp$.

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

The correctness of the honest path propagation rule is trivial: if a path is entirely corruption free, then value x, which is relayed through that path, is correct. Checking whether $tail(p) \neq u$ we ensure that at least one corrupted node will be included in a faulty path. 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.

4.4.2 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 (proof omitted).

Theorem 17 (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 due to the *dealer propagation rule*, since the dealer is honest. We next show the rest of the players will decide due to the *honest path propagation rule*.

Let v be any player in $V \setminus \mathcal{N}(D)$ and assume that no t-local pair cut exist in (G, D) exist. Let T be a t-local set and consider the execution σ_T of PPA where T is the corruption set. Let $\mathcal{P}$ be the set of all paths connecting D with v and are composed entirely by nodes in $V \setminus T$ (honest nodes). Observe that $\mathcal{P} \neq \emptyset$, otherwise T is a cut separating D from v and T is trivially a t-local pair cut, a contradiction. Since paths in $\mathcal{P}$ are entirely composed by honest nodes it is easy to see that v will receive the correct value through all paths in $\mathcal{P}$.

We next prove that under any t-local corruption set T' at least one path in $\mathcal{P}$ is completely corruption free.

Assume that $\exists T'$: t-local cover of $\mathcal{P}$. 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, under any t-local corruption set T' at least one path in $\mathcal{P}$ is entirely corruption free.

Consequently, in execution σ_T, node v will receive the correct value through every path in $\mathcal{P}$ along with the corresponding propagation trail and will decide on the correct value due to the honest path propagation rule, because $\mathcal{P}$ is not covered by any t-local set. Moreover honest nodes will not decide on the wrong value due to this rule, because the set of paths transmitting this value will always have a t-local cover (the corruption set T). □

Using the same arguments as in the proof of the necessity of condition $t < LPC(G, D)$ [24] 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.

Theorem 18 (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).*

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.

4.4.3 On the hardness of Broadcast in known networks

In order to run PPA we have to be able to deduce whether a corruption-free path exists among a set of paths broadcasting the same value. Formally, given a graph $G(V, E)$, a set of paths $\mathcal{P}$ and a node u (the one that executes `decision`(u)) we need to determine whether there exists a t-local cover T of $\mathcal{P}$. We call this problem the Local Path Cover Problem, $LPCP(G, D, u, t, \mathcal{P})$ and show that is NP-hard (proof omitted).

Theorem 19. *It is* NP*-hard to compute $LPCP(G, D, u, t, \mathcal{P})$.*

Proof. We will describe a reduction from $3SAT$ to $LPCP(G, D, u, t, \mathcal{P})$. For every variable x_i we construct a gadget G_{x_i} shown on the left of Figure 4.4. We will make use of a parameter μ that will serve as a constant corruption function (that is, our hardness result holds even for the uniform model). We will use several copies of the complete graphs $K_{\mu+1}$ and $K_{2\mu}$. Node D is connected to every vertex of a $K_{\mu+1}$ copy. Every vertex of that $K_{\mu+1}$ copy is connected with the 'upper' μ vertices of a $K_{2\mu}$ copy; let us call this 'upper' node set X_i. Symmetrically for the lower part, node u is connected to every vertex of another $K_{\mu+1}$ copy and every vertex of that $K_{\mu+1}$ copy is connected to the 'lower' μ vertices of $K_{2\mu}$, let us call this set X_i'. Now assuming that $\mathcal{P}$ contains those paths in G_{x_i} that are of length 5 and connect D to u (and no other path in G_{x_i}) it is easy to show that :

Lemma 20. *If $LPCP(G, D, u, \mu, \mathcal{P}) = 1$ with μ-local cover T, then either $X_i \subseteq T$ or $X_i' \subseteq T$.*

$T \cap G_{x_i}$ is a cut of G_{x_i}. Since the only μ-local cuts in G_{x_i} are X_i and X_i', the claim is immediate.

