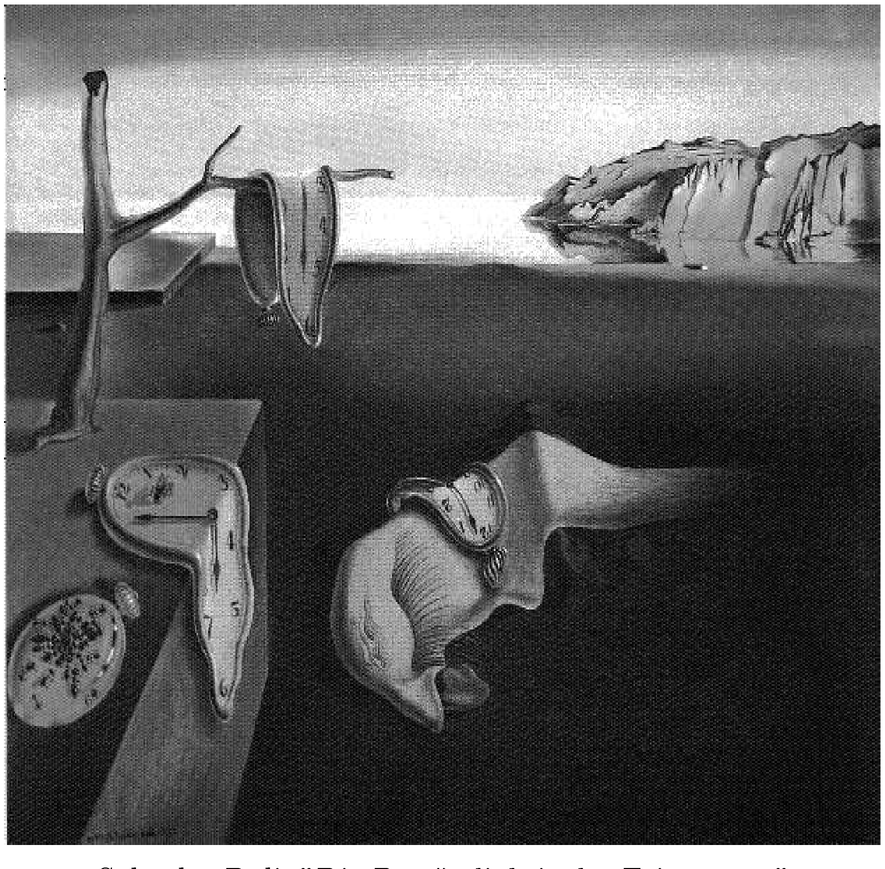
Institut für Automation
Abt. für Automatisierungssysteme

Technische Universität Wien

Projektbericht Nr. 183/1-114
March 2001

Consensus with Written Messages under Link Faults

Bettina Weiss and Ulrich Schmid


Salvador Dali, "Die Beständigkeit der Erinnerung"

Consensus with Written Messages Under Link Faults

BETTINA WEISS, ULRICH SCHMID
Technische Universität Wien
Department of Automation
Treitlstraße 1, A-1040 Vienna
Email: {bw, s}@auto.tuwien.ac.at
Phone: ++43-1-58801-18325, FAX: ++43-1-58801-18391

Abstract

This paper shows that deterministic consensus with written messages is possible in the presence of link faults. As in our analysis of consensus with oral messages (OMH), we circumvent the impossibility result of (Gray, 1978) by limiting the degree of inconsistency caused by link faults in the broadcasts of a single sender resp. the receptions of a single receiver. Relying upon a suitable perception-based *hybrid fault model, we prove that the $m+1$-round* Authenticated Hybrid Oral Messages *("Byzantine Generals") algorithm OMHA(m) of (Gong, Lincoln & Rushby, 1995) needs $n > 2f_\ell^s + f_\ell^r + 2(f_a + f_s) + f_c + m$ processes for tolerating at most f_ℓ^r receive link faults per process, f_ℓ^s broadcast link faults per process, and $f_a \leq m-1$, f_s, f_c arbitrary, symmetric, and manifest process faults. A considerably better fault-tolerance degree is established for their simple authenticated algorithm ZA(m), which needs only $n > f_\ell^s + f_\ell^r + f_a + f_s + f_c + 1$ processes for coping with the same number of faults. In case of broken signatures, OMHA degrades to OMH and hence requires an additional f_ℓ^{ra} in the above lower bound for n, where f_ℓ^{ra} is the number of non-omission link faults. For ZA, a process with a compromised signature must be considered as arbitrary faulty and hence be counted in f_a. Authenticated algorithms for consensus are therefore reasonably applicable even in wireless systems, where link faults and intrusions are the dominating source of errors.*

Keywords: *Fault-tolerant distributed systems, fault models, link faults, consensus, Byzantine generals, written messages, authentication.*

1 Motivation

Due to the well-known impossibility of deterministic consensus in presence of link faults [3], most existing work on this problem considers process faults only, see [6] for an overview. Still, in modern wireline and, in particular, wireless networks, the dominant cause of errors are message losses/corruptions. Those errors occur on the links and/or in the network interface of the receiving processes, hence cannot reasonably be ascribed to faulty processes[1]. Therefore, in order to reasonably use any distributed algorithm in a wireless network like our W2F fieldbus[2], a fault model incorporating both process and link faults is mandatory.

In [9], we demonstrated that the impossibility result of [3] can be circumvented by limiting the degree of inconsistency caused by link faults in the broadcasts of a single sender resp. the receptions of a single receiver: The *hybrid oral messages algorithm* (OMH) of [5] was shown to easily tolerate a large number of link faults, provided that enough processes are present and one additional round is used. An analysis of the resulting assumption coverage revealed that OMH can reasonably be employed even in wireless system architectures, where link fault probabilities up to 10^{-2} are quite common.

