
# Εθνικο Μετσοβιο Πολυτεχνειο

Δ.Π.Μ.Σ. Εφαρμοσμενες Μαθηματικες Επιστημες

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Λογικης & Επιστημης Υπολογισμων CoReLab


---

# Ασφαλής Εκπομπή σε Γενικά και Ασύρματα Δίκτυα

---


Μεταπτυχιακη Εργασια

του

**Δημήτρη Σακαβάλα**

**Επιβλέπων:** Άρης Παγουρτζής
Επίκουρος Καθηγητής Ε.Μ.Π.



Αθήνα, Οκτώβριος 2012


# Εθνικο Μετσοβιο Πολυτεχνειο

## Δ.Π.Μ.Σ. Εφαρμοσμενες Μαθηματικες Επιστημες

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Λογικης & Επιστημης Υπολογισμων CoReLab


---

# Ασφαλής Εκπομπή σε Γενικά και Ασύρματα Δίκτυα

---

Μεταπτυχιακη Εργασια

του


**Δημήτρη Σακαβάλα**


**Επιβλέπων:** Άρης Παγουρτζής
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 15 Οκτωβρίου 2012.

| | | |
|---|---|---|
| ..... | ..... | ..... |
| Ευστάθιος Ζάχος | Άρης Παγουρτζής | Δημήτριος Φωτάκης |
| Καθηγητής Ε.Μ.Π. | Επίκουρος Καθηγητής Ε.Μ.Π. | Λέκτορας Ε.Μ.Π. |

---

Αθήνα, Οκτώβριος 2012

# ΠΡΟΛΟΓΟΣ

Η παρούσα μεταπτυχιακή εργασία εκπονήθηκε στα πλαίσια της εκπλήρωσης των υποχρεώσεων για το διατμηματικό προγράμματα μεταπτυχιακών σπουδών 'Εφαρμοσμένες Μαθηματικές Επιστήμες' του Εθνικού Μετσόβιου Πολυτεχνείου.

Τα περιβάλλοντα κατανεμημένων υπολογισμών αποτελούνται από πολυάριθμες ανεξάρτητες υπολογιστικές συσκευές που συνεργάζονται για την επίτευξη ενός κοινού στόχου. Για το σκοπό αυτό οι συσκευές αυτές διατάσσονται σε ένα δίκτυο επικοινωνίας. Για να είναι αποτελεσματικά σε κρίσιμες εφαρμογές (όπως συστήματα ελέγχου πτήσης και συστήματα ελέγχου σε πυρηνικούς σταθμούς παραγωγής ενέργειας), τα περιβάλλοντα κατανεμημένων υπολογισμών πρέπει να είναι σε θέση να αντιμετωπίσουν τη δυσλειτουργία κάποιων συσκευών. Για παράδειγμα, κάποια συσκευή μπορεί να αρνείται να παραδώσει αναγκαίες πληροφορίες σε άλλα μέρη του συστήματος. Ακόμα χειρότερα η συσκευή μπορεί να στέλνει αντικρουόμενες και ασυνεπείς πληροφορίες σε άλλα μέρη του συστήματος. Η τελευταία μορφή δυσλειτουργίας καλείται *ενεργητική βλάβη* (ή Βυζαντινή βλάβη). Μια βολική θεώρηση για την μελέτη των δυσλειτουργιών ορισμένων συσκευών του συστήματος, είναι η υπόθεση ύπαρξης ενός κακόβουλου *αντιπάλου* ο οποίος διαφθείρει κάποιους από τους συμμετέχοντες (συσκευές) του κατανεμημένου συστήματος. Είναι πλέον ευρέως αποδεκτό πως ένα αναπόσπαστο μέρος ενός κατανεμημένου συστήματος, σε περιβάλλον όπου υποθέτουμε την ύπαρξη ενός αντιπάλου, είναι ένας μηχανισμός για την επίτευξη '*συμφωνίας*' μεταξύ των μη-διεφθαρμένων μερών του συστήματος. Σε έναν τέτοιο μηχανισμό όλοι οι μη-διεφθαρμένοι συμμετέχοντες πρέπει να είναι σε θέση να συμφωνήσουν σε κοινή τιμή χωρίς να ξέρουν ποια μέρη είναι διεφθαρμένα.

Τα διάφορα προβλήματα συμφωνίας έχουν προσελκύσει την έρευνα από τις αρχές τις δεκαετίας του 1980. Μία από τις σημαντικότερες παραλλαγές του προβλήματος είναι γνωστή στη βιβλιογραφία ως το πρόβλημα της *Ασφαλούς Εκπομπής* (ή το πρόβλημα των Βυζαντινών Στρατηγών). Στο πρόβλημα αυτό, θεωρούμε την ύπαρξη ενός καθορισμένου συμμετέχοντος, στην αρχική τιμή του οποίου θα πρέπει να συμφωνήσει κάθε μη-διεφθαρμένος συμμετέχον. Το πρόβλημα έχει μελετηθεί εκτενώς στο μοντέλο όπου είναι εξασφαλισμένη η διμερής επικοινωνία μεταξύ κάθε ζευγαριού συμμετεχόντων (πλήρες δίκτυο επικοινωνίας). Η περίπτωση των μη πλήρων δικτύων επικοινωνίας έχει μελετηθεί σε κάποιο βαθμό. Η έρευνα σε αυτήν την κατεύθυνση εισήγαγε νέες παραμέτρους για βελτιστοποίηση η οποίες αφορούν την τοπολογία του δικτύου και σχετικά προβλήματα παραμένουν άλυτα. Επιπλέον, θεωρήθηκαν μοντέλα αντιπάλου με τοπολογικούς περιορισμούς στο πλαίσιο των οποίων μελετώνται τα σχετικά όρια τα οποία καθιστούν το πρόβλημα επιλύσιμο. Τέλος, το πρόβλημα της Ασφαλούς Εκπομπής σε ασύρματα δίκτυα επικοινωνίας εξετάστηκε για πρώτη

φορά το 2004, και έχει μελετηθεί κυρίως σε πολύ ειδικές τοπολογίες δικτύων, π.χ. δίκτυα πλέγματος. Η εξέταση του προβλήματος στα ασύρματα δίκτυα δημιουργεί νέες ανάγκες και προκλήσεις για εφαρμογές ασφαλών κατανεμημένων υπολογισμών. Αφενός, η δομή αυτών των δικτύων επιτρέπει στους διεφθαρμένους συμμετέχοντες να δημιουργούν παρεμβολές σήματος σε άλλους παραλήπτες (δέκτες) μηνυμάτων, το οποίο έχει ως αποτέλεσμα την επιδείνωση της ποιότητας επικοινωνίας. Αφετέρου, οι συμμετέχοντες δεσμεύονται να εκτελούν τοπικές εκπομπές, το οποίο διευκολύνει σε μεγάλο βαθμό την επίτευξη συμφωνίας. Ένα ακόμα σημαντικό ζήτημα, είναι η μελέτη τοπικών κριτηρίων τα οποία μπορούν να χρησιμοποιηθούν από τους συμμετέχοντες για την επίτευξη Ασφαλούς Εκπομπής σε δίκτυα άγνωστης τοπολογίας.

## Ευχαριστίες

Παρουσιάζοντας την παρούσα μεταπτυχιακή εργασία, θα ήθελα να ευχαριστήσω όλους όσους συντέλεσαν στην ολοκλήρωσή της.

Θα ήθελα να ευχαριστήσω τον επιβλέποντα της εργασίας μου, κ. Άρη Παγουρτζή, Επίκουρο Καθηγητή στο Εθνικό Μετσόβιο Πολυτεχνείο, για την εμπιστοσύνη και το ενδιαφέρον που έδειξε τόσο κατά την ανάθεση της εργασίας, όσο και κατά τη διάρκεια διεξαγωγής των μαθημάτων του μεταπτυχιακού προγράμματος. Θα ήθελα επίσης να τον ευχαριστήσω για τη συμπαράσταση και την καθοδήγησή του καθ' όλη τη διάρκεια εκπόνησης της εργασίας μου.

Θα ήθελα να ευχαριστήσω τον κ. Στάθη Ζάχο, Καθηγητή του ΕΜΠ, και τον κ. Δημήτρη Φωτάκη, Λέκτορα του ΕΜΠ, που με στηρίξανε να κάνω αυτό που ήθελα στις μεταπτυχιακές μου σπουδές.

Επίσης θα ήθελα να ευχαριστήσω τον Υποψήφιο Διδάκτορα Χρήστο Λίτσα για τις τις συζητήσεις που είχαμε, αποτέλεσμα των οποίων είναι ένα μέρος της παρούσας εργασίας.

Τελευταίους και περισσότερο, ευχαριστώ τον πατέρα μου και τη μάνα μου για τη στήριξη, την υπομονή και τη χρηματοδότηση όλων αυτών των χρόνων που περάσανε.

# PREFACE

The current thesis has been elaborated in fulfillment of the thesis requirement for the Inter-Departmental Postgraduate program "Applied Mathematical Sciences" of the National Technical University of Athens (NTUA).

Distributed computing environments consist of several independent computing devices that act together to achieve a specific goal. For this purpose the interacting devices are arranged in a communication network. In order to be useful in mission-critical applications (such as flight control systems and control systems in nuclear power plants), distributed computing environments must be able to cope with failures of some devices. Generally, failures can have different forms. For example, one device of the system may refuse to deliver necessary information to other parts of the system. Even worse, the device may send conflicting and inconsistent information to other parts of the system. Such failures are called *active failures* (or byzantine failures). A convenient approach for arguing about failures is to assume the existence of an *adversary* which corrupts participants of a distributed system. It is widely accepted that an integral part of a mission-critical distributed system is a mechanism for reaching "agreement" between all non-corrupted parts of a system. In this setup, all non-corrupted devices should be able to agree on a common value without explicit knowledge of the corrupted parts.

The agreement problem has attracted research since the early 1980's. One of the major variations is known in the literature as the *Secure Broadcast* problem (Byzantine Generals problem), hereafter referred as Broadcast. In the Broadcast problem, we assume the existence of a designated participant on whose input value every non-faulty participant should agree. The problem has been extensively studied in the standard model in which pairwise communication is available between all pairs of participants (complete communication networks). The case of incomplete communication networks has been studied to some extent; the research introduced new parameters for optimization, relative to the communication network topology and related problems remain open so far. Moreover topologically restricted adversary models have been considered, and with their introduction the study of new feasibility and infeasibility bounds has been initiated. Finally, Broadcast in wireless communication networks has only been considered since 2004, and has mainly been studied under very special network topologies, e.g., grid networks. The consideration of the wireless network model brings up new needs and challenges for secure distributed computing applications. On the on hand, the structure of such networks allows the corrupted participants to cause interference to other receivers resulting in the deterioration of the communication quality. On the other hand,

participants are committed to perform local broadcasts, which greatly facilitates achieving agreement. Another important consideration, is the derivation of local criteria which the players can use in order to achieve Broadcast in networks of unknown topology.

# Contents

# Chapter 1

# Agreement in Unreliable Distributed Systems

## 1.1 Introduction

As communication networks grow in size, they become increasingly vulnerable to component failures. These networks consist of numerous interacting entities hereafter referred as *players*. Since distributed computing has become popular and widely used in contemporary networking, the provided solutions need to cope with erroneous and malicious components in the underlying communication network.

With the term *distributed system* we will refer to a software that executes a collection of protocols to coordinate the actions of multiple players on a given communication network $G$, such that all components cooperate together to perform a single or small set of related tasks. Typically, such network systems are characterized by a symmetric relationship between participating players and the lack of any central authority. Network functionality is achieved in a decentralized fashion, with autonomous devices cooperating to execute tasks such as demanding computations, distribution of digital content, or common decision making.

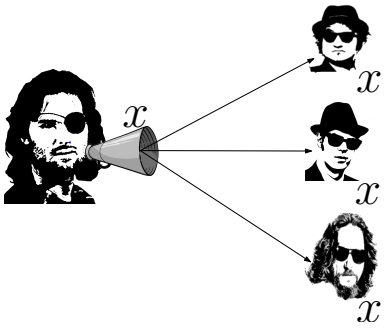
The study of distributed systems in the presence of adversarial behavior involves the design of algorithms that work correctly despite the existence of corrupted players in the network and without knowing their location. Several corruption types have been considered in the literature. Among all the corruption types, *active* (or *Byzantine*) corruption models a worst-case fault scenario, namely components can behave arbitrarily (even maliciously) as transmitters, by either stopping, rerouting,

or altering transmitted messages in a way most detrimental to the communication process.

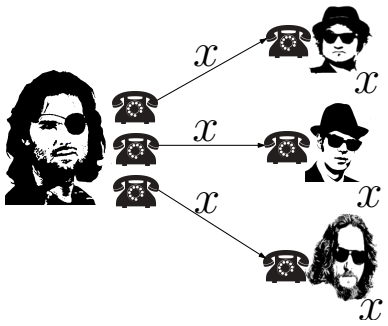
Agreement of the participating players in a certain value is of profound importance and its applications are many. There are two major variations of the problem, both introduced by L.Lamport, R.Shostack and M.Pease in [PSL80, LSP82],

- **Broadcast (Byzantine Generals):** Have some designated player, called the dealer, consistently send a message to all other players.
- **Consensus (Byzantine Agreement):** Make all players agree on the same output value given that every player starts with an input value. If all honest players hold the same input value then the output value is required to be the same as this input value

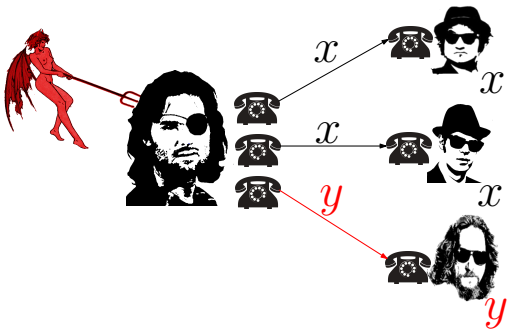
In the current thesis we will focus in the Broadcast problem, although we will see that the problems are closely related and results on the one problem easily imply similar results for the other.




(a) Ideal Broadcast




(b) Real Broadcast




(c) Real Broadcast with Corrupted Dealer

In Broadcast the difficulty of designing a solution can be primarily summarized in a scenario where the dealer is corrupted. As seen in the above pictures the dealer

may send conflicting values to the players in which case we demand that the players finally agree on the same arbitrary value. The purpose of a Broadcast protocol is to simulate the Ideal Broadcast through the communication of the players. Essentially, the major task is to circumvent errors without losing unanimity.

In the general case one does not know which players are corrupted. Moreover, in most cases one will never be able to find it out. To understand the reason for this, one should picture himself as a player $u$ in a distributed system that receives a message from a player $v$. Assume that $u$ wants to make sure that $v$ is honest. So, $u$ inquires what are the messages the rest of the system received. If player $u$ finds out that the message he has received differs from all the rest, then does this make $v$ a corrupted player? The answer is not necessarily positive. The possibility that the only reliable players in the system are $u$ and $v$ always exists. To overcome this logical pathology, one should make a decision assuming limitations on the frequency of occurrence of corruptions in the system. That is, if in the system these limitations don't hold, it will no longer matter what decision $u$ is going to make.

How should such limitations be expressed? In other work on analysis of systems with adversarial behavior, these limitations often take the form of probability distributions governing the occurrences of corruptions. Here, instead of using probabilities, we simply assume that the number of corruptions is bounded in advance, by a fixed number $t$. This is a simple assumption to work with, since it avoids the complexities of reasoning about probabilistic failure occurrences. In practice, this assumption may be realistic in the sense that it may be unlikely that more than $t$ corruptions will occur. However, we should keep in mind that the assumption is somewhat problematic: in most practical situations, if the number of corruptions is already large, then it is likely that more corruptions will occur. Assuming a bound on the number of corruptions implies that corruptions are negatively correlated, whereas in practice, corruptions are usually independent or positively correlated.

**Secure Multiparty Computation.** Agreement problems are special cases of the more general problem of *multi-party computation* (MPC), initially defined by Yao [Yao82]. In MPC we have $n$ players $v_1, \ldots, v_n$, each of which holds a private input, who wish to perform a joint computation on these inputs in a secure way. Roughly speaking, the security of the computation requires that the output is correctly computed on the inputs given by the players (correctness), and that the players learn no more information than what they can deduce from their specified outputs of the computation (privacy). The security of the computation should be guaranteed even when some players misbehave.

**Applications**

The agreement problem is of profound importance and its applications are many. In the footnote of the seminal paper of L.Lamport, R.Shostack and M.Pease, we see that they were sponsored by NASA, the Ballistic Missile Defense Systems Command and the Army Research Office. N. Lynch mentions in her book "Distributed Algorithms" [Lyn96], that the agreement problem is a simplified version of a problem that originally arose in the development of on-board aircraft control systems. In this problem, a collection of processors, each with access to a separate altimeter, and some of which may be faulty, attempt to agree on the airplane's altitude. Agreement algorithms have also been incorporated into the hardware of fault-tolerant multiprocessor systems; there, they are used to help a small collection of processors to carry out identical computations, agreeing on the results of every step. This redundancy allows the processors to tolerate the failure of a number of components. Agreement algorithms are also useful in processor fault diagnosis, where they can permit a collection of processors to agree on the failure of specific components (and should therefore be replaced or ignored).

## 1.2 The Communication Model

The players communicate with each other over channels. Virtually all protocols assume the existence of pairwise channels among the players. These channels can be assumed to be:

***Authenticated*:** The channel is resistant to tampering but not necessarily resistant to overhearing, i.e., messages, once sent, cannot be changed but can be read by an adversary. Moreover using an authenticated channel the receiver of a message always knows the identity of the sender.

***Secret/Confidential*:** The channel is resistant to overhearing , but not necessarily resistant to tampering.

***Secure*:** Authenticated and secret channel.

***Synchronous*:** The delay of messages in the channel is bounded by a known constant.

The topology of the network of channels can be *complete* or *incomplete*, i.e., the connectivity can be limited.

In the *asynchronous* channels model of communication where message delivery or bounds on transit time is not guaranteed, it is still possible to solve most of

the problems in a strictly weaker sense using asynchronous communication. As an example, assume that we want to tolerate that up to $t$ of the $n$ players are corrupted. If an honest player waits for messages from more than $n - t$ players, then it might potentially be waiting for a message from a corrupted player. This corrupted player might not have sent its message, and since no lower bound on message delivery is guaranteed, an unsent message cannot be distinguished from a slow message. The player might therefore end up waiting forever for the unsent message, and the protocol deadlocks. So, in an asynchronous protocol which must tolerate $t$ corruptions and must be dead-lock free, the honest players cannot wait for messages from more than $n - t$ players in each round. But this means that some of the honest players might not even be able to send their inputs to the other honest players, left alone having their inputs securely contribute to the result.

In the current thesis we consider synchronous networks with authenticated channels. In the first two Chapters we will study protocols and bounds concerning complete networks and in the last chapters incomplete networks will be considered.

## 1.3 The Adversary Model

The dishonesty of players is modeled by a *central adversary* that corrupts players. For concreteness, one may think of the adversary as a hacker who attempts to break into the players' computers. We distinguish corruption modes wrt the capacities of the adversary:

***Passive*:** Corrupted players gives all their internal data to the adversary, but continue executing the instructions of the protocol.

***Active/Byzantine*:** Corrupted players are under full control of the adversary and misbehave in arbitrary manner.

***Fail-Stop*:** Players follow the protocol instructions till the adversary instructs the player to crash; from then on, the player does not send any message to any other player. Note that a fail-corrupted player does not give his internal data to the adversary, unless he is passively corrupted at the same time.

If we allow the adversary to corrupt all subsets of the players of size at most $t$ for some $t < n$, with $n$ being the total number of players, then we call it a *threshold adversary* and we call t the *threshold*.

Finally, the adversary model also determines the adversary's computing power. Most common assumptions are that the adversary is either *unlimited*, or is *com-*

*putationally bounded* to probabilistic polynomial time computations in a security parameter $\kappa$.

In the current thesis we consider the existence of an active adversary. The results induced also hold for the other adversary types since an active adversary models a worst-case scenario on the corruption strength.