Now for every clause $c_i = c_{i_1} \vee c_{i_2} \vee c_{i_3}$ in C we construct the gadget shown on the right of Figure 4.4. Node D is connected to every vertex of $K_{\mu+1}$. Every vertex of $K_{\mu+1}$ is connected to the first literal of the clause, say l_{i_1}. Literal l_{i_1} is connected to l_{i_2}, and l_{i_2} to l_{i_3}. And symmetrically, node u is connected to every vertex of another copy of $K_{\mu+1}$ and every vertex of $K_{\mu+1}$ is connected to l_{i_3}. Let us call this subgraph of G, G_{c_i}. Assuming that all paths from D to u of length 6 that go through G_{c_i} are contained in $\mathcal{P}$ we show that:

Lemma 21. *if $LPCP(G, D, u, \mu, \mathcal{P}) = 1$ with μ-local cover T, then $l_{i_1} \in T$ or $l_{i_2} \in T$ or $l_{i_3} \in T$.*

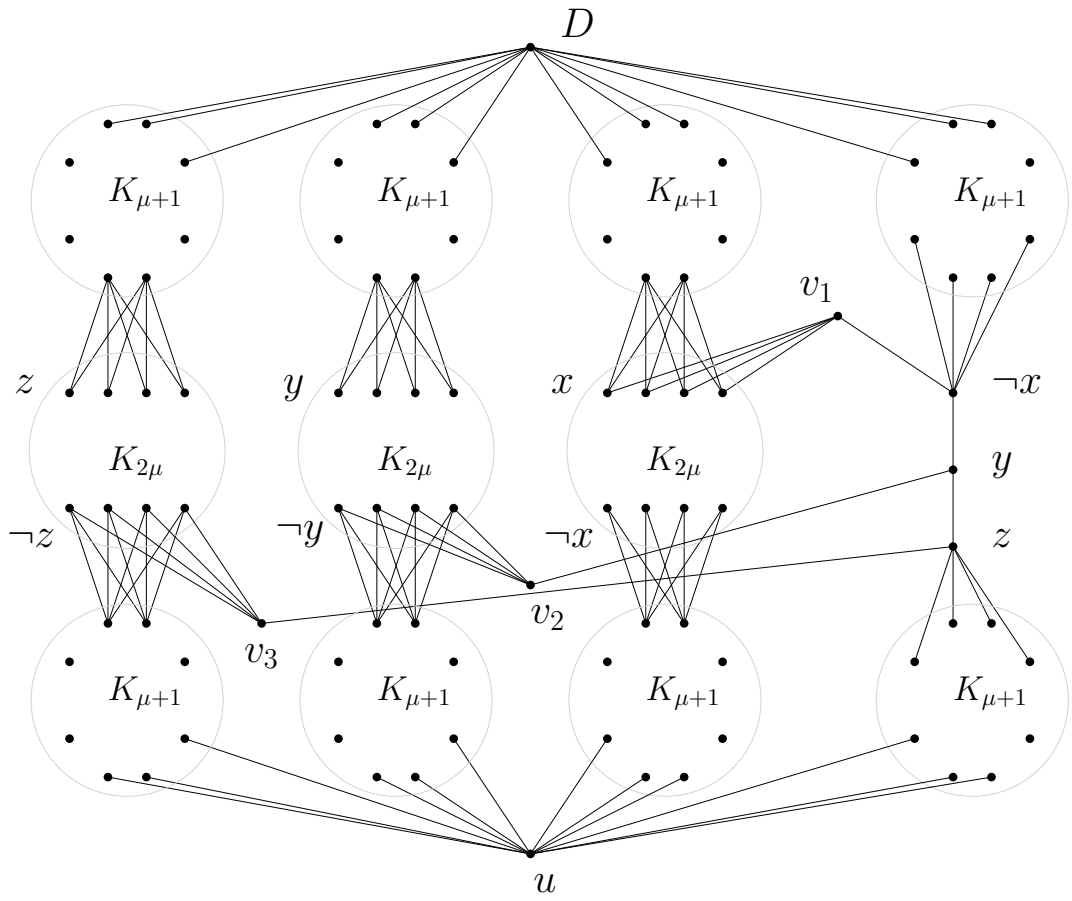

Figure 4.4: An instance of the reduction graph G for variables $\{x_1, x_2, x_3\}$ and clause $c_1 = \{x_1 \vee x_2 \vee \neg x_3\}$.