In this paper, we address the question of whether authentication can help with link faults, by considering the *hybrid written messages algorithms* OMHA and ZA developed in [2]. The remaining sections are organized as follows: In Section 2, we briefly review the perception-based hybrid fault model of [9]. Section 3 presents the Hybrid Oral Messages (OMH) algorithm of [5], along with the major results of our perception-based analysis in [9]. Section 4 explains authentication issues, Section 5 resp. 6 contains the analysis of OMH's authenticated version OMHA resp. the (superior) authenticated algorithm ZA. In Section 7, we consider what happens if signatures are broken, Section 8 is devoted to using signatures in broadcast networks. Some conclusions and directions of further research eventually round off the paper.

2 Fault Model

Deterministic fault models, like the one that at most f processes may behave Byzantine in each round, usually rest

[1]We use the terms process and node synonymously.

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

upon the total number of faults in the entire system. Channel and/or receiver-originating link faults, however, are difficult to accommodate in such models: Consider the model of [2], for example, where link faults are simply mapped to (sender-)process faults. If we grant each receiving process i, $1 \leq i \leq n$, just a single independent receive omission, it may, e.g., be the case that each process $i < n$ (resp. process n) drops the message from sender $i+1$ (resp. sender 1). Hence, all n processes must be considered faulty in this model. Even worse, since receiver-caused link faults often affect several consecutive messages, any f is quickly exceeded in real systems even in presence of less "exotic" fault patterns. The same argument obviously applies to fault models like the one of [7], where receive omissions are mapped to faults of the receiving processes. The particular distributed algorithm (in our case, consensus), however, might work very well in such situations.

This problem is avoided in the *perception-based fault model* introduced in [8], where the number of faults in the processes' *perception vector* —representing a particular receiver's local view of the system, as conveyed by the sending processes' messages— is considered. In this model, a sender process fault affects the sender's perception at all receiving processes in a correlated fashion, although probably inconsistently in case of a Byzantine fault. Therefore, a global fault assumption like "at most f processes may appear Byzantine" also implies "at most f perceptions may appear Byzantine", for any pair of non-faulty receiving processes. A perception-based fault model thus "preserves" the corresponding global one, which means that classic impossibility results like [1] remain valid.

A link fault, on the other hand, affects the sender's perception at the particular receiver only. Consequently, recalling the above example of $f_\ell = 1$ receive omissions per process, it is obvious that any two local views can differ only in at most $2f_\ell = 2$ perceptions, namely, the ones where either receiving process experienced its omission. Moreover, only at most $f_\ell = 1$ of the non-faulty perceptions present at some non-faulty process can be missing at any other non-faulty process.

For the analysis in this paper, we employ the particular hybrid perception-based fault model already employed in [9]. It is based upon the combination of a "link fault extension" of the oral messages model of [4] and the hybrid process fault model of [5].

Definition 1 (System Model [9, Def. 1]) *We consider a synchronous distributed system of n processes interconnected by a fully connected point-to-point network, which has the following properties:*

(A1^r) *If all processes $s_i \in \mathcal{S}$ of a set of non-faulty sender processes send a message containing V^{s_i} to some single receiver process r, at most f_ℓ^r of the values $V_r^{s_i}$ may differ from V^{s_i}. Let $f_\ell^{ra} \leq f_\ell^r$ be the maximum number of non-omissive, i.e., non-empty and hence value faulty, $V_r^{s_i}$ among those.*

(A1^s) *If a single non-faulty process s broadcasts (= successively sends) a message containing V^s to some set of non-faulty receiver processes $\mathcal{R}$, at most f_ℓ^s of the values $V_{r_i}^s$ may differ from V^s.*

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

(A3) *The absence of a message from sender s can be detected at any receiver r, which leads to $V_r^s = E$ for some distinguished value E.*

(P1) *In any execution, there may be at most f_a, f_s and f_c arbitrary, symmetric, and manifest faulty processes.*

Process faults are classified as follows:

- *A* manifest *faulty process s produces (detectably) missing messages or leads to a value that all non-faulty recipients can detect as obviously bad. All non-faulty receivers r deliver the value $V_r^s = E$ in this case.*
- *A* symmetric *faulty process s sends the same wrong —but not usually detectably bad— value X^s to every receiver. All non-faulty receivers r deliver $V_r^s = X^s$ —the value* "actually sent"— *in this case.*
- *An* arbitrary (asymmetric) *faulty process may inconsistently send any value to any receiver.*

Remarks:

1. Process and link faults occur independently of each other.
2. Faulty processes do not change their fault mode, i.e., must be counted in f_a, f_s or f_c according to their most severe behavior.
3. Each message reception resp. broadcast has its own "budget" f_ℓ^r resp. f_ℓ^s of link faults.
4. The particular links actually hit by link faults are usually different for each message reception/broadcast.
5. It will turn out that receive link faults (A1^r) resulting in an omission are easier to tolerate than those that produce a value fault. This is not the case for broadcast link faults (A1^s).
6. Broadcast link faults (A1^s) can also be interpreted as *restricted arbitrary process faults* with "degree of inconsistency" limited to f_ℓ^s. If at most f_ℓ^r processes suffer from such a fault, condition (A1^r) holds as well. Assuming at most f_ℓ^r restricted arbitrary process faults with a degree of inconsistency of at most f_ℓ^s hence also satisfies the requirements of Definition 1; consult [9] for some interesting consequences.

   Note carefully, however, that the two interpretations are not equivalent, since (A1^s) and (A1^r) admit faults that are not captured by the alternative process fault model. More specifically, a broadcast link fault may hit any sending process in the former, but is restricted to one of the f_ℓ^r faulty processes in the latter.

3 The Hybrid Oral Messages Algorithm

In this section, we will provide the definition of the *Hybrid Oral Messages* algorithm (OMH) introduced in [5], which is the basis of the authenticated algorithms OMHA and ZA considered in Section 5 and 6. For comparison purposes, we also restate the major results of OMH's perception-based analysis conducted in [9].

We consider the consensus problem in the usual "Byzantine Generals"-style, where the value v of a dedicated *transmitter* is to be disseminated consistently to the remaining $n-1$ receivers. Eventually, each non-faulty receiver p_i shall *deliver* a value v_{p_i} ascribed to the transmitter that satisfies the agreement and validity properties (B1) and (B2), as specified below. A fully-fledged consensus algorithm is obtained by using a seperate instance of Byzantine Generals for disseminating any process' local value and using a suitable choice function (majority) for the consensus result.

