Reaching (and Maintaining) Agreement in the Presence of Mobile Faults*

(EXTENDED ABSTRACT)

JUAN A. GARAY

IBM T. J. Watson Research Center
P.O. Box 704,
Yorktown Heights, NY 10598, USA.

Abstract. In this paper we consider a model where malicious agents can corrupt hosts and move around in a network of processors. We investigate the issue of fault mobility and the faults' power of disruption as a function of the fundamental parameter in such systems: the faults' *speed.* We do so by evaluating in a mobile-fault environment a classical testbed problem for fault-tolerant distributed computing: Byzantine agreement.
We present a family of mobile-fault models $\mathtt{MF}(\frac{t}{n-1}, \rho)$. In $\mathtt{MF}(\frac{t}{n-1}, \rho)$ there are a total of n processors, the maximum number of mobile faults is t, and their roaming pace is ρ (for example, $\rho = 3$ means that it takes a virus at least 3 rounds to "hop" to the next host). We define *MBA*, a version of the agreement problem that is adequate for such environments, and give a series of impossibility results. We show, in particular, that under no other asumptions *MBA* is not possible in $\mathtt{MF}(\cdot, 1)$ unless $t = 0$. We show one way of overcoming the above difficulties, and give two *MBA* protocols, one for $\mathtt{MF}(\frac{1}{6}, 1)$ and one for $\mathtt{MF}(\frac{1}{4}, 2)$. This improves on the result of Reischuk [14] who, in our parlance, presented a protocol for $\mathtt{MF}(\frac{1}{6}, 3)$. The protocols run in $O(n)$ rounds—the optimal for these models.

* Appeared in *Proc. 8th International Workshop on Distributed Algorithms*, LNCS (857), Springer Verlag, pp. 253-264, Terschelling (NL), September 94.

1 Introduction

We consider a model where malicious agents can corrupt hosts and move around in a network of processors. The importance of this issue is rapidly growing with the current trend of universalization of computer networking and open systems, and, specially, with the advent of the so-called "agent technology." Previous works have, in one way or another, addressed the issue of fault mobility. Early on, Reischuk [14] considered a model with malicious faults covering a fraction of the network, and designed a Byzantine agreement [13] protocol able to tolerate them as long as they remain stationary for a given interval of time. More recently, Ostrovsky and Yung [12] have proposed randomized methods to withstand the attack of mobile viruses. Canetti and Herzberg [3] use cryptographic techniques to achieve secure computation in a system where different sets of servers may be broken into at different times, but not all at once. Finally, Franklin and Yung [10] address the issue of privacy in a system with mobile eavesdroppers.

In this paper we investigate the issue of fault mobility and the faults' power of disruption as a function of the fundamental parameter in such systems: the faults' *speed*. We do so by evaluating in a mobile-fault environment the classical testbed problem for fault-tolerant distributed computing: Byzantine Agreement (BA) [13]. In BA, a network of processors is given some of which may be maliciously faulty. At the offset of the computation, each processor p holds an initial value $v_p \in \{0,1\}$. Regardless of the behavior of the faulty processors, required is a distributed protocol satisfying the following conditions:

- **Decision:** Every nonfaulty processor eventually irreversibly decides on a value $d_p \in \{0,1\}$.
- **Agreement:** The nonfaulty processors decide on the same value.
- **Validity:** If the initial values v_p of all nonfaulty processors are identical, then $d_p = v$ for all nonfaulty processors p.

Our work is motivated by the work of Reischuk [14], who stated the following condition (henceforth referred to as the "$\mathcal{R}$" condition):

($\mathcal{R}$) BA can only be achieved if in *each round* less than $\frac{1}{3}$ of the total number of processors are faulty.

