U N I V E R Z I T A K O M E N S K É H O
Fakulta matematiky, fyziky a informatiky
Katedra informatiky

RNDr. Richard Královič

Broadcasting with Dynamic Faults

Dizertačná práca

v odbore doktorandského štúdia:
11-80-9 teoretická informatika

školiteľ: doc. RNDr. Pavol Ďuriš, CSc.

Bratislava December 2007

Abstract

The thesis deals with the problem of broadcasting in distributed systems in presence of dynamic link failures. The first part of the thesis presents an overview of different computation models of distributed systems based on the point-to-point communication paradigm. Furthermore, different models of failures analyzed in previous theoretical research are presented, with focus on deterministic (i.e. non-randomized) models of dynamic link faults. Introduced models are classified according to a newly developed framework. Here, different models of faulty distributed computations can be described by combination of several independent aspects of distributed computation models.

The second part of the thesis presents a quest for a suitable deterministic model of dynamic faults. Several such models are introduced and analyzed in connection with the broadcasting problem. At first, the well known *constant k-bounded* model is presented, together with a short overview of corresponding previous results. In this model, the number of faulty links is bounded by the constant k in any time step of the distributed computation.

Next, the *fractional α-weakly-bounded* model is defined and analyzed. Here, contrary to the constant k-bounded model, the maximal number of faulty links depends linearly on the number of active links. This eliminates the undesirable property of the constant k-bounded model, which encourages the broadcasting algorithm to flood the communication network with as many messages as possible. In this thesis, previous results for broadcasting in fractional α-weakly-bounded model for value of $\alpha = 1/2$ are generalized for arbitrary value of $\alpha \in (0, 1)$.

However, if the number of active links is too small, the fractional α-weakly-bounded model provides too strong bounds on number of faults. To avoid this, the *α-fractional threshold model*, and its extreme case – the *simple threshold model*, are introduced and analyzed. The analysis of the simple threshold model yields interesting results about the ability of distributed systems to disseminate information even if there is a very weak guarantee on the number of non-faulty links. Presented results show that it is always possible to solve the broadcasting problem even in such extremely harsh environment. Furthermore, an efficient (i.e. polynomial-time) solution is provided for a large class of communication network topologies. Last but not least, several results for the more general α-fractional threshold model are presented. These results include efficient (logarithmic-slowdown) broadcasting algorithms for complete graphs under certain assumptions (either chordal sense of direction or $\alpha \lesssim 0.55$) and efficient algorithms for broadcasting to all but a constant number of vertices for complete graphs and hypercubes without any additional assumptions.

Abstrakt

Predkladaná práca sa zaoberá problémom broadcasting-u v distribuovaných systémoch s chybnými linkami. V prvej časti práce je uvedený prehľad rôznych výpočtových modelov distribuovaných systémov založených na tzv. point-to-point komunikácii. Súčasťou prehľadu je aj prezentácia rôznych modelov chýb, zameraná predovšetkým na deterministické (t.j. nie randomizované) modely chybných liniek. Pre lepšiu orientáciu v prezentovaných modeloch bol navrhnutý nový spôsob ich klasifikácie, ktorý umožňuje popísať jednotlivé modely kombináciou viacerých nezávislých čŕt.

Druhá časť práce je venovaná hľadaniu vhodného deterministického modelu dynamických chýb liniek. V práci je definovaných a analyzovaných niekoľko takýchto modelov. Najskôr je uvedený stručný prehľad výsledkov týkajúcich sa tzv. *konštantne ohraničeného* modelu, ktorému bola venovaná väčšina predchádzajúceho výskumu. V tomto modeli je počet chybných liniek v ľubovoľnom kroku výpočtu ohraničený konštantou.

Ďalším analyzovaným modelom dynamicky chybných liniek je tzv. *proporčne ohraničený model*, v ktorom maximálny počet chybných liniek závisí lineárne od počtu aktívnych liniek. Tento model odstraňuje neželanú vlastnosť konštantne ohraničeného modelu, v ktorom je optimálne zahlcovať sieť maximálnym možným množstvom správ. V predchádzajúcom výskume bol problém broadcasting-u v α-proporčne ohraničenom modeli analyzovaný iba pre hodnotu $\alpha = 1/2$. V dizertačnej práci sú tieto výsledky zovšeobecnené pre ľubovoľné hodnoty $\alpha \in (0, 1)$.

Proporčne ohraničený model má však inú neželanú vlastnosť, a to zaručene nulový počet chýb v prípade malého počtu aktívnych liniek. Na odstránenie tejto vlastnosti bol navrhnutý tzv. *proporčne-prahový* model a jeho extrémny prípad – tzv. *model s jednoduchým prahom*. Analýza modelu s jednoduchým prahom ukazuje, že problém broadcasting-u je vždy riešiteľný, dokonca aj pri najslabších možných obmedzeniach počtu chýb. Navyše, v mnohých prípadoch je možné broadcasting vykonať efektívne (v polynomiálnom čase od počtu uzlov distribuovaného systému).

Posledná skupina výsledkov uvedených v práci sa týka α-proporčne-prahového modelu. Tieto výsledky zahŕňajú efektívne algoritmy pre rozšírenie informácie do takmer všetkých uzlov distribuovaného systému (t.j. do všetkých uzlov okrem konštantného počtu) pre úplné grafy z hyperkocky. Výsledky pre úplné grafy sú pri použití určitých dodatočných predpokladov (chordálny zmysel pre orientáciu alebo hodnoty $\alpha \lesssim 0.55$) rozšírené aj na riešenie problému broadcasting-u.

Foreword

The *broadcasting problem* is one of the most fundamental tasks in the area of distributed algorithms. In this problem, the goal is to disseminate a piece of information known to only one node of the distributed system to every other node. Since the communication forms an integral part of all distributed algorithms, the broadcasting problem is an inherent component of many of them. Hence, the ability to design efficient broadcasting algorithms is essential for development of any efficient distributed algorithm. Furthermore, the analysis of the broadcasting problem allows one to compare the suitability of different communication topologies in distributed computing. Due to these reasons, the broadcasting problem is an important subject in the theoretical research.

The optimal solution of the broadcasting problem and its efficiency are deeply affected by the exact variant of the considered communication model. Many different communication models have been studied in previous research. One of the most popular class of models, which is used also in this thesis, is the *point-to-point* communication. Here, the distributed system consists of several autonomous nodes that are able to communicate only by sending and receiving messages via dedicated communication links. Naturally, the structure of the communication links has great influence on the properties of the distributed system. Hence, a significant amount of research is devoted to the study of the relationship between the communication links structure and the efficiency of the information dissemination.

It is usually not feasible to ensure a completely fault-free operation of large distributed systems. Hence, the fault tolerance is a very important feature of distributed algorithms. For this reason, it is necessary to take the desired fault tolerance properties into account when analyzing the relationship between broadcasting efficiency and communication link structure.

In the first part of the thesis, we present an overview of different communication models of distributed systems based on the point-to-point communication. We classify these models according to several independent aspects, such as the ability of the links to operate in a full-duplex mode, availability of the communication link topology knowledge to the nodes of the distributed systems, etc. These aspects can be arbitrarily combined to create a wide variety of communication models. In this part, we provide an overview of various models of faults, too, with focus on the deterministic models.

The rest of the work deals with the broadcasting problem in presence of *dynamic link faults*. Here, some communication links can fail during the computation. However, the set of faulty links can change during the computation, i.e. the link faults are transient only.

The most studied model of dynamic link faults so far is the *constant k-bounded model*. In this model, a constant number of k links can fail in any computation time step. Any set of k links can fail in any time, without any dependency on the previous failures of the computation. Such model, however, has a serious disadvantage: The optimal solution of the broadcasting problem is a greedy algorithm that floods the network with as many messages as possible. Such strategy causes a huge network load, which is often highly undesirable. This was the reason to introduce the *fractional α-weakly-bounded model* in paper [41], which I am a coauthor of. In this model, the maximal number of faulty links in a given time step depends linearly on the number of messages sent in that particular time step. Nevertheless, the fractional model considered in [41] has another weakness: If there are only few messages sent in a given time step, all of them are guaranteed to be delivered, which is a too strong assumption from the practical point of view.

Goals of the thesis: The first goal of the thesis was to generalize the results of [41], which dealt with a single value of $\alpha = 1/2$, for arbitrary values of the parameter α. This goal is fulfilled in Chapter 4: we show very natural counterparts of all results from [41] for arbitrary α values. The next goal was to eliminate the above-mentioned deficiency of the fractional model. For this reason, the *fractional threshold model*, and its corner-case variant, the *simple threshold model*, were introduced in paper [17], which I am a coauthor of, too. The third goal of the thesis was to analyse the broadcasting in the simple threshold and fractional threshold model. This goal is fulfilled in Chapters 5 and 6, where we present several results about broadcasting in the considered communication models. These results have been published in [17, 42], too.

All results presented in Chapters 4, 5, and 6 are original and have been developed by the author of this thesis together with the coauthors of [17, 41, 42].

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Concurrently with the development of both theoretical and applied computer science the advantages of parallel and distributed computing have been investigated. The term "distributed system" is used to describe a set of interconnected autonomous entities (e.g. computers, processors or processes) that cooperate on a specific task. Each of these entities (called also "nodes") is capable of both performing computation independently of other nodes and exchanging information with them.

The usage of distributed systems could provide several benefits. Among the most obvious ones, there is the possibility of communication between distant nodes. Indeed, the main goal of many practical applications (e.g. the world wide web, electronic mail, etc.) is the information exchange itself. This advantage is closely connected to the possibility of resource sharing. In a distributed system, one service (e.g. printer, back-up storage, etc.) can be used by many nodes of the system. Moreover, well designed distributed system is scalable, i.e. it is possible to expand the capacity of the system by adding more service providing nodes.

Other important advantages of distributed systems over single stand-alone ones are the possibility to increase performance through parallelization and the possibility to increase reliability through replication. To increase the performance of the system, the solved task is decomposed into multiple subtasks that can be solved by different nodes concurrently. As for reliability, distributed systems are often designed to be fault tolerant, i.e. that they function properly even when some of their components fail. In such systems, the important information is replicated in multiple nodes and in case of failure the remaining nodes can take over the tasks of the faulty component. This is important especially in large systems, where it is not realistic to assume that every component of the system is fault-free.

The price for these advantages is the fact that the distributed algorithms are more difficult to design. Even the simplest tasks, such as disseminating information from one node to all other nodes or gathering information from all nodes to one node, are not trivial in many settings. What makes things even more complicated is the lack of a universal computational model of distributed systems. While in the area of sequential algorithms there is one commonly used computation model (the random access machine) and all other reasonable models (e.g. Turing machines, register machines, etc.) are equivalent to

it within a polynomial slowdown, this is not the case in the area of distributed algorithms. There is a wide variety of frequently used distributed computation models and not much is known about their complexity relationship.

In this thesis, we focus on the distributed computation models based on the point-to-point communication. This class of models, extensively studied in the literature, is both simple and general enough to capture the properties of many real world applications. The point-to-point communication implies that the nodes can communicate only by exchanging messages, which can be transmitted through a set of dedicated links. This allows us to represent the communication infrastructure of the distributed system as a graph, whose vertices represent the nodes and edges represent the links of the system. It is obvious that the structure of this communication graph has huge impact on the performance of the underlying distributed system and must be taken into account in the design of the distributed algorithms.

Since the communication is inherent part of all distributed algorithms, it is necessary to solve the problem of information dissemination effectively. This holds for both computational and pure communicational tasks. Hence it is important to study the impact of the structure of communication graph on the efficiency of the information dissemination tasks.

One of the most frequently considered tasks of information dissemination is the problem of *broadcasting* (called also *one-to-all communication*). In this problem one node of the distributed system knows a piece of information that is required to be delivered to all other nodes. This simply formulated task is an important communication primitive in many more complex distributed algorithms. Moreover, broadcasting can be used to solve many other important communication tasks with reasonable efficiency. These tasks include e.g. the *gathering problem* (all-to-one communication), where each node knows a piece of information that has to be delivered to one dedicated *sink* node, and the *gossipping problem* (all-to-all communication), where each node needs to learn a piece of information from all other nodes.

In the sequel we focus on the influence of the structure of communication graph on the efficiency of broadcasting. Since the fault resilience is an important property of well-designed distributed systems, we assume that the computation is not fault-free and we consider various models of faults. We particularly focus on dynamic link faults. Here, the computation of the distributed system runs in synchronous time steps, and in each time step the algorithm sends some messages via some communication links. Some links may fail, i.e. the messages sent through the failing links are lost. The link failures are transient, i.e. different set of links may fail in different time steps of the distributed computation. We focus mainly on deterministic fault models, where the algorithm is given some deterministic worst-case guarantee on the number and location of faulty links. Hence, such model of faulty distributed computations can be viewed as a game between the distributed algorithm, which specifies how to send the messages, and the adversary, which selects the location of faulty links in each time step.

In Chapter 2, we give an overview of various models of distributed systems together with references to the literature considering them. We describe the possible models of fault-free computations and models of faults separately, as these can be arbitrarily combined.

We also split both the models of fault-free computations and the models of faults into several mutually independent aspects that can be arbitrarily combined . In Chapter 2, we also present complexity measures of distributed computations and an overview of classes of graphs that are commonly used as communication network topologies.

Next chapters are devoted to the analysis of the broadcasting in various models of faulty distributed computations. As already said, we focus on deterministic dynamic link failures. In Chapter 3, we present a short overview of the results about broadcasting in the constant k-bounded model. In this model, which is the oldest and most widely studied model of deterministic dynamic link faults so far, a constant number of k links may fail in every time step and the location of the failures in different time steps is completely independent.

The constant k-bounded model, however, exhibits a serious weakness: The optimal broadcasting strategy is always to flood the communication network with maximal possible number of messages. To avoid this undesirable property, the *fractional α-weakly-bounded model* has been introduced and analyzed in [41]. In this model, the number of link failures depends linearly (with coefficient α) on the number of active links. However, the analysis in [41] has been restricted to the value of $\alpha = 1/2$. We devote Chapter 4 to the analysis of the fractional α-weakly-bounded model; in particular, we generalize the results of [41] for arbitrary values of α.

Unfortunately, the fractional α-weakly-bounded model has a different flaw: If there are very few messages sent in a particular time step, all of them are always delivered. This guarantee, however, is too strong, as it is hard to justify a guarantee of completely fault-free operation of the distributed system, even if the network load is small.

To avoid both undesirable properties of the constant k-bounded model and the fractional α-weakly-bounded model, the *fractional threshold model*, and its corner case – the *simple threshold model* – have been introduced in [17]. In Chapter 5, we present results about the simple threshold model. In this model, the number of messages sent must be greater than certain threshold to guarantee a delivery of at least one message. Presented results are somewhat surprising, as they show that, even in this extremely harsh model, it is not only always possible to perform the broadcasting, but it is often possible to perform it in a time-efficient way.

The fractional threshold model is considered in Chapter 6. This model is similar to the simple threshold model in the fact that if the number of messages sent is below a certain threshold, all of them may be lost. However, if there are many messages sent, at least some linear fraction of them is delivered. We present results about information dissemination in complete graphs and hypercubes in this model, which have been published also in [42]. Last but not least, we sum up our work and show several remaining open problems in Chapter 7.

Chapter 2

Models

The solution of the fault tolerant broadcasting problem depends on the chosen computation model, which can be defined in many different ways. The computation model of a faulty distributed system can be viewed as a composition of two parts: the *model of the distributed system* and the *model of faults*. These parts of the model are independent, i.e. it is possible to combine any model of the distributed system with any model of faults.

In the following section, we discuss the models of the distributed system; various models of faults are defined in the next section.

2.1 Broadcasting Without Faults

Each distributed system consists of a set of nodes with certain communication capabilities. The nodes are independent entities, hence designing an algorithm for a distributed system involves specifying an algorithm that runs in every node of the system. It is required that all nodes run the same algorithm. However, the nodes may have unique identifiers (as discussed in 2.1.4), which can weaken this restriction.

In this thesis we consider the *point-to-point* model of the communication, which is widely studied in literature (see [1–5, 7, 9–12, 15, 16, 18–21, 25, 26, 28–36, 39, 43, 44, 46–49, 51, 53–56] or surveys [27, 37, 38, 50]).[1] In this model, the nodes of the distributed system can communicate only via a set of links, where each link connects exactly two nodes. Thus the distributed system can be viewed as a graph whose vertices represent nodes and edges represent links of the distributed system. We call this graph the *communication graph* of the distributed system. We assume that all links are symmetric, hence the communication graph is undirected. However, we sometimes work with directed edges of the communication graph, e.g. when dealing with the sense of direction (see Definition 2.7).

We assume that the nodes of the distributed system can communicate only by passing messages through the communication links. We consider the *store-and-forward* model, where each node must receive the whole message before it can use its contents. After re-

[1] Other models considered in the literature include e.g. the *radio network model* (see [52]), *ATM network model* (see [6] and references therein), etc.

ceiving the message, the node can store it and reuse it later (i.e. the buffering is allowed). Furthermore, we assume that each node can locally distinguish its own links.

An important aspect of the distributed system is its synchronicity. The cases most widely considered in the literature (see e.g. [45, 57]) are the extreme cases of this aspect: the *synchronous* mode and the *asynchronous* mode. In the asynchronous mode, nothing is known about the timing of the computation (this mode is studied e.g. in [4, 15]). In the synchronous mode, the algorithm runs in time steps that are synchronized between the nodes. The algorithm can exploit this, since it can make decisions based on the global clock of the distributed system. For other modes of synchronicity see e.g. [56].

In this work we consider only the synchronous models. The reason for this is that it is impossible to design fault tolerant asynchronous algorithms without any restriction on the timing of the computation. Indeed, in the asynchronous computation it is impossible to distinguish between a lost message and between a message that will be delivered later (see [22]).

Now we present the high-level definition of a broadcasting algorithm. This definition is later refined by specifying the capabilities of the algorithm.

Definition 2.1 *Let $G = (V, E)$ be a graph representing a distributed system. Let there be exactly one vertex $u \in V$ (called the* source*) that knows certain piece of information I. A distributed algorithm A is a* broadcasting algorithm *if it ensures that all vertices in V eventually receive the piece of information I.*

The above-mentioned definition of the broadcasting algorithm imposes no constraints on the termination of the algorithm, i.e. the nodes need not be able to detect that they can stop participating in the broadcasting protocol. However, this deficiency of the definition is not relevant in most cases. Since we consider only synchronous distributed systems, it is sufficient for the nodes to know an upper bound on the running time of the algorithm to be able to detect that all nodes have received the broadcasted piece of information.

The capabilities of the algorithm specified later in this section include the communication capabilities of the links and nodes (e.g. speed and duplexity of the links and ability of the nodes to use multiple links simultaneously), degree of synchronization of the nodes, ability of the nodes to adapt their behavior according to the messages they have received and the topology knowledge available to the nodes. After defining these refinements of the model we introduce the complexity measures of the distributed algorithms: the time complexity as the number of time steps required to finish the algorithm, the message complexity as the total number of messages sent, and the bit complexity as the total number of bits communicated by the algorithm. Furthermore, we present some widely studied classes of communication graphs in subsection 2.1.6.

An example of broadcasting is depicted on Figure 2.1. In this example each vertex sends only one message in each time step and the broadcast finishes in 3 time steps.

The classification of the models of distributed systems presented in this section does not cover all models that have been considered in the literature so far. For information about some such models, see e.g. survey [27] and references therein. However, most of the

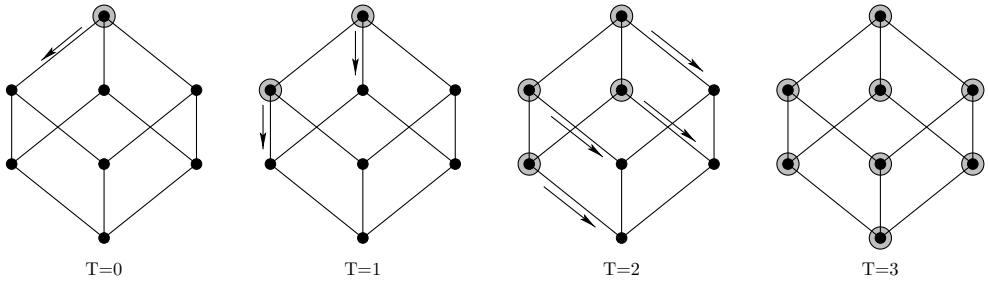


Figure 2.1: An example of broadcasting in a cube graph. Informed vertices are marked by gray circles.

models can be defined by imposing additional constraints on some model presented in this section.

2.1.1 Communication Modes

The communication between the nodes of the distributed system plays the most important role in broadcasting algorithms. Hence it is obvious that the design of the algorithm depends on the communication abilities of the nodes. In this section we present several aspects of these abilities.

An important aspect of the communication is the speed of the communication links:

Definition 2.2 *We define the following communication modes with respect to the speed of the communication links:*

- *In the* constant *mode, each message is delivered in one time step regardless of the length of the message. This mode is the most widely considered in the literature, e.g. in [2, 3, 5, 9–12, 16, 18–21, 25, 28, 29, 33–35, 44, 46, 47, 52–54].*
- *In the* linear *mode, the time of delivery depends on the size of the message: The message of length L is delivered in $\beta + L\tau$ time steps, where β (cost of the start-up) and τ (propagation time of data of unit length) are constants specified in the model. This mode is studied e.g. in [26], see also the survey [27].*

Since the constant mode is the most widely used one, we always consider that one, unless stated otherwise.

Another aspect of the communication is the duplexity of the links:

Definition 2.3 *We define the following communication modes according to the duplexity of the links:*

- *In the* one-way *mode (called also the* half-duplex *or the* telegraph *mode), during a given time step each link can be used to deliver a message only in one direction. In other words, if some node is active on some link, it can be active either as a sender or as a receiver, but not both at once. This is the mode used e.g. in [4, 21, 30, 34].*

- *In the* two-way *mode (called also the* full-duplex *or the* telephone *mode), each link can be used to deliver message in both directions in the same time. This means that one node can be active on one link both as a sender and as a receiver in one time step. This mode is considered e.g. in [16, 25, 33, 48].*

The two-way mode does not provide any advantage for broadcasting in most models that have been considered in the literature. Nevertheless, we assume full-duplex links, unless stated otherwise.

At last we distinguish the communication modes based on the capability of the nodes to use multiple links at once:

Definition 2.4 *We define the following communication modes according to the capabilities of the nodes:*

- *In the* one-port *mode (called also the* whispering *or the* processor-bound *mode), in one time step each node can be active on only one of its adjacent links. This model represents the situation where the bottleneck of the communication is the performance of the node. It is used e.g. in [2, 3, 5, 11, 12, 21, 26, 29, 33–35, 48, 54].*

- *In the* multi-port *mode (called also the* shouting *or the* link-bound *mode), in one time step each node can be active on any number of its adjacent links. This mode represents the situation where the bottleneck is the bandwidth of the links. It is used e.g. in [10, 18–20, 25, 26, 28, 31, 44, 46, 47, 53, 54].*

- *In the* k-port *mode (called also the* DMA-bound *mode), in one time step each node can be active on at most k of its adjacent links where k is a constant specified by the model. This mode is used e.g. in [9].*

In this thesis, we focus on the multi-port mode, unless stated otherwise.

2.1.2 Synchronization

As it has already been stated, we consider only synchronous computation models, i.e. the computation runs in fully synchronized time steps. However, there are two variants of the synchronous models depending on the synchronization of the source node with the other nodes:

- In the *wake-up mode*, only nodes that have received the information can send messages. The nodes that have not received the information yet are in the sleeping state until they are waken up by receiving a message. This mode represents the situation where the nodes other that the source does not know when the broadcast starts. This mode is considered e.g. in [15, 33, 44].

- In the *unrestricted mode*, all nodes can send messages. Moreover, all nodes wake up in the same time. This mode represents the situation where all nodes are fully synchronized through the global clock of the distributed system, i.e. all nodes know the global time. This mode is considered e.g. in [33].