The algorithm OMH as specified in Definition 2 below uses two primitives:

- The *hybrid-majority* of a set $\mathcal{V}$ of values provides the majority of all non-E-values in $\mathcal{V}$; if no such majority exists, some arbitrary, but functionally determined, value is returned.
- The *wrapper function* $R(v)$ encodes a statement "I am reporting v" as a unique value. Reporting is undone by means of the inverse function $R^{-1}(v)$, which must guarantee $R^{-1}(R(v)) = v$. Note that only E, $R(E)$, $R(R(E))$, $R(R(R(E)))$, $\ldots$ must actually be distinguishable here; for each legitimate value v, we can allow $R(v) = R^{-1}(v) = v$.

Consult [5] for a detailled discussion of the above primitives.

Definition 2 (Algorithm OMH [5]) *The Hybrid Oral Messages algorithm OMH is defined recursively as follows:*

OMH(0):

1. *The transmitter sends its value v to every receiver.*
2. *Each receiver p delivers the value v_p received from the transmitter, or the value E if a missing or manifestly erroneous value was received.*

OMH(m), $m > 0$:

1. *The transmitter sends its value v to every receiver.*
2. *For each p, let v_p be the value receiver p receives from the transmitter, or E if no value, or a manifestly bad value, is received.*

   Each receiver p acts as the transmitter in Algorithm OMH($m-1$) to communicate the value $R(v_p)$ to all[3] *receivers [including itself].*
3. *For each p and q, let v_q be the value receiver p delivers as the result of OMH($m-1$) initiated by receiver q in step 2 above, or else E if no v_q or a manifestly bad value was delivered. Each receiver p calculates the hybrid-majority value among all values v_q and applies R^{-1} to that value. The result is delivered as the transmitter's value.*

To solve Byzantine Agreement, an algorithm has to satisfy the following properties:

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

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

- v, if the transmitter is non-faulty,
- E, if the transmitter is manifest faulty,
- the value X^s actually sent, if s is symmetrically faulty,
- unspecified, if the transmitter is arbitrarily faulty.

In [9], we showed that OMH guarantees (B1) and (B2) under the system model of Section 2, provided that the conditions given in Theorem 1 are satisfied.

Theorem 1 (Agreement and Validity [9, Thm. 1]) *For any $m \geq f_a + \min\{1, f_\ell^s\}$ and any f_a, f_s, f_c, f_ℓ^s, f_ℓ^r, f_ℓ^{ra}, the algorithm OMH(m) satisfies agreement (B1) and validity (B2) if there are strictly more than $2f_\ell^s + f_\ell^r + f_\ell^{ra} + 2(f_a + f_s) + f_c + m$ participating processes.*

To apply a deterministic fault model like the one of Definition 1 in practice, one also has to address the question of assumption coverage. More specifically, for the particular system in mind, the *probability of failure* Q implied by a possible violation of the fault assumptions (f_a, f_s, f_c, f_ℓ^s, f_ℓ^r) needs to be evaluated. This is particularly important with respect to our link faults, which cause Q to increase with every message broadcast during the execution of the algorithm: According to (A1^s) resp. (A1^r), no message broadcast/reception may suffer from more than f_ℓ^s resp. f_ℓ^r link faults. Given the fact that the algorithm OMH sends many, many messages, the question arises whether Q can

[3]There are $n-1$ receivers in the first instance OMH(m) of the algorithm; the transmitter does not participate in any way in further recursive instances.

eventually be made as small a desired by choosing suitable values of f_ℓ^s and f_ℓ^r.

In [9], we derived an upper bound on the probability of failure Q_m of OMH(m) for a simple probabilistic model of link faults: We assumed that the probability of loosing or corrupting a single message at the link or the receiver's network interface is $0 < p < 1$, and that such events occur independently of each other. Theorem 2 gives the appropriate result, which shows that the probability of failure Q_m

- rapidly grows with m and hence with the number f_a of arbitrary faults,
- marginally grows with n and hence with the number f_s resp. f_c of symmetric resp. manifest faults,
- rapidly decreases with the number of tolerated link faults f_ℓ (and with decreasing p, of course).

Theorem 2 (Assumption Coverage OMH [9, Thm. 2]) *For $n - m - f_\ell - 2 \geq 1$ and $np < 1$ sufficiently small, the probability of failure Q_m of OMH(m) satisfies $Q_m \leq Q'_m$ with*

$$\begin{aligned} Q'_m &\leq \Big(1 + \frac{1}{n-m-f_\ell-2}\Big)[n-1]_{m+f_\ell+1}\frac{p^{f_\ell+1}}{(f_\ell+1)!} \\ &= \mathcal{O}\Big(n^m \cdot \frac{(np)^{f_\ell+1}}{(f_\ell+1)!}\Big) \end{aligned}$$

where a term of order at most $\mathcal{O}\Big(\frac{(Q'_m)^2}{[n-f_\ell-2]_m}\Big)$ has been neglected.

Note carefully that Theorem 2 is valid for the authenticated algorithms considered in this paper as well, since both OMHA and ZA send and receive messages exactly as OMH does. In Tables 1–4, we give numerical values for Q'_m for different values of m and f_ℓ and $n = 4f_\ell + 3m + 1$, which allows $f_\ell^s = f_\ell^r = f_\ell$, $f_a = m - 1$ and $f_s = f_c = 1$ or, alternatively, $f_s = 0$, $f_c = 2$ by Theorem 1.

f_ℓ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
1	0.64	1.	1.	1.	1.	1.
2	0.59	1.	1.	1.	1.	1.
3	0.52	1.	1.	1.	1.	1.
5	0.36	1.	1.	1.	1.	1.
7	0.22	1.	1.	1.	1.	1.
10	0.095	1.0	1.	1.	1.	1.
15	0.019	0.86	1.	1.	1.	1.
20	0.0036	0.37	1.	1.	1.	1.

