Proc. 15th IASTED Int. Conf. on Parallel and Distributed Computing and Systems (PDCS), Marina del Rey, CA, USA, Nov. 3–5, 2003, pp. 146–151 December 4, 2003

An optimal Byzantine Agreement algorithm with arbitrary node and link failures

Martin Biely
Embedded Computing Systems Group (E182/2)
Technische Universität Wien
Treitlstrasse 1
A-1040 Vienna
mb@ecs.tuwien.ac.at

Abstract

In a series of recent papers[1], we showed how to avoid the impossibility result of deterministic consensus in presence of link failures [Gray,1978] for well-known Oral and Written Messages Byzantine agreement algorithms. This is done by employing the perception-based fault model [Schmid,2001]. We will, however, use a simplified version of that model in this paper.

The Byzantine Agreement protocol presented in this paper is a polynomial time protocol with optimal resilience in respect to the simplified perception-based fault model. The most important feature of the Byzantine Agreement algorithm is, that it uses some of the processes to tolerate process failures as well as to overcome link failures, as we need only $n > 2f_a + \max(f_a, f_\ell^{r,a} + f_\ell^r + f_\ell^{s,a} + f_\ell^s)$, where f_ℓ^r, $f_\ell^{r,a}$, f_ℓ^s and $f_\ell^{s,a}$ are measures of the link failures allowed.

Keywords: Distributed Algorithms, Byzantine Agreement, link failures

1 Introduction

Although process failures models have always been applied most successfully in the analysis of fault-tolerant distributed algorithms, they do have limitations. Given the steadily increasing dominance of communication over computation in modern distributed systems, it becomes increasingly difficult to apply fault models that capture only process failures.

Indeed, due to the high reliability of modern processors, communication-related failures like receiver overruns (run out of buffers), unrecognized packets (synchronization errors), and CRC errors (data reception problems) in high-speed wireline and, in particular, all sorts of wireless networks are increasingly dominating process failures. Such *link failures* occur on the communication channel or in the network interface and can cause any data packet to be lost or even faulty. The resulting error, however, cannot reasonably be attributed to the innocent sender process. Declaring the receiver process as faulty would be overly conservative too, since a packet error does not usually imply a process failure (after all, its processor executes the particular algorithm correctly). Consequently, link failures should be a category of their own in a more realistic fault model.

Schmid et al. [17, 18] provided a comprehensive hybrid fault model that captures both process and link failures, and showed that it allows consensus with *Oral* and *Written Messages* (algorithms OMH, OMHA & ZA [7,11]) in synchronous distributed systems, despite the impossibility result of Gray [8, 12]. The key for escaping from the impossibility result is to limit the inconsistency caused by link failures in the broadcasts of a single sender and the receptions at a single receiver. Limiting the power of link failures that way, their effects can be masked transparently, provided that sufficiently many nodes are available. Still, the algorithms considered in [17] suffer from an exponential number of messages. In this paper, we present a message-efficient algorithm, i.e. polynomial in number and size of messages, allowing consensus even in systems with low communication bandwidth and high link failure rates, and increasing resilience in comparison to previous efforts to create polynomial time protocols [4].

The remainder of this paper is organized as follows: In Section 2, we present a simplified version of the perception-based hybrid fault model introduced in [17]. Section 3 is devoted to the Byzantine Agreement protocol itself. A discussion of the results and some directions of further research in Section 4 eventually conclude the paper.

2 The System Model

For the reasons mentioned in Section 1 it is a logical step to incorporate link failures into one's system model. These reasons were already valid 30 years ago, however the well-known result of Gray [8], where he claims that no deterministic algorithm that can solve consensus in presence of lossy links, cf. [12], has discouraged research in this direction.

[1] This research is part of the W2F-project, which targets at developing a wireline/wireless fieldbus based upon spread-spectrum (CDMA) communications, see *http://www.auto.tuwien.ac.at/Projects/W2F/* for details. W2F is supported by the Austrian START programme Y41-MAT.