Similarly as in the case of the duplexity of communication links, there is no difference in the complexity of broadcasting between the wake-up and unrestricted mode in most models that have been considered in the literature. Hence, we assume the wake-up mode, unless stated otherwise.

2.1.3 Adaptiveness

The computational resources of individual nodes may be scarce and the broadcasting algorithm should be made as simple and efficient as possible. It may be useful to precompute the whole broadcast in such situations. The algorithm then makes decisions based only on the global clock of the distributed system; it does not consider the messages it has received.

Definition 2.5 *Let A be a distributed algorithm. We say that A is an* adaptive *algorithm, if it makes decisions based on previous messages it has received. Otherwise, A is a* non-adaptive *algorithm.*

Non-adaptive algorithms have been considered e.g. in [5, 29, 34, 35] and adaptive ones e.g. in [15, 33, 44].

It is easy to see that if no faults occur, the source node is a-priori known and full topology knowledge is available, it is possible to restrict to non-adaptive algorithms without loss of generality. This is the reason why many publications (e.g. survey [38]) define the broadcasting problem as follows: Given a communication graph $G = (V, E)$, find sequences $\mathcal{S} = (S_1, \dots S_t)$ and $\mathcal{R} = (R_1, \dots R_t)$ such that $S_i \subseteq E$ is a set of (directed) edges active in time step i for sending information, $R_i \subseteq E$ is a set of (directed) edges active in time step i for receiving information, and the sets S_i and R_i satisfy the constraint imposed by the communication mode used.

If this definition of broadcasting (which corresponds to a non-adaptive algorithm) is used in presence of faults, it is possible that a node tries to send a message although it has not received any message yet. The transmission fails in this case and it is the responsibility of the non-adaptive algorithm to take such cases into account.

The above-mentioned facts imply that it does not make sense to distinguish the wake-up and the unrestricted model in case of non-adaptive algorithms: All transmissions of nodes that have not yet received any information fails, which is similar to the wake-up model. However, the nodes can not make decisions based on the time of receiving of the first message. In fact, it is straightforward to transform any non-adaptive algorithm to an equivalent adaptive algorithm working in the wake-up model.

2.1.4 Topology Knowledge

When designing broadcasting algorithms it is important to know what information about the communication graph of the distributed system is available. This *topology knowledge* can be of two types: Some knowledge can be given *a-priori*, in the time when the algorithm is designed. For example, in some situations we are interested only in some particular class

of graphs; the algorithm does not need to work correctly for other graphs (this is the situation e.g. in [9, 15, 18–20, 26, 31, 44, 46–48, 53, 54]):

Definition 2.6 *Let $\mathcal{G}$ be a class of graphs, let A be a distributed algorithm. The algorithm A is a* broadcasting algorithm *on $\mathcal{G}$ if it works correctly (according to the definition 2.1) on any distributed system with the communication graph $G \in \mathcal{G}$.*

On the other side, some information may be available to the algorithm in the run-time and the algorithm can make decisions based on it. This information can be of any kind. We present some previously considered types of such information:

Definition 2.7 *Some of the types of topology knowledge available in the run-time of the algorithm are:*

- Full knowledge of the topology. *In this scenario each node knows the whole communication graph. It knows its own location and location of all its neighbors in this graph as well. This kind of topology knowledge is exploited e.g. in [51].*
- Blind map. *Each node knows the whole communication graph, but it does not know its location in the graph. The blind map has been considered (although not in connection with broadcasting) e.g. in [14].*
- Identifiers. *Each node knows its own unique identifier. Sometimes the knowledge of the identifiers of neighbors is assumed, too (e.g. in [40, 51]). Note that full topology knowledge implies the knowledge of identifiers. Knowledge of the identifiers without full topology knowledge is used e.g. in [4, 40, 51].*
- Sense of direction. *Each node has associated some topological information (*label*) with each of its adjacent link. The sense of direction is usually defined separately for different topologies (see e.g. [57]). For example, it is the dimension of the link in case of hypercubes (used e.g. in [23]), orientation (up, down, left or right) in case of meshes, etc. A sense of direction has been defined for general graphs in [24].*

2.1.5 Complexity Measures

To compare different distributed algorithms it is necessary to introduce a measure of the quality of an algorithm. The most widely used measures are the message complexity, the bit complexity and the time complexity (see e.g. [45, 57]). However, other complexity measures are possible, too (see e.g. [38]).

Definition 2.8 *Let $G = (V, E)$ be a communication graph and A be a distributed algorithm.*

- *The message complexity $M_A(G)$ of the algorithm A on the graph G is the total number of messages sent by the algorithm A running on the graph G.*

- *The bit complexity $B_A(G)$ of the algorithm A on the graph G is the total number of bits of all messages sent by the algorithm A running on the graph G.*
- *The time complexity $T_A(G)$ of the algorithm A on the graph G is the number of time steps used by the algorithm A running on the graph G.*

The message (bit, time) complexity $M_A(n)$ ($B_A(n)$, $T_A(n)$) of the algorithm A is the maximum of the message (bit, time) complexity $M_A(G)$ ($B_A(G)$, $T_A(G)$) over all graphs G with n vertices, respectively.

In the sequel, we focus on the time complexity of distributed algorithms, which is the most widely studied complexity measure in connection with information dissemination (see e.g. [38]).

2.1.6 Common Topologies

Now we present some most widely studied *topologies*, i.e. classes of communication graphs. These include the *complete graph*, *tree*, *line*, *ring*, *mesh*, *torus* and *hypercube*. Definitions of some other well-known topologies, such as *cube-connected cycles*, *butterflies*, *shuffle-exchange* and *DeBrujin*, are presented in [38].

Definition 2.9 *The* complete graph with n vertices *is defined as follows:*

$$K_n = (V, E), \text{ where } V = \{0, \ldots, n-1\} \text{ and } E = \{\{i, j\} \mid i, j \in V \wedge i \neq j\}$$

The complete graph with a chordal sense of direction *is a complete graph, whose (directed) edge (i, j) is labelled with the number $j - i (\mathrm{mod}\, n)$.*

Informally, in the complete graph every two vertices are connected with an edge (see Figure 2.2). Furthermore, the distributed algorithm running on the complete graph with a chordal sense of direction can distinguish for each link, how far it goes on a fixed Hamiltonian cycle $(0, 1, \ldots, n-1)$. However, the sense of direction does not imply that the algorithm knows the identifiers of the vertices.

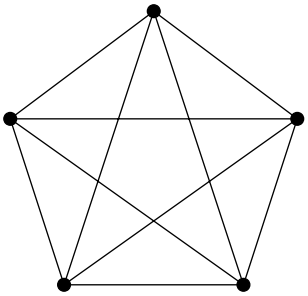
The ring is a connected 2-regular graph (see also Figure 2.3):[2]

Definition 2.10 *The* ring with n vertices *is defined as follows:*

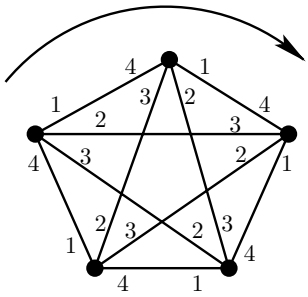
$$R_n = (V, E), \text{ where } V = \{0, \ldots, n-1\} \text{ and } E = \{\{i, i \pm 1 (\mathrm{mod}\, n)\} \mid i \in V\}$$

The ring with the sense of direction is a ring, whose (directed) edge $(i, i+1 (\mathrm{mod}\, n))$ is labelled with $+1$ and the edge $(i, i-1 (\mathrm{mod}\, n))$ is labelled with -1.

[2] An undirected graph is called *k-regular* if and only if each its vertex has degree exactly k.

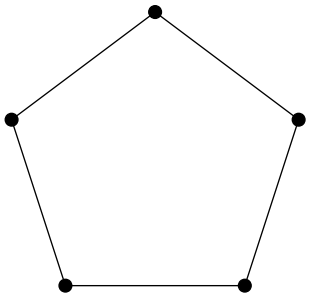


Complete graph K_5



Complete graph K_5 with chordal sense of direction

Figure 2.2: An example of a complete graph.



Ring R_5

Figure 2.3: An example of a ring.

Definition 2.11 *A* tree *is any acyclic connected graph.*

A rooted tree *is a tree with one dedicated vertex r called* root*. Let u and v be vertices connected with an edge. If the distance between r and v is smaller than the distance between r and u, we say that v is the* parent *of u, otherwise v is the* child *of u. A vertex of the tree is called a* leaf *if and only if it has no children.*

A k-ary tree *is a rooted tree such that each its vertex is either a leaf or has exactly k children.*

A complete tree *is a rooted tree such that all leafs has the same distance from the root. This distance is called the* height *of the tree.*

A line *is a complete unary tree.*

There is no standard definition of sense of direction for general non-rooted trees. In fact, the definition of the sense of direction for general graphs in [24] suggests that there is no need for any. The sense of direction for rooted trees is any labelling of the edges that provides the information which edge leads to the parent.

The d-dimensional torus of orders $k_1, \ldots, k_d$ is a graph whose vertices are d-tuples from $\mathbb{Z}_{k_1} \times \ldots \times \mathbb{Z}_{k_d}$ such that every pair of vertices that differ in exactly one position by exactly one (in the group $\mathbb{Z}_{k_i}$) is connected by an edge. An example of the torus is shown on Figure 2.5.

Line with 4 vertices Complete ternary tree of height 2

Figure 2.4: An example of a line and a complete tree.

Definition 2.12 *The* d-dimensional torus of orders $k_1, \ldots, k_d$ *is a graph with* $\prod_{i=1}^{d} k_i$ *vertices defined as follows:*

$$\begin{aligned} G &= (V, E) \\ V &= \{0, \ldots, k_1 - 1\} \times \ldots \times \{0, \ldots, k_d - 1\} \\ E &= \Big\{ \{[x_1, \ldots, x_d], [x_1, \ldots, x_{i-1}, x_i + \Delta (\mathrm{mod}\, k_i), x_{i+1}, \ldots, x_d]\} \mid \\ &\qquad [x_1, \ldots, x_d] \in V, i \in \{0, \ldots, d\}, \Delta \in \{-1, 1\} \Big\} \end{aligned}$$

The d*-dimensional torus* of order k *is the* d*-dimensional torus of orders* $\overbrace{k, \ldots, k}^{d}$.

The torus with the sense of direction is a torus, whose (directed) edges are labelled with the dimension and direction of the edge, i.e. the edge $([x_1, \ldots, x_d], [x_1, \ldots, x_{i-1}, x_i + \Delta (\mathrm{mod}\, k_i), x_{i+1}, \ldots, x_d])$ *is labelled by the tuple* (i, Δ).

The definition of the mesh is very similar to the definition of the torus. Informally, we just remove the "wrap-around" edges (see also Figure 2.5).

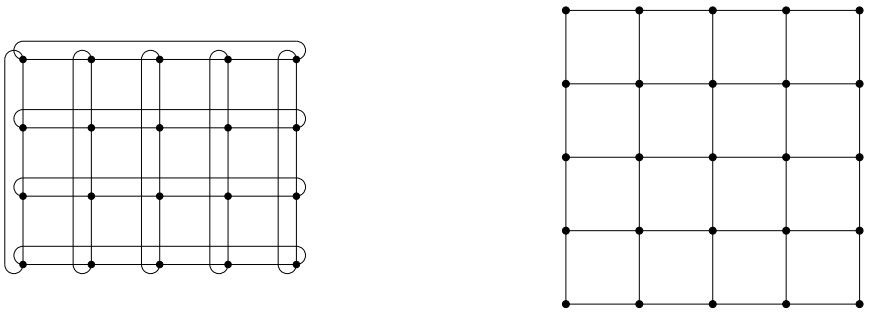
Definition 2.13 *The* d-dimensional mesh of orders $k_1, \ldots, k_d$ *is a graph with* $\prod_{i=1}^{d} k_i$ *vertices defined as follows:*

$$\begin{aligned} G &= (V, E) \\ V &= \{0, \ldots, k_1 - 1\} \times \ldots \times \{0, \ldots, k_d - 1\} \\ E &= \Big\{ \{[x_1, \ldots, x_d], [x_1, \ldots, x_{i-1}, x_i + \Delta, x_{i+1}, \ldots, x_d]\} \mid \\ &\qquad [x_1, \ldots, x_d] \in V, i \in \{0, \ldots, d\}, \Delta \in \{-1, 1\} \Big\} \cap V \times V \end{aligned}$$

The d*-dimensional mesh* of order k *is the* d*-dimensional mesh of orders* $\overbrace{k, \ldots, k}^{d}$.

The definition of the sense of direction is the same as for the torus.

The d-dimensional hypercube can be viewed as a special case of the torus, in fact it is a d-dimensional torus of order 2 without multiple edges (see also Figure 2.6):



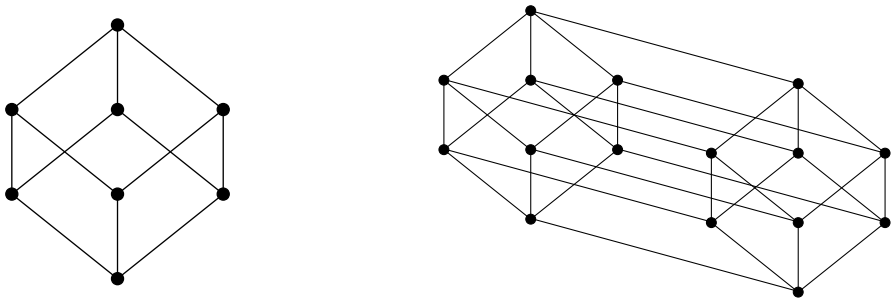
2-dimensional torus of orders 4,5 2-dimensional mesh of order 5

Figure 2.5: An example of a torus and a mesh.

Definition 2.14 *The* d-dimensional hypercube H_d *is a graph with* 2^d *vertices defined as follows:*

$$\begin{aligned} H_d &= (V, E) \\ V &= \{0,1\}^d \\ E &= \Big\{ \{[x_1,\ldots,x_d],[x_1,\ldots,x_{i-1},x_i+1(\mathrm{mod}\,2),x_{i+1},\ldots,x_d]\} \mid \\ &\qquad [x_1,\ldots,x_d] \in V, i \in \{0,\ldots,d\} \Big\} \end{aligned}$$

The hypercube with the sense of direction is a hypercube whose (directed) edges are labelled with the dimension of the edge, i.e. the edge $([x_1,\ldots,x_d],[x_1,\ldots,x_{i-1},x_i{+}1(\mathrm{mod}\,2),$ $x_{i+1},\ldots,x_d])$ *is labelled by the number* i*.*



3-dimensional hypercube 4-dimensional hypercube

Figure 2.6: An example of a hypercube.

2.2 Models of Faults

Similar to the model of the distributed system, the model of faults can be described by several independent aspects of the faults; any combination of these aspects yields a unique model. We discuss these aspects in the following section.

2.2.1 Various Aspects of Faults

1. *Randomized vs. bounded models*

   There are two main approaches in the modelling of the faulty computations. In the randomized model, the faults occur with a certain probability. The goal is to design a distributed algorithm that is correct *with high probability*. In the randomized models studied in the literature there is usually a fixed probability of failure of certain node/link. This is the case in e.g. [5, 11, 12, 16, 49, 52]. However, various variants of randomized models have been considered (e.g. both node and link failures, both permanent and transient failures, both Byzantine and omission failures, etc.).

   The second approach is to analyze the worst-case scenario. To do so, the model of faults imposes constraints on the number of faults that can occur. This approach has been used e.g. in [2, 4, 9, 10, 18–20, 25, 26, 28–36, 39, 43, 44, 46, 47, 51, 53, 54]. The computation can be then viewed as a game of two players: the algorithm specifies which messages are to be sent and the adversary chooses which faults happen. The exact rules of the game (e.g. the order of the players, the information available to the algorithm, the abilities of the adversary, etc.) are determined by other aspects of the faults.

2. *Node vs. link failures*

   Each distributed system consists of elements of two types: the nodes and the links. This implies that there are also two kinds of faults: the faults of the nodes (considered e.g. in [4, 12, 25, 26, 31–34, 36, 49, 53, 54]) and the faults of the links (considered e.g. in [2, 5, 7, 9–12, 16, 18–20, 25, 26, 28, 29, 31, 32, 35, 36, 44, 46, 47, 49, 53, 54]).

3. *Byzantine vs. crash/omission vs. detectable faults*

   This aspect specifies the power of the adversary: in the *Byzantine* faults model, the adversary can specify the behavior of the faulty entity arbitrarily. In case of the node failures this means that the faulty node does not need to follow the specified algorithm, but the adversary can maliciously choose the behavior most detrimental to the algorithm. In case of the link failures the adversary can discard or modify the message transmitted through the faulty link or create any fake message on it. Obviously, the Byzantine faults represent the most harmful type of faults, i.e. the type of faults that gives maximal power to the adversary. This type of faults is studied e.g. in [5, 44, 51, 52].

The *crash/omission* faults give less power to the adversary. In this model, the adversary can block the faulty entity, but it can not alter its behavior in any other way. In case of node failures this means that the node fail to send some messages or it can crash entirely. In case of link failures, some or all messages transmitted through the faulty link can be lost. However, the adversary can not modify the content of any message nor create any fake message. This type of faults is studied e.g. in [2, 10–12, 16, 18–20, 25, 26, 28, 29, 31–35, 39, 43, 44, 46, 47, 52–54].

The weakest variant of this aspect are *detectable* faults. In this model, the locations of faults are known to the nodes, hence it does not make sense to distinguish Byzantine and omission faults. This type of faults is considered e.g. in [4, 9, 31].

4. *Static vs. dynamic faults*

   Another important aspect of the model is the duration of the faults. The models working with *static* (or *permanent*) faults assume that there is a set of faulty entities of the distributed system which are faulty through the whole computation. This means that the adversary must choose some fixed location of the faulty entities before the start of the algorithm. Static faults are considered e.g. in [2, 4, 5, 9, 25, 26, 31–34, 36, 39, 43, 53].

   A minor modification of the static model yields the *partially static* model: The entities can fail in different time steps, but no entity recovers from the failure. Hence the adversary can choose location of new faults in every time step as long as it does not exceed the number of faults allowed by the model. Partially static faults are discussed e.g. in [1, 56].

   In the models working with the *dynamic* (or *transient*) faults, entities can recover from faults. This means that the adversary can choose different location of the faults in every time step of the computation as long as it satisfies other constraints imposed by the model. Hence it is obvious that dynamic faults give more power to the adversary than static faults. Dynamic faults are studied e.g. in [10, 16, 18–20, 28, 29, 35, 41, 44, 46, 47, 52].

2.2.2 Time-independent Dynamic Faults

An important subclass of models with dynamic faults are models with *time-independent* dynamic faults. These models impose constraints on the number and type of failures independently on the history of the computation. The adversary can choose the location of the faults according to these fixed rules and does not need to consider its previous decisions.

In the following chapters, we focus mainly on the subclass of models with bounded time-independent dynamic omission link failures – the $f(x)$-*bounded fault* model and the $f(x)$-*weakly-bounded fault* model:

Definition 2.15 *Let $f(x)$ be a function $\mathbb{N} \to \mathbb{N}$. The $f(x)$-*bounded fault *model is a model of faults with the following aspects:*

- *bounded model*
- *link failures*
- *omission faults*
- *dynamic time-independent faults*

Moreover, let m be the number of messages sent by the algorithm in some time step t, then the adversary can arbitrarily choose at most $f(m)$ messages that are lost. There are no other faults – all messages that are not chosen by the adversary are correctly delivered.

The definition of the $f(x)$-weakly-bounded fault is very similar. Actually, the only difference is that the adversary can block at most $f(m)$ messages, where m is the number of messages destined to uninformed nodes only:

Definition 2.16 *Let $f(x)$ be a function $\mathbb{N} \to \mathbb{N}$. The* $f(x)$-weakly-bounded fault *model is a model of faults with the following aspects:*

- *bounded model*
- *link failures*
- *omission faults*
- *dynamic time-independent faults*

Moreover, let m be the number of messages sent by the algorithm in some time step t that are destined to uninformed vertices*, then the adversary can arbitrarily choose at most $f(m)$ messages that are lost. There are no other faults – all messages that are not chosen by the adversary are correctly delivered.*

The $f(x)$-(weakly)-bounded fault models represent interesting class of models that can be defined by specifying the function $f(x)$.

Now we define several subclasses of $f(x)$-(weakly)-bounded fault models that have been already introduced in the literature and are particularly relevant for this thesis. The simplest and most extensively studied $f(x)$-(weakly)-bounded fault model is the case when $f(x)$ is constant:

Definition 2.17 *Let k be a constant. The special case of $f(x)$-bounded fault model when the function $f(x) = k$ is constant is called the* constant k-bounded fault model.

The constant k-bounded model has been introduced in [55] and studied e.g. in [10, 18–20, 28, 44, 46, 47].

As we show in Chapter 3, the simple greedy flooding algorithm is always optimal in this model. However, this is rarely the case in real networks, since number of faults usually increase with the congestion of the network.

To capture the dependency of the number of failures on the load of the network, another $f(x)$-(weakly)-bounded model has been introduced in [41].

Definition 2.18 *Let $\alpha \in (0,1)$ be a constant. The special case of $f(x)$-weakly-bounded fault model when the function $f(x)$ is defined as $f(x) = \alpha x$ is called the* fractional α-weakly-bounded fault model.

This model has been studied in [41] in connection with a simple class of non-adaptive algorithms. The fact that the definition uses $f(x)$-weakly-bounded model instead of $f(x)$-bounded model is not essential here. It has been chosen because the weakly-bounded model is easier to analyse and in some sense it balances the non-adaptivity restriction of the algorithms.

Although the greedy flooding algorithm is not optimal in the fractional α-weakly-bounded model, the model exhibits another undesirable property: If there is only one message sent in one time step, this message is guaranteed to be delivered. In some cases, the broadcasting algorithm can gain advantage from this unrealistic fact. Hence, a modification of the model has been proposed in [17] to avoid this property.

Definition 2.19 *Let $\alpha \in (0,1)$ and k be constants. The special case of $f(x)$-bounded fault model when the function $f(x)$ is defined as $f(x) = \max\{k-1, \alpha x\}$ is called the* (α, k)-fractional threshold *fault model.*

If k is equal to the edge connectivity of the underlying communication graph, we denote the model as α-fractional threshold *fault model.*

If there are many messages sent in one time step in this model, some fraction of them can be lost, exactly as in the case of the fractional model. On the other side, if there are too few messages sent, all of them can be lost, hence the delivery of any single message is never guaranteed.

Obviously, the threshold k can not be larger than the edge connectivity $c(G)$ of the underlying communication graph, otherwise it may not be possible to perform the broadcasting at all. In our work, we focus on the case when the value of k is maximal possible, i.e. $k = c(G)$. Hence, we shall deal only with the α-fractional threshold model, which has been analyzed in [42].

An interesting modification of the α-fractional threshold model is the case when α is infinitely close to 1, i.e. the case where the adversary has maximum power.

Definition 2.20 *Let $c(G)$ be the edge connectivity of the underlying communication graph. The special case of $f(x)$-bounded fault model when the function $f(x)$ is defined as $f(x) = \max\{c(G)-1, x-1\}$ is called the* simple threshold *fault model.*

In this model, at least $c(G)$ messages have to be sent in one time step to guarantee at least one delivery. Furthermore, only one delivered message is guaranteed in that case.

This model, which captures extremely harsh conditions for the broadcasting algorithms, has been analyzed in [17].

Both the α-fractional threshold model and the simple threshold model eliminate the above-mentioned undesirable features of the constant k-bounded model and the fractional α-weakly-bounded model. Hence, their introduction fulfills one of the goals of this thesis.

2.2.3 Time-dependent Dynamic Faults

As opposed to time-independent dynamic faults, some models define constraints on the number and type of failures depending on the failures that occurred in the past part of the computation. We call such fault models as models with *time-dependent dynamic faults*.