Table 1. *Value of probability of failure Q_m for $p = 0.1$ and $n = 4f_\ell + 3m + 1$.*

Whereas the probability of failure of OMH(m) given in Tables 1–4 is not bad, even in case of a typical "wireless loss probability" $p = 0.01$, it is nevertheless clear that an algorithm that uses less messages is preferable with respect

f_ℓ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
1	0.01	0.3	1	1	1	1
2	0.002	0.04	1	1	1	1
3	0.0002	0.006	0.3	1	1	1
5	2.10^{-6}	0.00009	0.005	0.3	1	1
7	2.10^{-8}	1.10^{-6}	0.00009	0.007	0.6	1
10	2.10^{-11}	2.10^{-9}	2.10^{-7}	0.00002	0.002	0.2
15	2.10^{-16}	2.10^{-14}	3.10^{-12}	4.10^{-10}	5.10^{-8}	8.10^{-6}
20	2.10^{-21}	2.10^{-19}	4.10^{-17}	7.10^{-15}	1.10^{-12}	2.10^{-10}

Table 2. *Value of (approximate) probability of failure Q'_m for $p = 0.01$ and $n = 4f_\ell + 3m + 1$.*

f_ℓ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
1	1.10^{-6}	0.00003	0.0009	0.03	1	1
2	2.10^{-9}	4.10^{-8}	2.10^{-6}	0.00007	0.004	0.2
3	2.10^{-12}	6.10^{-11}	3.10^{-9}	1.10^{-7}	7.10^{-6}	0.0004
5	2.10^{-18}	9.10^{-17}	5.10^{-15}	3.10^{-13}	2.10^{-11}	2.10^{-9}
7	2.10^{-24}	1.10^{-22}	9.10^{-21}	7.10^{-19}	6.10^{-17}	5.10^{-15}
10	2.10^{-33}	2.10^{-31}	2.10^{-29}	2.10^{-27}	2.10^{-25}	2.10^{-23}
15	2.10^{-48}	2.10^{-46}	3.10^{-44}	4.10^{-42}	5.10^{-40}	8.10^{-38}
20	2.10^{-63}	2.10^{-61}	4.10^{-59}	7.10^{-57}	1.10^{-54}	2.10^{-52}

Table 3. *Value of (approximate) probability of failure Q'_m for $p = 0.0001$ and $n = 4f_\ell + 3m + 1$.*

f_ℓ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
1	1.10^{-10}	3.10^{-9}	9.10^{-8}	3.10^{-6}	0.0001	0.007
2	2.10^{-15}	4.10^{-14}	2.10^{-12}	7.10^{-11}	4.10^{-9}	2.10^{-7}
3	2.10^{-20}	6.10^{-19}	3.10^{-17}	1.10^{-15}	7.10^{-14}	5.10^{-12}
5	2.10^{-30}	9.10^{-29}	5.10^{-27}	3.10^{-25}	2.10^{-23}	2.10^{-21}
7	2.10^{-40}	1.10^{-38}	9.10^{-37}	7.10^{-35}	6.10^{-33}	5.10^{-31}
10	2.10^{-55}	2.10^{-53}	2.10^{-51}	2.10^{-49}	2.10^{-47}	2.10^{-45}
15	2.10^{-80}	2.10^{-78}	3.10^{-76}	4.10^{-74}	5.10^{-72}	8.10^{-70}
20	2.10^{-105}	2.10^{-103}	4.10^{-101}	7.10^{-99}	1.10^{-96}	2.10^{-94}

Table 4. *Value of (approximate) probability of failure Q'_m for $p = 0.000001$ and $n = 4f_\ell + 3m + 1$.*

to our fault model. In [9], we therefore considered a modified algorithm $\overline{\text{OMH}}$ as well, which combines all round-k-messages that a process sends during OMH into a single message. Theorem 3 shows that the resulting probability of failure $\overline{Q}_m$ does no longer grow with m. Tables 5 and 6 contain a few numerical values for $\overline{Q}'_m$ for different values of m and f_ℓ and the same $n = 4f_\ell + 3m + 1$ as used before, cp. Tables 1 and 2.

Theorem 3 (Assumption Coverage $\overline{\textbf{OMH}}$) *For $n - m - f_\ell - 2 \geq 1$ and $np < 1$ sufficiently small, the probability of failure $\overline{Q}_m$ of $\overline{\text{OMH}}$(m) satisfies $\overline{Q}_m \leq \overline{Q}'_m$ with*

$$\overline{Q}'_m \leq \frac{[n+1]_{f_\ell+3} - [n-m]_{f_\ell+3}}{f_\ell + 3} \cdot \frac{p^{f_\ell+1}}{(f_\ell+1)!} = \mathcal{O}\Big(n \cdot \frac{(np)^{f_\ell+1}}{(f_\ell+2)!}\Big)$$

where a term of order at most $\mathcal{O}\big((\overline{Q}'_m)^2\big)$ has been neglected.

f_ℓ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
1	0.88	1.	1.	1.	1.	1.
2	0.85	1.	1.	1.	1.	1.
3	0.80	0.99	1.	1.	1.	1.
5	0.62	0.93	1.	1.	1.	1.
7	0.42	0.77	0.96	1.	1.	1.
10	0.20	0.43	0.71	0.91	0.99	1.
15	0.041	0.10	0.21	0.37	0.58	0.78
20	0.0078	0.019	0.041	0.08	0.14	0.24

Table 5. *Value of (approximated) probability of failure $\overline{Q}_m$ for $p = 0.1$ and $n = 4f_\ell + 3m + 1$.*

f_ℓ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
1	0.04	0.1	0.4	0.9	1.	1.
2	0.004	0.02	0.04	0.1	0.2	0.4
3	0.0004	0.002	0.004	0.01	0.02	0.04
5	5.10^{-6}	0.00002	0.00005	0.0001	0.0002	0.0005
7	5.10^{-8}	2.10^{-7}	5.10^{-7}	1.10^{-6}	3.10^{-6}	5.10^{-6}
10	5.10^{-11}	2.10^{-10}	4.10^{-10}	1.10^{-9}	3.10^{-9}	6.10^{-9}
15	4.10^{-16}	1.10^{-15}	4.10^{-15}	1.10^{-14}	2.10^{-14}	5.10^{-14}
20	4.10^{-21}	1.10^{-20}	3.10^{-20}	9.10^{-20}	2.10^{-19}	5.10^{-19}

Table 6. *Value of (approximate) probability of failure $\overline{Q}'_m$ for $p = 0.01$ and $n = 4f_\ell + 3m + 1$.*

4 Authentication

Consensus with written messages (introduced in [4]) assumes that no process can make undetectable modifications to messages and that the originator of a message is always known. It is generally agreed that electronic signatures can be used to achieve these goals, although there are some pitfalls [2].