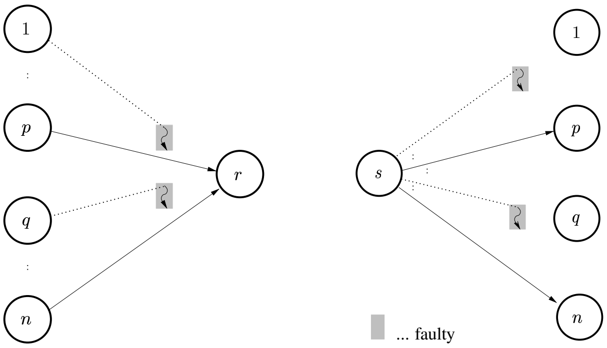

Figure 1. *The two sides of link failures: a receive failure that involves the messages from two senders* (left) *and a broadcast failure that affects the messages of two recipients* (right).

Only synchronous systems are considered in this paper, since ommisive link failures are relatively easy to handle in asynchronous systems, if the links satisfy the "fair loss" property. This would allow to simulate a perfect (but slow) link by suitable retransmission schemes (see, e.g. [1, 9]). This technique cannot be used in synchronous systems without increasing the duration of a round, according to the number of message losses that should be tolerated, resulting in impractically long execution times.

Schmid [16] showed that node and link failures are easily accommodated in a *perception-based fault model*, where the usual global, i.e., system-wide, number of failures is replaced by the number of failures that are observable in the nodes' local *perceptions* of the system. In [17, 18], this model was generalized from the single-round clock synchronization framework to general round-based synchronous algorithms.

As shown in [17, 18] Gray's result can be circumvented if the power of link failures is moderately restricted, in a way that allows to mask away their effects, if sufficiently many correct nodes are present. Two different views, resulting in two constraints, on link failures are required for this purpose:

Receive link failures: We restrict the number f_ℓ^r of senders that could appear faulty to any single receiver, see Figure 1 *(left)*.

Broadcast link failures: We restrict the number f_ℓ^s of receivers that could obtain a faulty message in the broadcast of a single sender, see Figure 1 *(right)*.

The System Model of Definition 1, below, formalizes this intuition. It is a simplified version of the model employed in [17, 18], as we consider only arbitrary process failures here, whereas the original model also deals with symmetric value, asymmetric omission and manifest failures.

Definition 1 (System Model) *We consider a synchronous distributed system consisting of n nodes executing one or more processes, which are interconnected by a fully connected point-to-point network made up of pairs of unidirectional channels.*

(P1) In any entire execution, there may be at most f_a arbitrary faulty nodes.[2]

(A1^s) If a single [faulty or non-faulty] process p sends a value V_p to some set of correct receiver processes $q_i \in \mathcal{R}$, at most f_ℓ^s of the delivered values $V_{q_i}^p$ may differ from the admissible receive values in rvals(V_p, p). Let $f_\ell^{s,a} \leq f_\ell^s$ be the maximum number of non-omissive, i.e., non-empty and hence value faulty, $V_{q_i}^p$ among those.

(A1^r) If all processes $p_i \in \mathcal{S}$ of a set of [faulty or non-faulty] processes send a message containing V_{p_i} to some correct receiver process q, at most f_ℓ^r of the delivered values $V_q^{p_i}$ may differ from the admissible receive values in rvals(V_{p_i}, p_i). Let $f_\ell^{r,a} \leq f_\ell^r$ be the maximum number of non-omissive, i.e., non-empty and hence value faulty, $V_q^{p_i}$ among those.

(A2) The receiver of a message knows who sent it.

(A3) The absence of a message from sender p can be detected at any receiver q at the end of a round, which leads to $V_q^p = \emptyset$.

Remarks:

1. A single round consists of (1) all the processes' local computations based upon the values received in the previous round, (2) the broadcasts (= successive sends) of the resulting messages according to (A1^s), and (3) the reception of those messages according to (A1^r).

