FAST BROADCASTING WITH BYZANTINE FAULTS

Michel Paquette [*,1] **Andrzej Pelc** [*,2]

[*] *Département d'informatique et d'ingénierie, Université du Québec en Outaouais, Hull, Québec J8X 3X7, Canada.*
E-mail: `michel.paquette@polymtl.ca, pelc@uqo.ca`

Abstract: We construct and analyze a fast broadcasting algorithm working in the presence of Byzantine component faults. Such faults are particularly difficult to deal with, as faulty components may behave arbitrarily (even maliciously) as transmitters, by either blocking, rerouting, or altering transmitted messages in a way most detrimental to the broadcasting process. We assume that links and nodes of a communication network are subject to Byzantine failures, and that faults are distributed randomly and independently, with link failure probability p and node failure probability q, these parameters satisfying the inequality $(1-p)^2(1-q) > 1/2$. A broadcasting algorithm, working in an n-node network, is called *almost safe* if the probability of its correctness is at least $1 - 1/n$, for sufficiently large n. Thus the robustness of the algorithm grows with the size of the network. Our main result is the design and analysis of an almost safe broadcasting algorithm working in time $O(\log^2 n)$ and using $O(n \log n)$ messages in n-node networks. The novelty of our algorithm is that it can cope with the most difficult type of faults, potentially affecting all components of the network (both its links and nodes), and that it is simultaneously robust and efficient. *Copyright (c) 2004 IFAC*

Keywords: fault tolerance, binary tree architectures, communication networks, distributed computer control systems, noisy channels, optimality, probabilistic models.

1. INTRODUCTION

As interconnection networks grow in size and complexity, they become increasingly vulnerable to component failures. Links and/or nodes of the network may fail, and these failures often result in delaying, blocking, or even distorting transmitted messages. It becomes important to design communication algorithms in such a way that the desired communication task be accomplished efficiently in spite of these faults, usually without knowing their location ahead of time. Such communication algorithms are called *fault-tolerant*.

No communication algorithm can work properly for all fault types and configurations. For example, if all links of a network are faulty, and faults result in permanently blocking all transmitted messages, there can be no hope of accomplishing any communication task. On the other hand, such massive failures are extremely rare in practice and they need not be of primary concern in algorithm design. Much more frequent are faults that damage a limited number of components. Hence an

[1] Supported in part by the M.Sc. CALDI Scholarship of the Research Chair in Distributed Computing of the Université du Québec en Outaouais.
[2] Supported in part by NSERC grant OGP 0008136 and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

important goal is to design communication algorithms that work properly, under some assumptions bounding the number of possible faults.

We are concerned with one of the fundamental communication procedures called *broadcasting*. One node of the network, called the *source*, holds a message which must be transmitted to all other fault-free nodes. (It is assumed that the source is fault free: otherwise, no broadcasting is possible). There are two fundamental qualities which are demanded of a good fault-tolerant broadcasting algorithm. One of them is *robustness*, i.e., the ability of the algorithm to work correctly in spite of the presence of faults of some assumed type, which are distributed in unknown locations but according to some prescribed assumptions. The other one is *efficiency*: this requirement is similar as in the fault-free situation. In this work we adopt two primary measures of efficiency of a communication algorithm: its *time*, i.e., the number of time steps used by the algorithm, and its *cost*, i.e., the number of messages used in the communication process.

1.1 The model and terminology

The communication network is modeled as a simple undirected graph. We assume that the network is complete, i.e., all pairs of nodes are connected by communication links. However, our algorithm works for much sparser networks as well, in fact for some networks of logarithmic degree. Communication is synchronous: every elementary transmission is assumed to take one *time step*. We adopt one of the communication models most widely used in the literature, the so called *one-port* or *whispering* model (Hedetniemi, Hedetniemi and Liestman, 1988; Fraigniaud and Lazard, 1994). In one time step, each node can communicate with at most one other node, i.e., pairs of nodes communicating simultaneously must form a matching.