The proof is by contradiction: if no l_{i_j} node belongs to T, then it must be $K_{\mu+1} \subseteq T$, contradicting the t-locality of T.

The last thing we need to establish is that if $X_i \subseteq T$ (respectively $X'_i \subseteq T$), no $\neg x_i$ (resp. x_i) literal of G_{c_j} is in T. We achieve this by adding a node v_{ij} connecting X_i (resp. X'_i) to $\neg x_i$ (resp. x_i) for each appearance of these literals in some G_{c_j}. The following holds because If both X_i and $\neg x_i$ are in T, then T is not μ-local since $|N(v_{ij}) \cap T| = \mu+1$.

Lemma 22. *If $LPCP(G, D, u, \mu, \mathcal{P}) = 1$ with μ-local cover T, then $X_i \subseteq T$ (resp. $X'_i \subseteq T$) $\Rightarrow \neg x_i \notin T$ (resp. $x_i \notin T$).*

So for graph G that is constructed as described above and for path set $\mathcal{P}$ consisting of the paths used for proving Lemmata 20 and 21 we have that $LPCP(G, D, u, \mu, \mathcal{P}) = 1$ iff there exists a truth assignment A which makes every clause in C true. The '$\Rightarrow$' direction follows from the lemmata proved above. The truth assignment A is constructed as follows: if $X_i \subseteq T$ (resp. $X'_i \subseteq T$) then $\neg x_i$ (resp. x_i) is true in A. The '$\Leftarrow$' direction comes naturally by setting T contain X_i if x_i is true by A, otherwise T contains X'_i; T also contains all literals in G_{c_j} that are set true by A. Then T is a μ-local cover of $\mathcal{P}$ and $LPCP(G, D, u, \mu, \mathcal{P}) = 1$. □

The above theorem implies that PPA may not be practical in some cases, since its decision rule cannot be always checked efficiently. It remains to show whether any other algorithm which has the same resilience as PPA can be efficient. The following theorem provides an indication that the answer is negative, by showing that algorithms which behave exactly as PPA w.r.t. decision are unlikely to be efficient.

Theorem 23. *Assuming* P $\neq$ NP*, no safe fully polynomial protocol* Π *can satisfy the following: for any graph G, dealer D, corruption function t, and admissible corruption set C executing protocol Π_C, a node u decides through PPA on a value x iff u will decide on x by running* Π *on* (G, D, t, C, Π_C).

Proof. We will show that if such Π existed then it would be a polynomial time solver for the 3-SAT problem. Let us consider what happens when Π is run on the graph G that we used in the proof of Theorem 19, with dealer D and the corrupted nodes being the ones that connect the "clause" gadgets with the "variable" gadgets (e.g. $C = \{v_1, v_2, v_3\}$ in Figure 4.4). The adversary protocol Π_C is: the corrupted nodes don't send or relay any messages.

The 3-SAT instance used to make G has a solution iff $LPCP(G, D, u, t, \mathcal{P}) = 1$, i.e. a μ-local cover C_1 on $\mathcal{P}$ exists, where $\mathcal{P}$ is the set of paths we used in the proof of Theorem 19. It can be seen from the decision rule of PPA that, while running PPA on G, u will not decide on any value iff a μ-local cover C_1 on $\mathcal{P}$ exists. Moreover a node u does not decide through on a value x iff u does not decide on x by running Π on (G, D, t, C, Π_C).

So u decides on x_D while running Π on G, with dealer D and corruption set C which runs the Π_C protocol iff 3-SAT does not have a solution. Apparently if Π existed then 3-SAT would have a polynomial time solver. □

4.5 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 devise a new, generalized version of PPA that can run with partial knowledge of the topology of the network. More specifically we assume that each player v only has knowledge of the topology of a certain connected subgraph G_v of G which includes v. Namely if we consider the family $\mathcal{G}$ of connected subgraphs of G we use the *topology view function* $\gamma : V \to \mathcal{G}$, where $\gamma(v)$ represents the subgraph over which player v has knowledge of the topology. We also define the *joint view* of a set S as the subgraph $\gamma(S)$ of G with node-set $V(\gamma(S)) = \bigcup_{u \in S} V(\gamma(u))$ and edge-set $E(\gamma(S)) = \bigcup_{u \in S} E(\gamma(u))$. 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).