Time-dependent dynamic faults have been analyzed e.g. in [35]. This paper deals with dynamic faults in the constant one-port communication model. In such model, however, even one fault per time step renders the broadcasting impossible. This has been the main reason to introduce the *linearly bounded* model of dynamic faults in [35]: The model of faults is bounded, with omission faults and faulty links. Given a constant $0 < \alpha < 1$, the adversary can block at most αi messages during the first i time steps of the computation, for every natural number i. The broadcasting algorithm that works correctly in this model is called as the *α-safe broadcasting algorithm*.

Only non-adaptive algorithms are considered in [35]. In the sequel we summarize results proved in this paper.

Theorem 2.2.1 (Theorem 2.1 and Theorem 2.2 in [35]) *Consider α-safe broadcasting algorithm on line with n vertices for fixed constant α. If $0 < \alpha < \frac{1}{2}$, time $O(n)$ is sufficient for this algorithm. If $\frac{1}{2} \leq \alpha < 1$, time $\Omega\left((\frac{1}{1-\alpha})^n\right)$ is required.*

Theorem 2.2.2 (Theorem 3.1 and Theorem 3.2 in [35]) *Define the* star with $n+1$ vertices *(denoted as S_n) to be the tree with central node v_0 and nodes $v_1, \ldots, v_n$ adjacent to it (i.e. S_n is a complete n-ary tree of height 1). The time complexity of α-safe broadcasting on S_n is $\Theta\left(\left(\frac{1}{1-\alpha}\right)^n\right)$ for any fixed constant $0 < \alpha < 1$.*

Theorem 2.2.3 (Theorem 4.1 in [35]) *For any tree T with maximum degree bounded by a fixed constant d and any fixed constant $\alpha < \frac{1}{d}$, the time complexity of α-safe broadcasting on T is $O(D)$, where D is the diameter of T.*

Theorem 2.2.4 (Theorem 4.2 in [35]) *The time complexity of α-safe broadcasting on a ring with n vertices is $O(n)$ for any fixed constant $0 < \alpha < 1$.*

Theorem 2.2.5 (Theorem 5.1 in [35]) *The time complexity of α-safe broadcasting on a r-dimensional hypercube is*

$$\left\lfloor \frac{\alpha}{1-\alpha}(r-1) \right\rfloor + r$$

for any $0 < \alpha < 1$.

Theorem 2.2.6 (Theorem 5.2 in [35]) *The time complexity of α-safe broadcasting on a complete graph with n vertices is*

$$\frac{1}{\alpha}\log n + O(\log\log n)$$

for any fixed constant $0 < \alpha < 1$; $\log n$ denotes a base-2 logarithm of n.

Although the analysis of time-dependent models of faults might be interesting and non-trivial, the time-independent models seems to be more natural in most cases. In fact, the main reason for introducing the linearly bounded model was the fact that no time-independent model of faults was feasible for the chosen communication model. Hence, we focus on the time-independent models in the rest of this thesis.

Chapter 3

Constant k-bounded Model

This chapter presents some previous results about broadcasting in presence of dynamic faults. We focus on the constant k-bounded model of faults in connection with the constant multi-port communication model. This model is one of the oldest models of dynamic faults. It has been introduced in [55] and analyzed in numerous papers, e.g. [10,18–20,28,44,46,47].

In the constant k-bounded model, the adversary can block at most k messages in each time step. Since the faults are dynamic, the adversary is not bound by its preceding decisions. The model works with omission faults, so the adversary can not alter any message nor create fake messages.

It is easy to check that in the considered model, the following greedy non-adaptive algorithm is optimal: each node v that has received the message broadcasts the message to all its neighbors in every time step.

Claim 3.1 *The greedy non-adaptive broadcasting algorithm is optimal in the constant k-bounded model.*

Proof: Consider broadcasting in communication graph $G = (V, E)$. Let V_i^G be the set of vertices informed in the i-th time step of the greedy algorithm playing against an arbitrary adversary. Let A be an arbitrary broadcasting algorithm, we show that there exists a strategy for the adversary playing against the algorithm A such that $\forall i : V_i^A \subseteq V_i^G$, where V_i^A is the set of vertices informed in the i-th time step of the algorithm A. This means that it is possible for the adversary to keep fewer vertices informed in the algorithm A than in the greedy algorithm, which implies that the time complexity of the greedy algorithm is not higher than the complexity of the algorithm A.

The claim $V_i^A \subseteq V_i^G$ is easily proved by induction on i. The base case is trivial: $V_1^A = V_1^G = \{v_s\}$, where v_s is the source node of the broadcast. Now assume that $V_i^A \subseteq V_i^G$, the adversary of the greedy algorithm has blocked the set of edges E_i^G in the time step i and the algorithm A is sending messages via the edges E_i^A in the time step i. The adversary of the algorithm A can block edges $E_i^G \cap E_i^A$, which clearly ensures that $V_{i+1}^A \subseteq V_{i+1}^G$.

□

The above-mentioned implies that without loss of generality it is possible to use a non-adaptive algorithm with half-duplex links. Moreover, it is sufficient to analyze the non-adaptive greedy algorithm.

3.1 General Graphs

In this subsection we provide results about constant k-bounded model that are valid for general graphs. The results are parameterized by the number of faults k and the diameter d of the communication graph.

Theorem 3.1.1 (Theorem 1 in [10]) *Let d be a fixed constant. The broadcasting time in the constant k-bounded model is $O(k^{\frac{d}{2}-1})$ on all graphs G with edge connectivity at least $k+1$ and diameter equal to d.*

Theorem 3.1.2 (Theorem 2 in [10]) *Let k be a fixed constant. The broadcasting time in the constant k-bounded model is $O(d^{k+1})$ on all graphs G with edge connectivity at least $k+1$, where d is the diameter of the graph G.*

The above-mentioned results are asymptotically tight:

Theorem 3.1.3 (Theorem 3 in [10]) *Let d be a fixed constant. There exists a class of graphs $\{G_k \mid k \geq 1\}$ with diameter d such that the time of the broadcasting on G_k in the constant k-bounded model is $\Theta(k^{\frac{d}{2}-1})$.*

Theorem 3.1.4 (Theorem 3 in [10]) *Let k be a fixed constant. There exists a class of graphs $\{G_d \mid d \geq 1\}$ such that G_d has diameter at most d and the time of broadcasting on G_d in the constant k-bounded model is $\Theta(d^{k+1})$.*

3.2 Complete Graphs

The broadcasting in complete graphs in constant k-bounded model is completely solved (together with the case of Byzantine faults) in [44]:

Theorem 3.2.1 *Let K_n be a complete graph with n vertices. The following holds for the broadcasting time in the constant k-bounded model in the graph K_n:*

- *The broadcasting time equals to 1 if and only if $k = 0$.*
- *The broadcasting time equals to 2 if and only if $1 \leq k < \frac{n}{2}$.*
- *The broadcasting time equals to 3 if and only if $\frac{n}{2} \leq k < n + \frac{1}{2} - \sqrt{n + \frac{1}{4}}$.*
- *The broadcasting time equals to 4 if and only if $n + \frac{1}{2} - \sqrt{n + \frac{1}{4}} \leq k < n - 1$.*

- *The broadcasting is impossible if* $n - 1 \leq k$.

Proof: The cases when $k = 0$ and $k \geq n - 1$ are trivial. The remaining cases are proved in [44] in Theorem 1 and Theorem 2. □

3.3 Tori

The broadcasting in the constant k-bounded model in tori is discussed in [46] and [19]. The results from [46] are strengthened in [19] to the following form:

Theorem 3.3.1 (Theorem 1 in [19]) *Let* d, k *be integers such that* k *is even and one of the following holds:*

- $k \geq 6$ *and* $d \geq k + 4$
- $k \geq d$ *and* $d \geq 10$

The d*-dimensional torus of order* k*, denoted as* $C_{d,k}$*, has the edge connectivity equal to* $2d$ *and diameter* $\frac{dk}{2}$*. The broadcasting time in the constant* $(2d-1)$*-bounded model in* $C_{d,k}$ *is equal to* $\frac{dk}{2} + 2$*.*

3.4 Hypercubes

The broadcasting in the constant k-bounded model in hypercubes is considered e.g. in [18,20,28] and [47]. The results from these papers are summarized in the following theorem:

Theorem 3.4.1 *The* d*-dimensional hypercube, denoted as* H_d*, is a graph with both edge connectivity and diameter equal to* d*. The broadcasting time* $T(d, k)$ *in the constant* k*-bounded model in the graph* H_d *satisfies the following:*

1. *If* $k = d - 1$*, then* $T(d, k) = \begin{cases} 1 & d = 1 \\ 3 & d = 2 \\ d + 2 & d > 2 \end{cases}$.
2. *If* $k = d - 2$*, then* $T(d, k) = \begin{cases} 1 & d = 2 \\ 3 & d = 3 \\ d + 1 & d > 3 \end{cases}$.
3. *If* $k \leq d - 3$*, then* $T(d, k) = d$.

Proof:

1. The cases when $d \leq 2$ are trivial. It is proved in Theorem 3 of [47][1] that $T(d,k) \geq d+2$ for $d > 2$. The remaining inequality $d > 2 \Rightarrow T(d,k) \leq d+2$ is proved in the Theorem 4.1 of [18].

2. The cases when $d \leq 3$ are trivial. The case when $d > 3$ is proved in Theorem 1 of [20].

3. This result is proved in Theorem 1 of [20].

□

[1]The proof of this claim as presented in [47] needs a minor correction: faults need to be concentrated around $y(j)$ for last *two* rounds of the broadcast. This implies that the claim does not hold for $d \leq 2$.

Chapter 4

Fractional α-weakly-bounded Model

In this chapter, we consider the α-weakly-bounded model of faults in connection with the constant multi-port communication model, which has been proposed in [41] for $\alpha = \frac{1}{2}$. In this model, at most $\lfloor \alpha m \rfloor$ messages can be lost in every time step, where m is the number of messages destined to uninformed vertices in a particular time step. Hence, the number of faults can grow with the number of messages sent, what means that sending excessive number of messages is no longer "for free" as in the constant k-bounded model. The main motivation for introducing the fractional α-weakly-bounded model was to avoid exactly this undesirable property of the constant k-bounded model.

In [41], the performance of the greedy non-adaptive algorithm used in the previous section is analyzed. Contrary to the constant k-bounded model, the greedy algorithm is not always optimal in the α-weakly-bounded model. However, this algorithm is appealing due to its simplicity and uniformity.

The example of non-optimality of the greedy algorithm is depicted on Figure 4.1. Let the broadcast start from vertex v. Let us suppose that the adversary blocks at most one half of active edges in each step. Then the broadcasting time of the greedy algorithm is 5. However, an algorithm which in the fist step tries to send a single message from v to w (this message cannot be blocked by the adversary), yields a broadcasting time 4.

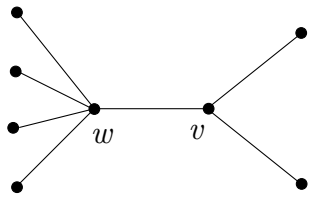


Figure 4.1: An example of non-optimality of the greedy broadcasting algorithm.

It is easy to see that there is no difference between half-duplex and full-duplex links in the considered model, because only links between informed and uninformed vertices matter.

The main interest of [41] is the following question: In which graphs is the broadcasting time of the greedy algorithm asymptotically equal to the time of the broadcasting without

faults (i.e. to the diameter of the communication graph)? It is shown that in complete graphs and complete trees this is not the case, but in the d-dimensional tori for fixed d it is.

The only value of α considered in [41] is 1/2. In this section, we present results of this paper and generalize them for arbitrary values of α, hence fulfilling the first goal of this thesis announced in the Foreword.

4.1 Results for $\alpha = 1/2$

The following theorem shows that the broadcasting time of the greedy algorithm in complete graphs is asymptotically slower than the time of broadcasting without faults. More precisely, the time of broadcasting on complete graph K_n increases from $\Theta(1)$ to $\Theta(\log n)$.

Theorem 4.1.1 (Theorem 1 in [41]) *Let T be the broadcasting time of the greedy algorithm in the $\frac{1}{2}$-weakly-bounded model on the complete graph K_n. The following inequality holds:*[1]

$$\lceil \log(n+1) \rceil - 1 \leq T \leq \lfloor \log(n-1) \rfloor + 1$$

The situation in complete trees is similar. The broadcasting time of the greedy algorithm on the complete d-ary tree of height h in presence of fractional 1/2-weakly-bounded faults is $\Omega\left((d \log d - c)^{\frac{h}{d}}\right)$ for fixed d and some constant c as opposed to $\Theta(h)$ in a fault-free scenario. The difference is even easier to see for the case $d = 2$, where the broadcasting time in presence of faults is at least $\frac{3}{2-\sqrt{2}} 2^{\frac{h}{2}} - O(h)$.

Theorem 4.1.2 (Theorems 2 and 3 in [41]) *Let T be the broadcasting time of the greedy algorithm in the $\frac{1}{2}$-weakly-bounded model on the complete d-ary tree of height h. The following fact holds:*

$$T = \Omega\left((d \log d - c)^{\frac{h}{d}}\right),$$

where $c = \log 3 - 1$ and d is fixed. Furthermore, if $d = 2$ (i.e. the tree is binary), the following inequality holds:

$$T \geq \frac{3}{2-\sqrt{2}} 2^{\frac{h}{2}} - O(h).$$

Unlike complete graphs and complete trees, the broadcasting in fractional α-weakly bounded model can not be slowed down asymptotically in multidimensional tori.

Theorem 4.1.3 (Theorem 6 in [41]) *Let d be a fixed integer. The broadcasting time of the greedy algorithm in the $\frac{1}{2}$-weakly-bounded model on the d-dimensional torus of order k for an even integer k and fixed d is $\Theta(k)$.*

[1] $\log x$ denotes a base-2 logarithm of x.

4.2 Results for Arbitrary α

All three above-mentioned theorems can be generalized for arbitrary value $\alpha \in (0, 1)$. In this section, we present the generalized theorems together with their proofs.

4.2.1 Complete Graphs and Complete Trees

In this subsection, we present results about complete graphs and complete trees. These results imply that, regardless of the value of α, complete graphs and complete trees are not robust in the fractional α-weakly bounded model in the sense that broadcasting time on these topologies can be slowed down asymptotically.

At first, we show the result for complete graphs.

Theorem 4.2.1 *Let T be the broadcasting time of the greedy algorithm in the α-weakly-bounded model on the complete graph K_n. The following inequality holds:*

$$\left\lceil \log_{1/\alpha}\big(n(1-\alpha)+\alpha\big) \right\rceil \leq T \leq \left\lfloor \log_{1/\alpha}(n-1) \right\rfloor + 1$$

Proof: Consider a broadcast $\{w\} = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_t = V$ and let $|V_i| = a_i$. With a_i informed vertices in step i, there are $a_i(n - a_i)$ active edges.

For the upper bound, note that the adversary may block at most $\lfloor \alpha a_i(n - a_i) \rfloor$ of them and the rest $\lceil (1 - \alpha)a_i(n - a_i) \rceil$ active edges deliver their messages to some uninformed vertices. However, only a_i of these messages may be destined to one particular vertex and it follows that a_{i+1} is at least[2] $a_i + \left\lceil \frac{\lceil (1-\alpha)a_i(n-a_i) \rceil}{a_i} \right\rceil = a_i + \lceil (1 - \alpha)(n - a_i) \rceil$.

On the other hand, there is a strategy for the adversary to obtain equality $a_{i+1} = a_i + \lceil (1 - \alpha)(n - a_i) \rceil$ – just block the edges in such a way as to inform as few vertices as possible.

So we have

$$a_i + (1-\alpha)(n-a_i) \leq a_{i+1} = a_i + \lceil (1-\alpha)(n-a_i) \rceil \leq a_i + (1-\alpha)(n-a_i) + 1$$

By solving recurrent inequalities

$$\begin{array}{ccccc} & & a_0 & = & 1 \\ n(1-\alpha) + a_i\alpha & \leq & a_{i+1} & \leq & 1 + n(1-\alpha) + a_i\alpha \end{array}$$

we obtain

$$n(1-\alpha^i) + \alpha^i \leq a_i \leq n(1-\alpha^i) + \frac{1-\alpha^{i+1}}{1-\alpha}.$$

Let $\{w\} = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_t = V$ be a broadcast on K_n with maximal length, i.e. t be a broadcast time on K_n. It holds that $n - 1 \geq a_{t-1} \geq n(1 - \alpha^{t-1}) + \alpha^{t-1}$ and $n \leq a_t \leq n(1 - \alpha^t) + \frac{1-\alpha^{t+1}}{1-\alpha}$. These inequalities imply that

$$\left\lceil \log_{1/\alpha}\big(n(1-\alpha)+\alpha\big) \right\rceil \leq t \leq \left\lfloor \log_{1/\alpha}(n-1) \right\rfloor + 1.$$

[2]Note that for an integer a and any x it holds that $\left\lceil \frac{\lceil x \rceil}{a} \right\rceil = \lceil \frac{x}{a} \rceil$.

□

In the rest of this subsection we focus on the broadcasting time in complete d-ary trees. We denote the complete d-ary tree of height n as $T_n^{(d)}$ and we assume that the broadcast is started from the root r of the tree. We show that the complete d-ary trees are another example of graphs where the broadcasting may be slowed down asymptotically for arbitrary value of α.

First we consider the case for $d = 2$ (i.e. complete binary trees) and $\alpha = \frac{1}{2}$, what is exactly the case of the second claim of Theorem 4.1.2. In the sequel, we present slightly improved version of this claim. Although the presented proof is very similar to the proof of Theorem 2 in [41], it is based on the same idea as the generalized result, and allows us to present its idea without diving into the technical details. For this reason, we present this proof here, too.

Theorem 4.2.2 *The broadcasting time for a complete binary tree $T_n^{(2)}$ and $\alpha = \frac{1}{2}$ is at least $\frac{\sqrt[3]{3}}{2\left(\sqrt[3]{3}-1\right)} 3^{\frac{n}{3}} - O(n)$.*

Proof: The aim of the proof is to present a strategy for the adversary and to analyze its performance. The strategy is "conservative" in a sense that whenever the adversary chooses to block an edge in a given step, he tries to keep this edge blocked in all subsequent steps until there is no other choice. The start of a broadcast is depicted on Figure 4.2: in time step 0 only the root is informed. In the first step, there are two active edges and the adversary blocks one of them, so there is one vertex which gets informed in step 1 and in all subsequent steps the adversary tries to block the left son of the root. Hence in step 2, with three active edges, there are two vertices which receive the information. The situation in the next step is as follows: there are two trees of height $n - 2$, e.g. 5 active edges, and one active edge which must be blocked by all means.

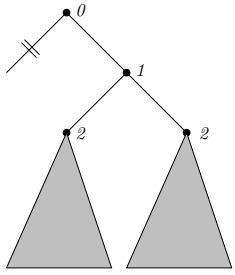


Figure 4.2: First steps of broadcast on a binary tree.

In order to present the general adversary strategy, let us consider a situation during the broadcast in which the uninformed vertices form $k + l$ complete binary trees. The root vertex of each tree is connected with an active edge to its informed parent. We call active edges from the root vertices of the first k of these trees *external edges*; they must be blocked whenever possible. Active edges that are not external are called *internal edges*.

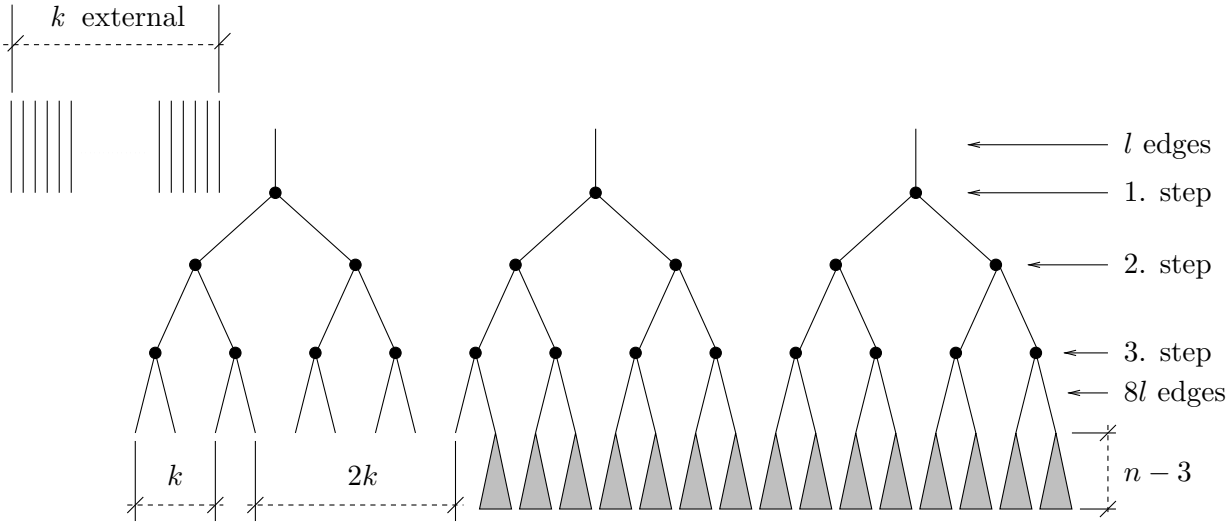


Figure 4.3: Adversary strategy in binary trees.

The strategy for the adversary is as follows: initially, there are $k+l$ active edges, $l \geq k$ and all l internal active edges lead to binary trees of height n. Three steps are taken by blocking only the k external edges, resulting in a situation with $8l$ internal and k external active edges (Figure 4.3). Now the same game is played with $4k$ external, $8l-3k$ internal edges and height $n-3$. When this broadcast is finished, the same game with $2k$ internal and $2k$ external edges and finally with k internal and k external edges is played. Note that in all three recursive calls the condition $l \geq k$ is preserved.

If $T(k,l,n)$ denotes the broadcasting time on k external, l internal edges and trees of height n, then it holds

$$T(k,l,n) = 3 + T(4k, 8l-3k, n-3) + T(2k, 2k, n-3) + T(k,k,n-3), \quad \text{for} \quad n \geq 2.$$

Now let's compute the term $T(k,l,n)$. As it does not depend on k,l it holds

$$T(k,l,n) = T(n) \geq 3\sum_{i=0}^{h-1} 3^i = 3\frac{3^h-1}{2}, \quad \text{where} \quad h = \lfloor n/3 \rfloor.$$

The whole adversary strategy on a tree of height n is as follows: first block one of the two active edges and use the above strategy for $k=1$, $l=1$ and height $n-1$. Then let the message spread along the single blocked edge leaving an analogous situation for a tree of height $n-1$. The resulting time is thus:

$$T'(n) = \sum_{i=1}^{n} 1 + T(n-i)$$

So we get

$$T'(n) \geq n + 3\sum_{i=1}^{n} \frac{3^{\frac{n-i-2}{3}}-1}{2} \geq \frac{3}{2}\sum_{i=0}^{n-3} 3^{\frac{i}{3}} - O(n)$$

Noting that $\sum_{i=0}^{n-3} 3^{\frac{i}{3}} = \frac{3^{\frac{n-2}{3}}-1}{\sqrt[3]{3}-1}$ yields the result. □

This strategy is easily generalized to d-ary trees in the following way:

Theorem 4.2.3 *Consider a complete d-ary tree $T_n^{(d)}$ of height n. Let $A = \lceil 1/\alpha \rceil$ and a be integer constants such that $a \geq \frac{\log(A+1)}{\log d}$. Let $p = \lfloor \log_A (1 + (A-1)d^a) \rfloor$. The broadcasting time is at least $a\frac{p^{\lfloor n/a \rfloor}-1}{(p-1)^2}$.*

Proof: In a similar way to the strategy in Theorem 4.2.2 we first describe a strategy on a forest of $k + l$ uninformed trees with k external and $l = k(A-1)$ internal edges. The adversary first spends a steps by blocking only the k external edges; at the end there are $l \cdot d^a = k(A-1)d^a$ internal edges.