Since we are concerned with communication in the presence of faults, we must specify the nature and the assumed distribution of faults which our algorithms can tolerate. We work with the most general fault type: *Byzantine faults*. Such faults are particularly difficult to deal with, as faulty components may behave arbitrarily (even maliciously) as transmitters, by either blocking, rerouting, or altering transmitted messages in a way most detrimental to the broadcasting process. The main advantage of assuming such general type of faults is that an algorithm working under this assumption, also works correctly, and with the same or better efficiency, in the presence of any more benign type of failures, such as crash or fail-stop faults. Hence, designing an efficient and robust communication algorithm for Byzantine faults increases its portability. Another issue to be specified is the distribution of faults. We work under the probabilistic distribution scenario, widely considered to accurately model realistic faulty environments (cf. surveys (Lee and Shin, 1994; Pelc, 1996)). Faults of links and nodes of the network are assumed to be distributed randomly and independently. with link failure probability p and node failure probability q. For our algorithm to work reliably, these constant parameters must satisfy the inequality $(1-p)^2(1-q) > 1/2$. It should be noted that this assumption is satisfied in most real cases: for example our algorithm can tolerate link and node failures occuring with probability up to 20%. Most realistic communication networks have much more reliable components, hence our algorithm will behave well in practice. It is assumed that the source of broadcasting is fault free: otherwise, no broadcasting is possible.

Since components of the network are subject to random failures, it is even possible that all components fail and hence, no broadcasting algorithm can work correctly at all times. Hence we seek algorithms that have high probability of correctness. A broadcasting algorithm, working in an n-node network, is called *almost safe*, if the probability of its correctness is at least $1 - 1/n$, for sufficiently large n. Thus the robustness of the algorithm grows with the size of the network, which is a desirable property, as networks encountered in real applications are often very large. For this reason, almost safe fault-tolerant communication algorithms have been widely studied in the literature (cf. the survey (Pelc, 1996)).

1.2 Our results

Since redundancy is a fundamental ingredient in all fault-tolerant communication algorithms, there exists a trade-off between robustness and efficiency of such algorithms. If we want a broadcasting algorithm to tolerate a lot of faults, especially Byzantine faults, messages must be sent along many alternate routes, and verified during the communication process, thus augmenting the running time and the cost of the algorithm. Our main result is the design and analysis of a broadcasting algorithm which is fast, uses relatively few messages, and has high probability of correctness. More specifically, we design an almost safe broadcasting algorithm working in time $O(\log^2 n)$ and using $O(n \log n)$ messages in n-node networks, for network components subject to Byzantine failures, under the assumption that link failure probability p and node failure probability q satisfy the inequality $(1-p)^2(1-q) > 1/2$.

Our algorithm is *oblivious*, in the sense that all transmissions are scheduled in advance: it is predetermined which pairs of nodes communicate in a given time step. The only thing that is decided during the algorithm execution, is the content of the messages, which depends on outcomes of a voting scheme used to mask erroneous transmissions. Such algorithms have the advantage of being easy to implement and of requiring small local memory in network nodes. It can be proved that every almost safe oblivious broadcasting algorithm must use $\Omega(n \log n)$ messages, hence our algorithm has cost of optimal order of magnitude. On the other hand, it is well known that any broadcasting algorithm, even working in a fault-free network, must have running time at least $\log n$. Hence the execution time of our algorithm exceeds the optimal one by at most a logarithmic factor.

1.3 Related work

Efficient broadcasting algorithms in various communication networks have been widely studied in the literature (see, e.g., surveys (Hedetniemi, Hedetniemi and Liestman, 1988; Fraigniaud and Lazard, 1994)). In particular, a lot of effort has been devoted to the study of broadcasting under the assumption that components of the network may be faulty (see the survey (Pelc, 1996)). Two commonly used fault models are the *bounded* model and the *probabilistic* model. In the bounded model, an upper bound k is imposed on the number of faulty components, and their worst-case location is assumed. Under this scenario, k-*tolerant* broadcasting is usually sought (see, e.g., (Gargano, 1992; Peleg and Schäffer, 1989)): the source message must reach all fault-free nodes provided that no more than k components (links, or nodes, or both, depending on the particular scenario) are faulty. In the probabilistic model, faults are assumed to occur randomly and independently of each other, with specified probabilities. Under this scenario, almost safe broadcasting is often sought, similarly as in the present paper (cf. (Berman, Diks and Pelc, 1997; Bienstock, 1988; Chlebus, Diks and Pelc, 1994; Diks and Pelc, 1992)).