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 path propagation rule.

Generalized path propagation rule: Player v receives the same value x from a set $\mathcal{P}$ of paths that are completely inside $\gamma(v)$ 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 12 (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 is t-local in the graph $\gamma(B)$.*

Definition 13 (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 $\forall u \in B$, $C_2 \cap N(u)$ 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 and are omitted.

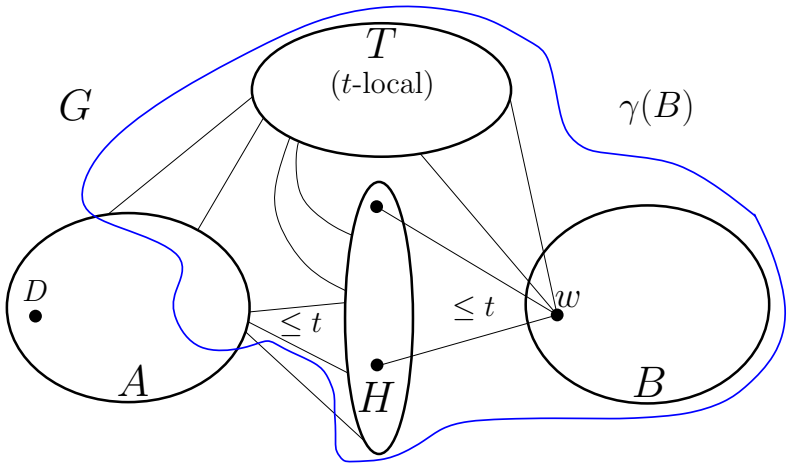

Figure 4.5: A graph where originally H is not t-local, but it seems t-local in $\gamma(B)$.

Theorem 24 (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 $GPPA(G, D, t, \gamma)$ is (γ, t)-locally resilient for G, D.*

Proof. Suppose no (γ, t)-plp2 cut exists. $T \cup N(D)$ is a cut on G non including node D. From the definition of (γ, t)-plp2 cut we have that there exists $u_1 \in V \setminus (T \cup \mathcal{N}(D) \cup D)$ s.t. $N(D) \cap N(u)$ is not t-local on $\gamma(u_1)$. But since all the honest nodes in $N(D) \cap N(u)$ have decided, u will receive the value x_D from paths starting from these nodes of length 1. Finding a t-local corruption set covering these paths is impossible since it would have to include all these nodes, and from 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 $C_k = (N(D) \setminus T) \cup \{u_1, u_2, ..., u_{k-1}\}$ be the set of the nodes that have decided until a certain round of the protocol. 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. $C_k \cap N(u_k)$ 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. □

Theorem 25 (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 4.1). $\gamma(B)$ 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 14 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 4.6.

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 would probably include exchange of topological knowledge among players.

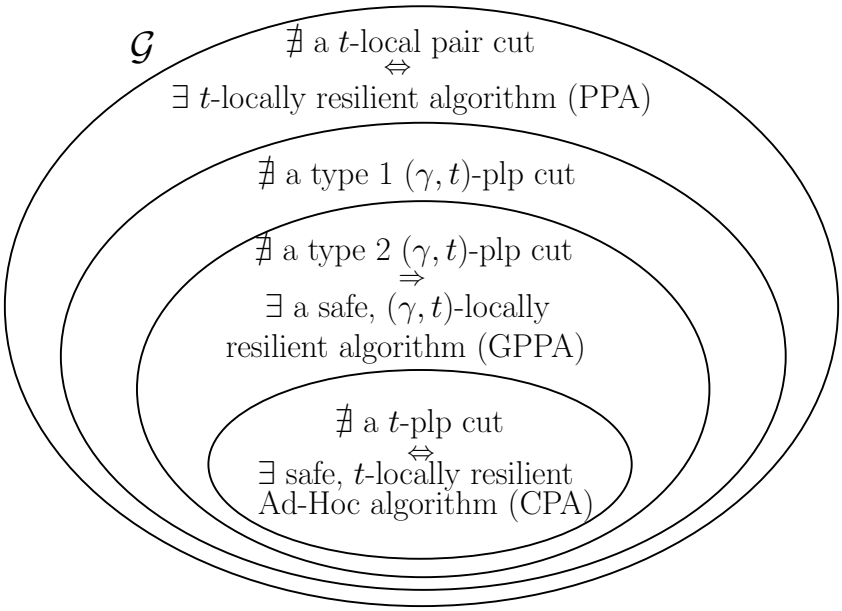

Figure 4.6: Overview of conditions concerning the existence of t-locally resilient algorithms with respect to the level of topology knowledge. Note that $\mathcal{G}$ refers to the family of pairs (G, D).