The assumptions placed on the authentication scheme are:

(SA1) A process cannot change the contents of a message.

(SA2) A process cannot forge a signature.

(SA3) A process can only relay a message it has previously received in the same execution run (this prevents a faulty process from replaying an earlier message).

(SA4) A valid signature cannot be mistaken for an invalid one (i.e., the signature does not introduce new errors).

Generally, every process p uses its signature σ_p to sign a message v, generating the signed message $\sigma_p(v)$. So all messages received in stage k of the algorithm bear the $k+1$ signatures of the previous transmitters. A value v that has been sent from process p_1 along the chain of processes $p_2 \dots p_k$ must be packed into a message $M = \sigma_{p_k} \cdots \sigma_{p_1}(v)$, which allows a node to recognize several more manifest faults upon reception of a message:

(M1) The message contains $v \neq E$ but has not been signed by the original transmitter first.

(M2) If a message arrives in round k, then it must either bear $k+1$ signatures, the first of which is from the original transmitter, or it must contain the value E. All other messages are manifest faulty.

(M3) The message arrives on the link from process p_i but has not been signed by p_i last.

(M4) The message contains a signature at least twice (i.e., one node has signed the message twice).

(M5) Two messages bear the same signature chain and contain different values.

The reaction of the node to these manifest faults depends on its own fault status. We assume the following:

- A non faulty node recognizes (M1)-(M5) and discards the message received in (M1)-(M4), reporting E instead. In (M5), it only recognizes the manifest fault upon reception of the second message and simply discards it. It will, however, forward the first message. If that behavior is not desired, we can circumvent the problem by demanding that all processes wait until they receive all messages from stage k of the algorithm before sending these values in the next stage of the recursion. In that case, the node will discard both messages in (M5) and report E instead.
- A *manifest* faulty node produces a manifest fault on all receivers regardless of what it receives.

- A *symmetric* faulty node may ignore (M1)-(M4) and send the manifest faulty messages. If the signatures are secure, then it may also ignore (M5) and send both messages it has received. However, if we assume that signatures are broken, then we must assume that a symmetric faulty node recognizes (M5) and does not send two different messages along, cf. Lemma 3.

- A *arbitrary* faulty node ignores (M1)-(M5) and sends along whatever it likes. In particular, it may send a message it should have recognized as manifest faulty.

Using signatures has a beneficial effect on process faults, because faulty processes cannot introduce new values into the system, as will be proved in Lemma 3 in Section 7. A faulty process that relays a message can only choose not to relay it at all, or to report that it has experienced a manifest fault.

How does authentication affect link faults? Obviously, a link fault can either produce a detectable fault or replace the original message with some other valid message sent by the same process. In case of a non-arbitrary faulty process, replacing the message has no effect since all valid messages are the same, and in case of an arbitrary faulty process, the value sent is not important anyway. So authentication does have a positive effect on the severity of link faults as well, since it prohibits value faults.

5 Algorithm OMHA

In this section, we will analyze a variant of the algorithm OMHA developed in [2] under the system model of Section 2. The original algorithm OMHA is the same as OMH except that every message sent in OMHA(m) with $m > 0$ must be signed.

As we have argued in the previous section, faulty processes cannot generate new values, so the only values that do occur are those originally sent by the transmitter, E, and various $R(E)$'s. In the hybrid fault model of [5], the fact that a faulty process can inject an $R(E)$ value is enough to make the performance of OMHA no better than that of OMH. But how does authentication affect link faults in OMHA? The previous section tells us that link faults do not introduce any new values if authentication is used. However, the original version of OMHA does not sign messages in OMHA(0), and at this stage, a link fault could insert a bogus $R(E)$ value. Therefore, we must assume that messages are signed even in OMHA(0).

Lemma 1 (Validity) *For any $m \geq min\{1, f_\ell^s\}$ and any $f_a, f_s, f_c, f_\ell^s, f_\ell^r$, algorithm OMHA(m) satisfies the validity property if there are strictly more than $2f_\ell^s + f_\ell^r + 2(f_a + f_s) + f_c + m$ participating processes.*

Proof: The proof is by induction on m. If $f_\ell^s = 0$, i.e., if there were no link faults in message broadcasts, every receiver simply uses OMHA(0), which obviously guarantees (B2) since the transmitter must not be arbitrary faulty. The induction starts at $m = 0$ in this case.

If $f_\ell^s > 0$, however, induction must start with $m = 1$ as the base case: According to the definition of OMHA(1), every (non-faulty) receiver p of step 1 of OMHA(1) uses OMHA(0) to disseminate its v_p to all other receivers q. Abbreviating the number of initially participating receivers by

$$n' \geq 2f_\ell^s + f_\ell^r + 2(f_a + f_s) + f_c + m, \qquad (1)$$

with $f_c' \leq f_c$ manifest faulty ones among those, there must be at least $n' - f_\ell^s - f_a - f_s - f_c'$ non-faulty receivers p of step 1 of OMHA(1) that get the same $v_p = \nu$ (recall that $\nu = E$ in case of a manifest faulty transmitter), despite the at most f_ℓ^s link faults according to (A1^s).