Byzantine faults have been widely studied in the context of network communication, of the consensus problem in distributed computing, and of multiprocessor fault diagnosis (cf., e.g., (Lynch, 1996) and the survey (Barborak, Malek and Dahbura, 1993)). In (Bao, Igarashi and Katano, 1995), broadcasting in hypercube networks was studied under the assumption that either nodes or links (but not both) are subject to randomly distributed Byzantine faults. In (Berman, Diks and Pelc, 1997), a $O(\log n)$-time broadcasting algorithm was designed under the assumption that links are subject to randomly distributed Byzantine faults but all nodes are fault free. On the other hand, in (Chlebus, Diks and Pelc, 1994) a $O(\log n)$-time broadcasting algorithm was proposed for randomly distributed link and node failures but only of *crash* type: such faults can only block messages but cannot distort them, which significantly facilitates the design of fault-tolerant broadcasting algorithms. To the best of our knowledge, our broadcasting algorithm is the first to cope with randomly distributed Byzantine faults of links and nodes at the same time.

2. THE BROADCASTING ALGORITHM

2.1 The underlying network

Although we work in the complete network, our broadcasting algorithm will in fact use very few links. This is a desirable feature, as sparse networks are easier to implement. We will construct an n-node network of logarithmic degree in which almost safe broadcasting is possible (cf. (Chlebus, Diks and Pelc, 1994; Berman, Diks and Pelc, 1997), where a similar network was used previously).

We use $\log n$ to denote the logarithm with base 2 and $\ln n$ to denote the logarithm with base e. For a positive constant c, we define an n-node network $G(n, c)$. Let $m = \lceil c \log n \rceil$ and $s = \lfloor n/m \rfloor$. For clarity of presentation, assume that n is a power of 2, m is odd and divides n, and $s = 2^{L+1} - 1$, for some integer $L \geq 0$. By definition, $L < \log n$. It is easy to modify the construction and the proof in the general case. Partition the set of all nodes into subsets S_1,..., S_s, of size m, called *supernodes*. In every supernode S_i, enumerate nodes from 0 to $m-1$. For any $i = 1, ..., s$ and any $j = 0, ..., m-1$, assign label (i, j) to the jth node in the ith supernode. We assume that node $(1, 0)$ is the source of broadcasting. Arithmetic operations on the second integers forming labels are performed modulo m.

Arrange all supernodes into a complete binary tree T with $L+1$ levels $0, 1, ..., L$. Level 0 contains the root and level L contains leaves of tree T. The supernode S_1, called ROOT, is the root of T. For every $1 \leq i \leq \lfloor s/2 \rfloor$, S_{2i} is the left child of S_i and S_{2i+1} is the right child of S_i in the tree T. For every $1 < i \leq s$, supernode $S_{\lfloor i/2 \rfloor}$ is the parent of S_i. If a supernode is a parent or a child of another supernode, we say that these supernodes are adjacent in T.

The set of edges of the network $G(n, c)$ is defined as follows. If supernodes S_i and S_j are adjacent in T then there is an edge in $G(n, c)$ between any node in S_i and any node in S_j. Moreover, every pair of nodes in the supernode S_1 forming the root of T are joined by an edge. These are the only edges in $G(n,c)$. The graph $G(n,c)$ is called a *thick tree*. It should be noted that a thick tree is not a tree but has a tree-like structure which will be exploited in the design of our broadcasting algorithm. Clearly, a thick tree is a graph of maximum degree $O(\log n)$.

2.2 The overview of the algorithm

The idea of our algorithm is the following. First the source message is propagated to all nodes of ROOT, using all other nodes as intermediaries. Every node of ROOT gets $m-1$ versions of the source message. Then it computes the message considered to be correct, by majority voting. Next, the source message is propagated down the thick tree, along its branches. Every supernode S_i, whose nodes already got the message, transmits first to its left child and then to its right child. This is done as follows. Every node of S_i sends its version of the source message to every node of the given child. After obtaining m versions of the source message from nodes of S_i, every node of the child of S_i computes the message considered to be correct, by majority voting. This is done until all nodes in all supernodes compute the version of the source message considered to be correct.

2.3 The description of the algorithm

We now proceed with a formal description of the broadcasting algorithm. For every node u, X_u denotes the content of register X in this node. In the beginning, the source message is held in register M of the source, i.e., it is $M_{(1,0)}$. For every node u, the message that it computes during the scheme execution, and considers to be the source message, is stored in its register M. Every node u outputs the final value of M_u as its version of the source message.