4.6 General Adversary

Hirt and Maurer in [14] 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 Z are in $\mathcal{Z}$ if $Z \in \mathcal{Z}$. Let us now redefine some notions that we have introduced in this paper 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*. We next generalize the notion of a t-local pair cut.

Definition 14 ($\mathcal{Z}$-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.*

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 2).

$\mathcal{Z}$-PPA Honest Path Propagation Rule: player v receives 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 can be easily shown using essentially the same proofs as for Theorems 17, and 18 and replacing the notion of t-local pair cut with that of $\mathcal{Z}$-pair cut.

Theorem 26 (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 27 (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).*

***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* $\mathcal{Z}_v = \{A \cap \mathcal{N}(v) : A \in \mathcal{Z}\}$.

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 the propagation rule of CPAin the following way.

$\mathcal{Z}$-CPA 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 $N \subseteq \mathcal{N}(v)$ s.t. $N \notin \mathcal{Z}_v$, it decides on value x.

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 15 ($\mathcal{Z}$-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 (proofs omitted).

Theorem 28 (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 $\forall u \in B : N(u) \cap A \in \mathcal{Z}_u$, 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 29 (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 $\mathcal{Z}' = \{\bigcup_{u \in B} Z \cap N(u) : Z \in \mathcal{Z}\} \cup \{C_2\}$.

For every node u in B we have:

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

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

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 14, 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. Regarding the computational complexity of $\mathcal{Z}$-CPA one can observe that it is polynomial if and only if for every player v there exists a polynomial (w.r.t. the size of G) algorithm $\mathcal{B}$ which given a set $S \subseteq \mathcal{N}(v)$ decides whether $S \in \mathcal{Z}_v$. Since $\mathcal{Z}$-CPA is clearly polynomial in round complexity and communication complexity, if such an algorithm $\mathcal{B}$ exists, $\mathcal{Z}$-CPA is fully polynomial.

4.6.1 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 [14] 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 section 3.2) 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. Analogously, the same result holds in networks of known topology, if we replace Condition 2 with the corresponding $\mathcal{Z}$-pair cut condition. Naturally, the above observations hold also in the special case of a locally bounded adversary.

4.7 Partial knowledge against a General Adversary

In this setting each player v only has knowledge of the topology of a certain connected subgraph G_v of G which includes v. Namely if we consider the family $\mathcal{G}$ of connected subgraphs of G we use the *view function* $\gamma : V \to \mathcal{G}$, 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 $S \subseteq G$. The output will correspond to the joint view of nodes in S. In addition each player knows the possible corruption sets in his view $\mathcal{Z}_u = \{z \cap V(\gamma(u)) | z \in \mathcal{Z}\}$.

Now 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. Let E, F, G be adversary structures and A, B, C be sets of nodes. Let $E^A = \{z \cap A | z \in E\}$ denote the restriction of the adversary structure E to the set of nodes A. The joint adversary structure from two restricted adversary structures can be obtained through the $\oplus$ operator.

Definition 16. *Let $\mathcal{Z}^A$ denote the space of adversary structures on the set of nodes A. Then operator $\oplus$ is a function of the form $\oplus : \mathcal{Z}^A \times \mathcal{Z}^B \to \mathcal{Z}^{(A \cup B)}$, for any A, B and is defined as follows:*

$$E^A \oplus F^B = \{z_1 \cup z_2 | (z_1 \in E^A) \wedge (z_2 \in F^B) \wedge (z_1 \cap B \subseteq z_2) \wedge (z_2 \cap A \subseteq z_1)\}$$

Next we are going to show that the $\oplus$ operator is commutative.