## 1.4 Security

Analogously to cryptographic primitives, security is typically defined with respect to a security parameter $\kappa$, allowing an error probability $\epsilon$ that is negligible in function of $\kappa$, i.e., it is tolerated that the respective task fails (the conditions of the problem are not satisfied) with probability at most $\epsilon$. The types of security considered in the literature are the following

***Computational/Cryptographic Security*:** Denotes security against an adversary who is bounded to probabilistic polynomial time computations.

***Unconditional / Information-Theoretic Security*:** Denotes security against an unlimited adversary (security derives purely from information theory).

***Perfect Security*:** Unconditional security with zero error probability $\epsilon$. (the conditions of the problem are satisfied in the absolute sense).

It can be assumed that, additionally, some (partially secret) data is consistently set up among the players. This could for example be attained by a precomputation phase involving a mutually trusted party to distribute some related information to the players. The shared data would typically be a PKI (Public Key Infrastructure). In this case we say that the players hold *consistently shared data*. In the literature the term *Cryptographic Security* is often used for Computational security with consistently shared data among players.

We will focus on perfectly secure protocols for the problems considered.

## 1.5 Efficiency and Resiliency

The complexity of MPC protocols can be stated with respect to the local computation complexity of the players and with respect to the amount of communication that is required among the players. Moreover the number of communication rounds

required for the completion of the protocol is an other measure of consideration. In synchronous networks, we assume that during each communication round, all nodes in parallel receive the latest messages from their neighbors, perform arbitrary local computation and finally send new messages to their neighbors. The measures considered for optimization are the following:

***Bit Complexity* (BC):** This is the total number of bits sent by all honest players during the protocol in the worst case, overall. We will also use the term *Message Complexity* for the total number of messages sent by all honest players.

***Round complexity* (RC):** This is the maximal number of subsequent execution steps (communication rounds) that are required by any correct player in the worst case.

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

***Resiliency*:** It means the number of corrupted players, up to which a protocol can tolerate. If a protocol can handle up to $t < an$ actively corrupted players, then we say that it is $a$-resilient.

## 1.6 Defining the Problem

The formal definition of the Broadcast problem follows:

**1.1 Definition** (Broadcast)**.** *Let $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ be a set of $n$ players, $X$ be a finite domain and $D \in \mathcal{V}$ be the dealer. Then we say, that $\Pi$ is a Broadcast (Byzantine Generals) protocol among players in $\mathcal{V}$ with values in $X$, where $D$ has as input a value $x_d \in X$ and all players finally decide on a value $y_i \in X$, if it satisfies the following conditions:*

1. Validity: *If the dealer is honest, then all honest players will decide on $x_d$;*
2. Consistency: *All honest players finally decide on the same value. i.e. $\forall v_i, v_j$ honest players, $y_j = y_i$ holds;*
3. Termination: *All players will eventually terminate the protocol;*

We also define the related problem of Consensus as we will take advantage its relation with Broadcast to develop protocols in later chapters.

**1.2 Definition** (Consensus)**.** *Let $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ be a set of $n$ players, $X$ be a finite domain and $D \in \mathcal{V}$ be the dealer. Then we say, that $\Pi$ is a Byzantine*

*General (Broadcast) protocol among players in $\mathcal{V}$ with values in $X$, where each $v_i$ has an initial input value $x_i \in X$ and finally decides on a value $y_i \in X$, if it satisfies the following conditions:*

1. Validity: *If all honest players have as input the same value $x$, then all honest players will decide on $x$;*
2. Consistency: *All honest players finally decide on the same value. i.e. $\forall v_i, v_j$ honest players, $y_j = y_i$ holds;*
3. Termination: *All players will eventually terminate the protocol;*

The Termination property in the context of the protocols and the bounds presented is usually immediately implied by the context and the proof of its validity will often be omitted.

**Interactive Consistency.** The Consensus problem has a closely related variant, named *Interactive Consistency* (IC), which was studied in [PSL80], before the introduction of the Broadcast and Consensus problems. In this version of the problem, instead of agreeing on a single value, the players have to agree on a vector of values, which has an entry corresponding to every player in the system. The output vectors of any two honest players must be equal. Furthermore, an entry in the vector that corresponds to a honest player has to be equal to the input value of that player. Clearly, an IC algorithm also solves the Consensus problem, e.g., by deciding on the majority value in the output vector. Moreover an algorithm for Broadcast solves the IC problem, by letting every player broadcast his input value, the required vector can then be composed trivially.

The definition of consensus can at most allow for a strict minority of corrupted players ($t < n/2$)–otherwise the corrupted players, by majority, would always be able to dictate the outcome independently of the inputs by the honest players.

**1.1 Proposition.** *Consensus among $n \geq 2$ players, secure against $t \geq n/2$ corrupted players is impossible.*

*Proof.*
Let $V_0 \cup V_1 = \mathcal{V}$ be a partition of the player set in to two sets of cardinalities $|V_0| = \max(n-t, 1)$, $|V_1| = \min(t, n-1)$; and let all players $v_i \in V_0$ hold the same input value $x_i = 0$ and all players $v_i \in V_1$ hold input value $x_i = 1$. The adversary can now choose either $\emptyset, V_0,$ *or* $V_1$ uniformly at random and corrupt the respective players by having them honestly follow the protocol. To make this clear, we supposed that an active adversary forces his players behave arbitrarily,

yet arbitrarily includes consistent behavior. Consequently, a honest player cannot distinguish between honest and corrupted players.

If, at the end, all players compute the same output value $x$ then validity is violated with probability at least 1/3 since the adversary corrupts $V_x$ with probability 1/3. If, at the end, the players compute different output values then consistency is violated with probability at least 1/3 since the adversary does not corrupt any player with probability 1/3.

□

**1.2 Theorem.** *If $t < n/2$, then Broadcast and Consensus are equivalent. In other words, if there is a protocol for broadcast, then we can have a protocol for Consensus too and visa versa.*

*Proof.*
" $\Rightarrow$" : Assume, that $\Pi$ is a broadcast protocol. Consensus can be achieved by having every player broadcast its input value using $\Pi$ and then, having every player decide on the majority of the received values. Since all values are distributed by broadcast and a majority of the players is honest, this protocol achieves consensus.

" $\Leftarrow$" : Assume that $\Pi$ is a Consensus protocol. Then broadcast can be simulated by having the sender distribute (multi-send) his input value to all players and having all players run a consensus protocol on the values received from the sender. Since a majority of the players is honest, this protocol achieves broadcast.

Consequently, Broadcast and Consensus are equivalent.

□

Usually, the Consensus problem is studied in the literature related with unanimity in complete graphs. This is due to the fact that a Consensus protocol implies a Broadcast protocol with the overhead of just one extra round as can be observed in Theorem 1.2. Moreover the majority of Broadcast protocols presented in the literature are of this certain form, namely, in the first round Dealer sends his initial value and then the players achieve consensus. We will refer to this certain kind of Broadcast Protocols as *Basic Broadcast* protocols.

**1.3 Definition.** *A* Basic Broadcast *protocol is of the form:*

- ***Round 1:*** *Dealer sends its initial value $x_d$ to all players.*
- ***Round* r** *($r \geq 2$):* *All the players run a Consensus protocol, with inputs the values received from the dealer in order to achieve agreement on the Dealer's value.*

## 1.7 Scenarios and Views

We will now formalize the description of a distributed system using definitions which facilitate our further study. Let $\mathcal{V}$ be the set of players (participants) in the distributed system. We denote by $X$ the initial input space and $M \supseteq X$ the message space in which every message sent in the system belongs.

**1.4 Definition** (Scenario ). *A scenario $\sigma$ is a function*

$$\sigma : \mathcal{V} \times \mathcal{V} \times \mathbb{N} \to \mathcal{M}$$

*Value $\sigma(v, w, i)$ is equal to the message that player $v$ sent to player $w$ in round $i$.*

Clearly, an execution of a distributed protocol can be described by the scenario function. If $\sigma(v, w, i) = \perp$ it means that no message has been sent by $v$ to $w$ in round $i$ or an erroneous message has been sent, treating both cases the same. We say that a scenario $\sigma$ is a $k$-round scenario if and only if

$$\min\{i \in \mathbb{N} \mid \sigma(v, w, r) = \perp,\ \forall v, w \in \mathcal{V},\ \forall r \geq i\} = k$$

**1.5 Definition** (Subscenario). *A $k$-round sub-scenario $\sigma^k$ of scenario $\sigma$ is the restriction $\sigma|_{V \times V \times \{1,\ldots,k\}}$ of the function $\sigma$. Obviously $\sigma^k$ describes the execution of a distributed protocol until a certain round $k$.*

**1.6 Definition** (View). *The view of a player $v$ over sub-scenario $\sigma$ is the function $view_v^\sigma : V \times V \times \mathbb{N} \to \mathcal{M}$ defined by*

$$view_v^\sigma(u, w, i) = \begin{cases} \perp & if\ u, w \neq v. \\ \sigma(u, w, i) & else. \end{cases}$$

*The range of function $view_v^\sigma$ consists of the messages sent or received by $v$ in sub-scenario $\sigma$. We will omit $\sigma$ and simply write $view_v$ when the scenario is implied by the context or does not concern us.*

**1.7 Definition** (Indistinguishable scenarios). *Scenario $\sigma$ is* indistinguishable *from $\sigma'$ with respect to player $v$ ( $\sigma \overset{v}{\sim} \sigma'$) if and only if*

$$view_v^\sigma(u, w, i) = view_v^{\sigma'}(u, w, i),\ \forall u, w \in \mathcal{V} \text{ and } i \in \mathbb{N}$$

Let $\mathcal{S}$ be the family of sub-scenarios implied by a distributed system $G$, $VIEW_v$ the family of $view_v$ functions implied by *scenarios* in $\mathcal{S}$ and $subVIEW_v$ the family of $view_v$ functions implied by sub-scenarios in $\mathcal{S}$. Also let $ResVIEW_v$ be the family of all the restrictions of all the functions which are elements of $subVIEW_v$.

**1.8 Definition** (Decision)**.** *The $decision_v$ of player $v$ is a function*

$$decision_v : VIEW_v \to X$$

*the value that player $v$ decides in scenario $\sigma$ according to its view. We will also use the notation $decision_v(\sigma)$ since a scenario $\sigma$ implies a unique function $view_v^\sigma$ and the simplified notation $decision_v$ when the scenario is implied by the context.*

We will use the Corollary below in our proofs. It's validity is immediately implied by the definitions of indistinguishability of scenarios and the $decision_v$ function.

**1.3 Corollary.** *Given two scenarios $\sigma, \sigma'$ and a player $v$ it holds that,*

$$\sigma \overset{v}{\sim} \sigma' \Rightarrow decision_v(\sigma) = decision_v(\sigma').$$

**1.9 Definition** (Correctness Rule)**.** *A function $R_v : subVIEW_v \to ResVIEW_v$ which produces the value*

$$R_v(view_v^{\sigma^k}) = view_v^{\sigma^{k+1}}|_{\{v\}\times V\times\mathbb{N}}$$

*(i.e. the messages that $v$ is supposed to transmit in the next round).*

**1.10 Definition** (Honest and Corrupted)**.** *With respect to a given correctness rule $R$, a player $v$ is said to be* honest *(or correct) at round $k$ if in round $k$, $v$ sends the messages dictated by $R$ operating in the previous $k-1$ rounds, else we call $v$* corrupted *in round $k$. Player $v$ is an honest player in scenario $\sigma$ (denoted by $v \in \mathcal{H}$) if it is honest at each round of $\sigma$, else $v$ is corrupted in $\sigma$ (denoted by $v \in \mathcal{T}$).*

A **Broadcast protocol** on a class of scenarios $\mathcal{S}$ consists of a correctness rule $R$ (union of possibly distinct correctness rules $R_v$, $\forall v \in \mathcal{V}$ ) and a *decision* function (similarly a union of individual $decision_v$ functions).

# Chapter 2

# Broadcast Protocols

The design of protocols for the Broadcast problem has received a great deal of attention since its introduction in [PSL80]. The attempts on improving the existing protocols has essentially been an effort to improve the trade-off between the resilience, the bit complexity, the round complexity and the local computation complexity of the solutions.

In [PSL80] the authors presented a protocol that solves the problem in $t+1$ rounds whenever $n > 3t$. However, the protocol required the players to send exponentially long messages and perform exponentially many steps of computation. Polynomial-time Broadcast protocols for $n > 3t$ that halt in more than $2t$ (see, e.g. [DFF+82, TPS87] rounds have been known as of 1982. In 1985 Coan [Coa86] presented a family of Broadcast protocols for $n > 4t$ that, for every $d$, halt in $t + t/d$ rounds, and require messages of size $O(n^d)$. However Coan's protocols require exponential local computation. Bar-Noy, Dolev, Dwork and Strong later improved on this result, providing protocols with essentially the same round and communication behavior, but requiring only polynomial computation [BNDDS92]. These protocols thus provide a trade-off between the number of rounds required and the size of messages used, and prove that $2t$ rounds are not necessary for polynomial-time Broadcast protocols. Polynomial- time Broadcast protocols, operating in $t + 1$ rounds for $n = \Omega(t^2)$ where presented by Dolev, Reischuk and Strong in [DRS90]. In 1988 Moses and Waarts presented the first polynomial $(t + 1)$-round protocol with linear resilience: It required only that $n > 8t$, and was later improved to handle $n > 6t$ [MW88]. Berman and Garay presented a polynomial protocol for $n > 4t$ [BG93], which they later improved to handle $n > (3t + \epsilon)t$ for any $\epsilon > 0$ [BG91]. At the cost of requiring more processors $(\Omega(t \log t))$, Coan and Welch developed a polynomial protocol that uses one-bit messages and achieves asymptotically optimal total bit transfer [CW93]. Finally after the long series of

papers improving the trade-off between resiliency and round complexity, Garay and Moses presented the first fully polynomial Broadcast protocol in [GM98]. A partial list of Broadcast protocols and the parameters characterizing them is given in table 2.1.

| Protocol | $n$ | rounds | comm. | comp. |
|---|---|---|---|---|
| [PSL80] | $3t+1$ | $t+1$ | $exp(n)$ | $exp(n)$ |
| [DFF+82, TPS87] | $3t+1$ | $2t+c$ | $poly(n)$ | $poly(n)$ |
| [Coa86] | $4t+1$ | $t+\frac{t}{d}$ | $O(n^d)$ | $\exp(n)$ |
| [DRS90] | $\Omega(t^2)$ | $t+1$ | $poly(n)$ | $poly(n)$ |
| [BNDDS92] | $3t+1$ | $t+\frac{t}{d}$ | $O(n^d)$ | $O(n^d)$ |
| [MW88] | $6t+1$ | $t+1$ | $poly(n)$ | $poly(n)$ |
| [BG89] | $3t+1$ | $t+\frac{t}{d}$ | $O(c^d)$ | $O(c^d)$ |
| [BG93] | $4t+1$ | $t+1$ | $poly(n)$ | $poly(n)$ |
| [CW93] | $\Omega(t\log t)$ | $t+1$ | $poly(n)$ | $poly(n)$ |
| [BG91] | $(3+\epsilon)t$ | $t+1$ | $poly(n)\cdot O(2^{1/\epsilon})$ | $poly(n)\cdot O(2^{1/\epsilon})$ |
| [GM98] | $3t+1$ | $t+1$ | $poly(n)$ | $poly(n)$ |

Table 2.1

## 2.1 Exponential Information Gathering Algorithm

In this section, we present an algorithm for the Broadcast problem based on a strategy known as exponential information gathering (*EIG*). In exponential information gathering algorithms, players send and relay values for several rounds, recording the values they receive along various communication paths in a data structure called an *EIG tree*. At the end, they use a commonly agreed-upon decision rule based on the values recorded in their trees. The EIG algorithm of [BNDDS87], presented here, is a variation of the original Broadcast algorithm of [LSP82] which incorporates the *EIG tree* data structure. The algorithm requires $n > 3t+1$.

### EIG Tree Data Structure

In the EIG Algorithm each player incrementally constructs a tree-based data structure of height at most $t$ (each path from root to leaf contains at most $t+1$ nodes), called the *EIG tree* (Figure 2.1). The nodes of the EIG tree are labeled with player names as follows . The root is labeled $D$, for Dealer. Let $p$ be an internal node

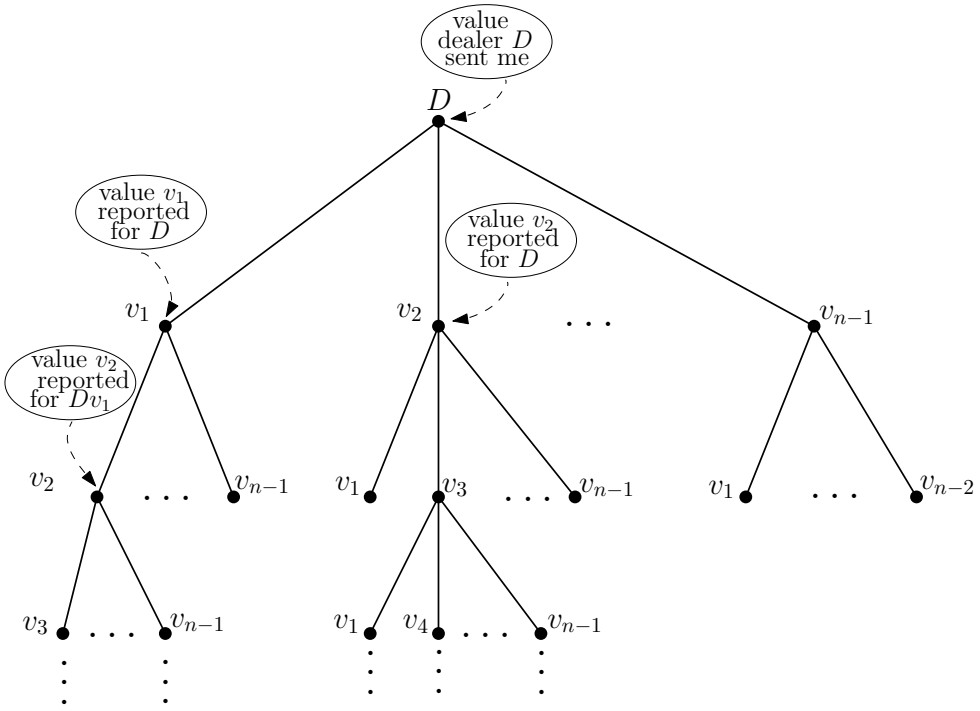



Figure 2.1: The Information Gathering Tree

in the tree. For every player name $v_i$ not labeling an ancestor of $p$, $p$ has exactly one child labeled $v_i$. With this definition no label appears twice in any path from root to leaf in the tree. Henceforth, a *sequence* is an ordered list of at most $t+1$ distinct player names, beginning with $D$. We often refer to a node in the tree by specifying the sequence of player names encountered in traversing the path from the root to the node. Let $a$ be such a sequence. The length of $a$, denoted $|a|$, is the number of names in the sequence. Note that if $|a|$ is an internal node then a has $n - |a| > 2t + 1$ children. The player corresponding to node $a$ is the player whose name labels node $a$, i.e., the last player name in the sequence $a$. The Information Gathering Tree maintained by player $v$ is called $tree_v$. The tree is built one level at a round. For each $1 \leq h \leq t+1$, the EIG tree at the end of round $h$ is called the *round $h$ tree*, and is of height $h-1$. Each player $v$ can store values in the nodes of $tree_v$, we refer to the value stored in the node with label $a$ as $tree_p(a)$ eliminating the subscript when no confusion will arise. The value stored in $tree_v(D)$ is the *preferred value*.

The EIG Algorithm is split into two phases : Information Gathering, and Data Conversion .

**Information Gathering:**

**Round 1**

1. Dealer sends its initial value $x_D$ to the $n-1$ other players and decides on $x_D$.

2. Each player $v$ stores the value $x_D$, received from the Dealer, in node $D$, i.e., $tree_v(D) := x_D$. A special default value of $\perp$ is stored if the Dealer failed to send a legitimate value in $X$.

**Round h**, $2 \leq h \leq t+1$

1. Each $v$ broadcasts the leaves of its round $(h-1)$ tree.

2. Every $v$ adds a new level to its tree, storing at node $D \ldots qr$ the value that $r$ claims to have stored in node $D \ldots q$ in its own $tree_r$. Again, the default value $\perp$ is used if an inappropriate message is received.