It hence follows that any non-faulty receiver q of step 1 of OMHA(0) obtains at least n_q' identical values $R(\nu)$ with

$$n_q' = n' - f_\ell^s - f_a - f_s - f_c' - f_\ell^{ra'} - f_\ell^{ro'}, \qquad (2)$$

where $f_\ell^{ro'}$ resp. $f_\ell^{ra'} \leq f_\ell^{ra}$ with $f_\ell^{ro'} + f_\ell^{ra'} \leq f_\ell^r$ denotes the number of omission resp. value faults caused by link faults according to (A1^r). Due to the signatures, both omission and value faults are detectable. Note carefully that (2) is also true for the transmitter ($q = p$), which must be non-faulty if counted here and must hence have "sent" itself the correct value $R(\nu)$. Since there are f_c' manifest faulty receivers, each non-faulty receiver q obtains a minimum of $f_c' + f_\ell^{ro'} + f_\ell^{ra'}$ values equal to E. Therefore, a non-faulty receiver q can get at most $n_q'' = n' - f_c' - f_\ell^{ro'} - f_\ell^{ra'}$ values different from E.

Hence, recalling (2), we find

$$\begin{aligned} 2n_q' - n_q'' &= n' - 2f_\ell^s - 2f_a - 2f_s - f_c' - f_\ell^{ro'} - f_\ell^{ra'} \\ &\geq f_\ell^r - f_\ell^{ro'} - f_\ell^{ra'} + (f_c - f_c') + m \\ &> 0 \end{aligned}$$

since $m = 1$ and $f_\ell^{ro'} + f_\ell^{ra'} \leq f_\ell^r$, which implies that $R(\nu)$ wins the hybrid-majority at any non-faulty receiver. Since R^{-1} is applied to the result, the final value ν is obtained as required.

Assuming now that the lemma is already true for $m-1 \geq \min\{1, f_\ell^s\}$, we will show that it is also true for m: The proof is almost the same as the one for the base case; we only have to replace the application of OMHA(0) by OMHA($m-1$) with n' participants: As above, we have at least $n' - f_\ell^s - f_a - f_s - f_c'$ non-faulty receivers p of step 1 of OMHA(m) that apply OMHA($m-1$) to consistently disseminate their $R(v_p) = R(\nu)$. Since both m and the number of participants decreased by one, we can apply the induction hypothesis to OMHA($m-1$) to conclude that any non-faulty receiver q actually delivers $R(\nu)$ in this step. Consequently, any non-faulty receiver q must have at least

$$\overline{n}_q' = n' - f_\ell^s - f_a - f_s - f_c'$$

values equal to $R(\nu)$ among the at most $\overline{n}_q'' = n' - f_c'$ non-E values it may have got at all. Since $m > 1$,

$$2\overline{n}_q' - \overline{n}_q'' = n' - 2f_\ell^s - 2f_a - 2f_s - f_c'$$

$$\begin{aligned} &\geq\ f_\ell^r + (f_c - f_c') + m \\ &>\ 0 \end{aligned}$$

as before, so $R(\nu)$ wins the hybrid-majority at any non-faulty receiver and the final value $\nu = R^{-1}(R(\nu))$ follows. $\Box$

Theorem 4 (Agreement and Validity) *For any $m \geq f_a + min\{1, f_\ell^s\}$, algorithm OMHA(m) satisfies agreement and validity if there are strictly more than $2f_\ell^s + f_\ell^r + 2(f_a + f_s) + f_c + m$ participating processes.*

Proof: The proof is by induction on m. In the base case $m = min\{1, f_\ell^s\}$, we have $f_a = 0$ such that the transmitter must not be arbitrary faulty. Hence, Lemma 1 holds and validity implies agreement.

Let us now assume that the assumption holds for OMHA($m-1$). If we look at OMHA(m), we only have to consider the case where the transmitter is arbitrary faulty (otherwise, validity already implies agreement). The transmitter sends its values to the receivers, which in turn call OMHA($m-1$). Since the transmitter is arbitrary faulty, OMHA($m-1$) is called with one process less and $f_a - 1$ arbitrary faults and the induction hypothesis applies. Hence, every non-faulty process uses the same set of delivered values for OMHA(m) and agreement is fulfilled. $\Box$

Remarks:

1. Note that the proof for agreement only uses the validity property and the fact that f_a is added to the m required by validity. Hence, every consensus algorithm of this type that achieves validity for a given m will also achieve agreement for $m' = m + f_a$.

2. We already mentioned that the original version of OMHA in [2] avoided signing the messages sent by OMHA(0). Recall that (A2) in Definition 1 assumes a point-to-point network where the transmitter of a message can be uniquely identified. If a link fault can only cause an omission or a manifest fault, Theorem 4 would remain valid for this original algorithm as well. However, if a link fault can substitute an $R(E)$ value for the real message, then the original algorithm performs no better than OMH. Hence, by Theorem 1, we would need strictly more than $2f_\ell^s + f_\ell^r + f_\ell^{ra} + 2(f_a + f_s) + f_c + m$ processes for this variant of OMHA.

Since OMHA just adds signatures to OHM, both algorithms send and receive the same messages. Therefore, the results of OHM's assumption coverage analysis (Theorem 2 and 3 as well as Tables 1–6) are also valid for OMHA.

6 Algorithm ZA

In this section, we will analyze the authenticated algorithm ZA of [2] under our perception-based fault model. The algorithm ZA has been derived from the flawed algorithm Z of [10] and provides a much better fault-tolerance degree than OMHA. However, its correctness depends critically upon Assumptions (SA1)–(SA4) (but see Section 7) and upon the fact that the transmitter must be known.