2. Properties (A1^s) and (A1^r) are assumed to be independent of each other, however, both must be fulfilled simultaneously in any feasible execution. Any node has an individual "budget" f_ℓ^r resp. f_ℓ^s of link failures that may hit arbitrary inbound resp. outbound links. Without restricting link failure patterns, however, this can only be guaranteed if $f_\ell^s = f_\ell^r$ and $f_\ell^{s,a} = f_\ell^{r,a}$.

2.1 Other approaches to link failures

The perception-based fault model is not the only work that tries to circumvent the impossibility result of Gray [8]. A (rather small) number of approaches to account for link failures or failures of the receiver's network interface in

[2] The full model [3,17,18] features the following classes of hybrid node failures: arbitrary faults (f_a), symmetric value failures (f_s), asymmetric omission failures (f_o) and manifest failures (f_m). The results presented in this paper were derived under the full model and later simplified for brevity. The original analysis [3] resulted in $n > f_m + 2f_s + f_o + 2f_a + \max(f_a + f_o, f_\ell^r + f_\ell^s + f_\ell^{r,a} + f_\ell^{s,a})$

processor-centric models have been proposed. These are, however, insufficient as the following discussion will show.

If, for instance, link failures are simply mapped to sender process failures, as in [7], we can find many failure patterns for which all $f = n$ processors must be considered faulty. Consider, for instance, a system of four processes $n = 4$ where each process p_x fails to send its value to $p_{((x+1) \bmod 4)}$ then $f_\ell = 1$ with f_ℓ denoting the number of incoming links that may be faulty per round for each process.

Similar arguments can be found for the more detailed send/receive-omission fault model of [13], where receive omissions are mapped to a failure of the receiving processor. Although it has been observed that only the number of processors that commit a send omission (but not the number of processors committing a receive omission) needs to be counted in f, agreement is only shown to hold for a processor that did not commit either type of failure, which implies that in the example above, no processor would remain that could reach agreement.

In both models, the situation deteriorates due to the fact that, over time, any receiver's f_ℓ receive omissions could hit different links. Since processor failures are usually considered persistent, this will result in the "exhaustion" of the number of failures f that are tolerated by the system. This exhaustion does not occur with our model, since different links can be faulty in every round, resulting in the fact that all links may appear faulty within an execution.

Sayeed, M. Abu-Amara and H. Abu-Amara considered an asynchronous system with arbitrary link failures only [15]. Their work is not really comparable to ours, since it deals with link failures only and cannot tolerate a single processor failure, which is, as Fischer, Lynch and Peterson [6] showed, not possible in asynchronous systems using a deterministic algorithm and without using failure detectors.

Their algorithm can, however, tolerate up to $t = \frac{n-2}{2}$ Byzantine faulty channels, where t is the total number of link failures throughout an execution. Considering only Byzantine link failures in the perception-based fault model, it is possible to tolerate $\mathcal{O}(nf_\ell)$ such failures in each round. Additionally in our model these links failures need not hit the same links in every round.

The only other work – we know of – that considers link failures in a dynamic way is [14]. Pinter and Shinahr assume that omissions may occur on a maximum of l different links system-wide per round. Their algorithm is identical to the "* algorithm", which was developed by Dolev, Fischer, Fowler, Lynch and Strong [5], except that it needs $a+l+2$ additional rounds, where a is the number of arbitrary processor failures the algorithm can tolerate. To achieve this the algorithm requires $n > 3(a+l)$ processors.

The main difference to our perception-based fault model is that Pinter and Shinahr bounded the total number of faulty links per round (system-wide) with $l \leq \frac{n}{3}$, whereas the total number of omission faulty links tolerated by algorithms that work under the perception-based fault model is $\mathcal{O}(nf_\ell) \gg n$. The protocol presented in Section 3.2 allows for f_ℓ link failures per round per processor and f_a arbitrary failures, if $n > 2f_a + \max(f_a, f_\ell^r + f_\ell^s)$ ($f_\ell^{r,a} = f_\ell^{s,a} = 0$). Since the number of total link failures in a round is nf_ℓ in the perception-based fault model, but only l in the model of Pinter and Shinahr, it is obvious that this resilience against link failures is much better.