   Intuitively, $v$ stores in node $D \ldots qr$ the value that "$r$ says $q$ says $\ldots$ the source said".

**Data Conversion:**

During this phase each player $v$ applies a recursive function to $tree_v$ to obtain a new preferred value. The value obtained by applying the conversion function to the subtree rooted at a node $a$ is called the *converted value for* $a$. The specific data conversion function, $resolve$, used in the EIG algorithm is essentially a recursive majority vote and is defined as follows for all sequences $a$:

$$resolve(a) = \begin{cases} tree(a) & \text{, if } a \text{ is a leaf;} \\ m & \text{, If } m \text{ is the majority of } resolve \text{ applied} \\ & \quad \text{to the children of } a; \\ \perp & \text{, If } a \text{ is not a leaf and no majority exists.} \end{cases}$$

The value obtained by player $v$ in computing $resolve(a)$ is denoted $resolve_v(a)$. We occasionally drop the subscript $v$ when no confusion will arise. Summarizing, we have :

**2.1 Algorithm : *Exponential Information Gathering***

1. Gather information for $t+1$ rounds;
2. Compute the converted value for $D$ using the data conversion function $resolve$;
3. Decide on $resolve(D)$

We now give a proof of correctness for this algorithm . After data conversion, a node $a$ is said to be *common* if each honest player computes the same converted value for $a$. Thus, the Validity and Consistency conditions of the Broadcast problem can be alternatively defined as:

- *Validity:* When the dealer is honest, $resolve_v(D) = tree_v(D)$ for every honest player $v$;
- *Consistency:* Node $D$ is common in every execution, i.e., for all honest $v$ and $w$, $resolve_v(D) = resolve_w(D)$.

Recall that if $v$ is honest, then $tree_v(D)$ is precisely the value received from the dealer during the first round. Thus, the second condition implies that if the dealer is honest then all honest players, including the dealer, decide on the dealer's initial value .

The following lemma asserts that if a sender of a value $x$ is honest, then all honest players will eventually compute the same converted value for the corresponding node.

**2.1 Lemma** (Validity Lemma)**.** *For any $1 \leq h \leq t+1$, consider the $h$-round EIG tree. Let $a = \beta q$ be a sequence of length at most $h$ in which $|\beta| \geq 0$ and $q$ is an honest player. If data conversion is applied to the $h$-round tree, then there is a value $x$ such that $a$ is common with converted value $x$ and for every honest player $v$, $tree_v(a) = x$.*

*Proof.*
Let $v$ be an honest player. Note that since $q$ is honest, $tree_v(a) = tree_q(\beta)$. If $|\beta| = 0$ then $q = D$. In this case, we interpret $tree_s(\beta)$ to be the dealer's initial value.

The lemma is proved by reverse induction on the length of $a$. If $|a| = h$ then, since $a$ is a leaf, $resolve_v(a) = tree_v(a)$ for all honest players $v$. Thus, $a$ is common.

Assume the lemma for sequences of length $k$, where $1 < k \leq h$. Let $a$ be a sequence of length $k-1$. Let $r \notin a$ be an honest player. By induction, $resolve_v(ar) = tree_v(ar)$. Moreover, since $v$, $q$, and $r$ are all honest,

$$tree_v(ar) = tree_r(a) = tree_q(\beta) = tree_v(a).$$

Thus, all but $t$ of the children of $a$ in $tree_v$ have common converted value equal to $tree_v(a)$. However, $a$ is internal it has at least $2t+1$ children, hence $resolve_v(a) = tree_v(a)$. This completes the proof. □

From the Validity Lemma (with $h = t+1$ and $a = D$) we immediately have:

**2.1 Corollary.** *If the dealer is honest then after data conversion $D$ is common and $resolve_v(D) = tree_v(D)$ for all honest players $v$.*

There are at most $t$ corrupted players and every path in the EIG tree is of length $t+1$, so every path from root to leaf contains an honest player. It therefore follows by the Validity Lemma that every path contains a common node, independent of the honesty of the source. When every root-to-leaf path contains a common node we say the EIG tree has a *common frontier*. It remains to show that the existence of a common frontier guarantees consistency. This is immediate from the following lemma.

**2.2 Lemma** (Frontier Lemma)**.** *Let $a$ be a node . If there is a common frontier in the subtree rooted at $a$, then $a$ is common (i.e., $a$ itself constitutes a common frontier of the subtree).*

*Proof.*
To prove the lemma, suppose it failed in some execution of the algorithm and suppose $a$ were a counterexample of maximal length: thus $a$ would not be common but the subtree rooted at $a$ would have a common frontier. If the subtree rooted at a leaf has a common frontier, then the leaf is common. Hence, $a$ cannot be a leaf. If a subtree has a common frontier, either its root is common or the subtree rooted at each child of its root has a common frontier. Hence, by the length maximality of $a$, each of its children is common. But then every honest player computes the same value for $resolve(a)$, and $a$ is common, contradicting the assumption that $a$ is not common . □

The following corollary follows directly from 2.2

**2.2 Corollary.** *If there is a common frontier, then $D$ is common.*

**2.3 Proposition.** *The EIG Algorithm achieves Broadcast in $t+1$ rounds provided that $n \geq 3t+1$.*

*Proof.*
Validity and Consistency are ensured by Corollaries 2.1,2.2 respectivelly. □

**Complexity**

Despite the simplicity of the algorithm, the message size and the amount of local computation required grow exponentially in $t$. More specifically, for any $1 \leq h \leq t+1$, the h-round EIG tree has $O(n^{h-1})$leaves, yielding messages of size $O(n^{h-1})$ in round $h+1$.

As we will see in later chapters the EIG algorithm is optimal in resiliency and round complexity, but is clearly impractical due to its exponential bit complexity.

## 2.2 Reducing the communication cost

Regarding the reduction of the bit complexity of Broadcast, the first communication - efficient $t+1$ round protocol was presented in the breakthrough work of Moses and Waarts [MW88], though their protocol was 1/8-resilient. Their result was improved by Berman, Garay and Perry in [BGP89] where they also presented the first protocol that doesn't require exponential bit complexity and is 1/3-resilient. Many protocols that were given later are based on the latter.

We examine the Consensus protocol of [BGP89] which using the equivalence between the two problems can be easily transformed in a Broadcast protocol with an overhead of 1 extra round. This protocol includes three sub-protocols ( *Weak Consensus, Graded Consensus, King Consensus* ),where each one of them achieves a weaker kind of Consensus. In order to have Consensus we run these three sub-protocols repeatedly.

The first sub-protocol is *Weak Agreement* where player $v_i$ has an input value $x_i \in \{0,1\}$ and decides on an output value $y_i \in \{0,1,\perp\}$. Running Weak Agreement we achieve the following:

**Weak consistency:** If an honest player $v_i$ decides on $y_i \in \{0,1\}$ then every other honest $v_j$ decides on $y_i \in \{y_i, \perp\}$.

**2.2 Protocol : *WeakConsensus*$(\mathbf{x_1}, \ldots \mathbf{x_n}) \to (\mathbf{y_1}, \ldots, \mathbf{y_n})$**

**Input:** Initial values $x_1, \ldots, x_n$ of the $n$ players.

**Output:** Decision values $y_1, \ldots, y_n$ of the $n$ players.

1. `Every` $v_i \in \mathcal{V}$ sends $x_i$ to all $v_j$.
   Let $c_m$ be the copies of a message $m \in \{0, 1\}$ received by player $v_j$ in this round.
2. `Every` $v_j$ computes:
$$y_j = \begin{cases} m & \text{if } c_m \geq n - t \\ \perp & else \end{cases}$$
3. `Every` $v_j \in \mathcal{V}$ returns $y_j$

**2.3 Lemma.** *The above protocol achieves* Weak Consistency *and Validity.*

*Proof.*
*Validity:* Suppose that all honest players have the same input $x$. In step 2 all honest players collect the value $x$ at least $n - t$ times, consequently all honest players receive the value $1-x$ at most $t < n-t$ (since $t < n/3$) and they all decide on $y_i = x$.

*Weak Consistency:* Let $v_i$ be an honest player who computed the output value 0. That means that he received at least $n - t$ times the value 0 after the first step. That means that at least $n - 2t$ honest players sent him this value. Now let $v_j$ be another honest player. It is clear that he also received 0 at least $n - 2t$ times and the value 1 at most $n - n + 2t = 2t < n - t$ times. So he computes either $y_j = 0$ or $y_j = \perp$.

□

The second sub-protocol is *Graded Consensus*. In this protocol every player $v_i$ computes two different values, the output value $y_i$ and the Grade value $g_i \in \{0, 1\}$. The latter value shows the level of consistency achieved, i.e., $g_i = 1$ means that Consistency is achieved and $g_i = 0$ means that it is unsure if Consistency is achieved. More formally:

**Graded Consistency:** If a honest player $v_i$ decides on $y_i \in \{0, 1\}$ with $g_i = 1$ then every other honest $v_j$ decides on $y_j = y_i$.

**2.3 Protocol : *GradedConsensus*$(\mathbf{x_1}, \ldots, \mathbf{x_n}) \to ((\mathbf{y_1}, \mathbf{g_1}), \ldots, (\mathbf{y_n}, \mathbf{g_n}))$**

**Input:** Initial values $x_1, \ldots, x_n$ of the $n$ players.

**Output:** Decision pairs $(y_1, g_1), \ldots, (y_n, g_n)$ of the $n$ players.

1. $(z_1, \ldots, z_n) := WeakConsensus(x_1, \ldots, x_n)$
2. `Every` $v_i \in \mathcal{V}$ sends $z_i$ to all $v_j$.
   Let $c_m$ be the copies of a message $m \in \{0, 1\}$ received by player $v_j$ in this round.
3. `Every` $v_j$ computes:

$$y_j = \begin{cases} 1 & \text{if } c_1 > c_0 \\ 0 & else \end{cases}$$

$$g_j = \begin{cases} 1 & \text{if } c_{y_j} \geq n - t \\ 0 & else \end{cases}$$

4. `Every` $v_j \in \mathcal{V}$ returns $(y_j, g_j)$

**2.4 Lemma.** *The above protocol achieves Graded Consistency and Validity remains.*

*Proof.*
*Validity:* Here Validity has the meaning that if all honest players have the same input $x$, then after the protocol these players have $(x, 1)$ as output. It is clear from the previous sub-protocol that after the first step all honest players compute $z_i = x$, where $x$ is their common initial input. Arguing in the same way as in the Weak Agreement protocol it can be easily shown that Validity remains after the fourth step.

*Graded Consistency:* Let $v_i$ be a honest player who computed $(y_i, 1)$. It is clear that at least $n - 2t$ honest players sent him $z_k = y_i$. Now let $v_j$ be another honest player. It is certain that he also received the value $y_i$ from $n - 2t$ honest players after step 2. The remaining $t + 1$ honest players sent him either $y_i$ or $\perp$ and that due to the Weak Consistency after the first step. He receives, consequently, the value $1 - y_i$ in step 2 at most $t$ times. But $t < n - 2t$ and, therefore $v_j$ computed $y_j = y_i$ as output.

□

The last sub-protocol is *King Consensus*. Here any player $v_k$ is chosen to be the *king*. The purpose of this protocol is that if the king is honest then all honest players decide on the same output, regardless of the king's input.

**King Consistency:** If the king $v_k$ is honest, then all honest players compute the same output $x \in \{0, 1\}$.

---

**2.4 Protocol : KingConsensus$(\mathbf{v_k}, \mathbf{x_1}, \ldots \mathbf{x_n}) \rightarrow (\mathbf{y_1}, \ldots, \mathbf{y_n})$**

---

**Input:** Initial values $x_1, \ldots, x_n$ of the $n$ players and player's $v_k$ id.

**Output:** Decision values $y_1, \ldots, y_n$ of the $n$ players.

1. $((z_1, g_1) \ldots, (z_n, g_n)) := GradedConsensus(x_1, \ldots, x_n)$
2. The king $v_k$ sends $z_k$ to all players.
3. `Every` $v_j$ computes
$$y_j = \begin{cases} z_j & \text{if } g_j = 1 \\ z_k & else \end{cases}$$
4. `Every` $v_j$ returns $y_j$

---

**2.5 Lemma.** *The above sub-protocol achieves King Consistency and Validity remains.*

*Proof.*
*Validity:* If all honest players have the same input $x$, then due to the Graded Consistency of the first step these players have $z_i = x$ with $g_i = 1$. Therefore all honest players decide on $x$ after step 3.

*King Consistency* We assume that the king $v_k$ is uncorrupted. If all honest players computed in step 1 $g_i = 0$, then they all decided on $y_i = z_k$ in step 3. If any honest player computed $g_i = 1$, then, because of the Graded Consistency of step 1, all honest players, $v_k$ included, computed the same $z_i$ , thus they compute the same output $y_i$. □

Due to the King Consistency definition, it is clear that if we ensure that the king is honest then Consensus will be achieved. Therefore we run the KingConsensus protocol $t + 1$ times, each time with a different king:

**2.5 Protocol : *Consensus*$(\mathbf{x_1}, \ldots, \mathbf{x_n}) \rightarrow (\mathbf{y_1}, \ldots, \mathbf{y_n})$**

1. `For` $k := 1$ *to* $t + 1$

$$(x_1, \ldots, x_n) := KingConsensus(v_k, x_1, \ldots x_n)$$

2. `Every` $v_j$ sets $y_j := x_j$
3. `Every` $v_j$ returns $y_j$

Observe that when an honest player is chosen to be the king, by King Consistency all honest players decide on the same output value $v$ which will be their input value for the next round. Due to the fact that the KingConsensus sub-protocol maintains Validity the final decision value of each honest player will remain $v$.

The corresponding Broadcast protocol is now trivial due to the equivalence of the two problems

**2.6 Protocol : *Broadcast*$(\mathbf{x}, \mathbf{D}) \rightarrow (\mathbf{y_1} \ldots, \mathbf{y_n})$**

1. Dealer $D$ sends $x$ to all players
2. $(y_1, \ldots, y_n) = Consensus(x_1, \ldots, x_n)$,
   with $x_i$ the value that player $v_i$ received from the Dealer.
3. `Every` $v_j \in \mathcal{V}$ returns $y_j$

**2.4 Theorem.** *The above protocol achieves Broadcast (Consensus) with resiliency* $n > 3t$, $BC = O(n^2 t)$ *and* $RC = 3t + O(1)$.

*Proof.*
Consistency and Validity are achieved due to the properties of the King Consensus protocol and the fact that we run this protocol $t + 1$ times with $t + 1$ different kings (at least one king is honest).

In each sub-protocol every player sends a bit to every player, which means $n^2$ bits, and we run each sub-protocol $t + 1$ times. This results in $BC = O(n^2 t)$. Moreover every time we run the King Consensus we need 3 rounds,one for each sub-protocol to be completed, which results in total round complexity $RC = 3t + O(1)$. (When

we want Consensus the protocol requires $3t+3$ rounds and when Broadcast $3t+1$, because the sender is assumed to be one of the $n$ players and, thus, we need to run the King Consensus protocol only $t$ times.

□

## 2.3 General Broadcast using Binary Broadcast

In the current thesis we will mainly focus on protocols and proofs concerning the case of Binary Broadcast, i.e., the special case of the problem where the dealer sends and players decide on values from the input space $X = \{0, 1, \perp\}$. Next we will show how the special case is connected with the general form of the Broadcast problem.

We will first show how to use an algorithm that achieves Consensus for inputs $x \in \{0, 1\}$ (Binary Consensus) as a subroutine for solving General Consensus with arbitrary initial values in the input space $X$. We will then extend the result in the case of achieving General Broadcast using a Binary Broadcast algorithm as a subroutine.

### 2.3.1 Achieving General Consensus

For the Consensus problem, the overhead is just 2 extra rounds, $2n^2$ extra messages, and $O(n^2 b)$ bits of communication, where $b = \max_{x \in X} |x|$. This can lead to substantial savings in the total number of bits that need to be communicated, since it is not necessary to send values in $X$, but only binary values, while executing the subroutine. We call the algorithm *TurpinCoan*, after its designers [Coa87]. The algorithm assumes that $n > 3t$. As earlier, we pretend that each player can send messages to itself as well as to the other players.

In the Algorithm 2.7 each player $v \in \mathcal{V}$ has local variables $x, y, z$, and $vote$, where $x$ is initialized to the player's input value and $y, z$ and $vote$ are initialized arbitrarily.

**2.7 Algorithm : *TrupinCoan***

- **Round 1:** $\forall v \in \mathcal{V}$, player $v$ sends its value of $x$ to every $u \in \mathcal{V}$ (including itself). Let $c_m$ be the copies of a message $m \in X$ received by player $v$ in this

round. Then $v$ computes $y$,

$$y = \begin{cases} m & \text{if } c_m \geq n - t \\ \perp & else \end{cases}$$

- **Round 2:** $\forall v \in \mathcal{V}$, player $v$ sends its value of $y$ to every $u \in \mathcal{V}$ (including itself). Let $c_m$ be the copies of a message $m \in X$ received by player $v$ in this round. Then $v$ computes $vote$,

$$vote = \begin{cases} 1 & \text{if } \exists m \in X \text{ } s.t. \text{ } c_m \geq n - t \\ 0 & else \end{cases}$$

  Also $v$ sets $z$ equal to the value $m \neq \perp$ that occurs most often among the messages received by $v$ at this round, with ties broken arbitrarily; if all messages are equal to $\perp$, then $z$ remains undefined.

- **Round r, r $\geq$ 3:** The players run the binary Consensus subroutine using the values of $vote$ as their input values. If player $v$ decides on value 1 in the subroutine and if $z$ is defined, then the final decision of the algorithm is $z$, otherwise it is the default value $x_0$.

---

A key fact about the TrupinCoan algorithm is

**2.6 Lemma.** *There is at most one value $x \in X$ that is sent in round 2 by honest players.*

*Proof.*
Suppose for the sake of contradiction that honest players $v$ *and* $w$ send messages at round 2 containing values $x_v$ and $x_w$ respectively, where $x_v, x_w \in X$, *and* $x_v \neq x_w$. Then $v$ receives at least $n-t$ round 1 messages containing value $x_v$. Since there are at most $t$ corrupted players, and honest players send the same round 1 messages to all players, it must be that $w$ receives at least $n - 2$ messages containing the value $x_v$. Since $n > 3t$, this means $w$ receives at least $t + 1$ messages containing $x_v$.

But also, since $w$ sends $x_w$ in round 2, $w$ receives at least $n - t$ round 1 messages containing $x_w$, for a total of at least $(t + 1) + (n - t) > n$ messages. But the total number of round 1 messages received by $w$ is only $n$, so this is a contradiction.

□

**2.5 Theorem.** *The* TurpinCoan *algorithm solves general Consensus problem when given a Binary Consensus algorithm as a subroutine, if* $n > 3t$.

*Proof.*
Termination is easy to see. To show validity, we must prove that if all honest players start with the same initial value, $x$, then all honest players decide on $x$.

So suppose that all honest players start with $x$. In round 1 every $v \in \mathcal{H}$ successfully broadcasts round 1 messages containing $x$ to all players. Since $|\mathcal{H}| > n - t$, all honest players set their $y$ variables to $x$ at round 1. Then in round 2, each honest player receives at least $n-t$ messages containing $x$, which implies that it sets its $z$ variable to $x$ and its *vote* variable to 1. Since all the honest players use input 1 for the Binary Consensus subroutine, they all decide on value 1 in the subroutine, by the validity condition for the Binary Consensus algorithm. This means that they all decide $x$ in the main algorithm, which shows validity.

Finally, we show consistency. If the subroutine's decision value is 0, then $x_0$ is chosen as the final decision value by all honest players and consistency holds by default.

So assume that the subroutine's decision value is 1. Then by the validity condition for the subroutine, some honest player $v$ must begin the subroutine with $vote_v = 1$. This means that player $v$ receives at least $n-t$ round 2 messages containing some particular value $x \in X$, so since there are at most $t$ corrupted players, $v$ receives at least $n - 2t$ round 2 messages containing $x$ from honest players. Then if $w$ is any honest player, it must be that $w$ also receives at least $n-2t$ round 2 messages containing $x$ from those same honest players. By Lemma 2.6, no value in $X$ other than $x$ is sent by any honest player in round 2. So process $w$ receives no more than $t$ round 2 messages containing values in $X$ other than $x$ (and these must be from corrupted players). Since $n > 3t$, we have $n - 2t > t$, so $x$ is the value that occurs most often in round 2 messages received by $w$. It follows that process $w$ sets $z := x$ in round 2.