Hence the $k\left(1 + (A-1)d^a\right) \geq kA^p$ active edges can be divided into $p+2$ groups $E_0, \ldots, E_p, E_r$, such that the group E_0 contains the $z_0 = k$ external edges, the group E_i, $1 \leq i \leq p$ contains $z_i = kA^i - kA^{i-1}$ edges and (possibly empty) E_r contains the rest, i.e. $z_{p+1} = k\left(1 + (A-1)d^a\right) - kA^p$ edges.

In the sequel, the adversary plays the game with groups $E_0, \ldots, E_{i-1}$ as external and E_i as internal edges for each i, starting with $i = p$ and ending with $i = 1$. Edges in group E_r are ignored; they are never blocked. Because $\sum_{j=0}^{i-1} z_j = kA^{i-1}$ and $z_i = kA^{i-1}(A-1)$ holds for each i, the condition $l = k(A-1)$ is preserved in all recursive calls.

Using this strategy, the total time of broadcast with k external and $l = k(A-1)$ internal edges is:

$$T(k, l, n) = a + \sum_{i=1}^{p} T\left(\sum_{j=0}^{i-1} z_j, z_i, n-a\right)$$

Again, this term does not depend on k,l so we get a bound

$$\begin{aligned} T(n) &= a + \sum_{i=1}^{p} T(n-a) = a + pT(n-a) && \text{for } n \geq a \\ T(n) &= n && \text{for } n < a \end{aligned}$$

Since $a \geq \frac{\log(A+1)}{\log d}$ it holds that $p \geq 2$. Hence, by solving the recurrence we have $T(n) \geq a\frac{p^{\lfloor n/a \rfloor}-1}{p-1}$.

The main strategy starts with a steps without blocking any edge; this yields $d^a \geq A$ active edges. Now the above described procedure is applied with 1 external and $A-1$ internal edges, the remaining active edges are ignored. Next, the same is repeated for height $n-a$, i.e. for the total time of the broadcast $T'(n)$ it holds that

$$T'(n) = \sum_{i=1}^{\lfloor n/a \rfloor} a + T(n - ai).$$

After substituting $T(n-ai)$, solving and simplifying, one gets the following:

$$T'(n) \geq a\left\lfloor \frac{n}{a} \right\rfloor \left(1 - \frac{1}{p-1}\right) + \frac{a}{p-1} \sum_{i=1}^{\lfloor n/a \rfloor} p^{\lfloor n/a \rfloor - i} \geq a\frac{p^{\lfloor n/a \rfloor} - 1}{(p-1)^2}$$

□

Corollary 4.2.4 *Let $A = \lceil \frac{1}{\alpha} \rceil$ be fixed integer constant. The broadcasting time on a complete d-ary tree $T_n^{(d)}$ of height n is $\Omega\left(2^{\sqrt[n]{\lceil \log(A+1)/\log d \rceil}}\right)$.*

Proof: Consider the notation from the preceding theorem. Let $a = \lceil \log(A+1)/\log d \rceil$. Obviously a meets the requirement of the theorem. Applying the previous theorem yields:

$$T'(n) \geq a\frac{p^{\lfloor n/a \rfloor} - 1}{(p-1)^2} = \Omega\big(p^{n/a}\big)$$

Since $p \geq 2$ it holds that

$$T'(n) = \Omega\big(2^{\sqrt[n]{\lceil \log(A+1)/\log d}}\big).$$

□

4.2.2 Tori

The rest of this section is devoted to examples of graphs in which the adversary cannot slow down the broadcasting asymptotically. We show that multidimensional tori of even order are such graphs for arbitrary values of α. This result is a generalization of Theorem 4.1.3, originally presented in [41] as Theorem 6.

[41] deals also with the case of infinite tori. The infinite d-dimensional torus (or equivalently the infinite d-dimensional mesh) is a straightforward modification of the finite d-dimensional torus: the set of vertices is the Cartesian product of $\mathbb{Z}$ instead of $\mathbb{Z}_k$ (hence $V = \mathbb{Z}^d$) and an edge connects vertices $[x_1, \ldots, x_i, \ldots, x_d]$, $[x_1, \ldots, x_i + \Delta, \ldots, x_d]$ for each $i \in \{1 \ldots d\}$, $\Delta \in \{-1, 1\}$. For the sake of completeness, we present the generalization of the analysis of infinite tori for arbitrary values of α, too. In fact, the analysis of infinite tori is simpler than the analysis of their finite counterparts, and can be used to demonstrate the core ideas of the analysis more easily.

Let us adopt the following notation. For a set $S \subseteq V$ denote by $S^{(r)}$ the r-neighborhood, i.e. $S^{(r)} = \{x \in V \mid dist(x, S) \leq r\}$ and denote by ∇S the vertex boundary, i.e. $\nabla S = S^{(1)} - S$. For a vertex v denote by $\mathcal{B}_r(v)$ the ball with radius r centered at v, i.e. $\mathcal{B}_r(v) = \{v\}^{(r)}$.

In the sequel we shall use the following isoperimetric inequality due to Bollobás and Leader:

Theorem 4.2.5 *[8] Let k be even and let A be a subset of $\mathbb{Z}_k^n$ with*

$$|A| = |\mathcal{B}_r(\mathbf{0})| + \beta|\nabla\mathcal{B}_r(\mathbf{0})|,$$

where $0 \leq \beta < 1$. Then

$$|A^{(1)}| \geq |\mathcal{B}_{r+1}(\mathbf{0})| + \beta|\nabla\mathcal{B}_{r+1}(\mathbf{0})|.$$

Furthermore, this theorem holds also for infinite tori. Denote $B_r = |\mathcal{B}_r(\mathbf{0})|$ and $\nabla B_r = |\nabla\mathcal{B}_r(\mathbf{0})|$. Now let us consider a broadcast $\{w\} = V_0 \subseteq V_1 \subseteq \cdots$ in a d-dimensional torus. The following lemma shows the limitation of the adversary: regardless of his strategy, the number of informed vertices increases.

Lemma 4.2.6 *Let* $A = 1/\left(1 + 2\frac{\alpha}{1-\alpha}d\right)$ *and* $|V_i| = B_r + \beta_i\nabla B_r$ *for some* r, $0 \leq \beta_i < 1$. *Then* $|V_{i+1}| = B_r + \beta_{i+1}\nabla B_r$ *such that*

$$\beta_{i+1} \geq \beta_i\left(1 + A\frac{\nabla B_{r+1}}{\nabla B_r} - A\right) + A$$

Proof: Suppose that in the i-th step the adversary blocks z vertices from ∇V_i. Clearly, there must be at least one edge for each of them in ∂V_i. Since the adversary may block at most $\alpha|\partial V_i|$ edges, it holds that $z \leq \alpha|\partial V_i|$, hence there are at least $|\partial V_i|(1-\alpha) \geq z\frac{1-\alpha}{\alpha}$ edges not blocked in i-th step.

Because the degree of each vertex is $2d$, for each not blocked vertex from ∇V_i there are at most $2d$ not blocked edges from ∂V_i. As there are $|\nabla V_i| - z$ vertices not blocked in i-th step, it follows that $(|\nabla V_i| - z)2d \geq z\frac{1-\alpha}{\alpha}$ and hence $z \leq |\nabla V_i|/\left(1 + \frac{1}{2d}\frac{1-\alpha}{\alpha}\right)$.

Using that $\nabla V_i = V_i^{(1)} - V_i$ and $\nabla B_r = B_{r+1} - B_r$ and applying Theorem 4.2.5 we get $|\nabla V_i| \geq \nabla B_r + \beta_i(\nabla B_{r+1} - \nabla B_r)$. It is obvious that $|V_{i+1}| = |V_i| + (|\nabla V_i| - z) \geq |V_i| + |\nabla V_i|A$, hence we have

$$|V_{i+1}| \geq B_r + \beta_i\nabla B_r + (\nabla B_r + \beta_i(\nabla B_{r+1} - \nabla B_r))\, A$$

After regrouping and using $|V_{i+1}| = B_r + \beta_{i+1}\nabla B_r$ we get that

$$\beta_{i+1} \geq \beta_i + A + A\beta_i\frac{\nabla B_{r+1}}{\nabla B_r} - A\beta_i = \beta_i\left(1 + A\frac{\nabla B_{r+1}}{\nabla B_r} - A\right) + A$$

□

Now we need to bound the term $\nabla B_{r+1}/\nabla B_r$ by a positive constant in order to show that the adversary may slow down the broadcasting only by a constant factor. In infinite tori it is immediate:

Theorem 4.2.7 *Consider a faulty broadcast in an infinite torus of fixed dimension* d. *Then in each step* r *the number of informed vertices is* $\Theta(B_r)$.

Proof: First note that in an infinite torus it holds $\nabla B_{r+1} > \nabla B_r$ for all r and hence by Lemma 4.2.6 $\beta_{i+1} \geq \beta_i + A$. This means that in any faulty broadcast at most $\left\lceil\frac{1}{A}\right\rceil = \left\lceil 1 + 2\frac{\alpha}{1-\alpha}d\right\rceil = A'$ steps are needed to ensure that $\beta_{i+A'} \geq 1$. Hence it holds $|V_{A'r}| \geq B_r$ for all r. Since B_r is a polynomial in r of degree d, the fraction $B_{A'r}/B_r$ converges to a constant (depending on d). □

For the upper bound on the broadcasting time on finite grids some more reasoning is necessary:

Lemma 4.2.8 *In a d-dimensional torus of an even order k it holds that $\nabla B_r \leq (2d-1)\nabla B_{r+1}$ for any $r < \frac{dk}{2}$.*

Proof: From the definition of ∇B_r follows that $\nabla B_r = |\nabla \mathcal{B}_r|$, where $\nabla \mathcal{B}_r$ is the set of vertices of the grid with distance exactly $r+1$ from the vertex $\mathbf{0}$.

Consider any vertex v from $\nabla \mathcal{B}_{r+1}$. As its degree is $2d$, there are at most $2d-1$ edges incident to v that leads to vertices from $\nabla \mathcal{B}_r$, hence there are at most $(2d-1)\nabla B_{r+1}$ edges between $\nabla \mathcal{B}_r$ and $\nabla \mathcal{B}_{r+1}$. Since each vertex from $\nabla \mathcal{B}_r$ is incident with at least one edge leading to $\nabla \mathcal{B}_{r+1}$, it holds that $\nabla B_r \leq (2d-1)\nabla B_{r+1}$.

□

Note that in a d-dimensional torus of an even order k there is only one vertex with distance $\frac{dk}{2}$ from the vertex $\mathbf{0}$ and no vertex with greater distance from $\mathbf{0}$. Hence the Lemma 4.2.6 is applicable during the whole broadcast:

Theorem 4.2.9 *In a d-dimensional torus of even order k the broadcasting time is $\Theta(k)$ for fixed d.*

Proof: The proof is similar as in the infinite grid case. From Lemma 4.2.8 it follows that $\frac{\nabla B_{r+1}}{\nabla B_r} \geq \frac{1}{2d-1}$. By using Lemma 4.2.6 we obtain the following:

$$\beta_{i+1} \geq \beta_i \left(1 - A + \frac{A}{2d-1}\right) + A; \quad A = \frac{1}{1 + 2\frac{\alpha}{1-\alpha}d}$$

Let $C = 1 - A + \frac{A}{2d-1}$. Since $C > 0$, by solving the recurrence (with boundary condition $\beta_0 = 0$) we get:

$$\beta_i \geq \frac{2d-1}{2d-2}\left(1 - C^i\right) = \frac{2d-1}{2d-2}\left(1 - \left(\frac{4\alpha d^2 - 2\alpha d - \alpha + 1}{4\alpha d^2 + 2(1-2\alpha)d + \alpha - 1}\right)^i\right)$$

Solving for $\beta_i \geq 1$ we get $C^i \leq \frac{1}{2d-1}$. Since $C < 1$, it yields:

$$i = \left\lceil \frac{\log(2d-1)}{-\log C} \right\rceil = \left\lceil \frac{\log(2d-1)}{\log(4\alpha d^2 + 2(1-2\alpha)d + \alpha - 1) - \log(4\alpha d^2 - 2\alpha d - \alpha + 1)} \right\rceil$$

Hence if there are at least B_r informed vertices, after i steps there will be at least B_{r+1} informed vertices, so the broadcasting time is at most $i\frac{dk}{2}$. Because d is constant the result follows. □

Chapter 5

Simple Threshold Model

In the previous chapter, we have discussed broadcasting in the fractional α-weakly-bounded model of faults. However, this model has a serious shortcoming: if there are less than $1/\alpha$ messages sent in one time step, none of them can be lost. The broadcasting algorithm can take advantage of this feature, as can be seen on the example of non-optimality of the greedy algorithm (see Figure 4.1).

The above-mentioned undesirable property of the α-weakly-bounded model of faults was the reason for introducing the (α, T)-*fractional threshold* model in [17] (see Definition 2.19). In this model of faults, at most $\max\{T-1, \alpha x\}$ messages may be lost in any time step, where x is the total number of messages sent in that time step. Hence, if the number of messages sent is less than the given threshold T, all of them may be lost. However, if there are many messages sent, some constant fraction α of them may be destroyed.

The paper [17] focused on the analysis of adaptive broadcasting algorithms in the most restrictive setting of the fractional threshold model. It is easy to see that maximal reasonable value of T is the edge connectivity of the communication graph (denoted as $c(G)$) minus one. Indeed, if $T \geq c(G)$, the adversary can cut the graph into disconnected components and the broadcasting is not possible at all. On the other side, the value of α can be arbitrarily close to 1, hence there is no single "worst-case" setting for the fractional threshold model. For this reason, so called *simple threshold* model of faults has been introduced in [17] (see Definition 2.20), which represents the case of α infinitesimally close to 1. In this model, at least $c(G)$ messages have to be sent such that delivery of at least one of them is guaranteed. Furthermore, no other guarantees are provided by the model: If there are more than $c(G)$ messages sent, all but one of them may be lost. Moreover, the model provides no fault-detection ability. If a node sends some messages, it can not detect which of them have been delivered. For sure, the destination node can send an acknowledgement in the next time step, but such acknowledgement is just an ordinary message and can be lost.

The simple threshold model of faults is extremely harsh for any distributed algorithm. In fact, it is not easy to see if it is possible to perform the broadcasting in this model at all. However, several surprisingly positive results have been presented in [17]: Not only it is

always possible to finish the broadcasting, but it is possible to do so fast (i.e. in polynomial time) for many topologies.

The complete overview of the results proven in [17] can be found in Table 5.1. The paper deals with rings, complete graphs, hypercubes and arbitrary communication graphs with various topology knowledge. Presented results prove that the broadcasting can be performed in polynomial time on any communication graph with edge-connectivity bounded by $O(\log n)$, where n is number of its vertices. Many interesting topologies, such as hypercubes, butterfly graphs, multidimensional tori with fixed dimension, etc., satisfy this requirement. Furthermore, all presented algorithms provide explicit termination, i.e. when the algorithm terminates at an entity, it will not process any more messages (and, in fact, no messages should be arriving anyway).

Topology	Condition	Time complexity
ring	n not necessarily known	$\Theta(n)$
complete graph	with chordal sense of direction	$O(n^2)$
complete graph	unoriented	$\Omega(n^2)$, $O(n^3)$
hypercube	oriented	$O(n^2 \log n)$
hypercube	unoriented	$O(n^4 \log^2 n)$
arbitrary network	full topology knowledge	$O(2^{c(G)}nm)$
arbitrary network	no topology knowledge except $c(G)$, n, m	$O(2^{c(G)}m^2n)$

Table 5.1: Summary of results presented in this chapter. The communication graph G has n vertices, m edges and is $c(G)$-edge-connected.

In this chapter, we present full proofs of the results of [17], what contributes to the completion of the third goal of this thesis. As in [17], we assume the constant multi-port communication model, presented algorithms are adaptive, and they work in the wake-up synchronization mode (recall Section 2.1.2).

5.1 Preliminaries

In the following section, we introduce some additional terminology used in this chapter, as well as techniques common to all presented results.

Algorithms presented in this chapter rely heavily on the fact that the vertices can distinguish their neighbors, hence we can assume that each vertex knows distinct local identifiers of its incident links. We call these labels as *ports*.

Given the non-acknowledged message-passing communication, and the fact that vertices can use only ports to distinguish neighbors, it is often convenient to consider a directed graph G' obtained from the communication graph G by replacing each edge by two opposite *arcs* (i.e. oriented edges). We sometimes abuse the notation and use G and G' interchangeably, and speak about sending a message along an arc.

Results presented in this chapter consider the case of anonymous networks. All our algorithms, however, construct unique vertex identifiers. The initiator is used to break symmetry, and vertices are assigned names based on the messages they receive. One possible way to do so is to use the sequence of ports used to reach a vertex v from the initiator as the identifier of v. There are usually many such sequences, but it is possible to assign one of them as the identifier of v at the moment v is informed.

Once the vertex identifiers are available, it is easy to construct arc identifiers: an arc is described by the vertex it is outgoing from, and the corresponding port. In the sequel we shall sometimes abuse notation and identify vertices (arcs) with their identifiers.

5.2 Rings

The ring is a 2-connected network, i.e. $c(G) = 2$. Hence, at least two messages must be sent in a time step to ensure that at least one of them is delivered.

We first present the algorithm assuming the ring size n is known, and then show how it can be extended to the case when n is unknown.

At any moment of time, the vertices can be either *informed* or *uninformed*. Since the information is spreading from the single initiator vertex s, informed vertices form a connected component. The initiator splits this component into the left part and the right part. Each informed vertex v can easily determine whether it is on the left part or on the right part of the informed component – this information can be delivered in the message that informs the vertex v.

The computation is organized in phases where each phase takes four communication rounds. In each phase, an informed vertex can be either *active* or *passive*. A vertex is active if and only if it has received, in previous phases, messages from only one of its neighbors. A passive vertex has received a message from both neighbors. This implies that, as long as the broadcast has not yet finished, there is at least one active vertex in both left and right part of the informed component (the left-most and the right-most informed vertices must be active; note, however, that also the intermediate vertices might be active).

The computation consists of $n-1$ phases. The goal of a phase is to ensure that at least one active vertex becomes passive. Each phase consists of the following four steps, which are formally defined in Algorithm 1:

1. Each active vertex sends a message to its possibly uninformed neighbor.
2. Each active vertex in the right part sends a message to its possibly uninformed neighbor. Each vertex in the left part that received a message in step 1 replies to this message.[1]
3. Same as step 2, but left and right parts are reversed.
4. Each vertex that received a non-reply message in steps 1–3 replies to that message.

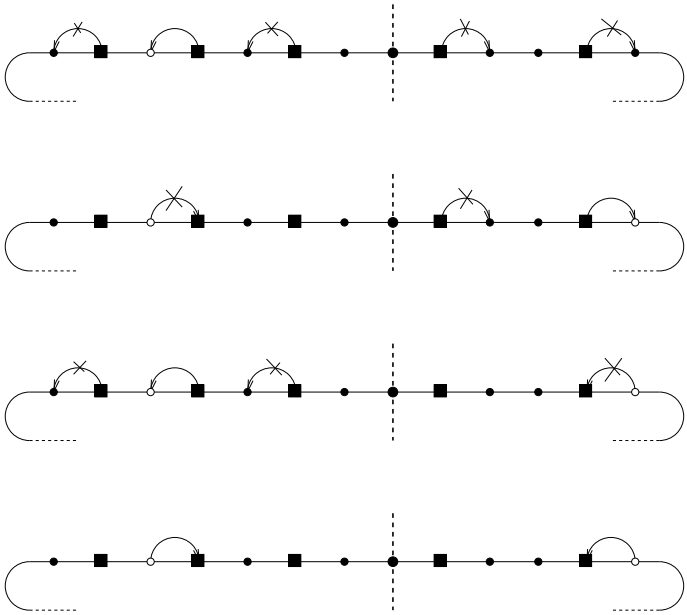


Figure 5.1: A sample phase of Algorithm 1. The initially active vertices are squares. In the first step, the only surviving message is delivered to the white vertex on the left, which then replies in the next step. At the end of the phase, two white vertices are sending acknowledgement, so at least one acknowledgement is delivered during that phase.

To avoid corner cases at the initiator of the broadcast, the initiator is split into two virtual vertices such that each of them starts in active state (i.e. the initiator acts as if it belonged both to the left and to the right part).

Lemma 5.2.1 *At least one reply message is delivered during each phase.*

Proof: Since there is at least one active vertex in both the left and the right part of the informed component, at least two messages are transmitted in step 1 and thus at least one of them is delivered. Assume that this message passed in the left part. Then there will be at least two messages transmitted in step 2 leading to different vertices (the reply message in the left part, and the discover message in the right part) and therefore at least one message passes in step 2 (or step 3 if the delivered message of step 1 was in the right part). If no reply message passed in step 2 (or 3), a discover message must have passed in the right (left) part. Therefore, at least two replies will be sent in step 4 and at least one of them will pass. □

Initially, there are two active (virtual) vertices (the left- and right- part of the initiator). Lemma 5.2.1 ensures that during each of the subsequent phases, at least one previously active vertex becomes passive. Since passive vertices never become active again, it follows that after at most $n-1$ phases, there are $n-1$ passive vertices. Once there are $n-1$ passive

[1]Note that passive vertices reply to such message, too.

Algorithm 1 Broadcasting in Rings.

```
1: state ∈ {uninformed, active, passive}
2: location ∈ {left, right}
3: dir-to-init ∈ Neigh

4: procedure PHASE
5:     time t:
6:     if state = active then send (discover, location) to opposite of dir-to-init
7:     end if

8:     time t + 1:
9:     if location = right then
10:        if state = active then send (discover, location) to opposite of dir-to-init
11:        end if
12:    else
13:        if received (discover) in time t then reply with (ack)
14:        end if
15:    end if

16:    time t + 2:
17:    if location = left then
18:        if state = active then send (discover, location) to opposite of dir-to-init
19:        end if
20:    else
21:        if received (discover) in time t then reply with (ack)
22:        end if
23:    end if

24:    time t + 3:
25:    if received (discover) in time t ... t + 2 then reply with (ack)
26:    end if

27:    if state = uninformed and received (discover, l) via dir in time t ... t + 3 then
28:        state := active;
29:        location := l;
30:        dir-to-init := dir;
31:    end if
32:    if state = active and received (ack) in time t ... t + 3 then state := passive;
33:    end if
34: end procedure
```

vertices, the remaining two must be informed (both are neighbors of a passive vertex), i.e. $n-1$ phases are sufficient to complete the broadcast.

Note also that the algorithm does not require distinct IDs or ring orientation (it can compute them, though, as it is initiated by a single vertex). Hence, we have proven the following theorem:

Theorem 5.2.2 *There is $4(n-1)$-time fault-tolerant broadcasting algorithm for (anonymous, unoriented) rings of known size.*

If n is unknown, Algorithm 1 cannot be directly used, as it does not know when to terminate. This is not a serious obstacle, though. Assume that the algorithm is run without a time bound, and each discover message also contains a counter how far is the vertex from the initiator. After at most n phases there will be a vertex v that has received discover messages from both directions. From the counters in those messages v can compute the ring size n. In the second part of the algorithm v broadcasts n (and the time since the start of the second broadcast) using the algorithm for known n; when that broadcast is finished, the whole algorithm terminates. In order to make this work, we have to ensure that there is no interaction between the execution of the first broadcast and the second broadcast. That can be easily accomplished by scheduling the communication steps of the first broadcast in odd time steps and the second broadcast in even time steps. Furthermore, we need to consider a special case when there are *two* vertices that receive discover messages from both directions in one time step. In this case, the second broadcast has two initiators. However, it is not difficult to see that our broadcasting algorithm works in this case, too.