Theorem 30. *Operator $\oplus$ is commutative.*

Proof. A binary operation $*$ is called commutative if $a * b = b * a$. For any adversary

structures E, F and node sets A, B:

$$\begin{aligned} E^A \oplus F^B &= \{z_1 \cup z_2 | (z_1 \in E^A) \wedge (z_2 \in F^B) \wedge (z_1 \cap B \subseteq z_2) \wedge (z_2 \cap A \subseteq z_1)\} \\ &= \{z_2 \cup z_1 | (z_2 \in F^B) \wedge (z_1 \in E^A) \wedge (z_2 \cap A \subseteq z_1) \wedge (z_1 \cap B \subseteq z_2)\} \\ &= F^B \oplus E^A \end{aligned}$$

So operator $\oplus$ is commutative. □

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

Lemma 31. *For any node sets A, B, C it holds that*

$$\begin{gathered} (z_1 \cap B \subseteq z_2) \wedge (z_2 \cap A \subseteq z_1) \wedge (z_1 \cup z_2 \cap C \subseteq z_3) \wedge (z_3 \cap A \cup B \subseteq z_1 \cup z_2) \\ \Leftrightarrow \\ (z_2 \cap C \subseteq z_3) \wedge (z_3 \cap B \subseteq z_2) \wedge (z_2 \cup z_3 \cap A \subseteq z_1) \wedge (z_1 \cap B \cup C \subseteq z_2 \cup z_3) \end{gathered}$$

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.

□

Theorem 32. *Operator $\oplus$ is associative.*

Proof. A binary operation $*$ is called associative if $(a * b) * c = a * (b * c)$ for any well defined a, b, c. For any adversary structures E, F, G and node sets A, B, C:

$$\begin{aligned}(E^A \oplus F^B) \oplus G^C &= \{z_1 \cup z_2 | (z_1 \in E^A) \wedge (z_2 \in F^B) \wedge (z_1 \cap B \subseteq z_2) \wedge (z_2 \cap A \subseteq z_1)\} \oplus G^C \\ &= \{z_1 \cup z_2 \cup z3 | (z_1 \in E^A) \wedge (z_2 \in F^B) \wedge (z_3 \in G^C) \wedge (z_1 \cap B \subseteq z_2) \\ &\wedge (z_2 \cap A \subseteq z_1) \wedge (z_1 \cup z_2 \cap C \subseteq z_3) \wedge (z_3 \cap A \cup B \subseteq z_1 \cup z_2)\}\end{aligned}$$

$$\begin{aligned}E^A \oplus (F^B \oplus G^C) &= E^A \oplus \{z_2 \cup z_3 | (z_2 \in F^B) \wedge (z_3 \in G^C) \wedge (z_2 \cap C \subseteq z_3) \wedge (z_3 \cap B \subseteq z_2)\} \\ &= \{z_1 \cup z_2 \cup z3 | (z_1 \in E^A) \wedge (z_2 \in F^B) \wedge (z_3 \in G^C) \wedge (z_2 \cap C \subseteq z_3) \\ &\wedge (z_3 \cap B \subseteq z_2) \wedge (z_2 \cup z_3 \cap A \subseteq z_1) \wedge (z_1 \cap B \cup C \subseteq z_2 \cup z_3)\}\end{aligned}$$

But from lemma 31 it follows that:

$$E^A \oplus (F^B \oplus G^C) = (E^A \oplus F^B) \oplus G^C$$

So operator $\oplus$ is associative. □

Theorem 33. *Operation $\oplus$ is idempotent.*

Proof. Given some operation $*$ we say that it is idempotent iff $a * a = a$ for any possible a.

$$\begin{aligned}E^A \oplus E^A &= \{z_1 \cup z_2 | (z_1 \in E^A) \wedge (z_2 \in E^A) \wedge (z_1 \cap A \subseteq z_2) \wedge (z_2 \cap A \subseteq z_1)\} \\ &= \{z_1 \cup z_2 | (z_1 \in E^A) \wedge (z_2 \in E^A) \wedge (z_1 = z_2)\} \\ &= \{z_1 | (z_1 \in E^A)\} \\ &= E^A\end{aligned}$$