The elementary procedure SEND(u,X,v,Y) will be used for adjacent nodes u,v, for the following action: u sends X_u to v, and v assigns the received value to its register Y. If u, v, and the link joining u with v are fault free, the effect of SEND(u,X,v,Y) is the same as the assignment $Y_v := X_u$. Otherwise, an arbitrary value is assigned to Y_v.

Another elementary subroutine is VOTE(v,V,A), where V is a vector of values. This subroutine is performed locally by node v, and it serves to compute the correct version of the source message, with high probability. It outputs the majority value of all terms of a vector V of records held in v, and stores it in record A, if such a majority exists. Otherwise, it stores a default value.

We now describe the main procedures used in our broadcasting algorithm.

Procedure ROOT-Propagation spreads the source message among nodes of ROOT. After exchanging their versions of the source message, all nodes in ROOT vote on all received values, to compute the correct version of the source message, with high probability. Let $R_1,...,R_{m-2}$ be the partition of all edges in ROOT$\setminus\{(1,0)\}$ into pairwise disjoint perfect matchings, and let $R_i(v)$ denote the node matched with v by matching R_i.

```
**procedure** ROOT-Propagation
**for** $j := 1$ **to** $m-1$ **do**
    SEND$((1,0), M, (1,j), W)$
**for** $i := 1$ **to** $m-2$ **do**
    **for all** $j = 1, ..., m-1$ **in parallel do**
        SEND$((1,j), W, R_i((1,j)), B[j])$
        $B[j]_{(1,j)} := W_{(1,j)}$
        //$R_i((1,j))$ stores the message coming
        //from $(1,j)$ in record $B[j]$. Also, every
        //node $(1,j)$ assigns the value of its record
        //$W$ to the record $B[j]$ in its vector $B$.
**for all** $j = 1, ..., m-1$ **in parallel do**
    VOTE$((1,j), B, M)$
    //Every node in ROOT computes the majority of
    //values from vector $B[1,...,m-1]$
```

Procedure GROUP-Transmission transmits the message from all nodes of one supernode to all nodes of another. Then all nodes in the receiving supernode vote on all received values, to compute the correct version of the source message, with high probability. Index addition in the procedure formulation is done modulo m.

```
**procedure** GROUP-Transmission $(i,k)$
**for** $r := 0$ **to** $m-1$ **do**
    **for all** $j = 0, 1, ..., m-1$ **in parallel do**
        SEND$((i,j), M, (k, j+r), B[j])$
**for all** $j = 0, 1, ..., m-1$ **in parallel do**
    VOTE$((k,j), B, M)$
```

Procedure TREE-Propagation (l) reliably transmits the source message from level l to level $l+1$ of the thick tree.

```
**procedure** TREE-Propagation $(l)$
**for all** supernodes $S_i$ on level $l$ **in parallel do**
    GROUP-Transmission$(i, 2i)$; //to left child
    GROUP-Transmission$(i, 2i+1)$; //to right child
```

Now Algorithm FBBF (Fast Broadcasting with Byzantine Faults) can be succinctly formulated as follows.

```
Algorithm FBBF
ROOT-Propagation
for $l := 0$ to $L-1$ do
    TREE-Propagation($l$).
```

3. THE ANALYSIS OF ALGORITHM FBBF

Let $q' = (1-p)^2(1-q)$ and $c = \frac{80\ln(2)q'}{(2q'-1)^2}$. By assumption, $q' > 1/2$. We consider Algorithm FBBF for the thick tree $G(n,c)$, for this value of c.

Proposition 3.1. Algorithm FBBF works in time $O(\log^2 n)$ and uses $O(n \log n)$ messages.

Proof. Since the size of every supernode is $O(\log n)$, the procedures ROOT-Propagation and GROUP-Transmission each take time $O(\log n)$. Since the number of levels in the thick tree is $O(\log n)$ as well, it follows that the execution time of Algorithm FBBF is $O(\log^2 n)$. In order to estimate the number of messages, observe that each pair of nodes, adjacent in the thick tree, exchange at most two messages. Since the maximum degree of the thick tree is $O(\log n)$, the number of edges in it is $O(n \log n)$, and consequently Algorithm FBBF uses $O(n \log n)$ messages. □

It remains to prove the main result of this section, concerning the correctness of Algorithm FBBF. In the proof we will use the following probabilistic lemma known as Chernoff's bound (cf. (Hagerup and Rüb, 1989/90)). It will be needed to estimate the probability that the majority of a set of pairwise disjoint paths joining two nodes, are fault free. Since paths are pairwise disjoint, the events that they do not contain faulty components are independent, and Chernoff's bound can be used.