Theorem 5.2.3 *There is an $O(n)$-time fault-tolerant broadcasting algorithm for (anonymous, unoriented) rings of unknown size.*

5.3 Complete Graphs

As the connectivity of complete graphs is $n-1$, we assume that least $n-1$ messages must be sent to ensure that at least one passes through.

5.3.1 Complete Graphs with Chordal Sense of Direction

Chordal Sense of Direction in complete graphs is a form of topology knowledge where the vertices are numbered $0, 1, \ldots, n-1$ and the link from vertex u to vertex v is labelled by $v - u \bmod n$ (see Definition 2.9 and Figure 2.2). The vertices do not necessarily need to know their ID, the link labels are sufficient. Indeed, any broadcasting algorithm running on complete graphs with chordal sense of direction can compute valid vertex identifiers in the following way: The initiator has label 0, and all other vertices are labelled by the edge label of the link by which they can be reached from the initiator. Note that each vertex, upon receiving a message, is able to compute its identifier even if the message was not sent from the initiator. Moreover, every vertex can compute the identifiers of all its neighbors.

The broadcasting algorithm consists of two parts. The purpose of the first part is to make sure that at least $\lceil n/2 \rceil$ vertices are informed; the second part uses these vertices to inform the remaining ones. The algorithm is executed by informed vertices. Each message contains a time counter, so a newly informed vertex can learn the time and join the computation at the right place.

The first part

of the algorithm consists of phases $0, 1, \ldots, \lceil n/2 \rceil - 2$. During phase 0 the initiator sends messages to all its neighbors. The goal of a phase k is to ensure that there are at least $k+1$ informed vertices distinct from the initiator; this ensures that after the first part, there are at least $\lceil n/2 \rceil$ informed vertices.

Consider a phase k and suppose that there are exactly k informed vertices distinct from the initiator at the beginning of phase k. Let $d = \lfloor \frac{n-1}{k+1} \rfloor$, and consider $k+1$ disjoint intervals $I_0, \ldots, I_k$ each of size d, consisting of non-initiator vertices. The phase will consist of $k+1$ rounds. The idea is that during the i-th round, the informed vertices (including initiator) try to inform an additional vertex in the interval I_i by sending messages to all vertices in I_i. If I_i does not contain any informed vertices, and at least one message is delivered, then a new vertex must be informed. The problem is, however, that only $d(k+1)$ messages are sent, which may not be sufficient to guarantee delivery. To remedy this, the i-th round will span over d steps. In a j-th step, all informed vertices send messages to all vertices in I_i and to the j-th vertex of $I_{i \oplus 1}$ (the addition is taken modulo $k+1$). Now, in each step there are $(k+1)(d+1)$ messages sent, so at least one must be delivered. Hence we can argue that, during phase k, a new vertex is informed if there is an interval I_i that does not contain any informed vertices, followed by interval $I_{i \oplus 1}$ that contains at least one non-informed vertex. However, the existence of such I_i follows readily from the fact that there are only k informed vertices distinct from the initiator and $d \geq 2$.

Algorithm 2 Complete graphs with chordal sense of direction - Part I.

```
1: the initiator sends message to all other vertices    // phase 0
2: for 1 ≤ k ≤ ⌈n/2⌉ − 2 do    // phase k
3:     d = ⌊(n−1)/(k+1)⌋
4:     for 0 ≤ i ≤ k do    // round i
5:         for 1 ≤ j ≤ d do    // step j
6:             all informed vertices send messages to vertices
7:             {di + 1, di + 2, ..., di + d, d((i + 1) mod(k + 1)) + j}
8:         end for
9:     end for
10: end for
```

Lemma 5.3.1 *After phase k there are at least $k+1$ informed vertices distinct from the initiator.*

Proof: By induction on k. The statement holds for phase 0, as the initiator sends $n-1$ messages and at least one of them will be delivered. Consider now the situation after phase $k-1$ with exactly k informed vertices distinct from the initiator. We show that at least one vertex will be informed during phase k.
Divide the interval $[1, n-1]$ into $k+2$ parts in such a way that each of the first $k+1$ parts is of size $d = \lfloor \frac{n-1}{k+1} \rfloor$. During step j of the round i of the phase k all informed vertices (including the initiator) send messages to all vertices in part i and to the j-th vertex of part $(i+1) \bmod (k+1)$. As $(k+1)(d+1) \geq n-1$ messages are sent, at least one of them is delivered. Hence it is sufficient to prove the following claim:

Claim 5.1 *There exists $0 \leq i \leq k$ such that there is no informed vertex in the i-th part and there is at least one non-informed vertex in the $(i \oplus 1)$-st part, where $\oplus$ is addition modulo $k+1$.*

Let there be exactly e parts $I_{z_1}, I_{z_2}, \ldots, I_{z_e}$ of size d containing no informed vertices. Assume that the claim does not hold, hence there are at least e disjoint parts $I_{z_1 \oplus 1}, I_{z_2 \oplus 1}, \ldots, I_{z_e \oplus 1}$ of size d containing only informed vertices. Since each of the remaining $k+1-2e$ parts contains at least one informed vertex, there are at least $ed + k + 1 - 2e$ informed vertices. So it must hold $ed + k + 1 - 2e \leq k$ which is equivalent to $e(d-2) + 1 \leq 0$. As $e \geq 0$, this yields $d = \lfloor \frac{n-1}{k+1} \rfloor < 2$, and hence $\frac{n-1}{k+1} < 2$. However, this contradicts to $k \leq \lceil \frac{n}{2} \rceil - 2$. □

Each phase k consists of $k+1$ rounds with d steps each, therefore every phase takes $O(n)$ time steps. Since there are $O(n)$ phases, the first part of the algorithm finishes in $O(n^2)$ time.

The second part

of the algorithm starts with at least $\lceil n/2 \rceil$ informed vertices and informs all remaining ones. The algorithm is as follows: consider all pairs $[i, j]$ such that $1 \leq i, j \leq n-1$, sorted in lexicographic order. In each step t, all informed vertices consider the t-th pair $[i, j]$ and send messages to vertices i and j. Since at least $2\lceil n/2 \rceil \geq n-1$ messages are sent, at least one of them is delivered. This ensures that a new vertex is informed whenever both i and j were uninformed. In this manner, all but one vertex can be informed (at any moment the two smallest uninformed vertices form a pair that has not been considered yet).

To inform the last vertex, all $n-1$ informed vertices send in turn messages to vertices $1, 2, \ldots, n-1$.

Theorem 5.3.2 *There is an $O(n^2)$ time fault-tolerant broadcasting algorithm for complete graphs with chordal sense of direction.*

Proof: The first part consist of $\lceil n/2 \rceil - 2$ phases, with each phase taking $O(n)$ time steps. The second part consists of $n(n-1)/2$ steps and informing the last vertex takes $n-1$ steps. □

Note that the algorithm did not exploit all properties of the chordal sense of direction; it is sufficient for the informed vertices to agree on the IDs of the vertices, and to be able

to determine the ID of the vertex on the other side of a link. These properties are fulfilled by the definition of a weak sense of direction from [24]. Without diving into the details, we mention that a local edge-labelling is a weak sense of direction if it admits so called coding function: a function that, given a sequence of edge labels, computes a local name in such a way that for any vertex v, and any two paths starting from v, these two paths lead to the same vertex exactly if the coding function returns the same local name. It is not difficult to see that any weak sense of direction enables the algorithm to compute appropriate vertex identifiers. Therefore, we get:

Corollary 5.3.3 *There is a $O(n^2)$ time broadcasting algorithm for complete graphs with weak sense of direction.*

5.3.2 Unoriented Complete Graphs

The algorithm in the previous section strongly relies on the fact that the vertices know the IDs of the vertices on the other side of the links. In this section, we use very different techniques to develop an algorithm that works for unoriented complete graphs (i.e. the only structural information available is the knowledge that the graph is complete; of course, local orientation – being able to distinguish incident ports – is also required).

The main idea of the algorithm is that of exploring an area. Informally, an area is a set of arcs that are known to have already delivered the message. Indeed, once the area is large enough, every vertex must have received the message. More formally, the knowledge of an area is described by a set of triples of the form (a, p, b), meaning that the port p of vertex a delivered a message to vertex b. This knowledge is stored in the vertices of the communication graph; the knowledge of an informed vertex u (i.e. the set of corresponding triples) is denoted as $K(u)$. Throughout the algorithm, the knowledge of the sending vertex v will always be piggybacked on every message sent by v. This means that if u and v are vertices such that at step t a message passed from u to v via the port p of u, and $K(u)$ and $K(v)$ are their knowledge at time t, then the knowledge of v at time $t+1$ is a superset of $K(u) \cup K(v) \cup \{(u, p, v)\}$.

An informal view of the algorithm is as follows (see Fig. 5.2): once a message arrives to a vertex v, v tries to expand the knowledge of the area by sending the message along "exploring" arcs (i.e. arcs that are not in the known area). Although the knowledge of v does not change in case of successful delivery, the knowledge of the destination vertex will contain the full knowledge of v plus the used exploring arc. Hence, the goal of the algorithm is to ensure delivery of the message along the exploring arcs.

To do so, v must make sure that at least $n-1$ messages are sent during each step. Hence, v might be forced to send also some messages along arcs that are in the known area. These messages are "helper" messages: upon the receipt of a helper message, a vertex v' sends messages along its exploring arcs, and may send additional helper messages. The algorithm is constructed in such a way that after the helper messages have been sent, the vertex continues to send the exploring messages for a certain number of steps. The timing

is chosen in such a way that, finally, only exploring messages are being sent, which ensures progress.

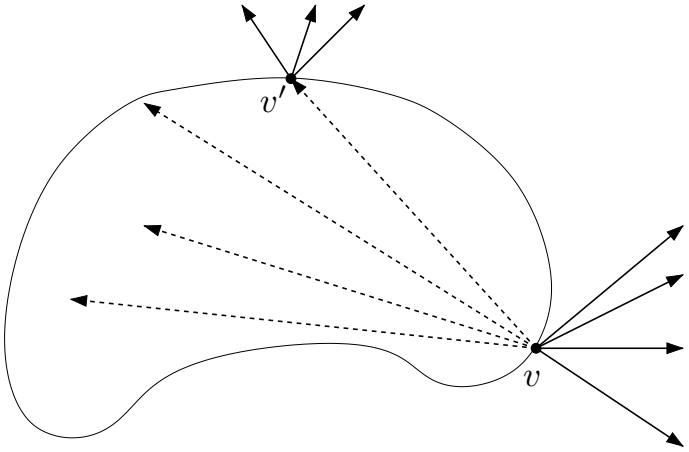


Figure 5.2: Main idea of the algorithm. Vertex v tries to expand the current area by sending exploring messages. However, in order to ensure that enough messages are sent, it sends also helper messages along arcs already included in the current area. If such helper message arrives to a vertex v', it starts sending exploring messages to help v reach the required number of messages. If the number is still not sufficient, v' may decide to send other helper messages.

This simple approach is complicated by the fact that in a given step there may be many independent initiators having (possibly intersecting) areas. Hence, care must be given to ensure that the computations of these multiple initiators do not collide. To remedy this situation, we introduce the "family tree" of messages. In our algorithm, a message is sent only as a response to another (exploring or helper) message (with the obvious exception of the first step). Hence, there is a natural notion of an ancestor of a message. An example of such message-tree is given on Figure 5.3.

Since a message always carries all available area knowledge, it is clear that when a particular message is sent, all arcs traversed by some ancestor of the message in the message-tree are in the known area.

Let T be any message. The *exploring ancestor* of a message T is the closest ancestor of T which is an exploring message.[2] The *helper tail* of message T is the path (i.e. sequence of messages) between the exploring ancestor of T and T itself. Note that all messages on the helper tail are helper messages except the first one.

We present a fault-tolerant broadcast algorithm that satisfies the following invariants:

I1 Let T be a message that is sent over an arc $\langle a, b\rangle$. If T is exploring, then it holds that no ancestor of T has been sent over $\langle a, b\rangle$. Conversely, if T is helper, there exists some ancestor of T that has been sent over $\langle a, b\rangle$.

[2]We assume that T is not an ancestor of itself, hence T and the exploring ancestor of T are always distinct.

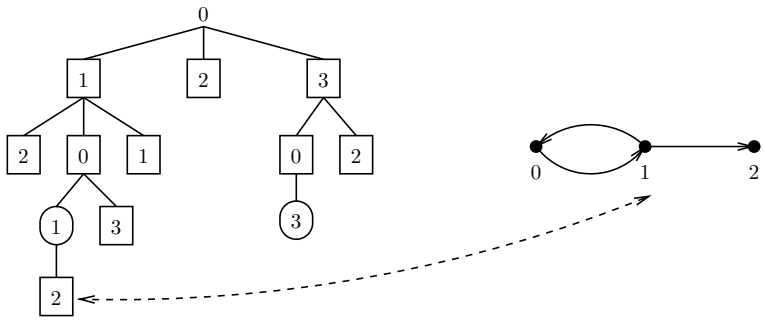


Figure 5.3: *Left part:* An example of a message-tree. Vertices are messages, and edges correspond to the ancestry relation. Messages can be either exploring (squares) or helper (circles). Numbers inside the vertices denote the nodes processing corresponding messages. *Right part:* The knowledge carried by the marked message.

I2 Let T be a helper message. Then the helper tail of T contains at most n vertices.

I3 Let T be a helper message. Then T is sent exactly one step later than the parent of T.

I4 Let T be an exploring message. Then T is sent at most $n + 1$ steps later than the exploring ancestor of T.

I5 Let T be an exploring message delivered in a step t. If the broadcast is not finished yet, at least one exploring message is delivered in some of the steps $t + 1, \ldots, t + n$.

These invariants imply that the broadcast completes in $O(n^3)$ time: the invariant *I5* ensures that the algorithm can not stop before the broadcast is finished. Consider a path from root to a leaf in the message-tree. Invariant *I1* ensures that the leaf is an exploring message and that there are at most $O(n^2)$ exploring messages on this path. Invariant *I4* implies that there are at most $n + 1$ consecutive helper messages on the path. Hence the overall time of the broadcasting algorithm is $O(n^3)$.

The algorithm works as follows. In the first step of the algorithm, the initiator sends exploring messages through all its outgoing arcs. All these messages are children of some virtual root message in the message-tree. In each subsequent step t, each vertex gathers all received messages in this step and processes them in parallel using the procedure PROCESS described in Algorithm 3.

If some processor should send more than one message through an arc in one step, it (arbitrarily) chooses one of them to send and discards the remaining ones.

Lemma 5.3.4 *The presented algorithm satisfies invariants I1, I2, . . . , I5.*

Proof:

Algorithm 3 Complete graphs without sense of direction.

procedure PROCESS(T) *// process message T*
  Let P be the set of all arcs outgoing out of the vertex processing T
  Let $A \subseteq P$ be the set of arcs that have never been traversed by any ancestor of T
  If $A = \emptyset$, the broadcast is finished.
  Let S be the set of vertices acting as a source of a helper message in the helper tail of T.
  Let $B \subseteq P - A$ be the set of arcs that lead to a vertex in S.
  Let $C = P - (A \cup B)$
  // Note that since only arcs already traversed by (an ancestor of) T
  // are considered, the vertex processing T can indeed compute B and C.

  for the first step of processing T **do**
    Send new helper messages with parent T to all arcs in C
    Send new exploring messages with parent T to all arcs in A
  end for

  Let l be the length of the helper tail of T.
  for subsequent $n - l$ steps of processing T **do**
    Send new exploring messages with parent T to all arcs in A
  end for
end procedure

I1 Since exploring (helper) messages are sent only at lines 10 and 14 (at line 9), the definition of A at line 3 (C at line 7) ensures the first (second) statement of this invariant, respectively.

I2 Helper message T can not be sent to a vertex that sent some message from the helper tail of T. This is ensured by the definition of C at line 7. (Note that this does not hold for exploring message T.) Hence, the helper tail of any message can contain each vertex at most once.

I3 The helper message can be sent only at line 9, i.e. immediately after receiving its parent message.

I4 Let T be an exploring message, message U be the parent of T, l be the length of the helper tail of U and V be the first message on the helper tail of U. It is necessary to prove that T is sent at most $n+1$ steps later that V.
The invariant *I2* implies that $l \leq n$. The invariant *I3* ensures that U is sent exactly l steps later than V. Hence, if T is sent at line 10, it is sent $l+1$ steps later than V. Otherwise T is sent at line 14 at most $l+1+n-l$ steps later than V.

I5 Assume the contrary, i.e. some exploring message is delivered in step t, only helper messages are delivered in subsequent n steps and the broadcast is not finished. Invariants *I2* and *I3* ensure that no helper messages can be delivered in the step $t+n$ or later. Therefore, the last message is delivered in the step u, for some $u < t+n$. Since exactly $n-1$ messages are sent in the first step of processing message T (see lines 9 and 10), it holds that $t < u$.
Let T be any message delivered in the step u to vertex p. Let $k < n$ be the number of messages in the helper tail of T. Let $S \neq T$ be a message in the helper tail of T, delivered to a vertex q. From construction it follows that in step u the message S is still processed by q at line 14. Because the broadcast is not finished yet, q sends at least one message in step u; combining for all such vertices in the helper tail of T yields $k-1$ messages sent in the step u. Processor p sends $|P|-(|A|+|B|) \geq (n-1)-(|A|+k-1)$ messages at line 9 and A messages at line 10 in the step u. Summing up, there are at least $n-1$ messages sent in step u, therefore there is at least one message delivered in step $u+1$, a contradiction.

□

Combining Lemma 5.3.4 with the discussion about the invariants we get

Theorem 5.3.5 *There exist a $O(n^3)$ fault-tolerant broadcasting algorithm for unoriented complete networks.*

5.3.3 Lower Bound for Unoriented Complete Networks

The $O(n)$ algorithm for rings is obviously asymptotically optimal. An interesting question is: How far from optimal are our algorithms for oriented and unoriented complete networks? In this section we show that

Theorem 5.3.6 *Any fault-tolerant broadcasting algorithm on unoriented complete networks must spend $\Omega(n^2)$ time.*

Proof: In the course of the computation there are two kinds of ports: the ports that have never been traversed by any message in any direction are called "*free*", the ports that are not free are called "*bound*". The lower bound proof is based on the following simple fact:

Let p be a free port of vertex u in time t. Let v be any vertex such that no bound port of u leads to v. Then it is possible that port p of u leads to vertex v.

Indeed, if p would lead to v, the first t steps of computation would be the same. Hence, the computation can be viewed as a game of two players: the algorithm chooses a set of ports through which messages are to be sent. The adversary chooses one port through which the requested message passes. If this port is free, it chooses also the vertex to which this port will be bound.

We show now that it is possible for the adversary to keep the vertex n uninformed for $\frac{(n-1)(n-2)}{2} = \Omega(n^2)$ time steps. The idea is that some message has to traverse through all edges between vertices $1 \dots n-1$ before any message arrives to the vertex n.

Consider the time step $i < \frac{(n-1)(n-2)}{2}$ and assume that the vertex n is not informed yet. There exist at most $2i$ bound ports, since in each time step at most one edge, i.e two ports are bounded. This means that at least $(n-1)(n-1) - 2i \geq n$ ports of vertices $1 \dots n-1$ are free.

The following cases can occur:

1. The algorithm sends some message through some bound port. The adversary passes this message, hence the vertex n stays uninformed.

2. The algorithm sends messages only through free ports.

   (a) The algorithm does not send messages from all vertices $1 \dots n-1$. Then there have to be at least 2 messages sent from one vertex. The adversary delivers one of these messages and binds the corresponding port to any vertex other than n. (Since there are at least two free ports, it is possible for the adversary to do so.)

   (b) The algorithm sends messages from all vertices $1 \dots n-1$. Since at least n ports of vertices $1 \dots n-1$ are free, at least one vertex w from $1 \dots n-1$, has 2 free ports. The adversary delivers the message sent from w, and binds corresponding port to any vertex other than n. (Again, since there are at least two free ports, it is possible for the adversary to do so.)

Hence it is possible for the adversary to keep the vertex n uninformed for the first $\Omega(n^2)$ time steps. □

Now assume a stronger computation model: each vertex immediately learns for any message it has sent whether this message has been delivered or not. It is interesting to note that our lower bound is valid also in this model. Furthermore, it is easy to see that the lower bound is tight in this model.

5.4 Arbitrary k-connected Graphs

In this section we consider k-edge-connected graphs and we assume the threshold is k, i.e. at least k messages must be sent to ensure that a message is delivered.

5.4.1 With Full Topology Knowledge

The algorithm runs in $n-1$ phases. Each phase has an initiator vertex u (informed) and a destination vertex v (uninformed), with the source s being the initiator of the first phase. The goal of a phase is to inform vertex v, which then becomes the initiator of the next phase; the process is repeated until all vertices are informed.

The basic idea is a generalization of the idea from rings. The ring algorithm tried to "push" the information simultaneously along the left and right part of the ring. Here, the initiator u chooses k edge-disjoint paths[3] $\mathcal{P} = \{P_1, \ldots, P_k\}$ from itself to v and then pushes the information through all the paths simultaneously. Let $P_i = (u_0 = u, u_1, \ldots u_{l_i} = v)$; consider an arc $e = \langle u_j, u_{j+1} \rangle$. This arc can be either *sleeping*, *active* or *passive*:

1. The arc e is *passive* if and only if a message has been received over both e and the arc opposite to e, i.e. $\langle u_{j+1}, u_j \rangle$.
2. The arc e is *active* if and only if it is not passive and a message has been received over the arc $\langle u_{j-1}, u_j \rangle$. In case $j = 0$ the arc e is active whenever it is not passive.
3. The arc e is *sleeping* if and only if it is not active nor passive.

One phase consists of several rounds, each round spanning over many communication steps. The goal of one round is to ensure that a progress over at least one arc has been made: at least one active arc becomes passive, at least one sleeping arc becomes active or the vertex v becomes informed.

The procedure ROUND() defined in Algorithm 4 is the core of the algorithm; it is performed in each round by every vertex $w \in \mathcal{P}$.

It is easy to see that the uninformed vertices never send any messages and that at any time each vertex can determine all active arcs incident to it. Synchronous communication

[3] since the graph is k-edge-connected and the vertices have full topology knowledge, the initiator can always find these paths

Algorithm 4 k-connected graphs

```
1: procedure ROUND(vertex w)
2:     Let A be the set of active arcs outgoing out of w at the beginning of the round
3:     for i:=0 to k do    // One subround:
4:         for B ⊆ {1...k} such that |B| = i do    // one iteration per time step
5:             Let C be the set of arcs outgoing out of w via which an activating message
6:             has been received in the current round    // not in the current time step
7:             for e such that opposite arc of e is in C do
8:                 send deactivating message through e    // all in one time step
9:             end for
10:            for e ∈ A such that e ∈ P_z ∧ z ∉ B do
11:                send activating message through e // all in the same time step as in 8
12:            end for
13:        end for
14:    end for
15: end procedure
```

and full topology knowledge ensure that all procedures (phases/rounds/subrounds) are started and executed simultaneously by all participating vertices.

Lemma 5.4.1 *During one round at least one active arc becomes passive, or a sleeping arc becomes active, or v is informed.*

Proof: By contradiction. Assume the contrary, we show that in such case, at the beginning of the i-th subround there will be at least i paths $\mathcal{P}' \subseteq \mathcal{P}$ such that on any path $P_j \in \mathcal{P}'$ there is an arc through which an activating message has been delivered in the current round. This would mean that in the k-th subround there are at least k deactivating messages sent and therefore at least one of them will be delivered and an active arc will become passive, a contradiction.