So operation $\oplus$ is idempotent. □

Theorem 34. *Let V be a finite set and $S = \{(E, A) | E \subseteq 2^A \wedge A \subseteq V\}$. Then $< S, \oplus >$ is a semilattice.*

Proof. A set L with some operations $*$ is a semilattice if the operation $*$ is commutative, associative and idempotent. From the previous theorem all these properties hold for the $\oplus$ operation and the set S. □

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

Lemma 35. *For any adversary structures E, F and node sets A, B let $G = E^A \oplus F^B$ where $G \in \mathcal{Z}^{A\cup B}$. It holds that $G'^A = E^A$ and $G'^B = F^B$.*

Theorem 36. *For any adversary structures E, F and node sets A, B let $G = E^A \oplus F^B$ where $G \in \mathcal{Z}^{A\cup B}$. It holds that $\forall G' \in \mathcal{Z}^{A\cup B}$: if $G'^A = E^A$ and $G'^B = F^B$ then $G' \subseteq G$.*

Proof. Suppose that there existed some G' s.t. $\exists z \in G' : z \notin G$. For z we have $z_1 = z \cap A \in E^A$ and $z_2 = z \cap B \in F^B$. Also $z_1 \cap B = (z \cap A) \cap B \subseteq z \cap B = z_2$ and symmetrically $z_2 \cap A \subseteq z_1$. But then from definition $z \in G$ which is a contradiction and no such G' exists. □

Corollary 37. *For any adversary structures E and node sets A, B:*
$E^{(A\cup B)} \subseteq E^A \oplus E^B$.

What theorem 36 tells us is that the $\oplus$ operation gives the maximal adversary structure possible that seems indistinguishable when considered under the perspective of any of the two initial domains, namely $\mathcal{Z}^A$ and $\mathcal{Z}^B$. This is exactly what we wanted initially to describe. An operation that captures the worst case scenario relatively to our initial knowledge.

So using these symbols we have $Z_u = Z^{\gamma(u)}$. For a given adversary structure Z and a view function γ let

$$Z_B = \bigoplus_{v\in B} Z^{\gamma(u)}$$

Then Z_B exactly captures the maximal adversary structure possible, restricted in $\gamma(B)$, relatively to the initial knowledge of players in B. Also notice that using corollary 37 we get $Z^{\gamma(B)} \subseteq Z_B$. The interpretation of this inequality in our setting, is that what nodes in B conceive as the worst case adversary structure indistinguishable to them relative to their initial knowledge, is always bigger or equal than the actual adversary structure in their scenario. This result follows our intuition.

Chapter 5

Conclusions

In this thesis a number of fundamental results on Byzantine Agreement, as well as some new results on the topologically restricted setting, were presented. In particular, on the first part emphasis was given on possibility and impossibility results on the classic setting. Most of these results are based on the indistinguishably argument i.e. players are limited on what they can distinguish, or in a positive manner, players need enough information in order to be able to distinguish the scenario they are in.

On the second part new results were presented on a model were the information players have about the adversary are topologically restricted. This model is relevant when information about the adversary in the whole communication graph are either impossible to obtain or hard to process. In this setting a number of older relevant results and open questions were closed and a new model capturing exactly this limitation was presented, the partial knowledge model. A number of open questions arise from this work.

- Necessary and sufficient criteria for Broadcast on known topology and ad-hoc networks are NP-hard to compute. It remains open to define and study meaningful approximation objectives.

- We conjecture that in the known topology locally bounded setting no safe, fully polynomial algorithm can achieve optimal resilience. We have provided an indication towards proving this in Subsection 4.4.3.

- Regarding the partial knowledge model discussed in Section 4.5, GPPA is not of optimal resilience. Devising such an algorithm would be of great interest. One direction towards this, is to consider discovering the network topology under a Byzantine adversary, as studied in [21, 10].

- In the *ad hoc* general adversary setting, we proved that $\mathcal{Z}$-CPA is unique, thus having optimal resilience. We conjecture that it is also unique w.r.t. polynomial time complexity, i.e., if a safe protocol achieves Broadcast in polynomial time then so does $\mathcal{Z}$-CPA.