Definition 3 (Algorithm ZA [2]) *The algorithm ZA is defined recursively as follows (we assume that E is assigned whenever a message was not received or manifest faulty or incorrectly signed):*

ZA(0):

1. *The transmitter sends its value to every receiver.*
2. *Each receiver delivers the value obtained from the transmitter, or some fixed value E.*

ZA(m), $m > 0$:

1. *The transmitter signs and sends its value to every receiver.*
2. *For each process p, let v_p be the value p has obtained from the transmitter, or E. Each receiver p acts as the transmitter in algorithm ZA($m-1$) to send the value v_p to the $n-1$ receivers [including itself].*
3. *For each process p and q, let v_q be the value p has obtained from receiver q in step (2) using algorithm ZA($m-1$), or E if no such value of a manifest faulty one was delivered. Each receiver p calculates the majority value among all non-E values v_q it has received; if no non-E-value exists, E is delivered, if no majority exists, some arbitrary but fixed value is used.*

Note that the strength of the signed algorithm lies in the fact that the only values that can occur are the values sent by the original transmitter and E. So if the transmitter is not arbitrary faulty, then every process can only receive the original value and E.

Lemma 2 (Validity) *For any $m \geq min\{1, f_\ell^s\}$ and any f_a, f_s, f_c, f_ℓ^s, f_ℓ^r, algorithm ZA(m) satisfies the validity property if there are strictly more than $f_\ell^s + f_\ell^r + f_a + f_s + f_c + 1$ participating processes.*

Proof: Let us assume a non-faulty transmitter which sends value v. Then we only have to show that every good receiver obtains at least one v in the first $min\{1, f_\ell^s\}$ rounds, because once it has obtained the value, it will also deliver it. Recall that any transmitter "sends" its value to itself in step 2 of ZA(m) as well.

If $f_\ell^s = 0$, then we allow $m = 0$. However, since the transmitter is non-faulty and the links are non-faulty as well (note that $f_\ell^s = 0$ implies $f_\ell^r = 0$), every good receiver will obtain v in ZA(0) and will deliver it.

Now let $m \geq 1$. In ZA(m), the transmitter signs and sends its value v to all receivers. If we assume n' non-faulty receivers, at least $n' - f_\ell^s$ of these will receive v, and at most f_ℓ^s will receive E. In the second round, the $n' - f_\ell^s$ receivers will broadcast v to the other $n' - 1$ non-faulty receivers. So each non-faulty receiver gets v from at least $n' - f_\ell^s - f_\ell^r$ processes and all we have to do is to ensure that the number of non-faulty processes is $n' > f_\ell^s + f_\ell^r$, so the number of processes must be $n > f_\ell^s + f_\ell^r + f_a + f_s + f_c + 1$ (the +1 comes from the transmitter). As soon as a process has obtained at least one v, it will deliver it, so for all $m \geq 1$, any non-faulty receiver will deliver v. □

Theorem 5 (Agreement and Validity) *For any $m \geq f_a + min\{1, f_\ell^s\}$, algorithm ZA(m) satisfies agreement and validity if there are strictly more than $f_\ell^s + f_\ell^r + f_a + f_s + f_c + 1$ participating processes.*

Proof: As argued in Remark 1 on Theorem 4, the proof is the same as that for agreement in OMHA. □

Remarks:

1. We have changed the definition of ZA so that every receiver relays the message to all other receivers including itself in step 2 of ZA(m). In the original paper [2], the message was only relayed to the other $n - 2$ receivers. However, the receiver needs the own value for step 3, so it should "send" that value to itself as well.

2. Since ZA does not distinguish between E and $R(E)$ as OMHA does, using signatures means that the only possible values a process ever encounters are those originally sent by the transmitter and E values. Link faults are also recognized as manifest faults in all subsequent stages of the algorithm. Contrary to OMHA, where the algorithm benefits from signing messages in OMHA(0), link faults can only insert the values E or a valid signed message v in ZA(0), so we do not require signatures in this last stage if the message has been signed at least once (hence, we require $m \geq min\{1, f_\ell^s\}$).

Like OMHA, ZA also sends and receives the same messages as OMH. The results of OMH's assumption coverage analysis, namely, Theorem 2 and 3, hence remain valid for ZA as well. Note carefully, however, that the numerical results in Tables 1–6 assume $n = 4f_\ell + 3m + 1$ and not ZA's minimum setting $n = f_\ell + m + 1$.

7 Broken Signatures

In the original Byzantine Generals paper [4], it was assumed that only messages from non-faulty processes cannot be forged. If we translate that to the hybrid fault model, then messages from arbitrary faulty processes can be forged, i.e., their signature is not secure. If we take this one step further, we can assume that the signatures of all arbitrary faulty processes as well as the signatures of at most f_b non arbitrary faulty processes (i.e., processes which are not arbitrary faulty) are common knowledge. Knowing a signature allows a node to generate a message in the name of some other node, although this does not imply that it is able to eventually generate a valid chain of signatures.

Lemma 3 (Signatures) *At the end of the execution run, there are no two messages $M = \sigma_{p_k} \dots \sigma_{p_1}(v)$ and $M' = \sigma_{p_k} \dots \sigma_{p_1}(v')$ with $v \neq v'$, if at least one signature σ_{p_i} in M, M' is from a non arbitrary faulty node p_i with an unbroken signature.*

Proof: Let us assume that two such messages M and M' exist. If we consider one node p_i which has signed both messages, then this node must have received both $\sigma_{p_{i-1}} \dots \sigma_{p_1}(v)$ and $\sigma_{p_{i-1}} \dots \sigma_{p_1}(v')$. But according to (M4) in Section 4, every non arbitrary faulty node at most signs the first message, but not the second one. So if p_i has signed both messages, it must be arbitrarily faulty. If the node has not signed the messages, then someone else must have done so in its name, so its signature must have been broken. □

If we recall how the consensus algorithms we have analyzed work, we see that in stage 1 of the algorithm, that is, in OMHA(1) or ZA(1), each non-faulty process uses the hybrid-majority of an input set whose chain of signatures only differs in the last signature. The chains may have different lengths, though, but if a chain has less than m signatures, then it must contain the value E. For all chains with length m, we can deduce from Lemma 3 that each process will work with the same input set if there is at least one non arbitrary faulty process in the chain whose signature has not been broken. So we will solve the problem by treating nodes with compromised signatures like arbitrary faulty nodes.