Since the subroutine's decision value is 1, this means that $w$ decides on $x$. Since this argument holds for any honest player $w$, consistency holds. □

**Complexity Analysis.** The number of rounds is $r + 2$, where $r$ is the number of rounds used by the Binary Consensus subroutine. The extra communication used by *TurpinCoan*, in addition to that used by the subroutine, is $2n^2$ messages, each of at most $b$ bits, for a total of $O(n^2 b)$ bits.

### 2.3.2 Achieving General Broadcast

The case of achieving General Broadcast using a Basic Binary Broadcast protocol (Definition 1.3) as a subroutine is trivial. We simply use Algorithm 2.7 with the following modifications:

- Insert Round 0 before round 1 in which $\forall v \in \mathcal{V}$, Dealer $D$ sends initial value $x_D$ to $v$, and $v$ stores the received value in its local variable $x$, i.e. $x := x_D$.
- In round $r, r \geq 3$ all the players run the Basic Binary Broadcast subroutine using the values of *vote* as the values received from the Dealer. The final decision is taken in the same manner as in the *TrupinCoan* Algorithm.

In this case, the round and bit complexity remains the same as in the General Consensus algorithm.

In the general case where we want to achieve Broadcast using an arbitrary Binary Broadcast subroutine further modifications are required. Specifically:

- Insert Round 0 before Round 1 in which $\forall v \in \mathcal{V}$, Dealer $D$ sends initial value $x_D$ to $v$, and $v$ stores the received value in its local variable $x$, i.e. $x := x_D$.
- 1. In round $r, r \geq 3$, each $v \in \mathcal{V}$ broadcasts its binary value $vote_v$ using the Binary Broadcast subroutine.
  2. Every $v \in \mathcal{V}$ decides on the value which was broadcasted by the majority of the players breaking ties with respect to the lexicographic order. Obviously all honest players will decide on the same value.
  3. If player $v$ decides on value 1 in the subroutine and if $z$ is defined, then the final decision of the algorithm is $z$, otherwise it is the default value $x_0$.

These further modifications induce an increase in both round and bit complexity of the Algorithm 2.7. Assume that the Binary Broadcast subroutine used has round complexity $R$ and bit complexity $B$. Then the resulting algorithm for General Broadcast has round complexity $n \cdot R + 3$ and bit complexity $n \cdot B + n^2 b$ where $b = \max_{x \in X} |x|$. It can be easily seen that validity, consistency and termination properties are preserved in both modifications.

# Chapter 3

# Limits of Achieving Broadcast

## 3.1 Resiliency

Next we give an example suggesting (though not proving) that three players cannot solve the Broadcast problem, if there is the possibility that even one of them might be corrupted.

Suppose that players $v_0, v_1$ and $v_2$ solve the Broadcast problem, tolerating one fault. Suppose, for example, that they decide at the end of two rounds and that they operate in a particular, constrained manner: at the first round, the Dealer $v_0$ broadcasts its initial value, while in the second round, each player reports to each other what was told to it in the first round by the Dealer. We consider the $view(v_1)$ of the honest player $v_1$.

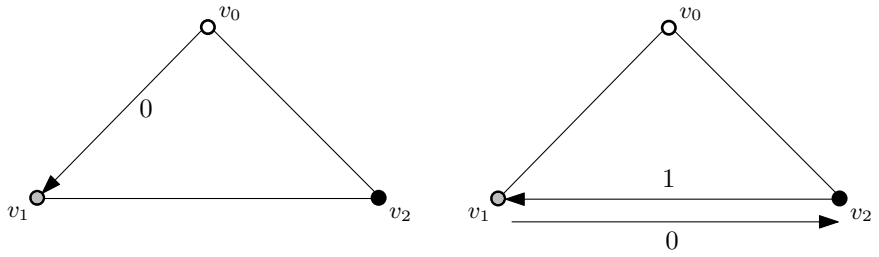



Figure 3.1: View of player $v_1$

Assume that player $v_1$, as shown in Figure 3.1, receives value 0 from the Dealer $v_0$ in the first round and value 1 from player $v_2$ in the second round whereas being honest, he sends value 0 to $v_2$ in the second round. Player $v_1$, knowing that at most one of the $v_0, v_2$ is corrupted (the problem is vacuous for the case of $n < t+2$) has to decide on a value that satisfies both conditions of the Broadcast problem.

Considering $view(v_1)$ there are two distinct scenarios $\sigma_1$ and $\sigma_2$ s.t. $\sigma_1 \overset{v_1}{\sim} \sigma_2$ (indistinguishable with respect to $v_1$).

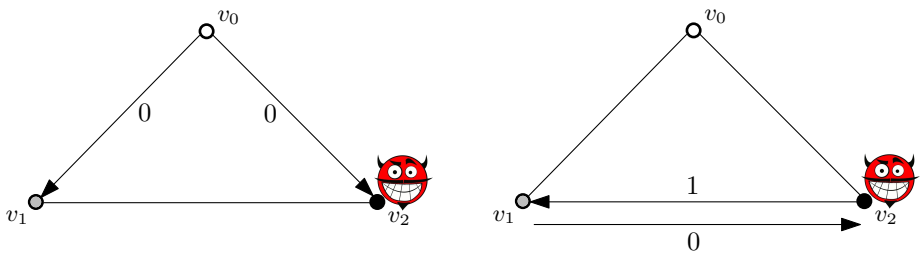



Figure 3.2: Scenario $\sigma_1$
Player $v_2$ is corrupted and reports the false value 1 to $v_1$ in round 2.

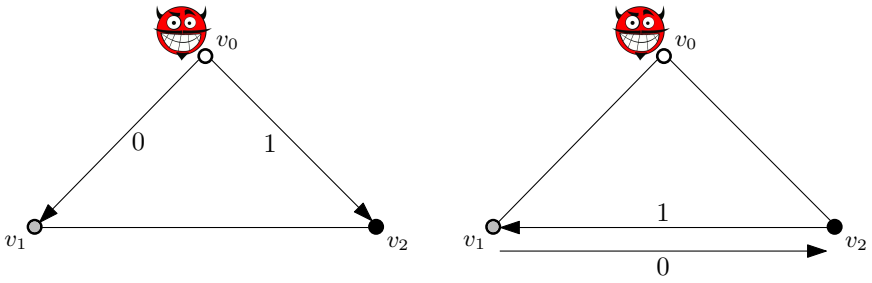



Figure 3.3: Scenario $\sigma_2$
Dealer $v_0$ is corrupted and sends value 1 to player $v_2$ in round 1.

If $v_1$ decides on 1 and scenario scenario $\sigma_1$ holds then the validity condition of Broadcast is violated, thus $v_1$ should decide on 0. Assume that the actual scenario is $\sigma_2$; therefore $v_2$ as an honest player faces the same problem by symmetry: he receives 1 from the dealer and 0 from player $v_1$. The above arguments imply that $v_2$ will decide on 1 in order to avoid violation of the validity condition. These forced decisions contradict the consistency condition of Broadcast. For each decision that $v_1$ can make there is a scenario in which Broadcast is not achieved, which contradicts the assumption that players $v_0, v_1$ and $v_2$ solve the Broadcast problem.

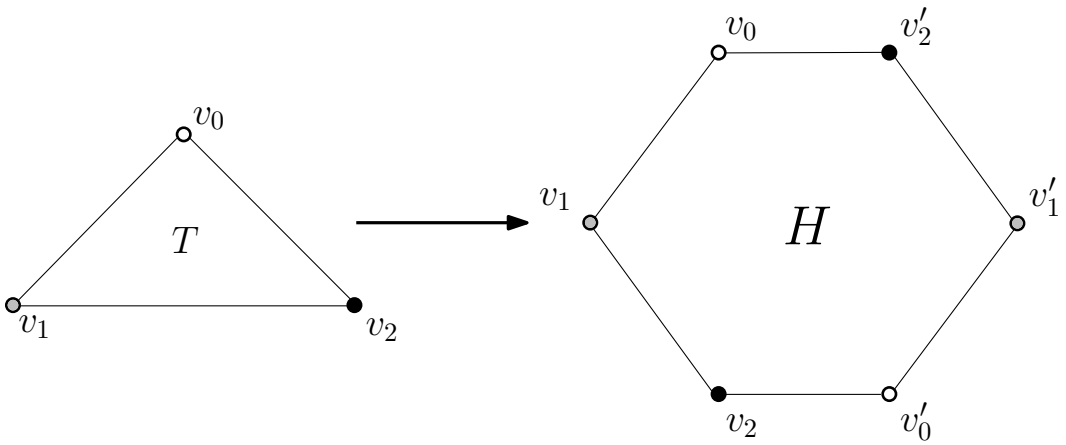



Figure 3.4: Combining two copies of $T$ to get $H$

We have shown that no algorithm of this particularly simple form can achieve Broadcast.

This example does not constitute a proof that three players cannot achieve Broadcast with the possibility of a single fault. This is because the argument presupposes that the algorithm uses only two rounds and sends particular types of messages. But it is possible to extend the example to more rounds and arbitrary types of messages. In fact, the ideas can be extended to show that $n > 3t$ players are needed to achieve Broadcast in the presence of $t$ faults.

**3.1 Lemma.** *Three players cannot solve the Broadcast problem in the presence of one fault ($n = 3$ and $t = 1$).*

*Proof.*
By contradiction. Assume there is a three-player algorithm $A$ that solves the Broadcast problem for the three players $v_0, v_1$ and $v_2$ (arranged in the complete network system $T$, as required for Broadcast), even if one of these three may be faulty. We assume that $v_0$ is the dealer with initial value $x_0$. We construct a new network system $H$ using two copies of $T$ and show that $H$ must exhibit contradictory behavior. It follows that the assumed algorithm $A$ cannot exist.

Specifically, we take two copies of each player in $T$ and configure them into a single hexagonal system $H$. For every player, we build an identical copy of it. Namely, for $k \in \{0, 1, 2\}$, we build player $v'_k = v_{k+3}$ , which is identical to player $v_k$. We connect player $v_{k \mod 6}$ with players $v_{(k+1) \mod 6}$ and $v_{(k-1) \mod 6}$, $\forall 0 \leq k \leq 5$. The arrangement is shown in Figure 3.4.

In $H$, we do not assume that the players know the entire (hexagonal) network graph, but rather that each player just has local names for its neighbors. For

example, in $H$, player $v_0$ knows that it has two neighbors, which it knows by the names $v_1$ and $v_2$, even though one of them is really $v_2'$. In particular, notice that the network system $H$ topologically appears to each player just like the network $T$.

*Claim.* In the new system $H$ and without the presence of an adversary, for every pair of adjacent players $v_k$ and $v_{k+1 \mod 6}$ their view is indistinguishable from their view as two players $v_{k \mod 3}$ and $v_{(k+1) \mod 3}$ in the original system $T$ with respect to an adversary who corrupts the remaining player $v_{(k+2) \mod 3}$ in an admissible way. That is, $\forall \sigma_H$ scenario of $H$ and $\forall k \in \{0, \cdots 5\}, \exists \sigma_T$ scenario of $T$ in which $v_{(k+2) \mod 3}$ is corrupted s.t.

$$\sigma_H \overset{v_k}{\sim} \sigma_T \text{ and } \sigma_H \overset{v_{k+1 \mod 6}}{\sim} \sigma_T$$

*Proof of Claim.* With respect to the players $v_0, v_1$, $v_2$ is "split" into two different copies, $v_2, v_2'$, where $v_0$ is connected with $v_2'$ and $v_1$ is connected with $v_2$. By assumption, when running this system (every player executes algorithm $A$), the players $v_0$ and $v_1$ achieve Broadcast independently from the behavior of the players $v_2$ and $v_2'$. Furthermore, by arranging the six players in $H$, this "splitting" is simultaneously achieved with respect to every pair $v_k$ and $v_{(k+1) \mod 6}$. Hence, for every such pair, their joint view is indistinguishable from their view (as players $v_{k \mod 3}$ and $v_{(k+1) \mod 3}$) in the original system where the adversary corrupts $v_{(k+2) \mod 3}$ by simply simulating all the remaining players of the new system. For example, with respect to pair $(v_0, v_1)$ the corresponding adversary strategy for the original system is to corrupt $v_2$ , simulate the correct players $v_2, v_0', v_1', v_2'$ the new system, and make $v_2$ behave to $v_1$ like $v_2$ in the new system and to $v_0$ like $v_2'$ in the new system — in other words, the adversary simulates the subsystem encircled by the dashed line in Figure 3.5

The new system involves two players of the type corresponding to the Dealer, namely, $v_0$ and $v_0'$, and these are the only players that enter an input. Let now $v_0$ and $v_0'$ be initialized with different inputs, i.e., assume that $v_0$ has input $x_0 \in \{0, 1\}$ and that $v_0'$ has input $x_0' = 1 - x_0$. We study the example where $x_0 = 0$ and $x_0' = 1$ and $\sigma_H$ is resulting scenario of $H$. System $H$ is not required to exhibit any special type of behavior. However, note that $H$ with any particular input assignment (initial values of $v_0, v_0'$) does exhibit some well-defined behavior. We will obtain a contradiction by showing that, for the particular input assignment indicated above, no such well-defined behavior is possible.

We first consider the scenario $\sigma_H$ from the point of view of players $v_0$ and $v_1$. To players $v_0, v_1$, it appears as if they are running in the triangle system $T$, in an scenario $\sigma_{T_2}$ in which player $v_2$ is corrupted. Since $\sigma_{T_2}$ is a scenario of $T$ in which only player $v_2$ is corrupted and Dealer $v_0$ broadcasts the value 0, and since $T$ is

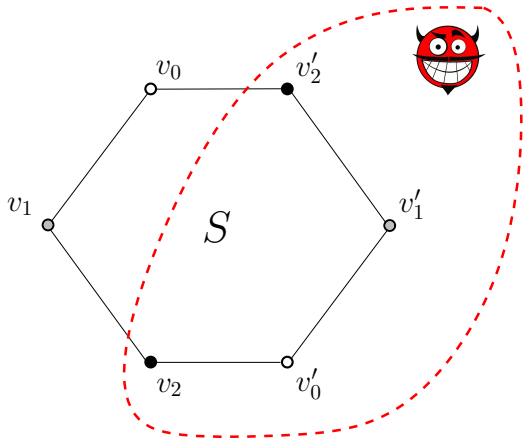



Figure 3.5

assumed to solve the Broadcast problem, the validity condition implies that player $v_1$ must decide on value 0. Finally since $\sigma_H$ is indistinguishable from $\sigma_{T_2}$ to $v_0, v_1$, player $v_1$ decides on 0 in $\sigma_H$ as well. Namely the above Claim and the validity condition implies:

$$\sigma_H \overset{v_0}{\sim} \sigma_{T_2} \text{ and } \sigma_H \overset{v_1}{\sim} \sigma_{T_2} \Rightarrow decision_{v_1} = 0 \tag{1}$$

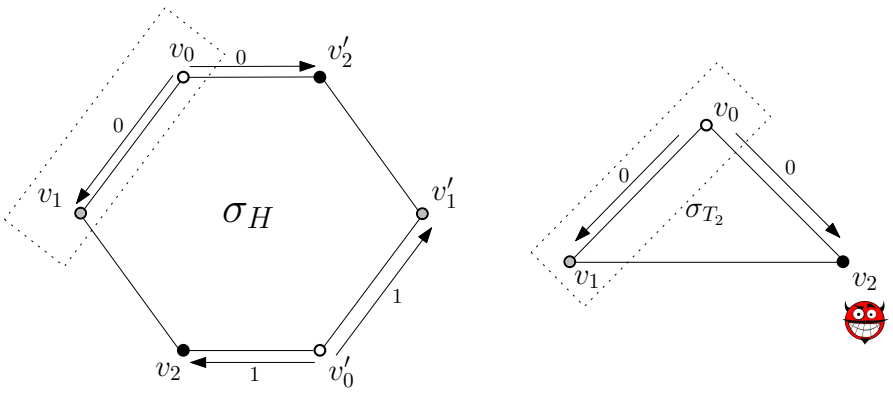



Figure 3.6: Scenarios $\sigma_H$ and $\sigma_{T_2}$ are indistinguishable to $v_0$ and $v_1$

.

Similarly, for the pair of players $v_0', v_2$ and the scenario $\sigma_{T_1}$ of $T$ in which player $v_1$ is corrupted, implied by the above Claim and the validity condition of Broadcast we have,

$$\sigma_H \overset{v_0'}{\sim} \sigma_{T_1} \text{ and } \sigma_H \overset{v_2}{\sim} \sigma_{T_1} \Rightarrow decision_{v_2} = 1 \tag{2}$$

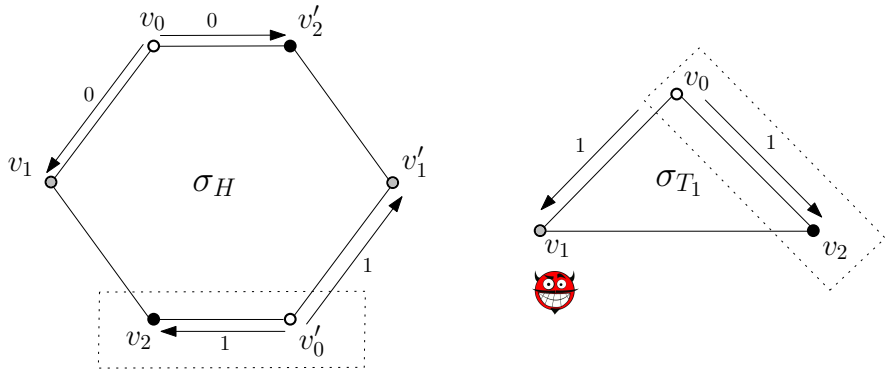



Figure 3.7: Scenarios $\sigma_H$ and $\sigma_{T_1}$ are indistinguishable to $v_0'$ and $v_2$

Finally, for the pair of players $v_1, v_2$ and the scenario $\sigma_{T_0}$ of $T$ in which player $v_0$ is corrupted, implied by the above Claim and the consistency condition of Broadcast we have,

$$\sigma_H \overset{v_1}{\sim} \sigma_{T_0} \text{ and } \sigma_H \overset{v_2}{\sim} \sigma_{T_0} \Rightarrow decision_{v_1} = decision_{v_2} \tag{3}$$

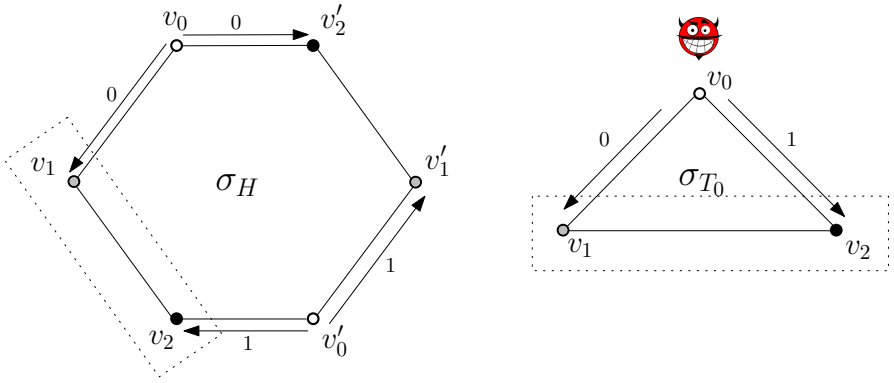



Figure 3.8: Scenarios $\sigma_H$ and $\sigma_{T_0}$ are indistinguishable to $v_1$ and $v_2$

Obviously relations (1), (2) and (3) yield a contradiction. □

**3.1 Theorem.** *There is no solution to the Broadcast problem for $n$ players in the presence of $t$ corrupted players, if $3 \leq n \leq 3t$*

*Proof.*
Assume for the sake of contradiction that there is a Broadcast protocol $A$ for $n$ players $v_0, \cdots, v_{n-1}$, with dealer $v_0$, that tolerates $t \geq n/3$ corrupted players. We show how to transform $A$ into a Broadcast protocol $B$ for three players, $v_0, v_1, v_2$, tolerating one corrupted player. Let $\mathcal{V} = \{v_0, \cdots, v_{n-1}\}$ be the set of players and

$\mathcal{V}_0 \cup \mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ be a partition of $\mathcal{V}$ s.t. for each set $\mathcal{V}_i$, it holds that $1 \leq |\mathcal{V}_i| \leq t$. Assume wlog, that player $v_i \in \mathcal{V}_i$. We let each player $v_i$ in $B$ simulate the behavior of every player $v \in \mathcal{V}_i$ in protocol $A$, as follows.