One of the results we establish in this paper is the following, seemingly stronger, counterpart to the above condition (we state here the negative version). Let t be the bound on the maximum number of processors being corrupted and ρ the maximum *roaming* pace of the faults (for example, $\rho = 3$ means that it takes a virus at least 3 rounds to "hop" to the next host):

Proposition 1: *Let $\rho = 1$. Then* BA *can only be achieved if $t = 0$.*

More specifically, the contributions of this paper are as follows:

- We define a family of mobile-fault models $\mathtt{MF}(\frac{t}{n-1}, \rho)$. In $\mathtt{MF}(\frac{t}{n-1}, \rho)$ there are a total of n processors, the maximum number of mobile faults is t, and their roaming pace is ρ.
- We define the *Mobile-Fault Byzantine Agreement* problem (*MBA*), a version of *BA* that is adequate for the mobile-fault environment.
- We establish a couple of impossibility results in these models *à la* Proposition 1, and give one condition that allows us to overcome these difficulties.
- Subject to this proviso, we give two *MBA* protocols, one for $\mathtt{MF}(\frac{1}{6}, 1)$ and one for $\mathtt{MF}(\frac{1}{4}, 2)$. This improves on the result of [14] who, in our jargon, presented a protocol for $\mathtt{MF}(\frac{1}{6}, 3)$. The protocols run in $O(n)$ rounds—the optimal for these models.

We emphasize that our study should be viewed as an abstraction of situations with transient faults. The model could also capture some aspects of computer viruses, e.g., when the antidote rate is as fast as the viruses' propagation rate.

The remainder of the paper is organized as follows. In Section 2 we describe the environment, define the MF models and the *MBA* problem. In Section 3 we establish the lower bounds and necessary conditions for *MBA*. In Section 4 we present the *MBA* protocols. We conclude with some final remarks.

2 Model and Problem Definition

We are given a network of n processors numbered 1 through n that may communicate only by exchanging messages via a reliable point-to-point channel. In order to be able to concentrate more cleanly on the issue of mobility, we shall consider the standard model of synchronous computation. In such a model, the network computation evolves as a series of *rounds*, during which processors can send one message to all other processors, receive those messages, and perform some local computation.

We assume that there are malicious agents in the network that can "corrupt" the processors. The semantics of corruption is as follows: We assume that the malicious agents are very powerful in the sense that they can take control over the processors' CPUs and completely wipe out their memories. Furthermore, the agents are mobile, meaning that they can leave their current "hosts," and move on to "infect" new, different processors.[2] We shall refer to these agents as *faults*, or simply *agents*. Given an assignment of the faults to the processors at any given round, we will draw from the virus terminology the term *cured* to refer to the processors that were infected in the previous round, but no longer. We will refer to the remaining processors not possessed by the agents as *nonfaulty*, or *uninfected.*

[2] A consequence of this assumption is that processors that come "back to life" are completely lost in the dark, and will require the help of the currently correct processors in order to reconstruct the code of the protocol being executed, as well as the global state of the execution. This assumption can be weakened as if, for example, some form of secure memory is available.

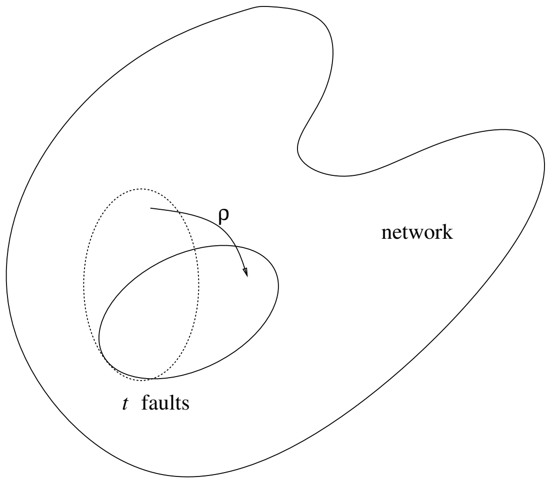

Fig. 1. Mobile fault model $\mathtt{MF}(\frac{t}{n-1}, \rho)$.