We prove that above statement about subrounds by induction on i. The statement trivially holds for $i = 0$, as there is nothing to prove. Assume (by induction hypothesis) that at the beginning of the i-th subround there are exactly i paths $\mathcal{P}'$ with an arc over which an activating message has been delivered in the current round (if there are more, the hypothesis is already true for $i + 1$). From the definition of an active arc and from construction it follows that unless the vertex v is informed, there is at least one active arc on each path P_j. Let us focus on the time step in the i-th subround when B contains exactly the numbers of paths from $\mathcal{P}'$ (i.e. $B = \{j | P_j \in \mathcal{P}'\}$). In this time step, at least $k - i$ activating and at least i deactivating messages are sent, therefore at least one of them must be delivered. As no activating message is sent over an arc $e \in \mathcal{P}'$ and no deactivating message is delivered (by assumption that no active arc becomes passive), an activating message must be delivered on a path not in $\mathcal{P}'$. Hence, the invariant is ensured for the subround $i + 1$, too. □

Theorem 5.4.2 *There is a fault-tolerant broadcasting algorithm on k-connected graphs with full topology knowledge that uses $O(2^k nm)$ time, where n is the number of vertices and m is the number of edges in the graph.*

Proof: The correctness follows straightforwardly from construction and Lemma 5.4.1.

The time complexity of one round is 2^k, as it spends one time step for each subset of $\{1, 2, \ldots, k\}$. The number of rounds per phase is[4] $2m$, as all paths in $\mathcal{P}$ together cannot contain more than all m arcs and each arc can change its state at most twice (from sleeping to active to passive). Finally, the number of phases is $n - 1$ as $n - 1$ vertices need to be informed. Multiplying we get $O(2^k mn)$. □

Theorem 5.4.2 can be successfully applied to many commonly used interconnection topologies. However, better results can usually be obtained by carefully choosing the order in which the vertices should be informed, allowing for short paths in $\mathcal{P}$. One such example is oriented hypercubes (i.e. each link is marked by the dimension it lies in):

Theorem 5.4.3 *There is a fault-tolerant broadcasting algorithm for oriented d-dimensional hypercubes that uses $O(n^2 \log n)$ time, where $n = 2^d$ is the number of vertices of the hypercube.*

Proof: The basic idea is to use the algorithm for k-connected graphs, with the initiator of a phase choosing as the next vertex to inform its successor in (a fixed) Hamiltonian path of the hypercube.

The algorithm for one phase is the same as in the case of k-connected graphs with the following exception: it is possible to choose d edge-disjoint paths from vertex u to its neighbor vertex v such that each of these paths has length at most 3. This results in $\mathcal{P}$ containing only $O(d)$ arcs instead of $O(n \log n)$, thus reducing the cost of one phase from $O(n^2 \log n)$ to $O(nd) = O(n \log n)$. The resulting time complexity is therefore $O(n^2 \log n)$. □

5.4.2 Without Topology Knowledge

Finally, we show that the broadcasting on a k-connected graph with n vertices and m edges can be performed in time $O(2^k m^2 n)$ even in the case when the only known information about the graph are the values of n, m, and k. To achieve this, we combine the ideas used for complete graphs with those for arbitrary graphs with full topology knowledge. In particular, the vertices accumulate topology information (using local identifiers) in a fashion similar to the algorithm for complete graphs. The algorithm works in phases, where each phase is performed within one informed component, and uses the topology knowledge of that component. However, since there may be many phases active at the same moment, great care must be given to avoid unwanted interference. Now we present this algorithm in more detail.

[4] some topology-specific optimization is possible here

As in the case of complete graphs, we use the notion of area knowledge; the knowledge of an informed vertex u is denoted as $K(u)$. It is clear that if there is at least one vertex u such that $|K(u)| = 2m$, then all vertices are informed.

The broadcasting algorithm runs in $2m$ phases. At the beginning of phase i there is at least one vertex (phase initiator) with a knowledge of at least i triples, the goal of phase i is to ensure this invariant holds for phase $i + 1$.

Let a be an initiator[5] of a phase and $K(a)$ be its knowledge at the beginning of the phase. Then each arc $\langle b, c\rangle$ can be classified as *helper* or *exploring* (with respect to a, in the current phase), depending on whether the corresponding triple (b, p, c) is contained in $K(a)$ or not. The goal of a is to propagate its knowledge through some of its exploring arcs. Any vertex u that receives information through such arc becomes an initiator in the next phase. As u's knowledge at that moment is strictly larger than what $K(a)$ was at the beginning of the phase, the phase invariant holds.

The main idea of the algorithm for one phase is similar to the approach used in the case of full topology knowledge: The initiator chooses k edge-disjoint paths such that the last arc of each path is exploring and all inside arcs are helper arcs, and then pushes the information along these paths until at least one path is fully traversed. The fact that the last arc of a path is exploring ensures progress; the fact that inside arcs are helper arcs ensures that the participating vertices know where the paths continue (in fact, the initiator cannot construct paths with inside exploring arcs as it does not know what their endpoints are).

If there is a single initiator in the phase, this approach would work nicely with no changes to procedure Round(). However, each phase can have several initiators and their computations can disrupt each other. The algorithm for one phase thus consists of $n - 1$ subphases. The goal of a subphase is to ensure that either some knowledge is propagated (i.e. one path is fully traversed), or at least one initiator (but not all of them) is eliminated. Hence, after $n - 1$ subphases some knowledge is propagated.

The algorithm for one subphase is as follows. Each initiator v chooses k edge disjoint directed paths starting at v. We call these paths *threads* and denote them $T(v, j)$ for $j = 1, 2 \ldots, k$. Let $T(v, j) = (v = u_0, u_1, \ldots, u_l)$. The arcs (u_i, u_{i+1}) are called *forward arcs* of $T(v, j)$; analogously, arcs (u_i, u_{i-1}) are called *backward arcs*. The threads are chosen in such a way that all forward arcs of $T(v, j)$ are helper with respect to v, except the last arc (u_{l-1}, u_l), which is exploring. Note that v can always choose k such threads (from the k-connectivity of the graph; a thread might consist of a single exploring arc). An example of such situation is depicted on Figure 5.4.

In a subphase, the messages are propagated in the same way as in a phase of the Algorithm described in the section 5.4.1, until the threads of different initiators collide (that is, a vertex receives messages from different initiators). In such case, the threads "bounce" back to their initiators. The algorithm guarantees that if no knowledge has been propagated then each subphase initiator receives at least one bounced thread. Each bounced thread carries the ID of the initiator of the colliding thread, therefore each initiator knows

[5] a phase might have several initiators

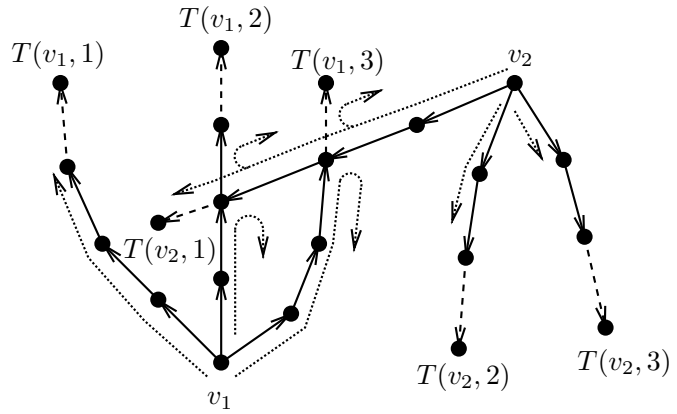


Figure 5.4: An example of a subphase. Both v_1 and v_2 constructed 3 threads. Helper arcs are marked as solid lines, exploring arcs are marked as dashed lines. Dotted lines show one possible propagation of messages.

the ID of at least one other initiator. Only the initiators which have not seen an ID higher then their own proceed to become initiators of the next subphase. Obviously, the initiator with the biggest ID survives and initiator with the smallest ID does not. This means that the number of initiators decreases, but there is always at least one initiator left.

Each subphase consists of $4m$ rounds. Let $p = \langle u_1, u_2 \rangle$ be an arc outgoing out of vertex u_1. At the beginning of each round the state of p with respect to a thread $T(v, j)$ is determined as follows (only messages received in the current subphase are considered):

1. The arc p is *passive* if and only if u_1 has received message of type *passive* via the opposite arc of p (i.e. from u_2).
2. The arc p is *forward-active* if and only if all of the following conditions hold:
   (a) p is not passive.
   (b) p is a forward arc in $T(v, j)$ and u_1 has received some message of the thread $T(v, j)$
   (c) u_1 has not received any message of a thread $T(v', j')$ such that $v' \neq v$.
3. The arc p is *backward-active* if and only if all of the following conditions hold:
   (a) p is not passive.
   (b) p is a backward arc of $T(v, j)$ and u_1 has received some message of the thread $T(v, j)$
   (c) u_1 has received a message from a thread $T(v', j')$ such that $v' \neq v$ **or** the vertex u_1 has received a message of the thread $T(v, j)$ of type *backward-active*.
4. Otherwise the arc p is *sleeping*.

An arc is called *active* if it is either forward-active or backward-active.

In each round each vertex tries to push *forward-active* messages through all its outgoing forward-active arcs and *backward-active* messages through all its outgoing backward-active arcs. An initiator receives a bounced thread if it receives a *backward active* message of that thread.

Each round ensures that if there is at least one initiator that has not received a bounced thread nor propagated its knowledge, then at least one active arc becomes passive or at least one new active message is delivered, i.e. some sleeping arc becomes active in the next round. Hence after applying at most $4m$ rounds the subphase is complete.

Algorithm 5 Algorithm for k-connected graphs without additional structural information.

```
1: procedure ROUND(vertex v)
2:     Let A is the set of arcs outgoing out of v active at the beginning of the round
3:     for i:=0 to k do    // One subround:
4:         for B ⊆ {1...k} such that |B| = i do
5:   // all messages for one B are sent simultaneously
6:             for e such that an activating message has been received through e
7:           in the current round do
8:                 if the activating message belongs to T(u, z) such that z ∈ B then
9:                     send a message of type passive through e
10:                end if
11:            end for
12:            for e ∈ A such that e ∈ T(u, z) ∧ z ∉ B do
13:                send an activating message of same type as the state of e through e
14:            end for
15:        end for
16:    end for
17: end procedure
```

The algorithm for one round, described as Algorithm 5, is very similar to the one used before. The correctness of the algorithm relies on the following facts:

1. No arc can be forward-active and backward-active at the same time. Indeed, if some vertex v has received messages from threads of different initiators, then it cannot have outgoing forward-active arcs. Moreover, the threads of the same initiator are edge-disjoint.

2. Each forward-active arc is forward-active on exactly one thread. (The same reasons.)

3. Each backward-active arc is backward-active on exactly one thread. From construction: if some arc outgoing out of the vertex v was backward-active on two threads $T_1 = T(v_1, j_1)$, $T_2 = T(v_2, j_2)$, then T_1 and T_2 share the predecessor u of v and the arc $\langle u, v\rangle$ would had been forward-active on both threads.

This implies that at most one active message is sent per arc per time step. If both an active and a passive message should be sent through one arc in the same time step, the passive message takes priority.

Lemma 5.4.4 *During one round either an initiator has propagated its knowledge, or all initiators have received a bounced thread, or an active arc becomes passive or a sleeping arc becomes active.*

Proof: By contradiction, analogous to the proof of Lemma 5.4.1. Assuming the contrary, we show that at the beginning of the i-th subround there are at least i threads $\mathcal{T} = T(v_1, j_1), \ldots, T(v_i, j_i)$ such that no two $j_1, \ldots j_i$ are equal and some active message belonging to any of these threads has been delivered in the current round. This means that in the k-th subround at least k passive messages are being sent and at least one of them is delivered, contradiction.

The proof is by induction on i, with the case $i = 0$ being trivial. Assume that at the beginning of the i-th subround there are exactly i threads satisfying above-mentioned conditions (if there are more than i such threads, the invariant obviously holds for the subround $i+1$). Let v be any initiator that has not received a bounced thread. It is easy to see that there is at least one active arc (either forward-active or backward-active) on each thread of v such that no passive message is to be sent on this arc. Indeed, let T be any such thread. If it has not collided with any other thread yet, then the last informed vertex of T has an outgoing forward-active arc (and a passive message is never sent through a forward-active arc). Otherwise at least one vertex of T has an outgoing backward-active arc such that no passive message is to be sent on it (the one that is closest to the thread initiator).

Now focus on the time step in the subround when B contains exactly the numbers of the threads from $\mathcal{T}$ (i.e. $B = \{j | T(u, j) \in \mathcal{T}\}$). In this time step at least $|B|$ passive and at least $k - |B|$ active messages are sent, therefore at least one of them must be delivered. Assuming no passive message is delivered yields that an active message on a new, $i+1$-th, thread is delivered. Hence the induction hypothesis is ensured for $i + 1$ as well. □

Putting the pieces together:

1. *The whole algorithm:* Consists of $2m$ phases.
2. *One phase:* The goal is to increase the maximal knowledge in the system by at least one arc. Consists of $n - 1$ subphases.
3. *One subphase:* The goal is to eliminate at least one phase initiator, unless the maximal knowledge is increased. Consists of $4m$ rounds.
4. *One round:* The goal is to make some sleeping arc active or some active arc passive, unless the subphase makes progress. Consists of $k + 1$ subrounds.

5. *One subround:* The goal is to deliver a new active message, thus increase the number of passive messages that are to be sent. The i-th subround of the round consists of $\binom{k}{i}$ time steps.

Combining together results in $2m \times (n-1) \times 4m \times 2^k = O(2^k m^2 n)$ overall time complexity.

Theorem 5.4.5 *There is a fault-tolerant broadcasting algorithm on k-connected graphs without sense of direction that uses $O(2^k m^2 n)$ time, where m is the number of edges and n is the number of the vertices of the graph.*

As the time complexity is only exponential in the connectivity of the graph, we get:

Corollary 5.4.6 *There is a polynomial-time broadcasting algorithm on any graph without sense of direction with edge connectivity $O(\log n)$.*

Directly applying Theorem 5.4.5 yields:

Corollary 5.4.7 *There is a broadcasting algorithm on d-dimensional hypercube without sense of direction that uses $O(n^4 \log^2 n)$ time.*

Chapter 6

Fractional Threshold Model

In the previous chapter, we have introduced the motivation for the α-fractional threshold model and focused on the extreme case where α is infinitesimally close to 1. In this chapter, we provide results for the general case, i.e. $\alpha \in (0, 1)$, hence completing the third goal of this thesis. Considered results have been published in [42], too. As in the previous chapter, our results assume the constant multi-port wake-up communication model and deal with adaptive broadcasting algorithms.

While trying to solve the broadcasting problem, we discovered an interesting feature of the fractional threshold model: Usually it is quite easy to inform vast majority of the vertices. But to inform the last remaining ones, all informed vertices have to cooperate very tightly, which is often very hard to achieve. On the other hand, it is often vital to finish the broadcasting communication task fast, even subject to some small error. These facts give a motivation to study a natural relaxation of the broadcasting problem in which we allow a small constant number of vertices to stay uninformed in the end. We call such relaxation as the *almost-complete broadcasting* problem.

Definition 6.1 *A broadcast is called* almost-complete *if there is a fixed constant c (independent on the network size) such that after the termination there are at most c uninformed vertices.*

Hence, to prove the existence of an almost-complete broadcasting algorithm for a family of graphs $\mathcal{G}$, one has to prove that there exists a constant c such that for each $G \in \mathcal{G}$ the broadcasting algorithm informs all but c vertices of G.

To avoid confusion, we refer to the standard broadcasting problem as the *complete broadcasting* in this chapter.

In this chapter, we analyze the problem of almost-complete broadcasting on complete graphs and hypercubes. We show how to solve this problem with only logarithmic slowdown, i.e. in time $O(D \log n)$, where D is the diameter of the communication graph and n is the number of its vertices. In particular, the time sufficient for almost-complete broadcasting is $O(\log n)$ for complete graphs and $O(\log^2 n)$ for hypercubes. Our results hold for anonymous networks without any topology knowledge (except for the size of the commu-

nication graph n). We also show that our results for complete graphs are tight, i.e. that the almost-complete broadcasting requires time $\Omega(\log n)$ on K_n.

Afterwards, we extend our results to the problem of complete broadcasting. We show that the complete broadcasting can be finished in time $O(\log n)$ on complete graphs if the chordal sense of direction is available, or the value of α is small ($\alpha \lesssim 0.55$).

Scenario	Almost complete broadcasting	Complete broadcasting
K_n, unoriented	$\Theta(\log n)$	$\Omega(\log n)$, $O(n^3)$
K_n, chordal sense of direction	$\Theta(\log n)$	$\Theta(\log n)$
K_n, $\alpha \lesssim 0.55$	$\Theta(\log n)$	$\Theta(\log n)$
H_d	$\Omega(d)$, $O(d^2)$	$\Omega(d)$, $O(n^4 d^2)$

Table 6.1: Results for the complete and almost complete broadcasting in the fractional model with threshold.

The overview of all results presented in this chapter is presented in Table 6.1. This table is augmented by the bounds inherited from the simple threshold model: It is easy to see that any algorithm working in the simple threshold model in time T is a correct algorithm in the fractional-threshold model working in time at most T. Hence, any upper bound on the time complexity of complete broadcasting in the simple threshold model can be inherited by the fractional threshold model.

6.1 Preliminaries

In this section, we present some definitions and techniques common to the current chapter.

As in the previous chapter, we consider the communication graph to be directed, i.e. we replace each edge of the communication graph by two opposite arcs (see Section 5.1). Arcs leading from an informed vertex can be classified as being either active, passive or hyperactive during the computation:

Definition 6.2 *Let e be an arc leading from an informed vertex. We call e* active *if it leads to an uninformed vertex. We call an arc e* passive*, if some message has been delivered via the opposite arc of e. Finally, we call an arc e* hyperactive *if it leads to an informed vertex, and is not passive.*

If the arc e is passive, the source vertex of e is aware of the fact that the destination vertex of e has already been informed. The main idea of our algorithms is to perform appropriate number of *simple rounds* defined as follows:

Definition 6.3 *A* simple round *consists of two time steps. In the first step, every informed*

vertex sends a message along each of its incident arcs, excluding the passive ones.[1] *In the second step, all vertices that have received a message send an acknowledgement (and mark the arc as passive). Vertices that receive acknowledgement mark the corresponding arc as passive.*

For the remainder of this chapter, let $0 < \alpha < 1$ be a known fixed constant, and let us denote

$$X := \frac{1}{\alpha(1-\alpha)}.$$

The rest of the current chapter is organized as follows. In the next section, we analyze the time complexity of almost-complete broadcasting on complete graphs. We present an algorithm that runs in time $O(\log n)$, as well as the lower bound $\Omega(\log n)$. In the next section, we focus on hypercubes, for which we present an algorithm with time complexity $O(\log^2 n)$. Finally, we show how to perform broadcasting in complete graphs equipped with chordal sense of direction, and in unoriented complete graphs for $\alpha \lesssim 0.55$, having the optimal asymptotic complexity $\Theta(\log n)$.

6.2 Complete Graphs

In a complete graph K_n, all n vertices have degree $n-1$, and $n-1$ is also the edge connectivity. Hence, in each step t the adversary can destroy up to $\max\{n-2, \lfloor \alpha m_t \rfloor\}$ messages, where m_t is the number of messages sent in the step t.

At first we preset the lower bound $\Omega(\log n)$ on the time complexity of the almost-complete broadcasting in complete graphs. In fact, we show the lower bound on the time of the almost-complete broadcasting in the $(\alpha, 0)$-fractional threshold model, i.e. without any assumptions on the size of the threshold. The main idea of the proof is identical to the proof of the lower bound on broadcasting in α-weakly-bounded model (see Theorem 4.2.1).

We need the following auxiliary lemma.

Lemma 6.2.1 *Let $0 \leq z_1 \leq \ldots \leq z_k$ be integers. It holds that*

$$\sum_{i=1}^{\lfloor \alpha k \rfloor} z_i \leq \alpha \sum_{i=1}^{k} z_i.$$

Informally, the sum of αk smallest integers can not exceed the sum of all integers multiplied by α.

[1] In this step, a message is sent via all active and hyperactive arcs. The former can inform new vertices, the latter exhibit only useless activity. However, the algorithm can not distinguish between active and hyperactive arcs.

Proof: Let $l := \lfloor \alpha k \rfloor$. We have

$$\begin{aligned} \sum_{i=1}^{k} z_i &\geq \sum_{i=1}^{l} z_i + (k-l) z_l \geq \sum_{i=1}^{l} z_i + (k-l) \frac{\sum_{i=1}^{l} z_i}{l} \\ &= \left(1 + \frac{k-l}{l}\right) \sum_{i=1}^{l} z_i = \frac{k}{l} \sum_{i=1}^{l} z_i \end{aligned}$$

This immediately yields

$$\sum_{i=1}^{\lfloor \alpha k \rfloor} z_i \leq \frac{\lfloor \alpha k \rfloor}{k} \sum_{i=1}^{k} z_i \leq \alpha \sum_{i=1}^{k} z_i$$

□

Theorem 6.2.2 *The time complexity of the almost-complete broadcasting in the* $(\alpha, 0)$*-fractional threshold model on the complete graph* K_n *is* $\Omega(\log n)$.

Proof: Consider an almost-complete broadcast $\{w\} = V_0 \subseteq V_1 \subseteq \cdots \subseteq V_t$ and let $|V_i| = a_i$ be the number of informed vertices in step i. We show a strategy for the adversary that maintains the following inequality:

$$a_i \leq n(1 - \alpha^i) + \frac{1 - \alpha^{i+1}}{1 - \alpha} \tag{6.1}$$

Assume that x_i messages have been sent in time step i, and y_i among them have been sent to uninformed vertices. Let $0 \leq z_1 \leq \ldots \leq z_k$ (where $k = n - a_i$) be the number of messages sent to individual uninformed vertices. Obviously $\sum_{j=1}^{k} z_j = y_i$.

By Lemma 6.2.1 we have that $\sum_{j=1}^{\lfloor \alpha k \rfloor} z_j \leq \alpha y_i$. Since the adversary may discard up to $\alpha x_i \geq \alpha y_i$ messages, he is able to keep at least $\lfloor \alpha k \rfloor$ vertices uninformed by blocking messages corresponding to $z_1, \ldots, z_{\lfloor \alpha k \rfloor}$. Hence, at most $k - \lfloor \alpha k \rfloor$ new vertices are informed in step i, i.e.

$$a_{i+1} \leq a_i + (n - a_i) - \lfloor \alpha (n - a_i) \rfloor = n - \lfloor \alpha (n - a_i) \rfloor \leq 1 + n(1 - \alpha) + \alpha a_i.$$

We have obtained the same recurrent inequality as in Theorem 4.2.1:

$$\begin{aligned} a_0 &= 1 \\ a_{i+1} &\leq 1 + n(1 - \alpha) + a_i \alpha \end{aligned}$$

The solution of this inequality is exactly Inequality 6.1.

Let t be the number of time steps necessary to inform all but c vertices when the adversary follows the above-mentioned strategy. It holds that

$$n - c \leq a_t \leq n(1 - \alpha^t) + \frac{1 - \alpha^{t+1}}{1 - \alpha} \leq n - n\alpha^t + \frac{1}{1 - \alpha}.$$

This yields that

$$\alpha^t \leq \frac{c + \frac{1}{1-\alpha}}{n}.$$

Since c and $\alpha \leq 1$ are fixed constants, we directly obtain $t = \Omega(\log n)$. □

Now we present an algorithm that informs all but a constant number of vertices in logarithmic time. The idea of the algorithm is very straightforward – just repeat simple rounds sufficiently many times. However, the arguments given in the analysis of a simple round below hold only if there are enough informed vertices participating in the round. To satisfy this requirement two steps of a simple greedy algorithm are performed, during which each informed vertex just sends the message to all vertices. After two steps of this algorithm, the number of informed vertices is as shown in Lemma 6.2.3.

Lemma 6.2.3 *After two steps of the greedy algorithm, at least*

$$1 + \min\left\{\frac{n}{2}, (n-1)(1-\alpha)\right\}$$

vertices are informed.