**Protocol B:** Player $v_0$ is the dealer in protocol $B$. Each player $v_i, i \in \{0, 1, 2\}$ simulates the steps of all the players in $\mathcal{V}_i$ as well as the messages between pairs of players in $\mathcal{V}_i$. For every message $m$ sent in protocol $A$ from $v \in \mathcal{V}_i$ to $u \in \mathcal{V}_j$ with $i \neq j$, $m$ is sent along with the identities of $v, u$ from $v_i$ to $v_j$. If any simulated player $v \in \mathcal{V}_i$ decides on a value $m$, then $v_i$ decides on the value $m$. (If there is more than one such value, then $v_i$ can choose any such value.)

We show that $B$ correctly achieves Broadcast for three players. Designate the corrupted players of $A$ to be exactly those that are simulated by corrupted players of $B$. For any scenario $\sigma_B$ of $B$ with at most one corrupted player, let $\sigma_A$ be the simulated scenario of $A$. Since each player of $B$ simulates at most $t$ players of $A$, there are at most $t$ corrupted players in $\sigma_A$. Since $A$ is assumed to be a Broadcast protocol for $n$ players that tolerates at most $t$ corrupted players, the usual validity, consistency , and termination conditions for Broadcast hold in $\sigma_A$. We argue that these conditions carry over to $\sigma_B$.

- **Termination**: Let $v_i$ be a correct player of $B$. Then $v_i$ simulates at least one player, $v \in \mathcal{V}_i$, of $A$, and $v$ must be correct since $v_i$ is. The termination condition for $\sigma_A$ implies that $v$ must eventually decide; as soon as it does so, $v_i$ decides (if it has not already done so).

- **Validity**: If dealer $v_0$ of $B$ is honest with initial value $m$ then the same $v_0$ dealer of $A$ acts honestly broadcasting $m$. Validity for $\sigma_A$ implies that $m$ is the only decision value for an honest player in $\sigma_A$. Then $m$ is the only decision value for an honest player in $\sigma_B$.

- **Consistency**: Suppose that $v_i$ and $v_j$ are honest players of $B$ Then they simulate only honest players of $A$. Agreement for $\sigma_A$ implies that all of these simulated players agree, so $v_i$ and $v_j$ also agree.

We conclude that $B$ achieves Broadcast for three players, tolerating one corruption. But this contradicts Lemma 3.1

□

**Observation.** Due to the equivalence of Consensus and Broadcast problems, Theorem 3.1 trivially implies the same resiliency bound for the Consensus problem.

## 3.2 Bit Complexity

**3.2 Theorem.** *Every Broadcast protocol which handles up to t corruptions ($t < n-1$), requires at least $n(t+1)/4$ messages to be sent.*

*Proof.*

Let $\sigma_0$ be the scenario in which all players are honest and the dealer transmits the value 0, and $\sigma_1$ the one in which all are honest and the dealer transmits value 1. Let $A(v)$ denote the set of all players that either receive messages from $v$ or send messages to $v$ in at least one of the to scenarios, that is

$$A(v) = \{u \in \mathcal{V} \mid \exists\ i \in \mathbb{N},\ \exists j \in \{0,1,\}\ s.t.\ \sigma_j(v,u,i) \neq \emptyset\ or\ \sigma_j(u,v,i) \neq \emptyset\}$$

Assume that there is a player $v$ with $A(v)$ containing less than $t+1$ players. Then let $\sigma'$ be the scenario $\sigma_1$ modified in a way that all players in $A(v)$ behave towards $v$ as in $\sigma_0$ and towards the rest players as in $\sigma_1$; all the other players behave as in $\sigma_1$. Observe that this is possible for the set $A(v)$ of corrupted players because it is the set of all the players communicating with $v$ in both scenarios $\sigma_0, \sigma_1$.

It is obvious now, that in $\sigma'$ player $v$ has the same view as in $\sigma_0$, while every other honest player $u$ has the same view as in $\sigma_1$ . Namely we have that

$$\sigma' \overset{v}{\sim} \sigma_0 \Rightarrow decision_v(\sigma') = 0$$
$$\sigma' \overset{u}{\sim} \sigma_1 \Rightarrow decision_u(\sigma') = 1,\ \ \forall u \in \{\mathcal{H} \setminus \{v\}\}$$

There is such a player $u \in \{\mathcal{H} \setminus \{v\}\}$ because $t < n-1$. The above imply that consistency of Broadcast is violated which leads to a contradiction. Hence, $|A(v)| \geq t+1, \forall v \in \mathcal{V}$. Consequently, each player "exchanges" in both scenarios $\sigma_0$ and $\sigma_1$ at least $t+1$. If we sum over all players we have a minimum of $n(t+1)/2$ that are required to be send in both scenarios, hence, there is a scenario $\sigma_0$ or $\sigma_1$ which requires at least $n(t+1)/4$ messages to be sent.

□

Theorem 3.2 gives a lower bound on the number of messages needed to be exchanged in a Broadcast protocol or else the *Message Complexity*. It directly implies that the *Bit Complexity* of Broadcast is at least $n(t+1)/4$ because messages consist of at least 1 bit. These bounds also hold for the minimum number of signatures required to be sent in the cryptographic model as proved in [DR85].

**3.3 Corollary.** *The* Bit Complexity *of any Broadcast protocol is at least* $n(t+1)/4$.

## 3.3 Round Complexity

Now, we are going to see, that deterministic Broadcast can be achieved after more than $t$ rounds. The first proof was given by Fischer and Lynch in [FL82] and it holds for the case of unauthenticated messages. However, Dolev and Strong, proved in [DS83] that, no matter what the context of the messages is , Broadcast can be achieved only after $t+1$ rounds. The latter result is very important for the cryptographic model, because it shows that we cannot expect a faster protocol, in other words $t+1$ rounds is a lower bound for any deterministic protocol.

**3.4 Theorem.** *Any Broadcast protocol, which tolerates up to* $t$ *corrupted players, cannot be achieved in fewer than* $t+1$ *rounds, provided that* $t < n-1$.

*Proof.*
We assume that there is a protocol achieving Broadcast in at most $t$ rounds. Let $C$ denote the class of $t$-round scenarios with $n$ players, where $t$ of them are corrupted and have a *critical sequence* $c_i$ s.t. all corrupted players appear on the sequence any corrupted player starts to act in an incorrect way at or after the round corresponding to its position in $c_i$. (i.e. player $v = c_i$, which is faulty, acts honestly until the $(i-1)$-th round). In other words the set of faulty players can only increase by one per round but of course all players take part in the procedure. Note, also, that in $C$ belong also scenarios with all players honest.

We define an equivalence relation on scenarios in $C$ by saying $\sigma$ and $\sigma'$ are equivalent ($\sigma \equiv \sigma'$), if for any correct players $v$ in $\sigma$ and $w$ in $\sigma'$, $decision_v(\sigma) = decision_w(\sigma')$ holds. It is clear that, since there are scenarios, where the players decide on 0 and scenarios where players decide on 1, not all scenarios in $C$ are equivalent. However, we will show that this is not true. Using this contradiction we will prove the theorem.

A player $v$ is called *hidden* at round $k$ in scenario $\sigma$ if it does not send any messages in round $k$ or later, i.e. $\sigma(v,u,i) = \perp,\ \forall u \in \mathcal{V},\ \forall i \geq k$. Using induction we will prove the following Claim,

*Claim.* Let $v$ be a player of scenario $\sigma \in C$ then :

(a) There is a scenario $\sigma' \in C$ with $\sigma' \equiv \sigma$, which is identical to $\sigma$ through round $k$ except from the messages sent by $v$, with $v$ correct correct in round $k$ and all the players correct after round $k$;

(b) If all other players are correct at round $k$, then there is a scenario $\sigma' \in C$ equivalent to $\sigma$, which is identical to $\sigma$ through round $k$ except from the messages sent by $v$, with $v$ hidden at round $k$ and all the other players correct after round $k$;

These two properties imply that any two scenarios, which have all processors correct at the first round, except from the dealer (who may be corrupted or may not) are equivalent. The reason is that there is an equivalent scenario, for both, with the dealer hidden and all the other players correct during the whole procedure. Since all processors behave correctly in the whole procedure, the initial two scenarios must, also, be equivalent. At last, due to the definition of $C$, where all scenarios which belong to $C$ can have at most one corrupted player (the dealer) in the first round, the above properties imply that $C$ is a single equivalence class, which is a contradiction, because there are scenarios, where all correct players decide on 0 and scenarios, where all correct players decide on 1.

Note that if a player is hidden at round $k$, then changing the messages he receives in round $k$ cannot affect the views of other players.

By correcting a message of a player we mean that we change the message to be compatible with a given correctness rule $R$. In short, we will show by induction that we can correct any player at any round or hide any player if all the others are correct in this round, and the resulting scenario will be in $C$ and equivalent to the initial scenario. The changes we are going to do will only be to messages transmitted by the particular player in a certain round and to messages at later rounds.

*Proof of Claim.*

Let $k = t$

(a) Let $v$ be a corrupted player at round $k$ of scenario $\sigma \in C$ . We observe that if we correct one message of $v$ at a time, then there is always a correct player who has the same view as before because $t < n - 1$. Thus every individual change/correction preserves equivalence with $\sigma$. This means that after all the changes the resulting scenario $\sigma'$ is equivalent to $\sigma$. The correct players remain correct and thus $\sigma' \in C$. Since all changes were only made to the messages sent by $v$, in the final result $\sigma'$, $v$ is correct and all other players are trivially correct after the last round.

(b) Let $v$ be a player at round $k$ of scenario $\sigma \in C$ and let all other players at round $k$ be correct. Proceeding as in (a), we change every message $e$ sent by $v$ to $e = \perp$ (remove message $e$), one at a time. Here we may change $v$ from correct to corrupted but since there were no other corrupted players in round

$k$ membership in $C$ is preserved (by replacing the $k$-th position of the critical sequence with $v$, i.e. $c_k = v$). The rest of the argument is the same as in (a).

We assume that the two properties (a) and (b) hold for all rounds after $k$ and we will prove that they hold for round $k$ as well.

(a) Assume that $v$ is a corrupted player at round $k$ in history $\sigma \in C$. The following steps will preserve membership in $C$, equivalence to $\sigma$ and change only messages sent by $v$ in round $k$ and messages at later rounds.

1. Correct all players after round $k$, which we can do and still remain in $C$, because of induction hypothesis (a);
2. While there are still incorrect messages sent by $v$ in round $k$ do
   i. Replace $c_{k+1} = s$, where $s$ is a receiver of an incorrect message $e$ of $v$;
   ii. Hide $s$ at round $k+1$, which can be done and still remain in $C$, because of induction hypothesis (b);
   iii. Correct message $e$, which can be done and still remain in $C$, because some correct player will still have the same view as before the change;
   iv. Correct all players at round $k+1$ (induction hypothesis (a));

   End

Since we have corrected $v$ and all other players after round $k$ with "legal" changes the resulting scenario $\sigma'$ belongs to $C$ and is equivalent to $\sigma$.

(b) Assume that in scenario $\sigma \in C$ all players are correct at round $k$. Let $v$ be a player. Then:

1. Correct all players at round $k+1$, which we can do and still remain in $C$, because of induction hypothesis (a);
2. Replace $c_k = v$;
3. While there are messages sent by $v$ do:
   i. Replace $c_{k+1} = s$, where $s$ is a receiver of a message $e$ of $v$;
   ii. Hide $s$ at round $k+1$, which we can do and still remain in $C$, because of induction hypothesis (b);
   iii. Remove $e$ (set $e = \perp$), which we can do and still remain in $C$, because some correct player will still have the same view as before the change;
   iv. Correct all players after round $k$, which we can do and still remain in $C$, because of induction hypothesis (a);

   End

4. Hide the player with label $v$ at round $k+1$, which we can do and still remain in $C$, because of induction hypothesis (b);

   Since we have hidden $v$ and corrected all other players after round $k$, with 'legal' changes the resulting scenario $\sigma'$ belongs to $C$ and is equivalent to $\sigma$ .

Following this procedure we have managed to show that all scenarios in $C$ are equivalent to any scenario, with hidden dealer and all the other players correct. Consequently, $C$ is a single equivalence class, which is a contradiction to the definition of $C$ . Thus, there is no deterministic Broadcast protocol with less than $t+1$ rounds.

□

# Chapter 4

# Broadcast in Generic and Wireless Networks

So far, we have considered the Broadcast problem only in complete graphs. For complete graphs with $n$ nodes, we showed in Theorem 3.1 that Broadcast can be achieved if and only if $n > 3t$. In this section, we consider the Broadcast problem in generic network graphs. We characterize exactly the topological restriction under which the problem is solvable by giving a lower bound on the connectivity of a graph. We present protocols that achieve Broadcast under this restriction, thus proving that the lower bound is tight. Moreover we present studies on stronger topological restrictions under which more efficient Broadcast protocols can be composed.

## 4.1 Connectivity Lower Bound

**Observation.** First, if the network graph is a tree with $n$ nodes where $n \geq 3$, we cannot hope to achieve Broadcast even in the presence of one corrupted player, for any corrupted player that is not a leaf could essentially "disconnect" the players in one part of the tree from the processes in another. The honest players in different components would not even be able to communicate reliably, much less reach agreement. Similarly, it should be plausible that if $t$ nodes can disconnect the graph, then achieving Broadcast is impossible in the presence of $t$ corrupted players.

To formalize this intuition, we use the following notion from graph theory. The connectivity of a graph $G$, $conn(G)$, is defined to be the minimum number of

nodes whose removal results in either a disconnected graph or a trivial 1-node graph. Graph $G$ is said to be $c$-connected if $conn(G) \geq c$. We use a classical theorem of graph theory known as Menger's Theorem.

**4.1 Theorem** (Menger's Theorem)**.** *A graph $G$ is $c$-connected if and only if every pair of nodes in $G$ is connected by at least $c$ node-disjoint paths.*

We also denote a graph by $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges of the form $(v, u)$ included in the graph $G$. Now we can characterize those graphs in which it is possible to achieve Broadcast in presence of a given number of corrupted players $t$. The characterization is in terms of both the number of nodes $n$ in the graph and the connectivity $conn(G)$. The proof of the impossibility part of the characterization uses methods similar to those used in Section 3.1 to prove the lower bound for the number of corrupted players.

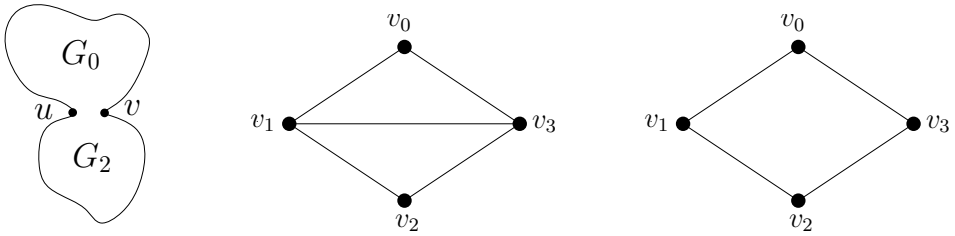
**4.2 Theorem.** *The Broadcast problem can be solved in an $n$-node network graph $G$, tolerating $t$ corrupted players, if and only if both the following hold:*

1. $n > 3t$
2. $conn(G) > 2t$

*Proof.*
The necessity of the resiliency condition ($n > 3t$) is obvious. Simply assume that there is an algorithm which achieves Broadcast in arbitrary networks (not necessarily complete) even if $n \leq 3t$. Then this algorithm could also be run in an n-node complete graph contradicting Theorem 3.1.

We next prove the sufficiency of both conditions, namely that achieving Broadcast is possible if $n > 3t$ and $conn(G) > 2t$. Since $G$ is $2t + 1$-connected, Menger's Theorem, Theorem 4.1, implies that there are at least $2t + 1$ node-disjoint paths between any two nodes in $G$. It is possible to implement reliable communication between any pair of honest players, $v_i$ and $v_j$, by having $v_i$ send a message along $2t + 1$ paths between itself and $v_j$. Since there are at most $t$ corrupted players, the messages received by $v_j$ along a majority of these paths must be correct. Once we have reliable communication between all pairs of correct players, we can solve Byzantine agreement just by simulating any algorithm that solves the problem in an n-node complete graph since $n > 3t$. Implementation of the above idea is explicitly given in Section 4.2.

We now show that achieving Broadcast is possible only if $conn(G) > 2t$. For simplicity we argue the case where $t = 1$.




(a) Network $G'$ (b) Network $G$ (c) Network $G''$

Figure 4.1: Networks with connectivity 2.

Assume there is a network $G$ with $conn(G) \leq 2$, in which Broadcast can be achieved in the presence of one corrupted player, using algorithm $A$. Then there are two nodes in $G$ that either disconnect $G$ or reduce it to one node. But if they reduce it to one node, it means that $n = 3, t = 1$ in which case Broadcast is impossible. So we can assume that the two nodes disconnect $G$ into subgraphs $G_1, G_2$ as shown in Figure 4.1(a).

For simplicity we study the case where the network $G$ is of the form shown in Figure 4.1(b). In the general case nodes $v_0, v_2$ might be replaced by arbitrary connected subgraphs and there might be several edges between each of processes $v_1$ and $v_3$ and each of the two connected subgraphs but the proof is similar to the one given. Also the link between $v_1$ and $v_3$ could also be missing, but impossibility of achieving Broadcast in $G = (\mathcal{V}, \mathcal{E})$ clearly implies impossibility in the subgraph $G'' = \big(\mathcal{V}, \mathcal{E} \setminus (v_1, v_3)\big)$, due to the fact that an algorithm which solves the Broadcast problem in $G''$ could also be used to solve the problem in $G$.

We construct a system $S$ by combining two copies of system $G$ (meaning network $G$ executing algorithm $A$ in a distributed manner). As in the proof of Lemma 3.1, $S$ with the given input assignment does exhibit some well-defined behavior. Again, we will obtain a contradiction by showing that no such behavior is possible.

Specifically, for every player, we build an identical copy of it. Namely, for $k \in \{0, 1, 2, 3\}$, we build player $v'_k = v_{k+4}$ , which is identical to player $v_k$.We connect player $v_{k \mod 8}$ with players $v_{(k+1) \mod 8}$ and $v_{(k-1) \mod 8}$, $\forall 0 \leq k \leq 7$. We also connect nodes $v_1, v'_1$ with $v_3, v'_3$ respectively. We assume that $v_1$ is the dealer in system $G$. The proof is similar in the case of any of $v_0, v_2, v_3$ being the dealer. Suppose, wlog that $v_1$ has input $x_1 = 0$ and $v'_1$ has input $x'_1 = 1 - x_0 = 1$; let $\sigma_S$ be the resulting scenario of $S$. The arrangement is shown in Figure 4.2.

