Deterministic Models of Communication Faults*

Rastislav Královič[1] and Richard Královič[1 2]

[1] Department of Computer Science, Comenius University,
Mlynská dolina, 84248 Bratislava, Slovakia.
[2] Department of Computer Science, ETH Zurich, Switzerland.

Abstract. In this paper we survey some results concerning the impact of faulty environments on the solvability and complexity of communication tasks. In particular, we focus on deterministic models of faults in synchronous networks, and show how different variations of the model influence the performance bounds of broadcasting algorithms.

1 Introduction

One of the main challenges in the design of complex systems such as computers is the issue of robustness. Indeed, systems comprised of a large number, albeit highly resilient, components experience the malfunction of some of them with probability high enough to be taken into account. Computers use e.g. sophisticated error-correcting procedures transparent to the software environment that maintain integrity of data stored in memory so that it can be viewed as a reliable storage medium.

Large scale communication networks consist of very high numbers of computing elements (computers, routers, etc.) and due to this complexity the issue of fault tolerance plays an important role in their design. Moreover, communication networks are accessible to a number of people, and are often targets of misuse and hacker attacks. The aim of the design is to develop the communication protocols in such a way that the network can serve its purpose (communication, cooperative computation, etc.) even with a number of components being permanently or temporarily out of operation.

In order to analyze the impact of faulty environments on the performance of the distributed system, two things must be specified: the model of the distributed system (network), and the model of the faulty environment. For the former we use a standard model of point-to-point communication in which the system is composed of independent entities (processors, processes, etc.) which can pairwise communicate by exchanging messages. It is natural to model such system by a graph, where vertices represent the processors, and edges connect those pairs of processors that can communicate. Having said this, there is still a number of issues that have to be addressed: e.g. are the vertices synchronous or asynchronous? Can a vertex send a message to all its neighbors simultaneously?

* The research has been supported by grant APVV-0433-06.

Do the vertices know the graph? It turns out that there is a number of parameters that significantly influence the results. Indeed, the situation in distributed computing is such that there is no simple and elegant model that would capture all the studied properties of communication. Thus, when speaking about the *network model*, a number of options must be specified. We briefly discuss these options in the next section.

Second thing that has to be formally modeled in order to obtain results about fault tolerance is the (faulty) environment. Standard approach of theoretical computer science uses the notion of an *adversary*: an entity that is responsible for introducing faults into the execution of the algorithm. What we call a *model of faults* is basically a set of restrictions that are imposed on the behavior of the adversary. It is always assumed that adversary produces the worst possible scenario within the limitations given by the model. Of course, without any restrictions on the adversary it is not possible to solve any non-trivial task. With some assumptions, certain tasks become solvable, and by restricting the adversary even further, more efficient solutions may be obtained.

There are two main approaches to the definition of the fault model: randomized, in which the behavior of the adversary is a random process (e.g. every message is lost with a given probability), and deterministic which states some worst-case bounds on the adversary's behavior (e.g. at most k messages are lost). Both approaches have advantages and weak-points, and we shall discuss them later in more detail.

Finally, one has also to specify the *task* performed by the network. In this paper we mostly consider the *broadcasting problem* (called also *one-to-all communication*) where the goal is to disseminate a piece of information known to only one vertex of the distributed system to every other vertex. This simply formulated task is an important communication primitive in many more complex distributed algorithms. Moreover, broadcasting can be used to solve 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. Other communication problems include, e.g. the *leader election* problem, where vertices start in identical states[3], and they all have to agree on one distinguished vertex, the *wakeup* problem etc.

As has just been mentioned, the setting of the research consists of three parts: the network model, the fault model, and the task. There are at least three approaches to the study of their relationship:

– Fix the network model (based on current technology), fault model (based on realistic assumptions), and task, and study the complexity (in terms of time or amount of communication) of the task.
– Fix the network model, and the task, and find the least restrictive fault model in which the task is still solvable.

[3] up to some topology knowledge as described by the network model

– Fix the fault model and the task and find the weakest possible network model in which the task is still solvable.

The rest of the paper is organized as follows: in the next two sections we give an overview of features that must be taken into account in the network model, and the model of faults, respectively. Then we present a case study to demonstrate the development of a particular fault model. We focus on the broadcasting problem and show how subsequent changes to the model influence the results. We finish by a brief mention of other considered problems, open questions, and possible future projects.

2 Network models

In this section we briefly describe some of the properties of the networks that can fundamentally influence the results. 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 models can be defined by imposing additional constraints on some model presented in this section.