Proof: In the first step the initiator sends $n-1$ messages. Let $l \geq 2$ be the number of informed vertices after the first step. In the second step, $l(n-1)$ messages are sent, and $\max\{n-2, \alpha l(n-1)\}$ of them are lost. We distinguish two cases:

Case 1: $\alpha l(n-1) \leq n-2$
In this case, at most $n-2$ messages are lost, i.e. at least $l(n-1) - n + 2$ are delivered. Among those delivered, at most $l(l-1)$ could have been sent to already informed vertices. Moreover, since each uninformed vertex has at most l informed neighbors, we get that the number of informed vertices is at least

$$l + \frac{l(n-1) - n + 2 - l(l-1)}{l} = n - \frac{n-2}{l}$$

Since $l \geq 2$ we get that the number of informed vertices after the two steps is a least $\frac{n}{2} + 1$.

Case 2: $\alpha l(n-1) > n-2$
This time, at most $\alpha l(n-1)$ messages are lost. Using similar arguments, we get that the number of informed vertices is at least

$$l + \frac{l(n-1)(1-\alpha) - l(l-1)}{l} = 1 + (n-1)(1-\alpha)$$

□

After these two steps, the algorithm performs a logarithmic number of simple rounds. To show that logarithmic number of simple rounds is sufficient to inform all but one vertex we first provide a lower bound on the number of acknowledgements delivered in each round, and then we show that each delivered acknowledgement decreases a certain measure function.

Theorem 6.2.4 *Let $\varepsilon > 1$ be an arbitrary constant. For large enough n it is possible to inform all but at most $X\varepsilon$ vertices in logarithmic time. Moreover, the number of remaining hyperactive arcs is at most $X(n-2)$.*

Proof: At the beginning, two steps of the greedy algorithm are executed. Then, a logarithmic number of simple rounds is performed. Now consider the situation at the beginning of the i-th round. Let k_i be the number of uninformed vertices, and h_i the number of hyperactive arcs. We claim that if $k_i > X\varepsilon$ or $h_i > X(n-2)$ then at least $[k_i(n-k_i)+h_i](1-\alpha)^2$ acknowledgements are delivered in this round. Since there are $k_i(n-k_i)+h_i$ messages sent in this round, in order to prove the claim it is sufficient to show that $\alpha(1-\alpha)[k_i(n-k_i)+h_i] \geq n-2$. Obviously, if $h_i > X(n-2)$, the inequality holds, so consider the case $k_i > X\varepsilon$. We prove that in this case $k_i(n-k_i) \geq X(n-2)$, i.e. $k_i^2 - nk_i + X(n-2) \leq 0$. Let $f(n) := 1/2\left(n - \sqrt{n^2 - 4X(n-2)}\right)$; the roots[2] of the equation $k_i^2 - nk_i + X(n-2) = 0$ are $f(n)$ and $n - f(n)$, so we want to show that $f(n) \leq k_i \leq n - f(n)$. Since $\lim_{n\mapsto\infty} f(n) = X$, we get that $k_i > X\varepsilon > f(n)$ holds for large enough n. Hence, the only remaining step is to show the inequality $k_i \leq n - f(n)$. From Lemma 6.2.3 it follows that $n - k_i > \min\{n/2, (n-1)(1-\alpha)\}$. Since $f(n) < n/2$, if $n - k_i > n/2$ it holds $k_i < n - f(n)$. So let us suppose that $n - k_i > (n-1)(1-\alpha)$, i.e. $k_i < 1 + \alpha(n-1)$. Let $n \geq \frac{\varepsilon+\alpha(1-\alpha)^2}{\alpha(1-\alpha)^2}$. Then it holds for large enough n that

$$k_i < 1 + \alpha n - \alpha \leq n - \frac{\varepsilon}{\alpha(1-\alpha)} = n - \varepsilon X \leq n - f(n).$$

We have proved that if $k_i > X\varepsilon$ or $h_i > X(n-2)$ then at least

$$[k_i(n-k_i)+h_i](1-\alpha)^2$$

acknowledgements are delivered in round i.

To conclude the proof we show that after logarithmic number of iterations we get $k_i \leq X\varepsilon$ and $h_i \leq X(n-2)$. Let $M_i := 2(n-1)k_i + h_i$; then every delivered acknowledgement decreases M_i by at least one: indeed, if the acknowledgement was delivered over a hyperactive arc, h_i decreases by 1. If, on the other hand, the acknowledgement was delivered over an active arc, the number of uninformed vertices is decreased by at least one, and the number of hyperactive arcs is increased by at most $2n-3$ (new hyperactive arcs are between the newly informed vertex and any other vertex, with the exception of the arc that delivered the acknowledgement, which is passive).

From Lemma 6.2.3 it follows that either $n - k_i > n/2$ or $n - k_i > (n-1)(1-\alpha)$. In the first case it follows that at least $(1-\alpha)^2[k_i(n-k_i)+h_i] > (1-\alpha)^2[k_i n/2 + h_i] \geq \frac{(1-\alpha)^2}{4} M_i$ acknowledgements are delivered. In the second case we get that at least $(1-\alpha)^2[k_i(n-k_i)+h_i] > (1-\alpha)^2[k_i(n-1)(1-\alpha)+h_i] \geq \frac{(1-\alpha)^3}{2} M_i$ acknowledgements are delivered. Let $c := \min\{\frac{(1-\alpha)^2}{4}, \frac{(1-\alpha)^3}{2}\}$, then obviously every iteration decreases the value

[2] Assume that n is large enough such that $f(n)$ is real number.

of M_i at least by factor c. Since the value of M at the beginning of the algorithm is $M_1 = O(n^2)$, $\log_{1/c} M_1 = O(\log n)$ steps are sufficient to inform all but a constant number (at most $X\varepsilon$) of vertices and to ensure that the number of remaining hyperactive arcs is linear (at most $X(n-2)$). □

6.3 Hypercubes

In this section we consider d-dimensional hypercubes. The hypercube H_d has 2^d vertices, and both diameter and edge connectivity are d. We present an algorithm that informs all but a constant number of vertices in time $O(d^2)$.

The general idea is the same as for complete graphs: first we perform two initialization steps to make sure there are enough informed vertices for the subsequent analysis to hold. Next, simple rounds are repeated for a sufficient number of times. The analysis, however, is more complicated in this case.

The next lemma covers the initialization steps. In the first step, the initiator sends a message to all its neighbors, and at least one of these messages is delivered. In the second step, the initiator sends a message to all its neighbors again; moreover, each of the vertices informed in the first step sends a message to all its neighbors except the initiator.

Lemma 6.3.1 *After the first two steps of the algorithm, at least* $\frac{1-\alpha}{2}(2d-1)$ *vertices are informed.*

Proof: In the first step, the initiator sends d messages. Since at most $d-1$ can be lost, some $r > 0$ of them are delivered. In the second step, the initiator sends again d messages, but at the same time, each of the informed vertices sends $d-1$ messages to all its neighbors except initiator. Hence, $d+r(d-1)$ messages are sent in the second step. Let us distinguish two cases:

If $d-1$ messages are lost, then $d+(r-1)(d-1)$ messages are delivered. r messages from the initiator can be delivered to the already informed vertices which leaves $d+(r-1)(d-1)-r$ messages that enter uninformed vertices. Since at most r messages can be destined to the same vertex, the number of informed vertices after two steps is at least $1+r+\frac{d+(r-1)(d-1)-r}{r} \geq (1/2)(2d-1)$.

If at most $\alpha[d+r(d-1)]$ messages are lost, then at least $(1-\alpha)[d+r(d-1)]-r$ messages arrive into uninformed vertices. Hence, there are at least $\frac{1-\alpha}{r}[d+r(d-1)]+r \geq \frac{1-\alpha}{2}(2d-1)$ informed vertices. □

For the rest of this section we suppose that there are at least $\frac{1-\alpha}{2}(2d-1)$ informed vertices. We show that after $O(d^2)$ simple rounds all but some constant number of vertices are informed, and there are only linearly many hyperactive arcs. At the end of this section, we shall be able to prove the following theorem.

Theorem 6.3.2 *Let* $\varepsilon \in (0,1)$ *be an arbitrary constant. For large enough* d *it is possible to inform all but at most* $X/(1-\varepsilon)$ *vertices of* H_d *within* $O(d^2)$ *time steps. Moreover, the number of remaining hyperactive arcs is at most* $X(d-1)$.

In our analysis, we need to assert that enough acknowledgements are delivered, given the number of informed vertices. To bound the number of sent messages, we rely heavily upon the following isoperimetric inequality due to Chung et. al. [13]:

Claim 6.1 **[13]** *Let S be a subset of vertices of H_d. The size of the edge boundary of S, denoted as $\partial(S)$ is defined as the number of edges connecting S to $H_d - S$. Let $\partial(k) = \min_{|S|=k} \partial(S)$, and let* log *denote the logarithm of base 2. It holds that*

$$\partial(k) \geq k(d - \log k)$$

The first step in the analysis is to prove that if there are enough uninformed vertices, or enough hyperactive arcs at the beginning of a round i, then sufficiently many acknowledgements are delivered in this round:

Lemma 6.3.3 *Consider a d-dimensional hypercube with k non-informed vertices and h hyperactive arcs. Let $\varepsilon \in (0,1)$ be an arbitrary constant, and let $k > X/(1-\varepsilon)$ or $h > X(d-1)$. Then in the second step of a simple round at least $\beta(h+\partial(k))$ acknowledgements are delivered, where $\beta = (1-\alpha)^2$.*

Proof: Let S be the set of informed vertices. In the first step of the round, $h + \partial(S)$ messages are sent. Since the edge boundary of informed and uninformed vertices is the same, at least $h + \partial(k)$ messages are sent. We prove that $\alpha(h + \partial(k)) \geq d - 1$, so in the first step at most $\alpha(h + \partial(k))$ messages are lost, and at least $(1-\alpha)(h + \partial(k))$ of them are delivered. Next we prove that $\alpha(1-\alpha)(h + \partial(k)) \geq d - 1$, so in the second step at least $(1-\alpha)^2(h + \partial(k))$ messages are delivered. Since $1 - \alpha < 1$, it is sufficient to prove that $\alpha(1-\alpha)(h + \partial(k)) \geq d - 1$.

If $h > X(d-1)$, then obviously $h + \partial(k) \geq X(d-1)$ and the statement holds. Next, let us consider the case when $h > X/(1-\varepsilon)$. We distinguish three cases and prove that in each case $\partial(k) \geq X(d-1)$.

Case 1: $k \leq 2^{\varepsilon d}$
In this case it holds $\partial(k) \geq k(d - \log k) \geq kd(1-\varepsilon)$. Since $k > X/(1-\varepsilon)$, we get $\partial(k) \geq Xd$.

Case 2: $2^{\varepsilon d} \leq k \leq 2^d\left(1 - \frac{1}{\boldsymbol{e}}\right)$
In this case $\partial(k) \geq k(d - \log k) \geq 2^{\varepsilon d}\left(d - d - \log\left(1 - \frac{1}{\boldsymbol{e}}\right)\right) = 2^{\varepsilon d} \log \frac{\boldsymbol{e}}{\boldsymbol{e}-1} \geq 0.6 \cdot 2^{\varepsilon d}$. Since X is constant, for large enough d it holds $\partial(k) \geq 0.6 \cdot 2^{\varepsilon d} > X(d-1)$.

Case 3: $2^d\left(1 - \frac{1}{\boldsymbol{e}}\right) \leq k$
First, let us consider a function $f(x) := x(d - \log x)$, for $x \in \langle 0, 2^d \rangle$. Since $f'(x) = d - 1/\ln 2 - \log x$,[3] $f(x)$ is increasing for $x \in \langle 0, 2^d/\boldsymbol{e} \rangle$ and decreasing for $x \in \langle 2^d/\boldsymbol{e}, 2^d \rangle$.

Obviously, the edge boundary of uninformed vertices $\partial(k)$ is the same as the edge boundary of informed vertices $\partial(2^d - k)$. Hence, we get $\partial(k) \geq f(2^d - k)$. Since $2^d - k \leq 2^d\frac{1}{\boldsymbol{e}}$, the minimum of $f(2^d - k)$ is attained for the minimal value of $2^d - k$. From Lemma 6.3.1 we know that $2^d - k > \frac{1-\alpha}{2}(2d-1)$, so $\partial(k) \geq f\left(\frac{1-\alpha}{2}(2d-1)\right) = \frac{1-\alpha}{2}(2d -$

[3] $\ln x$ denotes a natural logarithm of x.

1) $\left(d - \log \frac{1-\alpha}{2}(2d-1)\right) = (1-\alpha)d^2 - O(d \log d)$. Hence, for large enough d we get $\partial(k) \geq X(d-1)$. □

In the rest of the proof of Theorem 6.3.2 we show that $O(d^2)$ simple rounds are sufficient to inform almost all vertices. The analysis is divided into two parts. In the first part we prove that within $O(d^2)$ rounds at least $2^d/3$ vertices are informed. In the second part we show that another $O(d^2)$ rounds are sufficient to finish the algorithm.

We use the following auxiliary lemma:

Lemma 6.3.4 *Let $x \geq 2$. It holds that $\log \frac{x+1}{x} \geq \frac{1}{x}$.*

Proof: The statement is equivalent to:

$$\forall x \geq 2 : \frac{1}{x} \geq 2^{\frac{1}{x}} - 1$$

Substituting $y := \frac{1}{x}$:

$$\forall y \in (0, 1/2\rangle : y \geq 2^y - 1$$

For $y = 0$ the equality holds. Hence it is sufficient to prove that the derivative of the left side is larger than the derivative of the right side for $y \in (0, 1/2\rangle$, i.e. $1 \geq 2^y \ln 2$, which obviously holds. □

Lemma 6.3.5 *After performing $O(d^2)$ simple rounds on H_d at least $2^d/3$ vertices are informed.*

Proof: Let $l := 2^d - k$ be the number of informed vertices and b be the number of passive arcs at the beginning of some simple round. Obviously $b \leq ld$. Since the conditions of Lemma 6.3.3 are met, at least $\beta\partial(k)$ acknowledgements are delivered in one simple round. Furthermore, the edge boundary of informed vertices is also the boundary of uninformed vertices, so the number of delivered acknowledgements is at least $\beta\partial(l)$. Because every delivered acknowledgement adds one passive arc, the number of passive arcs grows at least to $b' = b + \beta\partial(l)$ after this round.

First, let us consider a function $f(x) := x(d - \log x)$, for $x \in \langle 0, 2^d \rangle$. Since $f'(x) = d - 1/\ln 2 - \log x$, $f(x)$ is increasing for $x \in \langle 0, 2^d/\boldsymbol{e} \rangle$ and decreasing for $x \in \langle 2^d/\boldsymbol{e}, 2^d \rangle$.

As $b/d \leq l \leq 2^d/3 \leq 2^d/e$ it holds that $\partial(b/d) \leq \partial(l)$. Hence we have the following lower bound on b':

$$b' \geq b + \beta\partial\left(\frac{b}{d}\right) \geq b + \beta\frac{b}{d}\left(d - \log\frac{b}{d}\right) \geq b\left(1 + \beta\frac{d - \log\frac{b}{d}}{d}\right)$$

The lower bound on b implies the inequality $\log \frac{b}{d} \leq d + \log(1/3))$. Hence it holds

$$b' \geq b\left(1 + \beta\frac{-\log(1/3)}{d}\right) = b\left(1 + \frac{1}{\frac{d}{\beta \log 3}}\right)$$

We have shown that the number of passive arcs grows exponentially with number of simple rounds performed. As it can not grow above $d2^d/3$ without informing at least $2^d/3$ vertices, we can estimate an upper bound on number of required simple rounds:

$$T \leq \frac{\log(d2^d/3)}{\log\left(1+\frac{1}{\frac{d}{\beta\log 3}}\right)}$$

For large enough d, Lemma 6.3.4 is applicable, hence proving the Lemma:

$$T \leq \log(d2^d/3)\frac{d}{\beta\log 3} = O\left(d^2\right)$$

□

Lemma 6.3.6 *Let $\varepsilon \in (0,1)$ be an arbitrary constant, and let $k_i \leq (2/3)2^d$ be the number of uninformed vertices and h_i be the number of hyperactive arcs of a d-dimensional hypercube at the beginning of round i. Then after $O(d^2)$ simple rounds there are at most $X/(1-\varepsilon)$ uninformed vertices and at most $X(d-1)$ hyperactive arcs.*

Proof: Similarly to the proof of Theorem 6.2.4 let us consider the measure $M_i := 2dk_i + h_i$. Requirements of the Lemma ensure that $M_i \leq O(d2^d)$. It is easy to see that M_i decreases with every acknowledgement delivered: if the acknowledgement is delivered over a hyperactive arc, the value of h_i decreases by 1. If it is delivered over an active arc, new vertex is informed, hence the value of k_i decreases by 1 and the value of h_i increases by at most $2d-1$.

We show that the value of M_i decreases by a certain multiplicative factor in every simple round as long as the requirements of Lemma 6.3.3 hold. In one simple round at least $\beta(h_i + \partial(k_i))$ acknowledgements are delivered, hence the value M_i decreases to at most:

$$\begin{aligned} M_{i+1} &\leq 2dk_i + h_i - \beta(h_i + \partial(k_i)) \leq h_i(1-\beta) + 2dk_i - \beta k_i(d - \log k_i) = \\ &= h_i(1-\beta) + 2dk_i\left(1 - \beta + \beta\frac{\log k_i}{d}\right) \end{aligned}$$

Using the inequality $\log k_i \leq d + \log(2/3)$ yields:

$$M_{i+1} \leq h_i(1-\beta) + 2dk_i\left(1 + \frac{\beta\log(2/3)}{d}\right)$$

Hence for large enough d it holds:

$$M_{i+1} \leq (h_i + 2dk_i)\left(1 + \frac{\beta\log(2/3)}{d}\right)$$

Since the requirements of the Lemma ensures that $M_i \leq (7/3)d2^d$, the requirements of Lemma 6.3.3 can hold for at most

$$T := \frac{\log\left(\frac{7}{3}d2^d\right)}{\log\left(1+\frac{\frac{1}{d}}{\frac{\beta}{\log(2/3)}}\right)}$$

time steps. According to Lemma 6.3.4 for large d it holds that

$$T \leq \log\left(\frac{7}{3}d2^d\right)\frac{d}{\beta\log(2/3)} = O\left(d^2\right)$$

which concludes the proof. □

Combining Lemma 6.3.1 with Lemma 6.3.5 and Lemma 6.3.6 completes the proof of Theorem 6.3.2.

6.4 Complete Broadcast in Complete Graphs

In Section 6.2 we have shown how to inform all but some constant number of vertices in a complete graph K_n in time $O(\log n)$. A natural question is to ask if it is possible to inform also the remaining vertices in the same time complexity. In this section, we partially answer this question. In particular, we show in the following subsection that if the graph is equipped with a chordal sense of direction, then the complete broadcasting can be performed in time $O(\log n)$. In the subsequent subsection, we show that if the constant $\alpha \lesssim 0.55$, complete broadcasting can be performed in time $O(\log n)$ without the sense of direction, too.

6.4.1 Chordal Sense of Direction

We show how to perform a complete broadcast on a complete graph with the sense of direction in time $O(\log n)$. The process consists of three steps. First, using Theorem 6.2.4, all but a constant number of vertices are informed. In the second phase the information is delivered to all but one vertex. In the last phase the remaining single vertex is informed.

The sense of direction is essential to our algorithm. Since there is a unique initiator of the broadcasting, all vertices can derive unique identifiers defined as the edge label of the link by which they can be reached from the initiator (compare Section 5.3.1). Furthermore, the sense of direction allows each vertex to know the identifier of a destination vertex of any of its incident arcs.

Lemma 6.4.1 *It is possible to inform all vertices but one on complete graphs with chordal sense of direction in time $O(\log n)$. Furthermore, after finishing the algorithm vertex 0 or vertex 1 knows a constant number of candidates for the uninformed vertex.*

Proof: The outline of the algorithm is as follows: At first the algorithm from Theorem 6.2.4 is performed, which ensures that all but a constant number of vertices are informed. Afterwards a significant group of vertices negotiate a common set U of *candidates* for uninformed vertices, such that all uninformed vertices are in U and the size of U is constant. The vertices then cooperate to inform all vertices in U but one. As a side effect, the set U will be known to vertex 0 or vertex 1, hence satisfying the second claim of the lemma. Now we present this algorithm in more detail:

Phase 1 Run the algorithm from Theorem 6.2.4. This phase takes $O(\log n)$ time and ensures that there are at most $X\varepsilon$ uninformed vertices and at most $X(n-2)$ hyperactive arcs.

Phase 2 Each vertex v that has at most $3X(1+\varepsilon)$ non-passive (i.e. active or hyperactive) links leading to the set of vertices U_v sends a message containing U_v to vertices with number 0 and 1.

Now we show that at least one of these messages is delivered. It is easy to see that there are at least $2n/3$ vertices satisfying the above-mentioned condition, otherwise there would be more than $n/3$ vertices with at least $3X(1+\varepsilon)$ non-passive links, so there would be more than $nX(1+\varepsilon)$ active or hyperactive arcs. But since the number of uninformed vertices is at most $k \leq X\varepsilon \leq n/2$ for large n, there are $k(n-k) \leq X\varepsilon(n-X\varepsilon)$ active arcs. So the total number of active or hyperactive arcs is at most $X\varepsilon(n-X\varepsilon) + X(n-2) \leq Xn(1+\varepsilon)$, which is a contradiction.

The rest of the algorithm will be time-multiplexed into two parts. In even time steps, the case that the vertex 0 received a message in phase 2 is processed. In odd time steps, the case that the vertex 1 received a message is processed analogously. Hence, we can restrict to the first case in the rest of the algorithm description. As there are only two cases the asymptotic complexity of the algorithm is unaffected by the multiplexing.

Phase 3 The vertex 0 received at least one message containing a set of possibly uninformed vertices. It is obvious that the set of uninformed vertices is a subset of every received message. Hence the set U can be defined as the intersection of the received messages: Indeed, every uninformed vertex is in U and the size of U is at most $3X(1+\varepsilon) = O(1)$. The set U is then distributed using the algorithm in Theorem 6.2.4 among at least $n - X\varepsilon$ vertices in time $O(\log n)$.

Phase 4 There are at least $n-X\varepsilon$ vertices aware of the set U. In this phase, they cooperate to inform all but one vertex in U, using an idea similar to the second part of the algorithm described in Section 5.3.1 (and also in Lemma 2 in [17]): every vertex aware of the set U iterates through all pairs $[i,j]$ $(i,j \in U)$ in lexicographical order; in each time step it sends the original message to both vertices i and j. Since in each time step at least $2n - X\varepsilon$ messages are sent, at least one of them is delivered (for large enough n). As all vertices process the same pair $[i,j]$ in every time step, this ensures

that a new vertex is informed whenever both i and j were uninformed. Hence, at the end of this phase all vertices but one are informed. The time complexity of this phase is $O(|U|^2) = O(1)$.

It is obvious that after finishing the Phase 4 the claim of the Lemma holds. □

Finally, we show how to inform the last remaining vertex, thus proving the following theorem:

Theorem 6.4.2 *It is possible to perform broadcasting on complete graphs with chordal sense of direction in time* $O(\log n)$.

Proof: We present an algorithm for solving the broadcasting problem:

Phase 1 The algorithm from Lemma 6.4.1 is used. This takes $O(\log n)$ time, all vertices but one are informed and the vertex 0 or the vertex 1 knows a set U of constant size containing candidates for the uninformed vertex.

The rest of the algorithm is multiplexed into two parts, treating these two cases separately. In the remaining of the description we assume that the vertex 0 knows the set U.

Phase 2 The algorithm from Lemma 6.4.1 is used to broadcast the set U, together with the original information, to all vertices but one. This takes $O(\log n)$ time again.

After the Phase 2 is finished, two cases are possible: Either the uninformed vertex of the Phase 2 is different from or is the same as the uninformed vertex of the Phase 1. In the former case all vertices are informed. The rest of the algorithm handles the latter case.