We first consider the scenario $\sigma_S$ from the point of view of players $v_0, v_1$ and $v_2$. To players $v_0, v_1, v_2$, it appears as if they are running in system $G$, in an scenario $\sigma_{G_3}$ in which player $v_3$ is corrupted (Figure 4.3). Since $\sigma_{G_3}$ is a scenario of $G$ in

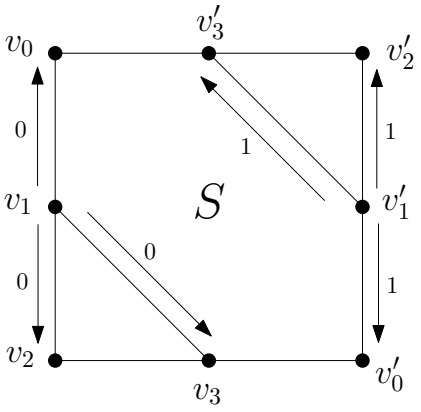



Figure 4.2: System S

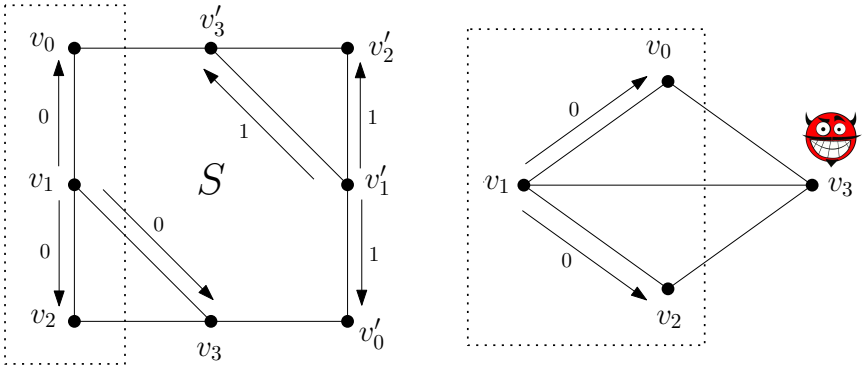



Figure 4.3: Scenarios $\sigma_S$ and $\sigma_{G_3}$ are indistinguishable to $v_0, v_1$ and $v_2$

which only player $v_3$ is corrupted and dealer $v_1$ broadcasts the value 0, and since $G$ is assumed to solve the Broadcast problem, the validity condition implies that player $v_2$ must decide on value 0. Finally since $\sigma_S$ is indistinguishable from $\sigma_{G_3}$ to $v_0, v_1, v_2$ (with reasoning similar to the claim of Lemma 3.1), player $v_2$ decides on 0 in $\sigma_S$ as well. Namely the validity condition of Broadcast implies:

$$\sigma_S \overset{v_0}{\sim} \sigma_{G_3} \text{ and } \sigma_S \overset{v_1}{\sim} \sigma_{G_3} \text{ and } \sigma_S \overset{v_2}{\sim} \sigma_{G_3} \Rightarrow decision(v_2) = 0 \tag{1}$$

Similarly, for the players $v'_0, v'_1, v'_2$ and the scenario $\sigma_{G_3}$ of $G$ in which player $v_3$ is corrupted (Figure 4.4), implied by the validity condition of Broadcast we have,

$$\sigma_S \overset{v'_0}{\sim} \sigma_{G_3} \text{ and } \sigma_S \overset{v'_1}{\sim} \sigma_{G_3} \text{ and } \sigma_S \overset{v'_2}{\sim} \sigma_{G_3} \Rightarrow decision(v'_0) = 1 \tag{2}$$

Finally, for the players $v_2, v_3, v'_0$ and the scenario $\sigma_{G_1}$ of $G$ in which the dealer $v_1$ is corrupted (Figure 4.5), implied by the consistency condition of Broadcast we have,

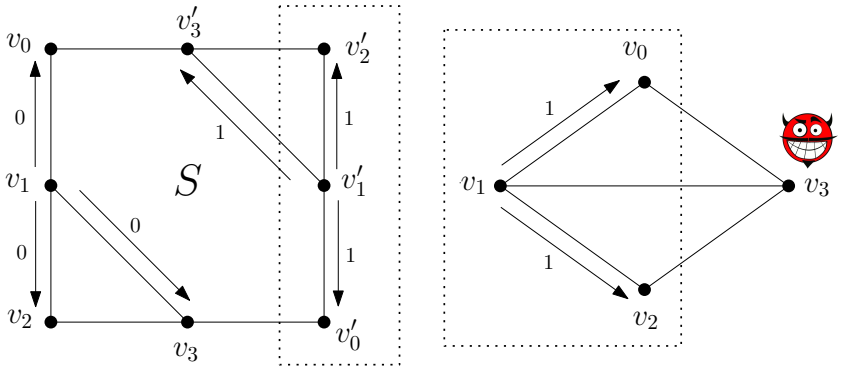



Figure 4.4: Scenarios $\sigma_S$ and $\sigma_{G_3}$ are indistinguishable to $v'_0, v'_1$ and $v'_2$

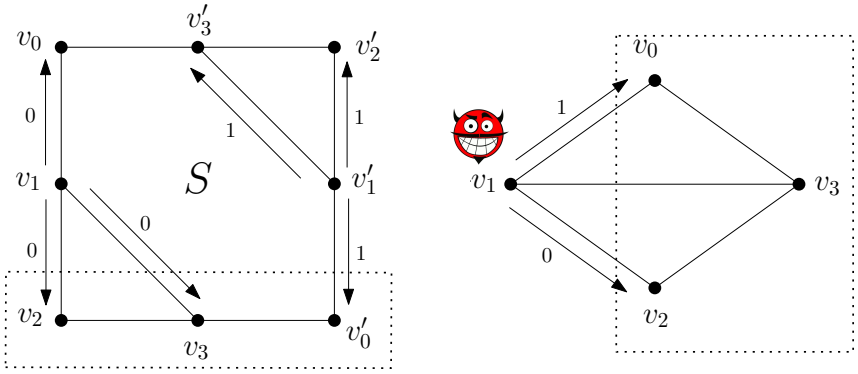



Figure 4.5: Scenarios $\sigma_S$ and $\sigma_{G_1}$ are indistinguishable to $v_2, v_3$ and $v'_0$

$$\sigma_S \overset{v_2}{\sim} \sigma_{G_1} \text{ and } \sigma_S \overset{v_3}{\sim} \sigma_{G_1} \text{ and } \sigma_S \overset{v'_0}{\sim} \sigma_{G_1} \Rightarrow decision(v_2) = decision(v'_0) \quad (3)$$

Obviously relations $(1), (2)$ and $(3)$ yield a contradiction.

We can apply the same methodology to prove the theorem in the case of graph $G'$, where $G_0$ and $G_2$ are connected subgraphs. We replace players $v_0, v_2, v'_0, v'_2$ with the subgraphs $G_0, G_2, G'_0, G'_2$ respectively where each $G'_i$ is an identical copy of $G_i$, $i \in \{0, 1\}$; namely for every $v \in G_i$ there is an identical $v'$ in $G'_i$ and if $(v, w)$ is an edge of $G_i$ then so is $(v', w')$ in $G'_i$. For every player $v \in G_0$ which is connected with $v_3$ in $G'$, we connect $v$ with $v'_3$ and $v'$ with $v_3$ in $S$. In the same way, for every player $v \in G_2$ which in connected with $v_3$ in $G'$, we connect $v$ with $v_3$ and $v'$ with $v'_3$ in $S$. For the players of $G_0, G_2$ originally connected with $v_1$, we connect them in a similar way with $v_1$ and $v'_1$ in $S$. To carry on the proof in the resulting system $S$ we work as before but instead of considering e.g. the view of players $v_0, v_1, v_2$ we consider the view of players $v \in G_0 \cup \{v_1\} \cup G_2$. System $S$ of this case is depicted in Figure 4.6.

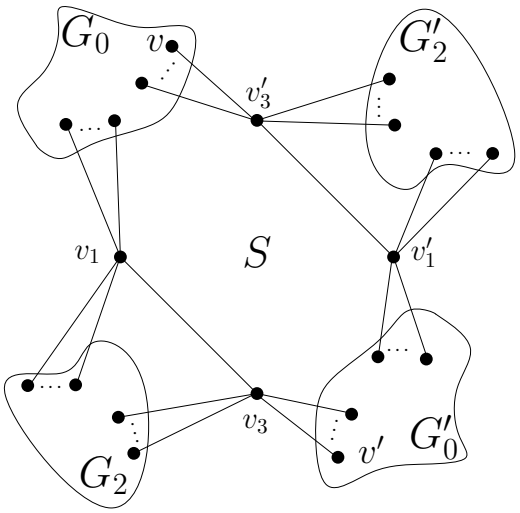



Figure 4.6: System $S$ in the case of $t = 1$.

In order to generalize the result to $t > 1$, we can use the same diagrams, with $v_1$ and $v_3$ replaced by $I_1$ and $I_3$ of at most $t$ nodes each and $v_0, v_2$ by arbitrary sets $I_0$ and $I_2$ of nodes. Removing all the nodes in $I_1$ and $I_3$ disconnects $I_0$ and $I_2$. The edges of Figure 4.1(b) can now be considered to represent bundles of edges between the different groups of nodes $I_0, I_1, I_2$ and $I_3$.

□

## 4.2 Solving the Problem in Generic Networks

We will next consider the design of Broadcast protocols in generic networks. Studying the problem of *Secure Message Transmission* (SMT) as introduced in [DDWY93] is essential in achieving Broadcast in incomplete networks. The protocol presented achieves *Secure Message Transmission* from any node to any other node of a generic network provided that it is at least $(2t + 1)$-connected. This protocol is partially based on the classic Dolev's protocol [Dol82]. In our case though, the computational complexity is polynomial. Furthermore, we show how this protocol can be executed in parallel in order to achieve secure transmission from any set of nodes to any other set of nodes of the graph. Using the above techniques we show that the protocol can be used as a subroutine for the simulation of any known protocol for complete networks. Combining the above with e.g. the protocol [GM98] yields a protocol for generic networks which remains polynomial with respect to

the three measures that we are interested in.

Essentially we present a reduction from the Broadcast problem in generic networks to the problem in the complete network model. More specifically we construct a protocol which takes as input a protocol solving the problem in complete networks, and produces as output a protocol solving the problem in generic networks. The following results has been first presented in the joint work of C.Litsas, A.Pagourtzis and D.Sakavalas [LPS12].

**Supplementary Notation**

Thereafter the paths of the graph will be represented by strings of the form $\sigma_i, p_i \in \mathcal{V}^*$. The neighborhood of a node $v$ will be denoted by $\mathcal{N}(v)$. Let $\sigma \in \mathcal{V}^*$ and $w \in \mathcal{V}$ then

- A player/node $w$ *participates* in a path $\sigma$ ($w \in \sigma$) if and only if

$$\exists \sigma_1, \sigma_2 \in \mathcal{V}^* \text{ s.t. } \sigma = \sigma_1 w \sigma_2$$

- The set of prefixes of $\sigma$ is the set

$$prefix(\sigma) = \{\sigma_1 \in \mathcal{V}^* | \exists \sigma_2 \in \mathcal{V}^*, \sigma_1\sigma_2 = \sigma\}$$

- The order of $w$ in string $\sigma$ is defined to be

$$ord_w(\sigma) = \begin{cases} |\sigma_1| & \text{if } \sigma = \sigma_1 w \sigma_2, \sigma_1, \sigma_2 \in \mathcal{V}^* \\ -1 & \text{if } w \notin \sigma. \end{cases}$$

The protocols (e.g. 2.1,2.6) that solve the problem in complete networks operate in rounds. Every single round consists of two phases: the *communication phase* where message are exchanged in parallel between nodes and the *internal computations phase* where every node processes the information it has received. Typically a Broadcast protocol as well as every protocol that involves communication between participants is of the following form.

**4.1 Protocol : *Secure Broadcast for Complete Networks***

1. `For` round $i = 1$ to $r$, $r \in \mathbb{N}$
   (a) **CommunicationPhase**($\mathcal{B}_\mathbf{i}$) :
   (b) **ComputationPhase**(**i**)
2. $\forall p \in \mathcal{P}$, $p$ returns $decision(p)$

**Note.** *Each* $CommunicationPhase(\mathcal{B}_i)$ *involves communication of the the pairs in the set* $\mathcal{B}_i \subseteq \mathcal{V} \times \mathcal{V}$. *Namely,*

$$CommunicationPhase(\mathcal{B}_i) : v_j \text{ transmits a message } m_{i,j} \text{ to } u_j, \forall (v_j, u_j) \in \mathcal{B}_i$$

*and each* $ComputationPhase(i)$ *involves internal computations performed by each node individually.*

In order to reduce the problem of Broadcast in general networks to the problem in complete networks we propose a sub-protocol to simulate authenticated message transmission between any two nodes of the network. Next we present a protocol that implements the above task:

**4.2 Protocol : *2-node transmission***

**Input:** Nodes $v, u$ and security parameter $t$.

**Objective:** Authenticated transmission of message $m$ from $v$ to $u$.

1. `If` $u \in \mathcal{N}(v)$ then node $v$ sends the message $m$ to node $u$ which decides on this value.
2. `Else`
   (a) `Every` node $w \in \mathcal{V}$ calculates the same set $\mathcal{P} = \{p_1, p_2, \ldots, p_{2t+1}\}$ of $2t + 1$ disjoint paths from $v$ to $u$ (*valid paths*).
   `Every` node $w \in \mathcal{V} \setminus u$ stores at most one single path $p_w \in \mathcal{P}$, which is the one that contains its name. Node $u$ stores the set $\mathcal{P}$ of *valid paths*.

(b) **Round 0** : $v$ sends the message $m_v = (m, v)$ to every one of his neighbors that happens to be a starting node of one of the disjoint paths of $\mathcal{P}$.

(c) `For` $i = 1 \ldots \max_{p \in \mathcal{P}} |p|$

**Round i :** `Every` node $w \in \mathcal{V} \setminus \{v, u\}$ with $p_w \neq \emptyset$ that received messages in round $(i-1)$ performs (all nodes in parallel):

`If` $w$ received in the previous step the message $m_x = (m', \sigma x)$ from node $x$, s.t.

$$\big(\sigma x w \in \text{prefix}(p_w)\big) \wedge \big(ord_w(p_w) = i\big)$$

then $w$ creates a new message $m_w = (m', \sigma x w)$ and relays it to the next node in $p_w$.

(d) Node $u$ finds the majority of the values he received through valid paths. If there are at least $t+1$ identical values (absolute majority) he decides on this value. If the majority is less than $t+1$ (relative majority) then $u$ decides on a default value $\perp$.

---

For the calculation of the common set $\mathcal{P}$, every node should execute an appropriate variation of the max-flow algorithm [AMO93] with the same input, thus also the same output.

Finally, $u$ receives at most $2t+1$ values from the disjoint valid paths. In case that $v$ is honest then given that there are at most $t$ corrupted generals we get that there are at least $t+1$ paths consisting purely of honest generals. Consequently, the majority of the received values is the message $m$. In every case the receiver $u$ will decide on the value that the transmitter $v$ intended it to receive. Observe that node $u$ may decide on $\perp$ only if $v$ is corrupted.

### Complexity

For the length of the longest computed path it holds that $\max_{p \in \mathcal{P}} |p| \leq n - 2t$ because in the worst case each of the $2t$ paths may consist of one internal node; thus the remaining path is possible to contain all the rest nodes of the graph. Thus,

$$RC = \max_{p \in \mathcal{P}} |p| \leq n - 2t \Rightarrow RC = O(n)$$

Let $\mathcal{M}_w$ be the set of messages received by $w$ during the protocol. Every message sent from a node has length $O(n \log n)$ (contains the message and at most $n - 2t$ nodes that form the path). Each $w \in \mathcal{V} \setminus u$ receives a message at most once, since

he only has to accept messages in round $ord_w(p_w) - 1$ from the node dictated by $p_w$. Similarly node $u$ will receive a total of $2t + 1$ messages. Thus the algorithm has bit complexity,

$$BC = \sum_{w \in \mathcal{V}} \sum_{m \in \mathcal{M}_w} |m| = \sum_{w \in \mathcal{V} \setminus u} \sum_{m \in \mathcal{M}_w} |m| + \sum_{m \in \mathcal{M}_u} |m| \leq$$

$$\leq \sum_{w \in \mathcal{V} \setminus u} n \log n + n \log n(2t+1) \leq n^2 \log n + n \log n(2t+1) \Rightarrow$$

$$\Rightarrow BC = O(n^2 \log n)$$

The local computations complexity for each node is bounded by the complexity of the algorithm used for the formation of the set $\mathcal{P}$ (essentially by the complexity of the max-flow algorithm, i.e. $O(n^3)$).

$$LCC = O(n^3)$$

### 4.2.1 Simulation of the Communication Phase

As can be observed in protocol 4.1, given any protocol e.g. [GM98] that solves the Broadcast problem in complete networks, it suffices to simulate its communication phase with a sub-protocol to obtain a solution in the generic network model. This *multiple authenticated message transmissions* sub-protocol must guarantee authenticated message transmission between every pair of the given set $\mathcal{B}$ corresponding to each $CommunicationPhase(\mathcal{B})$ and may be created by executing the *2-node protocol* in parallel. Specifically we give a modification of protocol 4.2 that operates in parallel for multiple sender-receiver pairs.

**4.3 Protocol : *Multi-node transmissions***

**Input:** Set $\mathcal{B}$ of node-pairs and security parameter $t$.

**Objective:** Authenticated transmission between every pair in $\mathcal{B}$.

1. `For` every $(v, u) \in \mathcal{B}$

   - `If` $u \in \mathcal{N}(v)$ then node $v$ sends the message $m_v$ to node $u$ which decides on this value.
     $\mathcal{B} := \mathcal{B} \setminus (v, u)$

## Precomputation

2. **Initialize:** $\mathcal{P} = \emptyset$
   `For` every $(v, u) \in \mathcal{B}$

   (a) `Every` $w \in \mathcal{V}$ computes the same set $P_{v,u}$ of $2t + 1$ disjoint paths connecting the pair $(u, v)$

   (b) $\mathcal{P} := \mathcal{P} \cup P_{u,v}$

   `Every` node $w \in \mathcal{V}$ computes and stores the set $\mathcal{P}_w = \{p \in \mathcal{P} | w \in p\}$

## Message Transmission

3. **Round 0** : `For` every $(v, u) \in \mathcal{B}$
   $v$ sends the message $(m_v, v)$ to every one of his neighbors that happens to be a starting node of one of the disjoint paths of $P_{v,u}$.

4. `For` $i = 1 \ldots \max_{p \in \mathcal{P}} |p|$

   **Round i :** `Every` node $w \in \mathcal{V}$ with $\mathcal{P}_w \neq \emptyset$ that received messages in round $(i - 1)$ performs (all nodes in parallel):

   $w$ accepts every message $(m, \sigma x)$ received from node $x$, provided that

   $$\exists p \in \mathcal{P}_w \text{ s.t. } \big(\sigma x w \in \text{prefix}(p)\big) \wedge \big(ord_w(p) = i\big)$$

   then $w$ creates a new message $(m, \sigma x w)$ for every one of those messages and relays it to the next nodes according to $\mathcal{P}_w$.

## Decision

5. `For` every $(v, u) \in \mathcal{B}$
   $u$ finds the majority of the values he received through valid paths of $P_{v,u} \subset \mathcal{P}_u$. If there are at least $t + 1$ identical values (absolute majority) he decides on this value for $m_v$. If the majority is less than $t + 1$ (relative majority) then $u$ decides on a default value $\perp$.

---

**Observation.** We can now simply replace the $CommunicationPhase(\mathcal{B})$ of any given protocol for complete networks, with the *multi-node transmissions* protocol 4.3 in order to obtain the corresponding solution for the problem in generic networks.

**Complexity**

Due to the parallel transmissions, in protocol 4.3, the number of rounds remains at most $n - 2t$, but the bit complexity is now $O(z \cdot n^2 \log n)$, where $z = |\mathcal{B}|$. Finally, the local computations complexity of each node is $O(z \cdot n^3)$ for the nodes to compute the set of disjoint paths for every given pair.

Given a protocol for complete networks with *round complexity* $r$, *bit complexity* $b$ and *local computations complexity* $c$, after the simulation of the communication phase we get a protocol for the generic network model with round complexity,

$$RC = O(r \cdot (n - 2t))$$

due to the $r$ executions of the *multi-node transmissions* protocol. Bit complexity,

$$BC = O(r \cdot (n^4 \log n)) \text{ and } BC = O(b \cdot (n^2 \log n))$$

because of the $r$ executions of the protocol 4.3, or $b$ executions of protocol 4.2 for $b$ bits to be transmitted over pairs. Finally the local computations complexity will be,

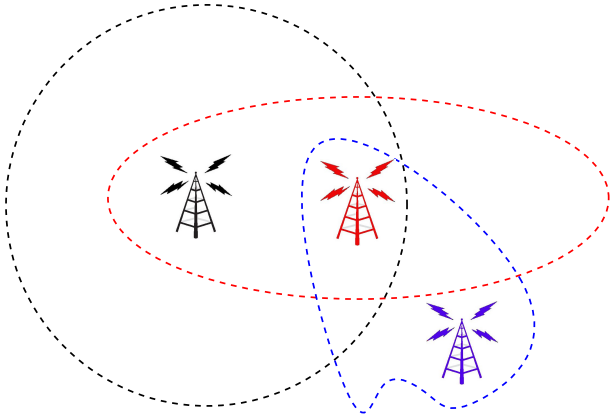
$$CC = O(c + n^5)$$

as the paths between every possible pair can be precomputed in the beginning of the protocol and not in every round.