We consider the *point-to-point* model of the communication, which is widely studied in the literature (see [1–5, 7, 8, 10–12, 14, 15, 18–21, 25, 26, 28–36, 39, 45, 46, 48–51, 53, 55–58] or surveys [27, 37, 38, 52]).[4] 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.

An important aspect of the distributed system is its synchronicity. The cases most widely considered in the literature (see e.g. [47, 60]) are the extreme cases: 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, 14]). 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. [58].

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[5]. Indeed, in the asynchronous computation it is impossible to distinguish between a lost message and a message that will be delivered later (see [22]).

[4] Other considered models include e.g. the *radio network model* (see [54]), *ATM network model* (see [6] and references therein), etc.

[5] one way to overcome this is to enhance the system with additional devices as e.g. *failure detectors* from [9]

For the synchronous systems the notion of *time complexity* is naturally defined – it is the number of time steps needed for the algorithm to terminate. In the *constant* mode (which we shall consider in this survey) each message is delivered in one time step regardless of the length of the message. This mode is the most widely studied one, e.g. in [2,3,5,8,10–12,15,18–21,25,28,29,33–35,46, 48,49,54–56]. On the other side, 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 by the model. This mode is studied e.g. in [26], see also the survey [27].

There are several properties of the communication links that must be taken into consideration. First is 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 [15,25,33,50]. We also distinguish the communication modes based on the capability of the nodes to use multiple links at once: 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 [2,3,5,11,12,21,26,29,33–35,50,56]. 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 [10,18–20,25,26,28,31,46,48,49,55,56]. 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 [8]. We focus on the multi-port mode, unless stated otherwise.

When designing algorithms for communication networks 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 [8,14,18–20,26,31,46,48–50,55,56]). 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:

– *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 [53].

- *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 [13].
- *Identifiers.* Each node knows its own unique identifier. Sometimes the knowledge of the identifiers of neighbors is assumed, too (e.g. in [40,53]). Note that full topology knowledge implies the knowledge of identifiers. Knowledge of the identifiers without full topology knowledge is used e.g. in [4,40,53].
- *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. [60]). 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].

3 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.

There are two main approaches to the modelling of the faulty computations. In the *randomized model*, the faults occur with a certain probability, i.e. the fault model describes the expected behavior of the adversary. 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, 15, 51, 54]. 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 main problem with these approaches is that in real life faults are usually not independent. Recently, a research direction aimed at analyzing dependency in randomized fault models has been introduced in [44]. The second approach is to analyze the worst-case scenario, i.e. the fault model sets rules for the adversary to obey. To do so, the model of faults usually imposes constraints on the number of faults that can occur. This approach has been used e.g. in [2, 4, 8, 10, 18–20, 25, 26, 28–36, 39, 45, 46, 48, 49, 53, 55, 56]. The main goal is to design such a set of rules that are not too restrictive and are realistic in the sense that they do not create unwanted special cases the algorithm can rely upon (e.g. if there is a guarantee that at least one message in every time step is non-faulty, the algorithm may send important messages one at a time – a strategy that has little justification in reality).

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, 51, 55, 56]) and the faults of the links (considered e.g. in [2, 5, 7, 8, 10–12, 15, 18–20, 25, 26, 28, 29, 31, 32, 35, 36, 46, 48, 49, 51, 55, 56]).

Another aspect is the type of faults that may occur. 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, 46, 53, 54]. 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 fails 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, 15, 18–20, 25, 26, 28, 29, 31–35, 39, 45, 46, 48, 49, 54–56]. 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, 8, 31].

Yet 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, 8, 25, 26, 31–34, 36, 39, 45, 55]. 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 [10,15,18–20,28,29,35,41,46,48,49,54].

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 this paper we focus on the time-independent 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] which 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 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.

4 Case study: broadcasting

In this section we demonstrate an evolution of a particular deterministic time-independent model of omission faults on a problem of broadcasting. For the network model we shall consider synchronous, full-duplex, all-port network. If there are no faults, then obviously a simple flooding algorithm delivers the message to all vertices in $d(G)$ time steps, where $d(G)$ is the diameter of the graph. The type of faults we are focusing on are omission faults on links.