Phase 3 If not all vertices are informed, then there is a single uninformed vertex v. Furthermore, every informed vertex knows the set U of constant size such that $v \in U$. Every informed vertex iterates through the set of U; in i-th time step of the current phase it sends the message to i-th member of U. Eventually, the uninformed vertex is processed. Since all $n-1$ informed vertices are doing the same, exactly $n-1$ messages are sent to the uninformed vertex, hence finishing the broadcast.

The time complexity of the Phase 1 and Phase 2 is $O(\log n)$; the time complexity of the Phase 3 is $O(|U|) = O(1)$. Hence the algorithm correctly solves the broadcasting on complete graphs in time $O(\log n)$. □

6.4.2 Without Sense of Direction

As a last result in this chapter we show that it is possible to perform broadcasting on complete graphs in time $O(\log n)$ for small values of α (i.e. $\alpha \lesssim 0.55$) even without the sense of direction. The idea is to use the algorithm from Theorem 6.2.4 to inform all but constantly many vertices. Next, instead of repeating 2-step simple rounds, $O(\log n)$-step extended rounds are repeated, such that each extended round informs a yet uninformed vertex. During an extended round messages are sent for $O(\log n)$ steps in such a way that in every step the number of hyperactive arcs is decreased by some factor[4] unless a new vertex is informed.

Theorem 6.4.3 *Let $1 - \alpha - 2\alpha^2 + \alpha^3 > 0$. Then it is possible to perform broadcasting on complete graphs without sense of direction in time $O(\log n)$.*

Proof: The algorithm is described as Algorithm 6.

Algorithm 6 Complete graphs without sense of direction

1: perform almost-complete broadcast according to Theorem 6.2.4
2: let k denote the number of uninformed vertices, let h denote the number of hyperactive arcs
3: **loop** L_1 times *// Perform L_1 extended rounds*
4: **loop** $L_2(n)$ times *// In each iteration h decreases by a constant factor*
5: $E :=$ set of all currently active or hyperactive arcs
6: $P := \emptyset$ *// Acknowledgements to be sent*
7: **loop** L_3 times
8: send the message via all arcs in $E \cup P$
9: $P := P \cup \left\{ e \mid \begin{array}{l} \text{a message has been delivered in this step} \\ \text{via the opposite arc of } e \end{array} \right\}$
10: **end loop**
11: **end loop**
12: **loop** $L_4(n)$ times *// Inform new vertex and decrease a*
13: perform one simple round
14: **end loop**
15: **end loop**

The values of L_1, $L_2(n)$, L_3 and $L_4(n)$ are specified in the analysis of the algorithm, such that $L_1, L_3 = O(1)$ and $L_2(n), L_4(n) = O(\log n)$.

At first, the algorithm from Theorem 6.2.4 is performed, ensuring that there are at most $k \leq X\varepsilon$ uninformed vertices and at most $h \leq X(n-2)$ hyperactive arcs (X and ε have the same meaning as in Theorem 6.2.4). The purpose of one iteration of the loop on lines 3–15 is to inform at least one uninformed vertex. Taking $L_1 := X\varepsilon = O(1)$ ensures that all vertices will be informed.

[4] in this part we need the assumption that α is small enough

The loop on lines 4–11 reduces the number of hyperactive arcs to zero unless a new vertex is informed. One iteration of this loop either informs a new vertex or reduces the number of hyperactive arcs from h to $(1-Y/2)h$, where $0 < Y < 1$ is a constant (depending on α) defined later. Hence the number of hyperactive arcs decreases exponentially with number of iterations of the loop and $\log_{1/(1-Y/2)} h$ iterations are sufficient to eliminate all hyperactive arcs. Since the condition $h \leq X(n-2)$ holds before every execution of the loop (this is provided either directly by Theorem 6.2.4 or by the loop on lines 12–14), we can define $L_2 := \log_{1/(1-Y/2)}(X(n-2)) = O(\log n)$.

Now we describe one iteration of the loop on lines 4–11. We distinguish two types of arcs that are hyperactive at the beginning of the considered iteration: An arc e is a *single hyperactive arc* if and only it is hyperactive and the opposite arc of e is passive at the beginning of the iteration. Otherwise (i.e. if both e and the opposite arc of e are hyperactive at the beginning of the iteration), e is a *double hyperactive arc*.

Let E be the set of all active or hyperactive arcs at the beginning of the iteration, and P be the set of all arcs opposite to arcs through which some message has been delivered in the current iteration. Furthermore, let k' be the number of uninformed vertices at the beginning of the current iteration, h' be the number of hyperactive arcs at the beginning of the current iteration and $p = |P - E|$ be number of arcs in P that were passive at the beginning of the current iteration. It clearly holds that $|E| = k'(n-k') + h'$ and that $k'(n-k') + h' + p$ messages are sent on every execution of line 8. Since at least $n-1$ messages are lost (because we may assume that no new vertex is informed), at most $\alpha(k'(n-k')+h'+p)$ of them are lost, i.e. at least $(1-\alpha)(k'(n-k')+h'+p)$ are delivered.

Now assume by contradiction that the number of hyperactive arcs does not decrease below $(1-Y/2)h'$, and no new vertices are informed during the current iteration of the loop on lines 4–11. Consider any message delivered over an arc e which is a double hyperactive arc or an arc in $P - E$; it is easy to see that the opposite arc of e is passive after the delivery and that it was hyperactive at the beginning of the iteration. This fact yields that at most $(Y/2)h'$ messages are delivered over a double hyperactive arc or an arc in $P - E$ on any execution of line 8.

Now we show a lower bound on the number of messages that pass over double hyperactive arcs or arcs in $P - E$ or single hyperactive arcs whose opposite arcs are not in $P - E$. Intuitively, every such message ensures some progress of the algorithm, since either an arc is made passive (in the first two cases) or a new arc is added to $P - E$ (in the third case). As no messages passes over active arcs by our assumption, and at most p messages pass over single hyperactive arcs whose opposite arcs are in $P - E$, there are at least $(1-\alpha)(k'(n-k')+h'+p) - p$ messages satisfying one of these three cases. Using the inequalities $k'(n-k') \geq n-1$ and $p \leq h'$ yields $(1-\alpha)(k'(n-k')+h'+p) - p \geq (1-\alpha)(n-2)+(1-2\alpha)h'$. Because $h' \leq X(n-2)$ which is equivalent to $(n-2) \geq \alpha(1-\alpha)h'$, we have $(1-\alpha)(k'(n-k')+h'+p)-p \geq (1-\alpha-2\alpha^2+\alpha^3)h'$. Defining $Y := 1-\alpha-2\alpha^2+\alpha^3$, which is positive and less than one by the assumption of the Lemma, we have shown that there are at least Yh' messages satisfying one of the three cases.

However, at most $(Y/2)h'$ of them satisfies the first two cases, hence there are at least $(Y/2)h'$ arcs added to P in every execution of line 9. So taking $L_3 := 2/Y + 1$ ensures that

P contains opposite arcs to all single hyperactive arcs at the beginning of the last iteration of the loop on lines 7–10. However, this is a contradiction with the fact that new arcs are added to P at line 9.

We conclude the proof with the analysis of the loop on lines 12–14. In the first iteration of the loop a new vertex is informed, because there are no hyperactive arcs left after the loop on lines 4–11 finished (unless the new vertex has already been informed in that loop). Due to Theorem 6.2.4, next $O(\log n)$ iterations are sufficient to ensure that $h \leq X(n-2)$, which is an invariant required by the loop on lines 4–11. Hence putting $L_4(n) := O(\log n)$ (according to Theorem 6.2.4) is sufficient to make the algorithm work correctly in time $L_1(L_2(n)L_3 + L_4(n)) = O(\log n)$. □

Chapter 7

Conclusion

In this thesis, we have analyzed the problem of broadcasting in faulty networks. At first, we presented an overview of different models of distributed computations and different models of network failures. As can be seen from Chapter 2, there is a vast amount of different models of distributed computations in presence of faults. Unfortunately, these models often exhibit substantially different properties, and not much is known about their mutual relationship. This contrasts to the situation in the classical sequential computation models (e.g. Turing machines, random-access machines, etc.), where it is often possible to simulate computations in one model by another one with reasonably small slowdown.

In Chapter 2, we presented a framework for describing the models of computations in presence of faults. We introduced several independent features of the models that can be almost arbitrarily combined. Our framework allows to define very large number of concrete models easily, just by enumerating the corresponding set of features. Many previously analyzed models of distributed computations can be expressed in this way.

In our work, we focused specifically on the broadcasting problem. As this is one of the most basic tasks of the information dissemination problems, it has diverse implications, both practical and theoretical. The broadcasting problem is an inherent part of many practical tasks in distributed systems (see e.g. [56]). Moreover, the analysis of the broadcasting problem on various communication topologies can provide good insight on the influence of the structure of the communication graph, the model of computation, and the type and amount of failures on the efficiency of the distributed algorithms. The importance of the broadcasting problem can also be seen from the cited references: The area of faulty broadcasting has been studied for a long time (the earliest cited publications were published in late eighties). On the other side, this area is still alive and attractive to researchers – a lot of papers relevant to this area were published in last few years.

The core of this thesis, Chapters 3, 4, 5, and 6, present a quest for the "good" deterministic model of faulty distributed computations. In Chapter 3, we presented some interesting results about the broadcasting in the *constant k-bounded model* – the oldest and most studied deterministic model of faulty distributed computations. However, there is a weak point in this model: To perform a broadcast, it is always optimal to flood the network with as many messages as possible.

In Chapter 4, we analyzed the *fractional α-weakly-bounded model*, which has been proposed in [41] to remove the above-mentioned undesirable property of the constant k-bounded model. After presenting the results of [41], which dealt with a fixed value of $\alpha = 1/2$, we provided their generalized counterparts for arbitrary values of α.

Our results provide both examples of "robust" topologies, where the broadcasting can not be slowed down asymptotically in comparison with the fault-free computation, and "fragile" topologies, where it can be slowed down. The former include multidimensional tori, the latter include complete graphs and complete trees.

The fractional α-weakly-bounded model, however, exhibits another inappropriate feature, in some sense exactly the opposite of the flooding problem: If there are too few messages sent, all of them are delivered for sure. In some situations, the broadcasting algorithm could exploit this. However, this guarantee is not well motivated; there is no good reason to assume a completely fault-free computation even if the network usage is small.

To fix both the problem of the constant k-bounded model and the fractional α-weakly-bounded model, we introduced the *fractional threshold model* in [17]. This model of faults forces the broadcasting algorithm to send many messages in each time step. On the other hand, the number of faults grows with the number of messages sent, hence the algorithm is penalized for the excessive network usage.

Before focusing on the fractional model, we analyzed the *simple threshold model*, which has been introduced in [17], too. This model of faults provides only minimalistic guarantees on the number of successful message transmissions: Only one message is guaranteed to pass in any single time step, and even this weak guarantee holds only if there are many messages sent in the corresponding time step. Although this model seems to be too harsh, there are good reasons to analyze it. Since it represents the most difficult model of faults (from the broadcasting algorithm's point of view), any upper bounds on broadcasting time obtained in it can be directly transferred to the constant k-bounded model or to the fractional threshold model.[1]

We presented results about the simple threshold model in Chapter 5. These results, published also in [17], are surprisingly positive. Although it is not easy to see if the broadcasting is always possible in the simple threshold model, we managed to show that this is indeed the case. Furthermore, the broadcasting can be performed in time polynomial in the size of the communication graph for many communication network topologies (see Table 5.1). In particular, we presented a broadcasting algorithm working in polynomial time on every communication graph with small edge connectivity (the edge connectivity has to be logarithmic in the size of the communication graph), even without any topology knowledge.

At last but not least, we dealt with the general case of the fractional threshold model in Chapter 6. We presented results about complete graphs and hypercubes, which have been published in [42], too.

Our experiences from the fractional threshold model suggest that it is usually easy

[1] Note that we are not mentioning the fractional α-weakly-bounded model here, since only the messages sent to uninformed vertices count when determining the maximal number of faults in this model.

to inform the vast majority of the nodes. Nevertheless, informing the remaining ones at the end of the broadcast requires tight cooperation of all informed vertices, which might be difficult to achieve. We presented simple broadcasting algorithms for both complete graphs and hypercubes for any fixed value of $\alpha < 1$ which inform all but constant number of vertices in logarithmic time. Furthermore, we were able to extend these algorithms to perform the complete broadcasting on complete graphs under certain conditions, in particular, if there is either chordal sense of direction available, or the value of α is small enough ($\alpha \lesssim 0.55$). For the complete overview of results of Chapter 6, see Table 6.1.

Several interesting problems connected to the models presented in this thesis remain open. In the simple threshold model, it is not known if it is possible to finish the broadcasting in polynomial time on any communication graph. Although our techniques for fast broadcasting on complete graphs and low edge-connectivity graphs can not be directly applied on the general case, their main ideas might be useful in the search for a polynomial broadcasting algorithm on arbitrary network topology. However, we have no strong arguments that such algorithm exists at all.

Another group of interesting open problems is connected with the fractional threshold model. One obvious question is to ask if it is possible to perform a complete broadcasting in complete graphs also for large values of α in polylogarithmic time. The difficulty of broadcast in the fractional dynamic model with threshold stems from the fact that, in order to inform the last few vertices, all informed vertices must cooperate very tightly. In general, the relationship between the almost complete and complete broadcast in various models is worth studying. Furthermore, it would be interesting to extend our results to more general classes of graphs.

We have presented some lower bounds in the considered models of broadcasting. Nevertheless, it might be interesting to develop more techniques for proving such lower bounds, especially in connection with the fractional threshold and simple threshold model.

Despite of the above-mentioned remaining open problems, the thesis fulfilled all its goals mentioned in the Foreword. We generalized the results of fractional α-weakly-bounded model for $\alpha = 1/2$ to arbitrary values of α. Next, we introduced the simple threshold and fractional threshold models, which eliminate the undesirable properties of both the constant k-bounded and the fractional α-weakly-bounded models. Furthermore, we provided several results about broadcasting in these models that gives a reasonable insight into their properties. Hence, results presented in this thesis help to understand the relation between the structure of the communication graphs and their fault resistance in different model of faults from the theoretical point of view.

Bibliography

[1] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Thrifty generic broadcast. In *DISC '00: Proceedings of the 14th International Conference on Distributed Computing*, pages 268–282, London, UK, 2000. Springer-Verlag.

[2] R. Ahlswede, L. Gargano, H. S. Haroutunian, and L. H. Khachatrian. Fault-tolerant minimum broadcast networks. *Networks*, 27, 1996.

[3] R. Ahlswede, H. Haroutunian, and L. H. Khachatrian. Messy broadcasting in networks. In Blahut, Costello, Maurer, and Mittelholzer, editors, *Communications and Cryptography: Two Sides of One Tapestry*. Kluwer Academic Publishers, 1994.

[4] A. Bagchi and S. L. Hakimi. Information dissemination in distributed systems with faulty units. *IEEE Transactions on Computers*, 43(6):698–710, 1994.

[5] P. Berman, K. Diks, and A. Pelc. Reliable broadcasting in logarithmic time with Byzantine link failures. *Journal of Algorithms*, 22(2):199–211, 1997.

[6] J.-C. Bermond, N. Marlin, D. Peleg, and S. Perennes. Directed virtual path layouts in atm networks. *Theoretical Computer Science*, 291(1):3–28, 2003.

[7] D. Bienstock. Broadcasting with random faults. *Discrete Applied Mathematics*, 20(1):1–7, 1988.

[8] B. Bollobás and I. Leader. An isoperimetric inequality on the discrete torus. *SIAM Journal on Discrete Mathematics*, 3(1):32–37, 1990.

[9] J. Bruck. On optimal broadcasting in faulty hypercubes. *Discrete Applied Mathematics*, 53(1–3):3–13, 1994.

[10] B. Chlebus, K. Diks, and A. Pelc. Broadcasting in synchronous networks with dynamic faults. *Networks*, 27, 1996.

[11] B. S. Chlebus, K. Diks, and A. Pelc. Optimal broadcasting in faulty hypercubes. In *FTCS*, pages 266–273, 1991.

[12] B. S. Chlebus, K. Diks, and A. Pelc. Waking up an anonymous faulty network from a single source. In *HICSS (2)*, pages 187–193, 1994.

[13] F. R. K. Chung, Z. Füredi, R. L. Graham, and P. Seymour. On induced subgraphs of the cube. *J. Combin. Theory Ser. A*, 49(1):180–187, 1988.

[14] A. Dessmark and A. Pelc. Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(1–3):343–362, 2004.

[15] K. Diks, E. Kranakis, and A. Pelc. Broadcasting in unlabeled tori. *Parallel Processing Letters*, 8(2):177–188, 1998.

[16] K. Diks and A. Pelc. Almost safe gossiping in bounded degree networks. *SIAM Journal on Discrete Mathematics*, 5(3):338–344, 1992.

[17] S. Dobrev, R. Královič, R. Královič, and N. Santoro. On fractional dynamic faults with threshold. In P. Flocchini and L. Gasieniec, editors, *SIROCCO*, volume 4056 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2006.

[18] S. Dobrev and I. Vrťo. Optimal broadcasting in hypercubes with dynamic faults. *Information Processing Letters*, 71(2):81–85, 1999.

[19] S. Dobrev and I. Vrťo. Optimal broadcasting in tori with dynamic faults. *Parallel Processing Letters*, 12(1):17–22, 2002.

[20] S. Dobrev and I. Vrťo. Dynamic faults have small effect on broadcasting in hypercubes. *Discrete Applied Mathematics*, 137(2):155–158, 2004.

[21] S. Even and B. Monien. On the number of rounds necessary to disseminate information. In *SPAA '89: Proceedings of the first annual ACM symposium on Parallel algorithms and architectures*, pages 318–327, New York, NY, USA, 1989. ACM Press.

[22] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[23] P. Flocchini and B. Mans. Optimal elections in labeled hypercubes. *Journal of Parallel and Distributed Computing*, 33(1):76–83, 1996.

[24] P. Flocchini, B. Mans, and N. Santoro. Sense of direction: definitions, properties, and classes. *Networks*, 32(3):165–180, 1998.

[25] P. Fragopoulou and S. G. Akl. Edge-disjoint spanning trees on the star network with applications to fault tolerance. *IEEE Transactions on Computers*, 45(2):174–185, 1996.

[26] P. Fraigniaud. Asymptotically optimal broadcasting and gossiping in faulty hypercube multicomputers. *IEEE Transactions on Computers*, 41(11):1410–1419, 1992.

[27] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. In *Proceedings of the international workshop on Broadcasting and gossiping 1990*, pages 79–133, New York, NY, USA, 1994. Elsevier North-Holland, Inc.

[28] P. Fraigniaud and C. Peyrat. Broadcasting in a hypercube when some calls fail. *Information Processing Letters*, 39(3):115–119, 1991.

[29] L. Gargano, A. L. Liestman, J. G. Peters, and D. Richards. Reliable broadcasting. *Discrete Applied Mathematics*, 53(1-3):135–148, 1994.

[30] L. Gargano and A. A. Rescigno. Communication complexity of fault-tolerant information diffusion. *Theoretical Computer Science*, 209(1–2):195–211, 1998.

[31] L. Gargano, A. A. Rescigno, and U. Vaccaro. Minimum time broadcast in faulty star networks. *Discrete Applied Mathematics*, 83(1-3):97–119, 1998.

[32] L. Gargano, U. Vaccaro, and A. Vozella. Fault tolerant routing in the star and pancake interconnection networks. *Information Processing Letters*, 45(6):315–320, 1993.

[33] L. Gasieniec and A. Pelc. Adaptive broadcasting with faulty nodes. *Parallel Computing*, 22(6):903–912, 1996.

[34] L. Gasieniec and A. Pelc. Broadcasting with a bounded fraction of faulty nodes. *Journal of Parallel and Distributed Computing*, 42(1):11–20, 1997.

[35] L. Gasieniec and A. Pelc. Broadcasting with linearly bounded transmission faults. *Discrete Applied Mathematics*, 83(1–3):121–133, 1998.

[36] Y. J. Han, Y. Igarashi, K. Kanai, and K. Miura. Broadcasting in faulty binary jumping networks. *Journal of Parallel and Distributed Computing*, 23(3):462–467, 1994.

[37] S. Hedetniemi, S. Hedetniemi, and A. Liestman. A survey of broadcasting and gossiping in communication networks. *Networks*, 18:319–349, 1988.

[38] J. Hromkovič, R. Klasing, B. Monien, and R. Peine. Dissemination of information in interconnection networks (broadcasting & gossiping). In D.-Z. Du and D. Hsu, editors, *Combinatorial Network Theory*, pages 125–212. Kluwer Academic Publishers, Netherlands, 1996.

[39] Z. Jovanović and J. Mišić. Fault tolerance of the star graph interconnection network. *Information Processing Letters*, 49(3):145–150, 1994.

[40] D. R. Kowalski and A. Pelc. Deterministic broadcasting time in radio networks of unknown topology. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 63–72, Washington, DC, USA, 2002. IEEE Computer Society.

[41] R. Královič, R. Královič, and P. Ružička. Broadcasting with many faulty links. In J. F. Sibeyn, editor, *SIROCCO*, volume 17 of *Proceedings in Informatics*, pages 211–222. Carleton Scientific, 2003.

[42] R. Královič and R. Královič. Rapid almost-complete broadcasting in faulty networks. In G. Prencipe and S. Zaks, editors, *SIROCCO*, volume 4474 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2007.

[43] S. Latifi. On the fault-diameter of the star graph. *Information Processing Letters*, 46(3):143–150, 1993.

[44] Z. Liptak and A. Nickelsen. Broadcasting in complete networks with dynamic edge faults. In F. Butelle, editor, *OPODIS*, Studia Informatica Universalis, pages 123–142. Suger, Saint-Denis, rue Catulienne, France, 2000.

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

[46] G. D. Marco and A. A. Rescigno. Tighter time bounds on broadcasting in torus networks in presence of dynamic faults. *Parallel Processing Letters*, 10(1):39–49, 2000.

[47] G. D. Marco and U. Vaccaro. Broadcasting in hypercubes and star graphs with dynamic faults. *Information Processing Letters*, 66(6):321–326, 1998.

[48] V. E. Mendia and D. Sarkar. Optimal broadcasting on the star graph. *IEEE Transactions on Parallel and Distributed Systems*, 3(4):389–396, 1992.

[49] A. Pelc. Broadcasting in complete networks with faulty nodes using unreliable calls. *Information Processing Letters*, 40(3):169–174, 1991.

[50] A. Pelc. Fault-tolerant broadcasting and gossiping in communication networks. *Networks*, 28(3):143–156, 1996.

[51] A. Pelc and D. Peleg. Broadcasting with locally bounded Byzantine faults. *Information Processing Letters*, 93(3):109–115, 2005.

[52] A. Pelc and D. Peleg. Feasibility and complexity of broadcasting with random transmission failures. In *PODC '05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 334–341, New York, NY, USA, 2005. ACM Press.

[53] D. Peleg. A note on optimal time broadcast in faulty hypercubes. *Journal of Parallel and Distributed Computing*, 26(1):132–135, 1995.

[54] P. Ramanathan and K. G. Shin. Reliable broadcast in hypercube multicomputers. *IEEE Transactions on Computers*, 37(12):1654–1657, 1988.

[55] N. Santoro and P. Widmayer. Distributed function evaluation in the presence of transmission faults. In *SIGAL '90: Proceedings of the International Symposium on Algorithms*, pages 358–367, London, UK, 1990. Springer-Verlag.

[56] A. Schiper. Group communication: From practice to theory. In J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, and J. Štuller, editors, *SOFSEM 2006: Theory and Practice of Computer Science*, volume 3831 of *Lecture Notes in Computer Science*, pages 117–136. Springer-Verlag, 2006.

[57] G. Tel. *Introduction to distributed algorithms.* Cambridge University Press, New York, NY, USA, 1994.