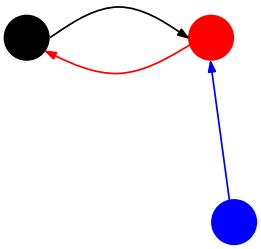
## 4.3 Broadcast in Wireless Networks

A large class of applications involves wireless networks in which nodes possessing radio transmitting/receiving devices are spread out on some physical surface (terrain), and two nodes can communicate if there are within transmission range of each other and signal interference is low. A common abstraction is to consider the network as a graph, and assume that the following holds

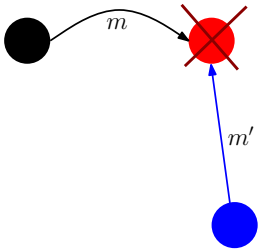
**Collision assumption** Communication is possible if a node receives a message from only one neighbor in a certain time-slot.



(a) Wireless Network



(b) Graph model




(c) Collision (Red receives nothing)

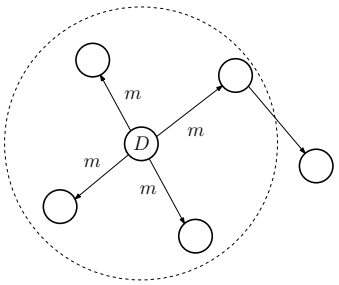
The theoretical distributed algorithms community has only recently devoted attention to wireless networks with adversarial behavior. The first such work is the

2004 paper of Koo [Koo04] which studies the Broadcast problem, in which the attention was restricted to very special graphs: those in which nodes are arranged in the integer grid and the neighborhood of every node consists of nodes at distance $r$ from it in one of the metrics $L_\infty, L_1$ or $L_2$. In [Koo04] a lower bound was established on $t$ for which broadcast is impossible. Bhandari and Vaidya [BV05] proved Koo's bound tight by exhibiting a matching algorithm. In 2006, Koo, Bhandari, Katz, and Vaidya [KBKV06] extend the model to allow for a bounded number of collisions and spoofed addresses.

### 4.3.1 A protocol for wireless networks

We modify the protocol presented in the previous section in order to develop a protocol especially designed for wireless networks. This modification has first been presented in [LPS12].

**Assumptions.** We consider a wireless network which provides authenticated communication between neighboring nodes and in which the collision assumption holds. We also assume that all nodes are incapable of deviating from the given transmission schedule imposed by the protocol. Finally we assume that there are at least $(2t + 1)$ disjoint paths connecting $D$ with $v, \forall v \notin \mathcal{N}(D)$.




We observe that the dealer $D$ in a wireless network is committed to behave honestly during the transmission of his message $m$. This is due to the fact that every message he transmits is received by all $v \in \mathcal{N}(D)$. Since communication channels are authenticated, every honest neighbor will correctly decide on $m$.

Obviously, Byzantine Generals problem is simplified in wireless networks since the honesty of the dealer yields a 1-round solution in a complete network. Specifically, in this round $D$ sends the message $m$ to every player $v$ and each $v$ accepts the value $m$ that he receives. Therefore, in a generic wireless network the problem reduces to every honest player correctly receiving the message of the dealer. The transmission of the message to all the players can be achieved with an appropriate modification of the *multi-node transmissions* protocol.

### 4.4 Protocol : *Wireless Broadcast*

**Input:** Dealer node $D$ and security parameter $t$.

**Objective:** Secure Broadcast of $D$'s message.

## Precomputation

1. **Initialize:** $\mathcal{P} = \emptyset$
   `For` every $v \in \mathcal{V} \setminus \mathcal{N}(D)$

   (a) `Every` $w \in \mathcal{V}$ computes the same set $P_v$ of $2t+1$ disjoint paths connecting the pair $(D, v)$

   (b) $\mathcal{P} := \mathcal{P} \cup P_v$

   `Every` node $w \in \mathcal{V}$ computes and stores the set $\mathcal{P}_w = \{p \in \mathcal{P} | w \in p\}$

## Message Transmission

2. **Phase 0:** $D$ transmits $(m, D)$ to every $w \in \mathcal{N}(D)$.
   `Each` $w \in \mathcal{N}(D)$ decides on value $m$.

3. `For` $i = 1 \ldots \max_{p \in \mathcal{P}} |p|$
   **Phase i :** `Every` node $w \in \mathcal{V}$ with $\mathcal{P}_w \neq \emptyset$ that received messages in round $(i-1)$ performs (all nodes in parallel):

   (i) General $w$ accepts every message $(m', \sigma x)$ received from node $x$, provided that

   $$\exists p \in \mathcal{P}_w \text{ s.t. } \big(\sigma x w \in \text{prefix}(p)\big) \wedge \big(ord_w(p) = i\big)$$

   (ii) $w$ creates message $m_w^j = (m', \sigma x w)$ for each accepted message.

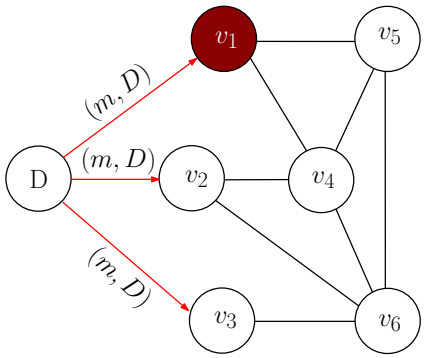
   (iii) Finally he concatenates all messages $m_w^j$ in a single message $m_w$ and transmits $m_w$ to every $v \in \mathcal{N}(w)$.

## Decision

4. `For` every $w \in \mathcal{V} \setminus \mathcal{N}(D)$
   $w$ finds the majority of the values he received through valid paths of $P_w \subseteq \mathcal{P}_w$. If there are at least $t+1$ identical values (absolute majority) he decides on this. If the majority is less than $t+1$ (relative majority) then $u$ decides on a default value $\perp$.

Below we give an example to illustrate the *Wireless Broadcast* protocol.

**4.1 Example.** Commander $D$ broadcasts message $m$ and player $v_1$ is corrupted. Each $v \in \mathcal{V}$ precomputes the the sets of disjoint paths $P_4, P_5.P_6$.




$$P_4 = \{Dv_1v_4, Dv_2v_4, Dv_3v_6v_4\},$$

$$P_5 = \{Dv_1v_5, Dv_2v_4v_5, Dv_3v_6v_5\},$$

$$P_6 = \{Dv_1v_4v_6, Dv_2v_6, Dv_3v_6\}$$

Initially $D$ transmits $(m, D)$ to all $v \in \mathcal{N}(v)$.

**1st phase:** Generals $v_1, v_2, v_3$ accept value $m$ they received from dealer and each transmits in a separate round $(m', Dv_1)$, $(m, Dv_2)$, $(m, Dv_3)$ respectively.

**2nd phase:** According to the computed paths, player $v_4$ transmits $(m, Dv_2v_4)||(m', Dv_1v_4)$ in order for $v_5, v_6$ to receive messages $m, m'$ respectively. Similarly $v_6$ transmits $(m, Dv_3v_6)$.

**3rd phase:** Finally players $v_4, v_5, v_6$ compute the $majority(m, m, m') = m$, of the messages received through valid paths $(P_4, P_5, P_6)$ and decide on value $m$.

■ presented below

### Observations

- In the wireless network model there is no need for the classic bounds for resiliency $(t < n/3)$ and connectivity $(t < k/2)$ to hold. Instead, the connectivity bound can be replaced by the weaker assumption that there are at least $(2t + 1)$ disjoint paths connecting $D$ with $v, \forall v \in \mathcal{V} \setminus \mathcal{N}(D)$.

  The necessity of this assumption, in the case we want to avoid further transmission of messages between pairs of players (which would increase the number of rounds significantly), is guaranteed by the results of [DDWY93].

- Due to the collision assumption each player must transmit in a separate round. In order to minimize the number of rounds, each player $w$ concatenates all messages to be relayed by him (of those he received in the previous phase) and transmits them to $\mathcal{N}(w)$ with a single transmission.

- The space requirements for each node $w$ include the storage of the set $\mathcal{P}_w$, for which we observe that,

$$|\mathcal{P}_w| \leq (n - |\mathcal{N}(D)| - 2) + (2t + 1) \leq n - 2 \Rightarrow |\mathcal{P}_w| = O(n)$$

because node $w$ will store at most one path for each $v \in \mathcal{V} \setminus (\mathcal{N}(D) \cup D \cup w)$ and $2t + 1$ in which it is the last node.

**Complexity**

As before, $\max_{p \in \mathcal{P}} |p| \leq n - 2t$ phases are needed for the messages to be relayed over the longest possible paths. Each phase $i$ includes $rounds(i)$ number of rounds for all the players that need to relay a message to transmit. In conclusion:

$$RC = \sum_{i=1}^{\max_{p \in \mathcal{P}} |p|} rounds(i) \leq \sum_{i=1}^{n-2t} rounds(i) \leq \sum_{i=1}^{n-2t} n = \\ = n \cdot (n - 2t) \Rightarrow RC = O(n^2)$$

Let $\mathcal{M}_v$ be the set of messages received by $v$ during the protocol, then

$$BC = \sum_{v \in \mathcal{V}} \sum_{m \in \mathcal{M}_v} |m| \overset{*}{\leq} \sum_{v \in \mathcal{V}} \sum_{m \in \mathcal{M}_v} n^2 \log n \overset{**}{\leq} \sum_{v \in \mathcal{V}} n^3 \log n = \\ = n^4 \log n \Rightarrow BC = O(n^4 \log n)$$

$(*)$ In the worst case every concatenated message player $w$ receives will contain one sub-message for every other player; therefore $\forall m, |m| \leq n^2 \log n$.
$(**)$ In total, player $w$ will accept $|\mathcal{P}_w|$ messages, thus $|\mathcal{M}_v| \leq n$.

Finally The internal computational complexity for each node is bounded by the complexity of the modified max-flow algorithm used for the computation of disjoint paths between the $n - 2t - 2$ pairs $(D, v)_{v \in \mathcal{V} \setminus \mathcal{N}(D)}$,

$$CC = O(n^4)$$

# Chapter 5

# Broadcast with Locally Bounded Adversary

Considering topological restrictions on the adversary's corruption capacity is of great importance in the study of protocols in incomplete networks. As is naturally expected, if we forbid the adversary to corrupt sets of players with certain properties then we can design protocols tolerating more corruptions than implied by the impossibility theorem. Moreover, as we will see later, if the restrictions involve only local conditions for every node, they imply local criteria which the players can use in order to achieve Broadcast in networks of unknown topology.

**t - Locally Bounded Adversary.** Such an example is the *t-locally bounded adversary model*, introduced in [Koo04], in which at most $t$-corruptions are allowed in the neighborhood of every node (hereafter $\mathcal{N}(v)$ will denote the neighborhood of node $v$ including itself). Namely

$$\forall v \in \mathcal{V},\ |\mathcal{N}(v) \cap \mathcal{T}| \leq t$$

Previous work on Broadcast in the $t$-locally bounded model has focused in the case when the dealer is honest. Specifically in [Koo04] the attention was restricted to very special wireless network graphs: those in which nodes are arranged in the integer grid and the neighborhood of every node consists of nodes at distance $r$ from it in one of the metrics $L_\infty, L_1$ or $L_2$. The *Certified Propagation Algorithm* (CPA) was first proposed by Koo for the solution of the problem. In 2005 Pelc and Peleg [PPP05] considered the $t$-locally bounded model in generic graphs and proved an upper bound for the number of corrupted players $t$ that can be tolerated in order to achieve Broadcast. In the latter paper, the writers also considered the CPA protocol of [Koo04] and proved a sufficient topological condition for

the protocol to achieve Broadcast in generic networks, leaving the deduction of a necessary condition as an open problem.

We will first focus in the *Broadcast problem with an honest Dealer* in order to emphasize the topological conditions which render the problem solvable.

### Terminology and Definitions

As we saw before in section 4.2, due to the structure of an incomplete network, the players may decide on the required value in different rounds unlike the most complete network Broadcast solutions given. For this reason we classify nodes at various stages during the execution into three classes:

**5.1 Definitions.** (Node Classes)

Uninformed: *The node did not get the message yet.*

Received: *The node has received the message (possibly in a number of copies and with a number of conflicting values) but cannot ascertain its correct value yet.*

Accepted: *The node has received the message and has ascertained its correct value (has decided on a value).*

**5.2 Definition** (Safe Algorithm)**.** *We call a Broadcast algorithm* safe*, if it never causes a node to accept an incorrect message.*

Note that a safe algorithm might still fail, particularly by not delivering the message to all the nodes of the network. These notions are made precise below.

**5.3 Definition** ($t$ - Local Set)**.** *A set $W$ of nodes is* $t$-local *if it contains at most $t$ nodes in each neighborhood, i.e., $|W \cap \mathcal{N}(v)| \leq t, \ \forall v \in \mathcal{V}$.*

**5.4 Definition** ($t$ - Locally Safe Algorithm)**.** *A Broadcast algorithm with dealer $D$ of graph $G$ is* $t$-locally safe*, if it never causes a node to accept an incorrect message under any $t$-local set of corrupted nodes.*

**5.5 Definition** ($t$-Locally Resilient Algorithm)**.** *A Broadcast algorithm with Dealer $D$ of graph $G$ is* $t$-locally resilient *if it achieves Broadcast under any $t$-local set of corrupted players.*

## 5.1 Impossibility of t-Locally Resilient Broadcast

In order to establish a lower bound on $t$ for which there is no $t$ -locally resilient algorithm, we need the following notions. First, in a graph $G = (\mathcal{V}, E)$, a *cut* $C \subseteq \mathcal{V}$ is a set of nodes whose removal disconnects the graph $G(\mathcal{V} \setminus C, E')$ induced by the remaining nodes into (at least) two components.

**5.6 Definition** ($t$ - Local Pair Cut)**.** *A cut $C \subseteq \mathcal{V}$ in a graph $G$ is a $t$* -local pair cut *if $C$ can be partitioned into two disjoint sets $C = C_1 \cup C_2$ such that $C_1$ and $C_2$ are $t$ -local.*

**5.7 Definition** (LPC)**.** *The* local pair connectivity *of a graph $G$, denoted $LPC(G)$, is the smallest nonnegative integer $t$ such that $G$ has a $t$-local pair cut.*

We next give an upper bound for the number of corruptions in order for the Broadcast problem in the $t$-locally bounded model to be solvable.

**5.1 Theorem** (Pelc, Peleg 2005)**.** *For every graph $G$ and integer $t \geq LPC(G)$, there is no $t$-locally resilient Broadcast algorithm on $G$.*

*Proof.*
Suppose that $\Pi$ is a $t$ -locally resilient Broadcast algorithm on $G$ for $t = LPC(G)$. Consider a $t$-local pair cut $C = C_1 \cup C_2$ and let $D$ (the dealer) and $v$ be nodes on two sides of the cut $C$ ($G'$ and $G''$ respectively). We show that $\Pi$ does not allow $v$ to correctly accept a message from $D$ in all scenarios, which contradicts its $t$-local resiliency.

Consider the following two scenarios $\sigma_1$ and $\sigma_2$ of $\Pi$. In $\sigma_1$ the initial value of $D$ is 1 and the corrupted nodes are precisely those in $C_1$. In each step $t \geq 1$ of scenario $\sigma_1$, every node in $C_1$ performs the action that it is instructed to perform in step $t$ of scenario $\sigma_2$ (where it is honest). In $\sigma_2$ the initial value of $D$ is 2 and the corrupted nodes are precisely those in $C_2$. In each step $t \geq 1$ of scenario $\sigma_2$ , every node in $C_2$ performs the action that it is instructed to perform in step $t$ of execution $\sigma_1$ (where it is honest). It follows that all nodes which are on the same side of the cut as $v$ perform identical actions in scenarios $\sigma_1$ and $\sigma_2$ of $\Pi$. Actually we have that

$$\forall v \in G'',\ \sigma_1 \overset{v}{\sim} s_2 \Rightarrow decision_v(\sigma_1) = desicion_v(\sigma_2)$$

Hence $v$ decides on the same value in $\sigma_1$ and $\sigma_2$, which cannot be correct in both scenarios (validity is violated in one of them), since $D$ sends different messages in each of them. □

In the work of Ichimura and Shigeno [IS10], the writers extend the results of [PPP05]. The following theorem indicates the difficulty of computing the parameter $LPC(G)$ for a graph $G$.

**5.2 Theorem** (Ichimura, Shigeno 2010)**.** *Computation of $LPC(G)$ is NP-hard*

*Proof.*
We use reduction of the SET SPLITTING PROBLEM, known as NP-hard [GJ79] to the problem of computing the parameter $LPC(G)$ for a graph $G$.

*SET SPLITTING PROBLEM*: Given a collection $\mathcal{S}$ of 3-element subsets of a finite set $X$, decide whether there is a partition of $X$ into two subsets $X_1$ and $X_2$ such that no subset in $\mathcal{S}$ is entirely contained in either $X_1$ or $X_2$.

Let $\mathcal{S}_+$ be the collection that results after adding dummy subsets $\{v\}$ to $\mathcal{S}$ such that the cardinality of $\{s \in \mathcal{S}_+ | v \in s\}$ is at least six for each $v \in X$. A complete graph with node set $\mathcal{S}_+$ and a copy of it are denoted by $K_{\mathcal{S}_+}$ and $K'_{\mathcal{S}_+}$, respectively. We construct a graph $G^{SSP}$ with node set and edge set respectively,

$$V(G^{SSP}) = V(K_{\mathcal{S}_+}) \cup V(K'_{\mathcal{S}_+}) \cup X$$
$$E(G^{SSP}) = E(K_{\mathcal{S}_+}) \cup E(K'_{\mathcal{S}_+}) \cup \{(v,s),(v,s') | v \in X, s \in \mathcal{S}_+, v \in s\}$$

where $s'$ is a node in $V(K'_{\mathcal{S}_+})$ which is a copy of $s \in \mathcal{S}_+$.

If a subgraph of $G^{SSP}$ deleting $C \subseteq V(G^{SSP})$ has at least two connected components and $X \setminus C \neq \emptyset$, then $C$ contains $\mathcal{N}(v) \cap V(K_{\mathcal{S}_+})$ or $\mathcal{N}(v) \cap V(K'_{\mathcal{S}_+})$ for some $v \in X$. Since each $v \in X$ has at least six neighbors in both $V(K_{\mathcal{S}_+})$ and $V(K'_{\mathcal{S}_+})$, $C$ is a t-local pair cut with $t \geq 3$.

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 the desired partitions. Therefore, we have $LPC(G^{SSP}) = 2$, if and only if the set splitting problem has a desired partition.

□

## 5.2 Feasibility of t-Locally Resilient Broadcast

For positive results, the Certified Propagation Algorithm of [Koo04] is one of the simplest Broadcast algorithms which is the only algorithm known, to the best of

our knowledge, that does not use any global knowledge of the network topology. CPA works as follows:

**5.1 Algorithm : *Certified Propagation Algorithm (CPA)***

1. The dealer $D$ sends its initial value $x_D$ all of its neighbors, decides on $x_D$ and terminates.

2. If a node is a neighbor of the source, then upon receiving the message $x_D$ from the dealer, decides on $x_D$, sends it to all of its neighbors and terminates.

3. If a node is not a neighbor of the source, then upon receiving $t+1$ copies of a message $m$ from $t+1$ distinct neighbors, it accepts $m$, sends it to all of its neighbors and terminates.