A fundamental parameter in such an environment is the *speed* v ($[v] = \frac{\#\ nodes}{\#\ rounds}$) with which the faults can travel in the network. For convenience, we will normalize to the speed required to move from one processor/node to another (for example, $v = \frac{1}{3}$ indicates that it takes a fault 3 rounds to "hop" to the next nonfaulty processor), and further, we will find it useful to talk about the inverse of this parameter, which we call ρ, informally, the *roaming* pace of the faults (in the previous example, $\rho = 3$). For simplicity, we only consider the case of $\rho \in \mathbf{N}$, the set of natural numbers. We will sometimes refer to the case of $\rho = 1$ as "full speed," the case of $\rho = 2$ as "half speed," etc.

We can now more formally define the *Mobile Fault* model as a family MF of models indexed by the tuple $(\frac{t}{n-1}, \rho)$. In $\mathtt{MF}(\frac{t}{n-1}, \rho)$ there are a total of n processors, the maximum number of mobile faults is t, and their roaming pace is ρ. (Granted that at first sight, this notation is not the prettiest. However, it does allow us to express in a handy manner that, for example, the number of faults is less than a third—of the size of the network—and that they move at half speed.) The situation is ilustrated in Figure 1. Notice that, in contrast with the traditional models of static faults (e.g., [13]) in which the only restriction is the bound on the maximum number of faults, there is no guarantee in the mobile-fault model that a processor that is initially correct will remain uninfected forever.

Intuitively, in this type of environment, as opposed to the traditional environment of faults that are static, things will have more of a continuous, longer execution flavor, since one cannot expect the (currently) noninfected processors to reach meaningful conclusions after a few rounds of communication. This is particularly ilustrated by the problem of achieving consistency in such an environment. We define the *Mobile-Fault* Byzantine Agreement problem (*MBA* for short) similarly to the *BA* problem of Section 1, together with the additional condition:

- **Consistency Maintenance:** Once Agreement is achieved among the currently uninfected processors, it is preserved among the (possibly different) uninfected processors.

This condition captures the contrast of MF with respect to the traditional model, in that if there is a moment (interval) in time in which a consistency state is achieved among the currently uninfected processors, then special efforts will have to be dedicated to *maintaining* such a state, by having the processors that participated in the decision "pass the token" to the processors that re-join the computation after the agents have left them. Albeit being more stringent, Consistency Maintenance resembles the correctness condition of *self-stabilizing* protocols (e.g., [5]). In our case, however, it is required that the correctness (consistency) state be maintained once it is reached, even if faults continue to occur.

3 Impossibility Results

We first establish the following, somewhat curious result:

Proposition 1. *In* $\mathtt{MF}(\cdot, 1)$*,* BA *can only be achieved if* $t = 0$*.*

The proof of this proposition extends the lower bound ideas of Dolev and Strong [7], which show that $t + 1$ rounds of communication are needed in the classical setting of static faults with a maximum of t faults. Roughly speaking, a class of execution histories is constructed that exhibit *serial faultiness*, in the sense that at each round a new processor misbehaves. In $\mathtt{MF}(\cdot, 1)$, this is achieved by having the single mobile agent "hopping" from one processor to the next. Same as in [7], a subhistory according to a processor, say p, is roughly defined as the view of (i.e., set of messages received by) p during the execution. (Note that p in these histories can be infected, and then cured. We assume that p is able to reconstruct whatever it missed while being possessed by the agent.) Now assume that *BA* can be achieved in k rounds (not necessarily $< n$). It can be shown by induction that it is possible to *hide* a processor at any round (a processor is *hidden* if it does not communicate at all from that round on), with the two resulting histories being equivalent, in the sense of two correct processors from both executions deciding on the same value. The end result is a big equivalence class of histories, regardless of the behavior of the source. We leave the details of the proof for the full version of the paper. We also note that Proposition 1 holds, *a fortiori*, for *MBA*. Now, the above result is fairly extreme: can't even tolerate one fault if it moves at full speed! One may wonder if the situation gets any better by slowing the faults down. This is what the following proposition—a generalization of Proposition 1—characterizes:

Proposition 2. *In* $\mathtt{MF}(\cdot, \rho)$*,* BA *is possible only if* $t < \rho$*.*

This is achieved by the t agents implementing serial faultiness through coordination. Note that one can now view the case of $\rho \geq t+1$ as corresponding to the classical setting. Given Propositions 1 and 2, the following condition constitutes one way of overcoming the above difficulties:

($\mathcal{I}$) At least one processor remains uncorrupted through the entire execution of the system.