Lemma 3.1. Let X be the random variable denoting the number of successes in a Bernoulli series of length m with success probability q. Let $0 < \epsilon < 1$. Then $Prob(X \leq (1-\epsilon)mq) \leq e^{-\epsilon^2 mq/2}$.

Theorem 3.1. Algorithm FBBF is almost safe.

Proof. We define the following events:

- C is the event that, at the end of Algorithm FBBF, all fault-free nodes get the correct source message. We denote by p_C the probability of event C.
- Let x be a branch of the thick tree. B_x is the event that, at the end of Algorithm FBBF, all fault-free nodes in all supernodes in the branch x, get the correct source message. We denote by p_{B_x} the probability of event B_x.
- C_{ROOT} is the event that, at the end of procedure ROOT-Propagation, all fault-free nodes in ROOT get the correct source message. We denote by $p_{C_{ROOT}}$ the probability of event C_{ROOT}.
- $C_{SOURCE,i}$, for $i = 1, ..., m-1$, is the event that, at the end of procedure ROOT-Propagation, a fault-free node $(1,i)$ gets the correct source message. We denote by $p_{C_{SOURCE,i}}$ the probability of event $C_{SOURCE,i}$
- Let supernode S_i be the parent of supernode S_k in the thick tree. CS_k is the event that, at the end of procedure GROUP-Transmission(i,k), all fault-free nodes in S_k get the correct source message. We denote by p_{CS_k} the conditional probability $Prob(CS_k|CS_i)$.

For any event E, we will use the notation $\overline{E}$ to denote the complement of E, and $\overline{p_E}$ to denote $1 - p_E$.

Consider the event $C_{SOURCE,i}$. Node $(1,i)$ is joined with the source by $m-2$ pairwise disjoint paths of length 2, and one path of length 1 (the joining edge). The probability that such a fixed path of length 2 does not contain faulty components, is $q' = (1-p)^2(1-q)$. The probability for the path of length 1 is even larger: $1-p$. For pairwise disjoint paths, these events are independent, hence Lemma 3.1 can be used to estimate the probability that the majority of these paths are free of faults. If this holds, event $C_{SOURCE,i}$ holds as well. Hence we get

$$\overline{p_{C_{SOURCE,i}}} \leq e^{\frac{-(2q'-1)^2(m-1)q'}{8q'^2}} \leq e^{\frac{-(2q'-1)^2 mq'/2}{8q'^2}}. \quad (1)$$

In view of $m = \lceil c \log n \rceil$ and $c = \frac{80\ln(2)q'}{(2q'-1)^2}$ we have

$$\overline{p_{C_{SOURCE,i}}} \leq e^{\frac{5\ln(2)\log(n)(-(2q'-1)^2}{(2q'-1)^2/8q'}\frac{q'}{8q'^2}}. \quad (2)$$

$$= e^{-5\ln(n)} = 1/n^5. \quad (3)$$

We know that $\overline{p_{C_{ROOT}}} \leq \sum_{i=1}^{m} \overline{p_{C_{SOURCE,i}}}$, since C_{ROOT} is the intersection of all events $C_{SOURCE,i}$, for fault-free nodes $(1,i)$. Hence $\overline{p_{C_{ROOT}}} \leq m \cdot 1/n^5 \leq 1/n^4$, for sufficiently large n.

Let supernode S_i be the parent of supernode S_k in the thick tree. Consider the event E_j that a single fixed fault-free node (k,j) in supernode S_k gets the correct source message at the end of procedure GROUP-Transmission(i,k). Suppose that all fault-free nodes in S_i got the correct source message prior to the execution of this procedure. Node (k,j) gets the source message through m disjoint paths, each consisting of one node from S_i and one joining link. The probability that both components in such a single path are fault free, is $(1-p)(1-q) > q'$. Hence the conditional probability $Prob(E_j|CS_i)$ is larger than $p_{C_{SOURCE,i}}$. Consequently, $Prob(\overline{E_j}|CS_i) \leq 1/n^5$, which implies $\overline{p_{CS_k}} \leq m \cdot 1/n^5 \leq 1/n^4$, for sufficiently large n.