More details about the full system model, including comparison with other hybrid fault models can be found in [2, 17]. The complete analysis of the algorithms presented can be found in [3].

3 A polynomial-time Byzantine Agreement

Introduced by Lamport, Shostak and Pease [10] in 1980, the Byzantine Agreement problem is widely known as a fundamental problem in fault-tolerant distributed computing. A Byzantine agreement algorithm is used to consistently disseminate a single *transmitter's* value m to a set of *receivers*. Within a fixed number of rounds, every non-faulty receiver p must *deliver* a value v_p ascribed to the transmitter t that satisfies the following properties:

(B1) (*Agreement*): If two processors p and q, are both non-faulty, then both deliver the same $v_p = v_q$.

(B2) (*Validity*): If processor p is non-faulty, the value v_p delivered by p is

- v, if the transmitter is non-faulty,
- any value, if the transmitter is arbitrary faulty.

The algorithm is based upon the same technique as the polynomial algorithm of [19, 20] and was already used under the perception-based fault model in [2, 4], which is based on the idea of simulating authentication and using an authenticated Byzantine Agreement protocol. The reasoning behind this is that authenticated algorithms are simpler than non-authenticated ones, especially the proof is simplified by the fact that it can be split into two parts.

An authenticated algorithm usually assumes that the following three properties are fulfilled by the underlying network. In our case this network is simulated by the broadcasting scheme of Section 3.2:

(C) (*Correctness*): If a non-faulty processor p broadcasts a certain message m in round k (by executing $\mathbf{broadcast}(p, m, k)$) then every correct processor $\mathbf{accepts}(p, m, k)$ in the same round.

(U) (*Unforgeability*): If a correct processor p never executes $\mathbf{broadcast}(p, m, k)$, then no correct processor ever $\mathbf{accepts}(p, m, k)$.

(R) (*Relay*): If a correct processor $\mathbf{accepts}(p, m, k)$ in round $r \geq k$, then every correct processor also $\mathbf{accepts}(p, m, k)$ in round $r + 1$ or earlier.

```
/* Round 0: transmitter t only*/
    if q = t then
        broadcast(t, v, 0)
    fi
/* Round 1...: all nodes */
    for r:=1 to f_a+1 do
        if accepted(p_x, v, r_x) from r nodes including t
          and v ∉ R then
            R = R ∪ {v}
            broadcast(v, q, r)
        fi
    done
    if |R| = 1 then
        decide(v : v ∈ R)
    else
        decide(∅)
    fi
```

Figure 2. *A simple authenticated Byzantine Agreement algorithm, code for node q.*

3.1 A simple BA algorithm with authentication

The Authenticated Algorithm of this section is somewhat similar to the authenticated algorithm of [19, 20], but it allows for any number of possible input values, not just "0" and "1" .

A node remembers all the values it received from the transmitter as wells as all the values the other nodes claim to have received from the transmitter. Depending on the number of values it receives, the algorithm either decides on the single received value, or on the default value $\emptyset$.

Theorem 1 (Authenticated Algorithm) *The algorithm of Figure 2 – in conjunction with a broadcasting scheme that fulfills Correctness (C), Unforgeability (U) and Relay (R) – achieves Byzantine agreement in at most $f_a + 1$ rounds under the system model of Definition 1, if $n > f_a$.*[3]

Proof: To prove this theorem we have to show Agreement (B1) and Validity (B2) based on the three properties (C), (U) and (R) of our broadcast protocol.

Starting with Validity (B2); Let the transmitter t be correct: Everyone adds v to $\mathcal{R}$ in round 1 due to (C), and no process will ever accept another message from t due to (U). Remember Validity is void in the case of a Byzantine faulty transmitter.

Let us now turn to Agreement (B1): Agreement depends on the values in $\mathcal{R}$, thus it suffices to show that the same values are added to $\mathcal{R}$ at all correct processors: Let p be the first correct processor to add some value v to $\mathcal{R}$ and let k be the round when that happens.