**Observation.** Notice that, since the adversary is $t$-locally bounded, a message sent by an honest node is always correct. Thus the above protocol is $t$-locally safe. The problem is, of course, that some nodes may not get the message at all, and moreover, some may get too few copies of the message, rendering them unable to guarantee its correctness and thus preventing them from forwarding it.

We will now present a class of graphs for which CPA is $t$-locally resilient. As proven in [PPP05] the condition defining the class is sufficient but not necessary. For a graph $G = (\mathcal{V}, E)$, a given dealer $D \in \mathcal{V}$ and node $v \in \mathcal{V}$, let $X(v, D)$ denote the number of nodes $w \in \mathcal{N}(v)$ that are closer to $D$ than $v$. Namely,

$$X(v, D) = \big|\{x \in \mathcal{N}(v) \mid d(D, x) < d(D, v)\}\big|$$

where $d(v, w)$ is defined to be the length of the shortest path connecting nodes $v$ and $w$. Also, let

$$\mathcal{X}(G, D) = \min\{X(v, D) \mid v \in \mathcal{V}, (v, D) \notin E.\}$$

The following proposition establishes an upper bound on $t$ for which CPA is $t$-locally resilient.

**5.3 Proposition.** *For every graph $G$, dealer $D$ and integer $0 \leq t < \mathcal{X}(G, D)/2$, CPA is $t$-locally resilient.*

*Proof.*
Consider a graph $G$ with dealer $D$ and let $0 \leq t < \mathcal{X}(G, D)/2$. The proof is by contradiction. Suppose that the Certified Propagation Algorithm does not work correctly under some $t$-local corruption pattern. Since CPA is safe, the incorrectness must be due to the fact that some node $v$ does not accept the dealer's message $x_D$. Let $v$ be the closest such node to the dealer. In $\mathcal{N}(v)$ there are at least $2t + 1$ nodes closer to $D$ than $v$. By the choice of $v$ all of them accept $x_D$ at some point. Since at most $t$ of these nodes can be corrupted, the rest of them, i.e., at least $t + 1$ nodes send the message $x_D$ to $v$. Hence $v$ accepts the message $x_D$, contrary to the assumption. □

To prove that the condition $0 \leq t < \mathcal{X}(G, D)/2$ is not necessary for CPA to be $t$-locally resilient, consider the following example of a graph $G$ with $X(G) = 1$ and for which CPA is 1-locally fault-tolerant.

**5.1 Example.**

For graph $G$, its set of nodes is partitioned into five disjoint groups $G_1, \ldots, G_5$, where $|G_1| = 1$, $|G_2| = |G_5| = 2$, and $|G_3| = |G_4| = 3$. Also let the dealer $D \in G_2$. The edges of the graph form complete bipartite graphs $(G_i, G_{i+1})$ for any $i \leq 4$, and a complete bipartite graph $(G_5, G_1)$. There are no other edges. (See Figure 5.1)

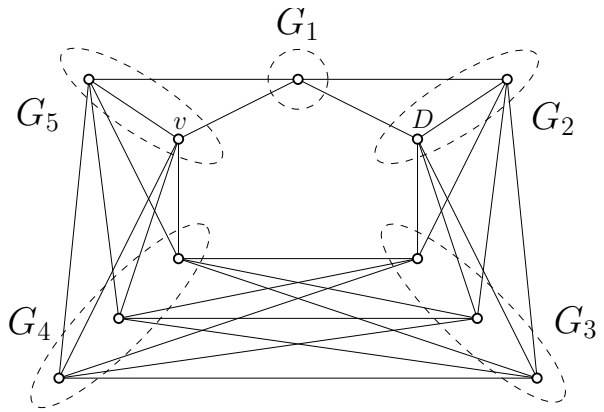



Figure 5.1: The graph $G$

Notice that $\mathcal{X}(G, D) = 1$. This is witnessed by the fact that $X(v, D) = 1$ for $v \in G_5$. If we execute CPA in $G$ with dealer $D$. Then the nodes in $G_1$ and in $G_3$ accept the dealer message $x_D$ as neighbors of $D$. There is at most one faulty node in $G_3$. Hence every node in $G_4$ and the non-dealer node in $G_2$ gets 2 copies of $x_D$ and thus it accepts $x_D$ and sends it to all neighbors. Finally nodes in $G_5$ receive at least to copies of $x_D$ from the two honest players of $G_4$ and thus accepts.

■

The condition given $0 \leq t < \mathcal{X}(G, D)/2$ of [PPP05] implies that the nodes of the graph should be divided in layers w.r.t. their distance from the dealer (level 1 being the neighborhood of $D$). If this certain conditions holds then every node in distance $k$ (level $k$) decides in the $k$ - th round, because it is certain to receive $t + 1$ correct values from honest nodes in the previous level. But as seen in the previous example, the situation could be different, namely, a node in level $k$ may collect $t + 1$ messages with the same value, from levels in greatest distance than $k$ from the dealer.

### 5.2.1 A Better Topological Parameter for CPA Broadcast

In [IS10], the writers introduced a new parameter for the feasibility of $t$-resilient CPA Broadcast. We present a parameter similar to that of [IS10], which seems simpler and more intuitive. The study presented next is a result of the join work of C.Litsas, A.Pagourtzis and D.Sakavalas. Considering the way that CPA works and the previous example we can generalize the previous topological restriction in order to obtain a better one for CPA to be $t$-locally resilient. Since CPA is trivially $t$-locally safe, it remains to introduce a condition that ensures that every node $v \in \mathcal{V}$ accepts a message. It is easy to see that in order for the dealer's message to be relayed further in the graph, there should be at least one player in every round $r > 2$ which has at least $t + 1$ honest accepted neighbors. Since the adversary is $t$-locally bounded, if there is at least one player in every round $r > 2$ which has at least $2t + 1$ accepted neighbors, then $t + 1$ of them will be honest. The situation is depicted in Figure 5.2. We formalize the latter notions below by introducing the notion of the *$l$-neighboring sequence* of sets $S_i$.

The *$l$-neighboring sequence* of a graph $G = (\mathcal{V}, E)$ for a given dealer-node $D$ is the following sequence $S_i$

$$\begin{aligned} S_1 =& \mathcal{N}(D) \\ S_2 =& S_1 \cup \{v \in \mathcal{V} : |\mathcal{N}(v) \cap S_1| \geq l\} \\ & \vdots \\ S_k =& S_{k-1} \cup \{v \in \mathcal{V} : |\mathcal{N}(v) \cap S_{k-1}| \geq l\} \\ & \vdots \end{aligned}$$

Also let

$$\widetilde{\mathcal{X}}(G, D) = \max\{l \in \mathbb{N} | \exists k \in \mathbb{N},\ s.t.\ S_k = \mathcal{V}\}$$

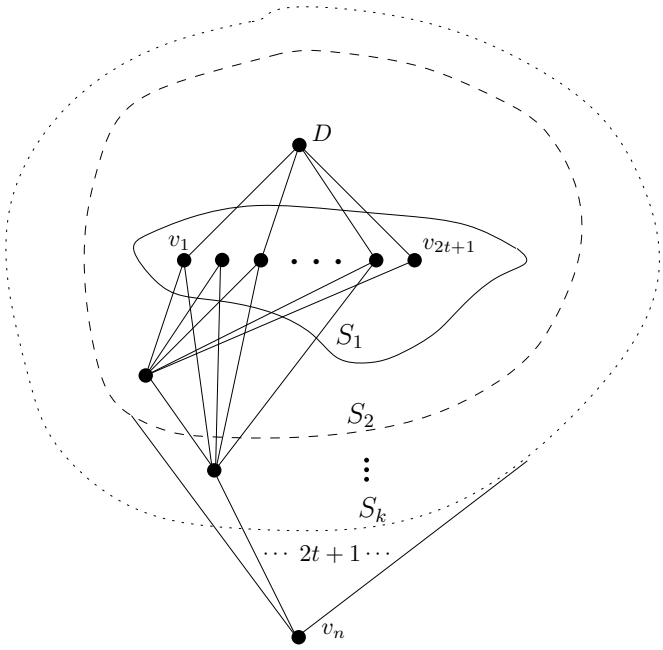



Figure 5.2

**5.4 Theorem.** *For every graph $G$, dealer $D$ and integer $0 \leq t < \widetilde{\mathcal{X}}(G,D)/2$, CPA is $t$-locally resilient.*

*Proof.*
Observe that $2t < \widetilde{\mathcal{X}}(G,D) \Rightarrow \exists k \in \mathbb{N},\ s.t.\ S_k = \mathcal{V}$ for the $(2t+1)$-neighboring sequence $S_i$. Let $k_{\min} = \min\{k \in \mathbb{N} | S_k = \mathcal{V}\}$. It suffices to show that for $i \leq k_{\min}$, every $v \in S_i$ accepts the dealer's message $x_D$. By induction on $i$:

For all honest $v \in S_1 = \mathcal{N}(D)$, $v$ accepts $x_D$ due to the CPA steps 1 and 2.
If for all honest $v \in S_m$, $v$ accepts $x_D$ at some round, then every honest $v \in S_{m+1}$ receives $|S_m \cap \mathcal{N}(v)| \geq 2t+1$ messages from his accepted neighbors in $S_m$, $t+1$ of which are honest. Thus he decides on $x_D$.

□

The following proposition shows that parameter $\widetilde{\mathcal{X}}(G,D)$ is more efficient than $\mathcal{X}(G,D)$ for the upper bound on $t$ for which CPA is t-locally fault-tolerant.

**5.5 Proposition.** *For any graph $G$ with dealer $D$, $\widetilde{\mathcal{X}}(G,D) \geq \mathcal{X}(G,D)$ holds.*

*Proof.*

Assume $\widetilde{\mathcal{X}}(G,D) < \mathcal{X}(G,D)$ for a graph $G = (\mathcal{V}, E)$ and dealer $D$. Then for $l = \mathcal{X}(G,D)$, it holds that

$$X(v,D) \geq l,\ \forall v \in \mathcal{V}$$

and $\nexists k \in \mathbb{N},\ s.t.\ S_k = \mathcal{V}$ for the $l$-neighboring sequence $S_i$ of $G$ and dealer $D$.

Consider the $l$-neighboring sequence for $G$ and dealer $D$,

$$\begin{aligned} S_1 =& \mathcal{N}(D) \\ S_2 =& S_1 \cup \{v \in \mathcal{V} : |\mathcal{N}(v) \cap S_1| \geq l\} \\ &\vdots \\ S_k =& S_{k-1} \cup \{v \in \mathcal{V} : |\mathcal{N}(v) \cap S_{k-1}| \geq l\} \\ &\vdots \end{aligned}$$

*Claim.* $\forall v \in \mathcal{V}$ with distance $d(v,D) = i$ it holds that $v \in S_i$

By induction on the distance $i$ we have:
For $i = 1$, trivially. For $i = 2$ every $v$ with distance $d(v,D) = 2$ has $X(v,D) \geq l$, thus $|\mathcal{N}(v) \cap S_1| = |\mathcal{N}(v) \cap \mathcal{N}(D)| \geq l$ since every neighbor of $v$ closer to $D$ than $v$ is in $\mathcal{N}(D)$. Thus it holds that $v \in S_2$.

Assume that the claim holds for $i = k$, namely, $\forall v \in \mathcal{V}$ with $d(v,D) = k$, $v \in S_k$. Then for every $v \in \mathcal{V}$ with $d(v,D) = k+1$ it holds that $X(v,D) \geq l$. For all $w$ neighbors of $v$ closer to $D$ than $v$ it holds that $d(w,D) = k \Rightarrow w \in S_k$, thus $|\mathcal{N}(v) \cap S_k| \geq l \Rightarrow v \in S_{k+1}$.

The above claim immediately implies the result, namely if the graph is connected then $\exists k \leq \max_{v \in \mathcal{V}} d(v,D).\ s.t.\ S_k = \mathcal{V}$, which yields a contradiction.

If the graph is not connected then $\widetilde{\mathcal{X}}(G,D) = \mathcal{X}(G,D) = 0$.

□

The parameter $\widetilde{\mathcal{X}}(G,D)$ also establishes a lower bound on $t$ for which CPA does not work correctly under any t-local set of corrupted players.

**5.6 Theorem.** *For any graph $G$ with $t \geq \widetilde{\mathcal{X}}(G,D)$, CPA is not $t$-locally resilient.*

*Proof.*
Assume that CPA is $t$-locally resilient, where $t \geq \widetilde{\mathcal{X}}(G,D)$.
Let $\widetilde{\mathcal{X}}(G,D)$ and $S_k$ be the $(t+1)$-neighboring sequence of $G$, with dealer $D$. Since $\widetilde{\mathcal{X}}(G,D) = \max\{l \in \mathbb{N} | \exists k \in \mathbb{N},\ s.t.\ S_k = \mathcal{V}\}$ and $t+1 > \widetilde{\mathcal{X}}(G,D)$, it holds that there exists $v \in \mathcal{V}$ for which

$$\nexists k \in \mathbb{N}\ s.t.\ |\mathcal{N}(v) \cap S_k| \geq t+1 \Rightarrow |\mathcal{N}(v) \cap S_k| \leq t,\ \forall k \in \mathbb{N}. \tag{5.1}$$

We will use the following claim to reach a contradiction,

*Claim.* Let $\mathcal{A}_i$ be the set of accepted players in the end of round $i$, then for every round $i$, $\exists k \in \mathbb{N},\ s.t.\ \forall w \in \mathcal{A}_i,\ w \in S_k$.

(Induction on the number of rounds)
$i = 1$ : For each $w \in \mathcal{A}_1$, it holds that $w \in \mathcal{N}(D) \Rightarrow w \in S_1$.

Assume that in round $i = r$, it holds that $\exists k \in \mathbb{N},\ s.t.\ \forall w \in \mathcal{A}_r,\ w \in S_k$.

For every $w \in \mathcal{A}_{r+1}$, either $w \in S_k$ (accepted players of previous round), either $w$ received at least $t+1$ identical messages from accepted players of previous rounds. In the second case we have that

$$|\mathcal{N}(w) \cap S_k| \geq t + 1 \Rightarrow w \in S_{k+1}.$$

Since CPA is $t$-locally resilient, player $v$ accepts a message at some point; this means that $v$ received at least $t + 1$ identical messages from accepted distinct neighbors. Using the previous claim we deduce that

$$\exists k \in \mathbb{N},\ s.t.\ |\mathcal{N}(v) \cap S_k| \geq t + 1$$

which contradicts the statement 5.1.

□

The parameter $\widetilde{\mathcal{X}}(G, D)$ provides us with upper and lower bounds on $t$ for which CPA is $t$-locally resilient. Theorems 5.4 and 5.6 imply that $\widetilde{\mathcal{X}}(G, D)$ approximates the largest $t$ such that CPA is $t$-locally resilient within a factor of two.

## 5.3 Conclusions

### Dealer Corruption

The $t$-locally bounded model, as presented in this chapter is easy to generalize in the case of a corrupted Dealer, simply by simulating the communication phase of a Broadcast protocol for complete networks. The simulation can be achieved using a combination of one-to-all transmissions implemented with CPA. Specifically there are efficient protocols for complete networks such as Protocol 2.6 presented in previous chapter, which only use one-to-all transmissions for the communication of the players. To modify such protocols in order to achieve Broadcast in the $t$-locally bounded model, it suffices for each player to run CPA as dealer, whenever he transmits a message to all the other players in the original protocol.

### Knowledge of the Topology

Since CPA does not use any global knowledge of network topology, the results presented hold regardless of whether the topology of the network is known or if the network is ad-hoc. CPA is the only Broadcast algorithm we know that does not use any global knowledge of the network topology. Since 2005, it remains an open problem to prove that the CPA algorithm is "unique" on ad-hoc networks, in the sense that for any $t$-local corruption set, if some other rule works then so does CPA.

Knowledge of topology turns out to be essential in the context of Broadcast protocols. The separation between broadcast algorithms knowing the topology of the network and ad hoc algorithms is evidenced by the result of [PPP05] that there exists a graph for which some 1-locally resilient algorithm exists, if the knowledge of the graph is assumed but no such algorithm exists otherwise.

### Wireless Networks

Results presented on the feasibility of $t$-locally resilient Broadcast apply in the wireless network model as well. In particular, in order to avoid collisions in the wireless model, all nodes must transmit separately in different rounds. As before, in the wireless model we need the assumption that the behavior of corrupted nodes is restricted to the content of messages and cannot affect the schedule (otherwise a corrupted node might create constant noise, thus preventing all nodes in its neighborhood from receiving the source message). An interesting extension of the model would be to increase the adversary's capacities by allowing for a bounded number of collisions and spoofed addresses. The latter was studied by Koo, Bhandari, Katz, and Vaidya [KBKV06] in the special case of the grid network model.

# Bibliography


[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[BG89] Piotr Berman and Juan A. Garay. Asymptotically optimal distributed consensus (extended abstract). In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *ICALP*, volume 372 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 1989.

[BG91] Piotr Berman and Juan A. Garay. Efficient distributed consensus with n = (3 + epsilon) t processors (extended abstract). In Sam Toueg, Paul G. Spirakis, and Lefteris M. Kirousis, editors, *WDAG*, volume 579 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 1991.

[BG93] Piotr Berman and Juan A. Garay. Cloture votes: n/4-resilient distributed consensus in t+1 rounds. *Mathematical Systems Theory*, 26(1):3–19, 1993.

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *FOCS*, pages 410–415. IEEE Computer Society, 1989.

[BNDDS87] Amotz Bar-Noy, Danny Dolev, Cynthia Dwork, and H. Raymond Strong. Shifting gears: changing algorithms on the fly to expedite byzantine agreement. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, PODC '87, pages 42–51, New York, NY, USA, 1987. ACM.

[BNDDS92] Amotz Bar-Noy, Danny Dolev, Cynthia Dwork, and H. Raymond Strong. Shifting gears: Changing algorithms on the fly to expedite byzantine agreement. *Inf. Comput.*, 97(2):205–233, 1992.

[BV05] Vartika Bhandari and Nitin H. Vaidya. On reliable broadcast in a radio network. In Marcos Kawazoe Aguilera and James Aspnes, editors, *PODC*, pages 138–147. ACM, 2005.

[Coa86] 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.

[Coa87] B. A. Coan. *Achieving consensus in fault-tolerant distributed computer systems: protocols, lower bounds, and simulations*. PhD thesis, Cambridge, MA, USA, 1987.

[CW93] Brian A. Coan and Jennifer L. Welch. Modular cosntruction of an efficient 1-bit byzantine agreement protocol. *Mathematical Systems Theory*, 26(1):131–154, 1993.

[DDWY93] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993.

[DFF+82] Danny Dolev, Michael J. Fischer, Robert J. Fowler, Nancy A. Lynch, and H. Raymond Strong. An efficient algorithm for byzantine agreement without authentication. *Information and Control*, 52(3):257–274, 1982.

[Dol82] Danny Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[DR85] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985.

[DRS90] Danny Dolev, Rüdiger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, 1990.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[FL82] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GM98] 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.

[IS10] 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.

[KBKV06] Chiu-Yuen Koo, Vartika Bhandari, Jonathan Katz, and Nitin H. Vaidya. Reliable broadcast in radio networks: the bounded collision case. In Eric Ruppert and Dahlia Malkhi, editors, *PODC*, pages 258–264. ACM, 2006.

[Koo04] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In Soma Chaudhuri and Shay Kutten, editors, *PODC*, pages 275–282. ACM, 2004.

[LPS12] C. Litsas, A. Pagourtzis, and D. Sakavalas. The byzantine generals problem in generic and wireless networks. *1st Conference of Cryptography and its Applications in the Armed Forces" , Hellenic Army academy, National and Kapodistrian University of Athens , Hellenic Mathematical Society*, 2012.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[MW88] Y. Moses and O. Waarts. Coordinated traversal: (t+1)-round byzantine agreement in polynomial time. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 246 –255, oct 1988.

[PPP05] Andrzej Pelc, Davdrzej Pelc, and Did Peleg. Broadcasting with locally bounded byzantine faults. *Inf. Process. Lett.*, 93(3):109–115, 2005.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[TPS87] Sam Toueg, Kenneth J. Perry, and T. K. Srikanth. Fast distributed agreement. *SIAM J. Comput.*, 16(3):445–457, 1987.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE Computer Society, 1982.