Fix a branch x of the thick tree. For sufficiently large n, we have

$$p_{B_x} = p_{C_{ROOT}} \cdot \prod_{i=1}^{L} p_{CS_i}. \tag{4}$$

$$p_{B_x} \geq (1 - \frac{1}{n^4})^{L+1} \geq (1 - \frac{1}{n^4})^{n}. \tag{5}$$

We have $(1-1/n^4)^{n^3} \to 1$ and $(1-1/n^2)^{n^2} \to 1/e$. Hence, for sufficiently large n,

$$(1 - 1/n^4)^{n^3} > (1 - 1/n^2)^{n^2}, \tag{6}$$

which implies

$$1 - 1/n^4 > (1 - 1/n^2)^{1/n}. \tag{7}$$

Thus we get $p_{B_x} \geq (1 - \frac{1}{n^4})^n \geq 1 - 1/n^2$. By definition of events C and B_x we have

$$\overline{C} \subset \bigcup \{\overline{B_x} : x \text{ is a branch in the thick tree}\}. \tag{8}$$

Since there are $2^L \leq n$ branches in the thick tree, we finally get, for sufficiently large n,

$$\overline{p_C} \leq n \cdot \frac{1}{n^2} = 1/n, \tag{9}$$

hence Algorithm FBBF is almost safe. □

4. CONCLUSION

We proposed an almost safe broadcasting algorithm working in time $O(\log^2 n)$ and using $O(n \log n)$ messages in n-node networks, subject to Byzantine faults of components. The fact that the algorithm is almost safe guarantees that its reliability grows with the size of the network. The novelty of our algorithm is that it copes with the most difficult type of faults potentially affecting all components of the network: both its links and nodes. Our algorithm has the optimal cost complexity, and its running time differs from the lower bound only by a logarithmic factor. The question of whether there exists an almost safe broadcasting algorithm working in time $O(\log n)$ under our Byzantine fault scenario remains a challenging open problem suggested by our result.

REFERENCES

Bao, F., Y. Igarashi and K. Katano (1995). Broadcasting in hypercubes with randomly distributed byzantine faults. *Proc. WDAG'95*, **972**, pp. 215–229.

Barborak, M., M. Malek and A. Dahbura (1993). The consensus problem in fault-tolerant computing. *ACM Computing Surveys*, **25**, pp. 171–220.

Berman, P., K. Diks and A. Pelc (1997). Reliable broadcasting in logarithmic time with byzantine link failures. *Journal of Algorithms*, **22**, pp. 199–211.

Bienstock, D. (1988). Broadcasting with random faults. *Disc. Appl. Math.*, **20**, pp. 1–7.

Chlebus, B.S., K. Diks and A. Pelc (1994). Sparse networks supporting efficient reliable broadcasting. *Nordic Journal of Computing*, **1**, pp. 332–345.

Diks, K. and A. Pelc (1992). Almost safe gossiping in bounded degree networks. *SIAM J. Disc. Math.*, **5**, pp. 338–344.

Fraigniaud, P. and E. Lazard (1994). Methods, problems of communication in usual networks. *Disc. Appl. Math.*, **53**, pp. 79–133.

Gargano, L. (1992). Tighter bounds on fault-tolerant broadcasting, gossiping. *Networks*, **22**, pp. 469–486.

Hagerup, T. and C. Rüb (1989/90). A guided tour of chernoff bounds. *Inf. Proc. Letters*, **33**, pp. 305–308.

Hedetniemi, S.M., S.T. Hedetniemi and A.L. Liestman (1988). A survey of gossiping, broadcasting in communication networks. *Networks*, **18**, pp. 319–349.

Lee, S. and K.G. Shin (1994). Probabilistic diagnosis of multiprocessor systems. *ACM Computing Surveys*, **26**, pp. 121–139.

Lynch, N.A. (1996). *Distributed Algorithms.* Morgan Kaufmann Publ., Inc.. San Francisco.

Pelc, A. (1996). Fault-tolerant broadcasting, gossiping in communication networks. *Networks*, **28**, pp. 143–156.

Peleg, D. and A.A. Schäffer (1989). Time bounds on fault-tolerant broadcasting. *Networks*, **19**, pp. 803-822.