If $k \leq f_a$: Due to (C) p's broadcast of value v will be accepted in round $k + 1$ at all other nodes, also due to (R) all nodes will have received the k messages, that caused p to add v to $\mathcal{R}$. Thus every correct node will have $k + 1$ messages including one from t in round $r + 1$, and will thus add v.

If $k = f_a + 1$: p must have accepted $f_a + 1$ messages containing v, including one from the transmitter. Due to (U) there must be $k = f_a + 1$ processors, which have sent these messages. But that implies that one of these processors, must have been non-faulty, and thus must have added v to $\mathcal{R}$ in a previous round. This contradicts the assumption that p was the first correct node to do so, thus k must be smaller than $f_a + 1$.

The claimed resilience is trivially correct, since the algorithm requires at least one non-faulty node, and (C), (U) and (R) rule out the existence of link failures. □

3.2 A Broadcast Primitive that simulates Authentication

Each round of the BA algorithm (Section 3.1) consists of 5 phases, since we split every call to $\mathbf{broadcast}()$ into 5 phases. A *phase* has the same properties as a round, but at a lower level of our protocol stack.

The primitive is based on ideas found in [19,20], but is extended to be able to mask out link failures transparently to the upper level Byzantine Agreement algorithm. This is done by tracking witnesses for the message the primitive is used to send.

The **acceptance_task** function could be split up into multiple tasks according to the "starting in phase x" label in the comments, (Lines 8, 13, 18 and 13) and then scheduled to be executed in every phase after phase x until the corresponding message was accepted. A process has to instantiate one acceptance_task per received $init$-message.

The $\mathbf{received}(message)$ $\mathbf{from} \geq x$ predicate has the following semantics: If the process has received the $message$ from more than x distinct processes in one phase, say k, then it returns $true$ this time and in all subsequent phases $l > k$. Alternatively if only messages from one specific phase shall be considered this is specially denoted as in Line 10. If only messages from one specific node shall be considered that node is referenced in the **from**-clause of the statement (as in Line 19).

It is important that the received messages are processed on a *per phase* basis and not accumulated over phases, since – due to the dynamic nature of the model – arbitrary links can produce $r f_\ell^{r,a}$ messages within r phases.

Theorem 2 (Correctness, Unforgeability and Relay) *Under the system model of Definition 1 with f_a, f_ℓ^r, $f_\ell^{r,a}$, f_ℓ^s, $f_\ell^{s,a} \geq 0$ the Broadcast Primitive of Figure-3 guarantees the three properties Correctness (C), Unforgeability (U) and Relay (R), if*

$$n > 2f_a + \max(f_a, f_\ell^r + f_\ell^s + f_\ell^{r,a} + f_\ell^{s,a}).$$[4]

[3] Under the full hybrid model we need $f_a + f_s + f_o + f_m + 1$ rounds and at least $f_a + f_s + f_o + f_m + 1$ processes.

[4] Under the full hybrid model we need at least $1 + f_m + 2f_s + f_o + 2f_a + \max(f_a + f_o, f_\ell^r + f_\ell^s + f_\ell^{r,a} + f_\ell^{s,a})$ processes.

```
1  /* phase 5r: sender of broadcast only (q=t) */
   broadcast(t, m, r) {
3      send(init, t, m, r) to all nodes
   }
5
   acceptance_task() {
7      𝕎_{t,m,r} = {}
   /* starting in phase 5r + 1: any node q */
9      if received(init, t, m, r)
               from t in phase 5r + 1 then
11         send(confirm(init), t, m, r) to all nodes
       fi
13 /* starting in phase k = 5r + 2: any node q */
       if received(confirm(init), t, m, r)
15             from≥ n − f_ℓ^s − f_ℓ^r − f_a then
           send(echo, t, m, r, q) to all nodes
17     fi
   /* starting in phase k = 5r + 3: any node q */
19     if received(echo, t, m, r, q) fromq then
           send(confirm(echo), t, m, r, q) to all nodes
21     fi
   /* starting in phase k = 5r + 4: any node q */
23     if received(confirm(echo), t, m, r, p)
               from≥ n − f_ℓ^s − f_ℓ^r − f_a then
25         𝕎_{t,m,r} = 𝕎_{t,m,r} ∪ {p}
       fi
27     if |𝕎_{t,m,r}| > f_a then
           send(echo, t, m, r, q) to all nodes
29     fi
       if |𝕎_{t,m,r}| > 2f_a then
31         accept(t, m, r)
           end_task()
33     fi
   }
```