Theorem 6 (ZA with Broken Signatures) *For any $m \geq f_a + f_b + min\{1, f_\ell^s\}$, algorithm ZA(m) satisfies agreement and validity if there are strictly more than $f_\ell^s + f_\ell^r + f_a + f_b + f_s + f_c + 1$ participating processes.*

Proof: For validity, it is easy to see that if the transmitter is not arbitrary faulty (and thus due to our assumption has no broken signature), then the only values that a non-faulty node considers are the value v sent by the transmitter and E. Therefore, the argument of Lemma 2 still holds.

For agreement, we now use enough rounds to ensure that every message received in ZA(0) has been signed by at least one non arbitrary faulty process or contains E. Therefore, Lemma 3 guarantees that no fictive messages can occur, and the algorithm is still the same as without broken signatures. So the proof of Theorem 5 still holds if we count broken signatures as arbitrary faults. □

Algorithm OMHA could be made tolerant to broken signatures in the same fashion as ZA. However, for OMHA it is probably cheaper with respect to the required number of nodes to simply let it degrade to OMH. So in fact, if there is a possibility that signatures might be compromised, then one should spend additional f_ℓ^{ra} nodes according to Theorem 1 or, preferably, simply dispose of authentication and use OMH instead.

8 Broadcast Networks

The consensus algorithms analyzed in this paper assume a point-to-point network. This implies that the sender of a message is always known. But what happens if we use those algorithms on a broadcast network? If we do not sign messages, we obviously loose the ability to identify the sender of a message, thus allowing faulty processes to impersonate non-faulty processes. So the oral messages algorithms would not work in this case. Written messages algorithms, however, should reasonably[4] work because they do not allow impersonation. In fact, without link faults, these algorithms would even benefit from the broadcast network, because arbitrary faulty processes are not possible anymore. Since only one message is sent by every process, every receiver must get the same value. So we can in fact set $f_a = 0$ and count all arbitrary faults as symmetric faults for any written messages algorithm analyzed under the hybrid fault model.

If link faults are possible, however, we find that they now have a lot more power on the algorithm than before. Whereas they can simply be caught by adding an appropriate multiple of f_ℓ^r and f_ℓ^s to the number of processes in the point-to-point case, we experience the unpleasant effect that they make arbitrary faults possible in the broadcast case. Consider a message from an arbitrary faulty process which is not received by f_ℓ^s receivers. If that process sends a second message containing a different value, which is not received by another f_ℓ^s receivers, then at most $2f_\ell^s$ receivers will only get one message and will assume that the message is valid. The rest do detect the second message from the same receiver and will use the value E due to the manifest fault. So the obvious solution is either to count arbitrary faults again, or to count sender link failures twice, i.e., require $4f_\ell^s$ instead of $2f_\ell^s$ additional processes.

Note that an arbitrary faulty process can do the worst damage by sending two messages. With a third message, again only f_ℓ^s receivers might not detect a manifest fault.

9 Conclusions

We analyzed two different authenticated algorithms for consensus (Byzantine Generals) under a hybrid perception-based fault model, which captures both process and link faults. For the $m+1$-round *Authenticated Hybrid Oral Messages* algorithm OMHA(m) of [2], we showed that $n > 2f_\ell^s + f_\ell^r + 2(f_a + f_s) + f_c + m$ processes are needed for tolerating at most f_ℓ^r receive link faults per process, f_ℓ^s broadcast link faults per process, and $f_a \leq m-1$, f_s, f_c arbitrary, symmetric, and manifest process faults. A considerably better fault-tolerance degree was established for the simple authenticated algorithm ZA(m) of [2], which needs only $n > f_\ell^s + f_\ell^r + f_a + f_s + f_c + 1$ processes for coping with the same number of faults. The impossibility result of [3] was circumvented by limiting the maximum number of link faults affecting the broadcast of a single message resp. the reception of a message from multiple senders to f_ℓ^s resp. f_ℓ^r. According to the results of the assumption coverage analysis in [9], this is not too severe a restriction even in today's wireless system architectures, where link fault probabilities up to 10^{-2} are quite common.

Our results show that the usefulness of authentication depends heavily upon the particular algorithm used. In fact, a consensus algorithm should be specifically designed for using written messages and not simply adapted from an oral messages solution: Whereas OMHA did not profit much from authentication, ZA benefits considerably — but also depends critically upon its strength. It turned out, however, that both algorithms can withstand intrusions to some extent: In case of broken signatures, OMHA degrades to OMH and hence requires an additional f_ℓ^{ra} in the lower bound for n. For ZA, a process with a compromised signature must be considered as arbitrary faulty and therefore counted in f_a. As far as link faults are concerned, authentication serves to identify and tolerate link value faults: Any algorithm that requires $f_\ell^r + f_\ell^{ra}$ processes to tolerate link faults will only require f_ℓ^r processes in the authenticated version. Apart from that, authentication is the only means to (more or less) safely employ algorithms like OMHA on top of broadcast networks.

We can hence conclude that written messages consensus algorithms are definitely more powerful than their oral messages counterparts. They can reasonably be employed even in wireless system architectures, where link faults and intrusions are the dominating source of errors. However, signing a message is a time-consuming operation, so algorithms that require few messages would be preferable. Part of our further research will address this problem.

[4]Besides of the problem of jamming.

References

[1] M. Fischer, N. Lynch, and M. Merrit. Easy impossibility proofs for the distributed consensus problem. *Distributed Computing*, 1(1):26–39, 1986.

[2] 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.

[3] J. 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.

[4] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[5] 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.

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

[7] 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.

[8] U. Schmid. A perception-based fault model for single-round agreement algorithms. Technical Report 183/1-108, Vienna University of Technology, Department of Automation, Oct. 2000. (Submitted to IEEE Transactions on Parallel and Distributed Systems).

[9] U. Schmid and B. Weiss. Consensus with oral messages: Link faults revisited. Technical Report 183/1-110, Department of Automation, TU Vienna, February 2001. (Submitted to DISC'01).

[10] P. M. Thambidurai and Y. K. Park. Interactive consistency with multiple failure modes. In *Proceedings 7th Reliable Distributed Systems Symposium*, Oct. 1988.