- Throughout this work a number of adversarial models were studied. Each one gives a different limited form to the adversary's power. Looking at these results from a protocol design perspective we would like to adopt an adversarial model that supports a topologically restricted view of the adversary (unlike the threshold model) and also allows a protocol to efficiently exploit this information. Unfortunately the locally bounded model does not seem to satisfy these criteria. So an interesting direction to explore is finding an adversarial model that matches these criteria and it is as expressive as possible.

Bibliography

[1] Amotz Bar-noy, Danny Dolev, Cynthia Dwork, and H Raymond Strong. Shifting gears: Changing algorithms on the fly to expedite byzantine agreement. In *Information and Computation*, pages 42–51, 1987.

[2] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, PODC '83, pages 27–30, New York, NY, USA, 1983. ACM.

[3] Piotr Berman, Juan A Garay, and Kenneth J Perry. Towards optimal distributed consensus. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 410–415. IEEE, 1989.

[4] Armando Castañeda, Yannai A. Gonczarowski, and Yoram Moses. Good, better, best! - unbeatable protocols for consensus and set consensus. *CoRR*, abs/1311.6902, 2013.

[5] Brian A Coan. A communication-efficient canonical form for fault-tolerant distributed protocols. In *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '86, pages 63–72, New York, NY, USA, 1986. ACM.

[6] Danny Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[7] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, January 1993.

[8] Danny Dolev, Michael J Fischer, Rob Fowler T, Nancy A Lynch, and H. Raymond Strong. An efficient algorithm for byzantine agreement without authentication. *Information and Control*, 52:257–274, 1982.

[9] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, January 1985.

[10] Shlomi Dolev, Omri Liba, and EladM. Schiller. Self-stabilizing byzantine resilient topology discovery and message delivery. In Teruo Higashino, Yoshiaki Katayama, Toshimitsu Masuzawa, Maria Potop-Butucaru, and Masafumi Yamashita, editors,

Stabilization, Safety, and Security of Distributed Systems, volume 8255 of *Lecture Notes in Computer Science*, pages 351–353. Springer International Publishing, 2013.

[11] Matthias Fitzi and Ueli M. Maurer. Efficient byzantine agreement secure against general adversaries. In Shay Kutten, editor, *DISC*, volume 1499 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 1998.

[12] Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Comput.*, 27(1):247–290, 1998.

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

[14] Martin Hirt and Ueli M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In James E. Burns and Hagit Attiya, editors, *PODC*, pages 25–34. ACM, 1997.

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

[16] Valerie King and Jared Saia. Byzantine agreement in polynomial expected time: [extended abstract]. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 401–410, New York, NY, USA, 2013. ACM.

[17] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In Soma Chaudhuri and Shay Kutten, editors, *PODC*, pages 275–282. ACM, 2004.

[18] M V N Ashwin Kumar, Pranava R. Goundan, K Srinathan, and C. Pandu Rangan. On perfectly secure communication over arbitrary networks. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, PODC '02, pages 193–202, New York, NY, USA, 2002. ACM.

[19] Chris Litsas, Aris Pagourtzis, and Dimitris Sakavalas. A graph parameter that matches the resilience of the certified propagation algorithm. In Jacek Cichon, Maciej Gebala, and Marek Klonowski, editors, *ADHOC-NOW*, volume 7960 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2013.

[20] Yoram Moses and Mark R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.

[21] Mikhail Nesterenko and Sébastien Tixeuil. Discovering network topology in the presence of byzantine faults. In *In: Proceedings of the 13th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2006, LNCS*, pages 212–226. Springer Verlag, 2006.

[22] 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*, pages 107–121, 2014.

[23] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, April 1980.

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

[25] Lewis Tseng and Nitin Vaidya. Iterative approximate byzantine consensus under a generalized fault model. In Davide Frey, Michel Raynal, Saswati Sarkar, RudrapatnaK. Shyamasundar, and Prasun Sinha, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 72–86. Springer Berlin Heidelberg, 2013.

[26] Lewis Tseng, Nitin H. Vaidya, and Vartika Bhandari. Broadcast using certified propagation algorithm in presence of byzantine faults. *CoRR*, abs/1209.4620, 2012.