Figure 3. *The Broadcast Primitive code for a broadcast started in round r. Code is for node q.*

Proof: The proof is divided into three parts, i.e., the three properties:

For *Correctness* we can assume that t is correct and all correct nodes, except f_ℓ^s receive the $init$-message, and thus send $confirm(init)$-messages. Consequently all correct nodes receive at least $n - f_\ell^s - f_\ell^r - f_a$ such confirmations, which will cause them to send $echo$ messages to all their peers. These $echo$ message from at least $n - f_a$ nodes will be confirmed – in the same way the $init$ message was – and thus $n - f_\ell^s - f_\ell^r - f_a$ confirmations for each of the $n - f_a$ *echo*-messages will be received at any correct node in phase $5r + 4$, which results in adding the originators of the *echo*-messages to the set of witnesses $\mathbb{W}$ and thus $|\mathbb{W}| \geq n - f_a$. Since $n - f_a > 2f_a$, the message sent will be accepted by all correct nodes at the end of phase $5r + 4$, i.e., within round r.

Unforgeability is shown by the following argument: If the transmitter never sends out an $init$-message, at most $f_\ell^{s,a}$ correct nodes will receive such a message, thus at most $f_\ell^{s,a} + f_\ell^{r,a} + f_a < n - f_\ell^s - f_\ell^r - f_a$ confirmations can be received by any correct node. Thus no correct node will ever send an *echo*-message. Again that means that at most $f_\ell^{s,a} + f_\ell^{r,a} + f_a < n - f_\ell^s - f_\ell^r - f_a$ echo confirmations can be received by any correct node. These bounds ensure that the witness set remains empty at all correct nodes. Hence no message is accepted.

To show the *Relay* property it suffices to show that any message accepted at one correct node p in phase k, will be accepted at all other correct nodes q in phase $k+2$. Since p has more than $2f_a$ nodes in its witness set, all other correct nodes must have more than f_a, since only arbitrary faulty nodes can cause one node to pass the bound of Line 24, and prevent another node to do so. Hence the difference in the size of the witness-sets at two correct nodes can be at most f_a.

Now since all correct nodes have more than f_a nodes in their witness-set, they will all send out echo messages, which will cause all correct nodes to add all correct nodes to their set of witnesses, and thus exceed the bound of Line 30.

The claimed resilience follows from the following argument: The required number of confirmations in lines 15 and 24 must be greater than the maximum number of forged messages a node can receive, i.e., $n - f_\ell^s - f_\ell^r - f_a > f_a + f_\ell^{s,a} + f_\ell^{r,a}$[5], thus

$$n > f_\ell^s + f_\ell^{s,a} + f_\ell^r + f_\ell^{r,a} + 2f_a.$$

On the other hand the acceptance bound of line 30 must be reachable for the correct nodes only, i.e., $n - f_a > 2f_a$, thus $n > 3f_a$. As both inequations must be fulfilled, we require the greater term of the two, which leaves us with the resilience term claimed above. This eventually completes the proof of Theorem 2.□

As for the claimed optimality, we argue as follows: In [17] Schmid shows that the lower bound for link failures only is $n > f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$. The central part of his proof is based on the fact, that Byzantine Agreement is not possible if some node p can withhold its data from some other node q for any number of rounds. This is shown by using two supernodes which can only communicate in one direction (due to link failures). However, we can place $2f_a$ nodes outside the two supernodes if f_a of them are Byzantine faulty. This would result in a lower bound of $n > 2f_a + f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$. However the lower bound for Byzantine Agreement in the presence of arbitrary node failures is $n > 3f_a$. [10]. Thus the minumum of these two terms is the lower bound. [3]

[5] Value faulty nodes can contribute up to f_a confirm messages, $f_\ell^{s,a}$ nodes might have received init or echo messages, that were never sent and thus send an confirmation, and each node may receive up to $f_\ell^{r,a}$ confirmations that were never sent.