The simplest model of faults in this setting is when a fixed number of k static faults is allowed, i.e. at the beginning of the computation the adversary chooses k links that will remain broken during the whole execution; the algorithm is not a-priori aware of which links are functional. This model corresponds to a situation when the faults are long-lasting compared to the execution of the algorithm (e.g. broken cable). The worst-case broadcasting time on a graph G will be $T(k, G) = \max_{G' \subseteq G}\{d(G')\}$, where the maximum is taken over all subgraphs G' obtainable from G by deleting at most k edges. In [59], Schoone, Bodlaender and van Leeuwen studied the worst case value of $T(k, G)$ over all graphs G with given diameter D, and delivered upper bound of the form $(k+1)D$ and a lower bound of the form $(k+1)D-k$ for even D and $(k-1)D-2k+2$ for odd $D \geq 3$.

Sometimes, however, transient faults are more appropriate to model the situation in the network (e.g. packet loss in wireless networks due to background noise). A next step in the evolution of the model is thus to allow the faults to be dynamic: in the *constant k-bounded model*, the adversary can block at most k messages in each time step. Since the faults are dynamic, they may occur on different links during various time steps. This model is one of the oldest models of dynamic faults. It has been introduced in [57] and analyzed in numerous papers, e.g. [10, 18–20, 28, 46, 48, 49].

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. Tight bounds on general graphs are due to Chlebusz, Diks, and Pelc:

Theorem 1. [10] *Let $\mathcal{G}_{d,k}$ be the class of graphs with diameter d and edge connectivity at least $k+1$. Let, for some class of graphs $\mathcal{G}$, denote $T(\mathcal{G})$ the worst case broadcasting time in the constant k-bounded model on graphs from $\mathcal{G}$.*

For a fixed constant d the following holds. $T(\mathcal{G}_{d,k}) = O(k^{\frac{d}{2}-1})$. Moreover, there exists a class of graphs $\mathcal{G}'_{d,k} \subseteq \mathcal{G}_{d,k}$ such that $T(\mathcal{G}'_{d,k}) = \Omega(k^{\frac{d}{2}-1})$.

For a fixed constant k the following holds. $T(\mathcal{G}_{d,k}) = O(d^{k+1})$. Moreover, there exists a class of graphs $\mathcal{G}''_{d,k} \subseteq \mathcal{G}_{d,k}$ such that $T(\mathcal{G}''_{d,k}) = \Omega(k^{\frac{d}{2}-1})$.

A number of special topologies has been investigated within this model. The broadcasting in complete graphs in constant k-bounded model is completely solved (together with the case of Byzantine faults) in [46]:

Theorem 2. *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 $n - 1 \leq k$.*

The broadcasting in the constant k-bounded model in tori is discussed in [48] and [19]. The results from [48] are strengthened in [19] to the following form:

Theorem 3. (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 $\frac{dk}{2} + 2$.

The broadcasting in the constant k-bounded model in hypercubes is considered e.g. in [18, 20, 28] and [49]. The results from these papers are summarized in the following theorem:

Theorem 4. *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 $T(d, k) = d$.*

The above results are optimistic in what they claim: the adversary cannot delay the progress of the algorithm too much. However, it seems that these positive results are mainly due to the fact that the constant k has to be relatively small[6] compared to the number of edges that the algorithm can use during its computation. Indeed, the lower bounds are based on isoperimetric inequalities: once a small constant number of vertices are informed, it can be guaranteed that the edge-boundary of the set of informed vertices is large enough to allow progress. This is a somewhat unrealistic behavior of the model, since adding

[6] at most the connectivity of the graph, in order to grant non-trivial results

more messages to the network does not introduce new errors; indeed, flooding the networks with vast amount of traffic is the best solution in this model, although not so much in real-life networks.

Hence, a next step was to introduce a model in which the number of faults could vary during each time step, with the intuition borrowed from the probabilistic models: if every message has a fixed probability p of failure, and there are m messages sent in a particular step, the average number of delivered messages is pm. In the *fractional α-bounded* model, if there are m messages sent during a particular time step, at most $\lfloor \alpha m \rfloor$ of them can be lost.

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

The main interest of [41] is the following question: In which graphs is the broadcasting time of the greedy algorithm[7] 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. More precisely, the time of broadcasting on complete graph K_n increases from $\Theta(1)$ to $\Theta(\log n)$. The following theorems were stated for $\alpha = 1/2$ in [41] and generalized to arbitrary α in [42]:

Theorem 5. *Let $T(n)$ be the broadcasting time of the greedy algorithm in the α-bounded model on the complete graph K_n. The following inequality holds:*

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

The situation in complete trees is similar; e.g. the broadcasting time of the greedy algorithm on the complete d-ary tree of height h in presence of fractional 1/2-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 result stated for arbitrary constant α is:

Theorem 6. *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 $p = \lfloor \log_A\left(1+(A-1)d^a\right)\rfloor$. The broadcasting time in fractional α-bounded model is at least $a\frac{p^{\lfloor n/a \rfloor}-1}{(p-1)^2}$.*

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

Theorem 7. *Let d be a fixed integer. The broadcasting time of the greedy algorithm in the fractional α-bounded model on the d-dimensional torus of order k for an even integer k and fixed d is $\Theta(k)$.*

[7] in the greedy algorithm an informed vertex sends the message to all its uninformed neighbors in every step

Although the greedy flooding algorithm is not optimal in the fractional α-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 feature: the algorithm may identify some "critical" messages and send each of them in a time step of its own which would guarantee their delivery. This means that perfect reliability can be traded for time complexity which is not what was intended when designing the model.

Hence, a modification of the model has been proposed in [16] to avoid this property. The *α-fractional threshold* model uses another motivation from the randomized models: not only the number of faults is proportional to the number of messages in transit, but there must be enough messages for the statistics to work, as well. To model this, if the number of messages m sent by the algorithm in one step is less than the connectivity of the graph,[8] all of them may be lost. Otherwise, at most a fraction α of them can be lost. It is important to note that the model provides no fault-detection mechanism. 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.

Before proceeding to the analysis of the α-fractional threshold model, let us first focus on its extreme setting when α is infinitely close to 1, i.e. the case where the adversary has maximum power. In this *simple threshold* model, at least $c(G)$ messages[9] have to be sent in one time step to guarantee at least[10] one delivery.

The motivation for the study of this model is twofold: first, since it is the extreme setting, any algorithm developed for the simple threshold model will work also for all other settings of fractional thresholds. Second, with an abundance of models and parameters, it is always useful to pay attention to the extreme cases in the hope of better understanding of the interplay among various parameters. Indeed, the simple threshold model can be viewed as an attempt to find the weakest restrictions to the adversary which still allow broadcasting to be performed.

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 [16]: 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 [16] can be found in Table 1. The paper deals with rings, complete graphs, hypercubes and arbitrary communication graphs with various topology knowledge. Presented results prove

[8] or some other chosen threshold

[9] where $c(G)$ is the connectivity of the graph

[10] Note that the "at least" part is important here. Indeed, if it is known that only one message is delivered, there are cases in which the algorithm may use this additional information about the adversary to its advantage (e.g. by using it to break symmetry).

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 sense of direction	$O(n^2)$[11]
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	$O(2^{c(G)} m^2 n)$

Table 1. Results about broadcasting time in simple threshold model from [16]. The communication graph G has n vertices, m edges and is $c(G)$-edge-connected.

In order to give a flavor of the techniques used, we present here a sketch of the algorithm for broadcasting in a ring. Even in this very limited scenario, a trivial algorithm would not work. 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. Obviously the initiator has to start by sending 2 messages. At least one of then is delivered, but since there is no implicit acknowledgement the initiator does not know which one. Moreover it must avoid sending a message to an already informed neighbor since in this case the adversary would deliver this message, and no progress would be made.

For the ease of presentation let us suppose that the size of the ring, n, is known to the vertices. 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 is delivered in the message that informs the vertex v.

The computation is organized in phases where each phase takes four time steps. Each 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

[11] This result can be improved to $O(n \log n)$.

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 some 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:

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.[12]
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.

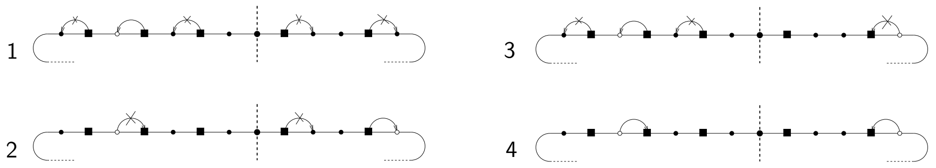

Fig. 1. A sample phase of the ring broadcasting algorithm. The initially active vertices are squares. In the first step, the only surviving message is delivered to the white vertex on the left, which then replies in the next step. At the end of the phase, two white vertices are sending acknowledgement,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

Initially, there are two active (virtual) vertices (the left- and right- part of the initiator). 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 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.

The previous results were done in the simple threshold model, i.e. in the extreme case where α is infinitesimally close to 1. In [43], results summarized in Table 2 have been obtained for complete graphs and hypercubes for arbitrary constant α.