4 Conclusion

We have presented a novel protocol, with optimal resilience. We showed that processors can be used to overcome the uncertainty originating from arbitrary node and (both omission and arbitrary) link failures at the same time. Thus tolerating link failures may, for some choices of $f_a > 0$ even come for free.

As far as future research is concerned one direction will definitely be the development of a universal hybrid lower bound for both link and hybrid node failures. Another direction of research could be improving existing algorithms or developing new ones that actually match that lower bound.

References

[1] M. K. Aguilera, W. Chen, and S. Toueg. Failure detection and consensus in the crash-recovery model. *Distributed Computing*, 13(2):99–125, Apr. 2000.

[2] M. Biely. Byzantine agreement under a perception based fault model. Master's thesis, Technical University Vienna, 2002.

[3] M. Biely. Towards an optimal algorithm for hybrid byzantine agreement. Technical Report 130, Department of Automation, Technical University Vienna, April 2003.

[4] M. Biely and U. Schmid. Message-efficient consensus in presence of hybrid node and link faults. Technical Report 183/1-116, Department of Automation, Vienna University of Technology, August 2001.

[5] D. Dolev, M. J. Fischer, R. Fowler, N. Lynch, and H. R. Strong. An efficient algorithm for Byzantine agreement without authentication. *Information and Control*, 52:257–274, 1982.

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

[7] L. Gong, P. Lincoln, and J. Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults. In *Proceedings Dependable Computing for Critical Applications-5*, pages 139–157, Champaign, IL, Sept. 1995.

[8] J. N. Gray. Notes on data base operating systems. In G. S. R. Bayer, R.M. Graham, editor, *Operating Systems: An Advanced Course*, volume 60 of *Lecture Notes in Computer Science*, chapter 3.F, page 465. Springer, New York, 1978.

[9] R. Guerraoui, R. Oliveira, and A. Schiper. Stubborn communication channels. Technical report, D'eoartment d'Informatique, Ecole Polytechnique Federale de Lausanne, Switzerland, 96.

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

[11] P. Lincoln and J. Rushby. A formally verified algorithm for interactive consistency under a hybrid fault model. In *Proceedings Fault Tolerant Computing Symposium 23*, pages 402–411, Toulouse, France, June 1993.

[12] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

[13] K. J. Perry and S. Toueg. Distributed agreement in the presence of processor and communication faults. *IEEE Transactions on Software Engineering*, SE-12(3):477–482, March 1986.

[14] S. S. Pinter and I. Shinahr. Distributed agreement in the presence of communication and process failures. In *Proceedings of the 14th IEEE Convention of Electrical & Electronics Engineers in Israel*. IEEE, Mar. 1985.

[15] H. M. Sayeed, M. Abu-Amara, and H. Abu-Amara. Optimal asynchronous agreement and leader election algorithm for complete networks with Byzantine faulty links. *Distributed Computing*, 9(3):147–156, 1995.

[16] U. Schmid. How to model link failures: A perception-based fault model. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'01)*, pages 57–66, Göteborg, Sweden, July 1–4, 2001.

[17] U. Schmid and B. Weiss. Synchronous byzantine agreement under hybrid process and link failures. Technical Report 183/1-124, Department of Automation, Vienna University of Technology, Nov. 2002. (replaces TR 183/1-110).

[18] U. Schmid, B. Weiss, and J. Rushby. Formally verified byzantine agreement in presence of link faults. In *22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 608–616, July 2-5, 2002.

[19] T. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2:80–94, 1987.

[20] S. Toueg, K. J. Perry, and T. K. Srikanth. Fast distributed agreement. *SIAM Journal on Computing*, 16(3):445–457, 1987.