5 Other problems in fractional threshold model

While trying to solve the broadcasting problem, an interesting feature of the fractional threshold model was pointed out in [43]: Usually it is quite easy to

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

Scenario	Broadcasting time
K_n, unoriented	$\Omega(\log n)$, $O(n^3)$
K_n, sense of direction	$\Theta(\log n)$
K_n, $\alpha \lesssim 0.55$	$\Theta(\log n)$
H_d ($n = 2^d$)	$\Omega(\log n)$, $O(n^4 \log^2 n)$

Table 2. Results for the broadcasting in the fractional model with threshold from [43].

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[13] c of vertices to stay uninformed in the end. We call such relaxation the *almost-complete broadcasting* problem. In [43] it was shown that the almost complete broadcasting can be solved in time $O(\log n)$ for complete graphs and $O(\log^2 n)$ for hypercubes.

In [17], the problem of leader election on rings has been investigated, and an $O(n \log n)$-time algorithm was developed in the case when all initiators start at the same time and there is no spontaneous wakeup, and an $O(n^2)$ algorithm in the general case.

6 Conclusion

This research leaves many open questions for further investigation. The line of the study is directed at specifying how various aspects of communication faults influence the performance of networks. One can thus consider a number of communication problems and study them under different models of faults. Also of interest is the relationship between the almost complete and complete broadcasts in various models.

For the fractional threshold model, non-constant values of α have not been considered. Results for more general classes of graphs would be beneficial. For the simple threshold model, one interesting question is whether the broadcasting can be done in polynomial time on all graphs. One can also try to compare the time complexity in the simple threshold model with the message complexity of a fault-free execution.

Another obvious question is to ask if it is possible to perform a complete broadcast in complete graphs also for large values of α in polylogarithmic time.

We finish by noting that there is a lack of any non-trivial lower bounds in the model of fractional faults with threshold.

[13] independent on the network size

References

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. J. Bruck. On optimal broadcasting in faulty hypercubes. *Discrete Applied Mathematics*, 53(1–3):3–13, 1994.
9. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.
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. A. Dessmark and A. Pelc. Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(1–3):343–362, 2004.
14. K. Diks, E. Kranakis, and A. Pelc. Broadcasting in unlabeled tori. *Parallel Processing Letters*, 8(2):177–188, 1998.
15. K. Diks and A. Pelc. Almost safe gossiping in bounded degree networks. *SIAM Journal on Discrete Mathematics*, 5(3):338–344, 1992.
16. 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.
17. S. Dobrev, R. Královič, and D. Pardubská. Leader election in simple threshold model. unpublished manuscript, Dept. of Computer Science, Comenius University, Bratislava, Slovakia, 2008.
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. Broadcasting with dynamic faults. Phd. thesis, Dept. of Computer Science, Comenius University, Bratislava, Slovakia, 2008.
43. 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.
44. E. Kranakis, M. Paquette, and A. Pelc. Communication in networks with random dependent faults. In L. Kucera and A. Kucera, editors, *MFCS*, volume 4708 of *Lecture Notes in Computer Science*, pages 418–429. Springer, 2007.
45. S. Latifi. On the fault-diameter of the star graph. *Information Processing Letters*, 46(3):143–150, 1993.
46. 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.
47. N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
48. 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.
49. G. D. Marco and U. Vaccaro. Broadcasting in hypercubes and star graphs with dynamic faults. *Information Processing Letters*, 66(6):321–326, 1998.
50. V. E. Mendia and D. Sarkar. Optimal broadcasting on the star graph. *IEEE Transactions on Parallel and Distributed Systems*, 3(4):389–396, 1992.
51. A. Pelc. Broadcasting in complete networks with faulty nodes using unreliable calls. *Information Processing Letters*, 40(3):169–174, 1991.
52. A. Pelc. Fault-tolerant broadcasting and gossiping in communication networks. *Networks*, 28(3):143–156, 1996.
53. A. Pelc and D. Peleg. Broadcasting with locally bounded Byzantine faults. *Information Processing Letters*, 93(3):109–115, 2005.
54. 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.
55. D. Peleg. A note on optimal time broadcast in faulty hypercubes. *Journal of Parallel and Distributed Computing*, 26(1):132–135, 1995.
56. P. Ramanathan and K. G. Shin. Reliable broadcast in hypercube multicomputers. *IEEE Transactions on Computers*, 37(12):1654–1657, 1988.
57. 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.
58. 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.
59. A. Schoone, H. Bodlaender, and J. van Leeuwen. Diameter increase caused by edge deletion. *J. Graph Theory*, 11:409–427, 1987.
60. G. Tel. *Introduction to distributed algorithms.* Cambridge University Press, New York, NY, USA, 1994.