We call the reader's attention to the diversity of contexts in which this condition reveals itself as necessary (e.g., failure detection [4], cryptographic protocols [3], etc.). Compare also with the "clean round" concept of [6]. In practice, condition $\mathcal{I}$ captures situations where not all places in the network are equally accessible to the faults. We also note that if the identity of the processor that cannot be corrupted by the faults is known beforehand to all processors, then the problem we are trying to solve becomes trivial. We will thus assume that this is not the case. We will sometimes refer to this processor as the "Incorruptible one" (`l'Inc`,[3] for short).

[**Note to the Byzantine-literate:** A first consequence of the $\mathcal{I}$ condition is that a direct application of the EIG (Exponential Information Gathering) protocol of Bar-Noy *et al.* [1] will not work. An essential requirement for this protocol to work is that the depth of the tree being constructed by the nonfaulty processors is small enough to guarantee a majority of nonfaulty processors at the leaf level of the tree, which is not the case here.]

Indeed, in contrast with the classical setting of static faults, solutions to *MBA* will now necessitate a time proportional to n, the size of the system, as opposed to t. Specifically:

Corollary 3. *Every* MBA *protocol requires* $\Omega(n)$ *rounds in its worst-case run.*

Finally, the lower bound on the number of faults of [13] ($n > 3t$) also holds for `MF` systems satisfying $\mathcal{I}$; we elaborate more on this point in Section 5. In the next section, we present protocols that rely on $\mathcal{I}$ to achieve *MBA* in $O(n)$ rounds.

4 Protocols for Mobile Byzantine Agreement

Given condition $\mathcal{I}$ and the observation from the previous section on the failure in `MF` of the classical full information protocol, a natural type of solution to look at is the *Phase King* (PK) paradigm of Berman and Garay [2], since its correctness relies on the eventual existence of a good processor (the "King"). The execution of a protocol adhering to this paradigm is divided into *phases*, each with a different king. In a phase, the following two steps take place:

1. Round(s) of universal exchange of messages among all processors; and

[3] From the Italian *l'Incorruttibile.*

2. all processors listen to the King.

The purpose of 1 is to eliminate possible discrepancies in the configuration of existing values, and discover the unique value, if such exists. In 2, processors that are not "overwhelmingly convinced" of the existence of a unique value, trust the king of the pase. The reader is referred to [2] for further details.

However, a direct use of the existing Phase King protocols for the static-fault models is not possible, since even if unanimity is achieved by the currently uninfected processors at a given round, that state may disappear as the agents move along. This brings us to the concept of *reconstruction* of information, or more specifically, to the use of the network as a collective memory device. The concept of network memory has previously been utilized by Ostrovsky and Yung [12].

Roughly, let a network object $\mathcal{O}$ be defined by the tuple <*round number*, *processor id*, *data*>. Then, given a big enough fraction of uninfected processors at all times, it is possible for a correct processor to store $\mathcal{O}$ in the network by sending copies to all processors, maintain it for a period of time (say, k rounds), and reconstruct it whenever needed. In this section we will be interested in storing objects for just one round. Specifically, we establish the following for $\texttt{MF}(\frac{1}{6}, 1)$:

Lemma 4 (Network Memory). *Let p be a noninfected processor at round r. Then in $\texttt{MF}(\frac{1}{6}, 1)$, every correct processor (cured or noninfected) can reconstruct at $r+1$ any object stored by p at r.*

Proof. At any given round, the number of correct processors (cured + uninfected) is at least $n - t \geq 5t + 1$. In round r, p stores m in the network by sending a copy of m to all processors. In round $r+1$, every uninfected processor echoes $\mathcal{O}$. The new object becomes:

$$\mathcal{O} = \begin{cases} < r, p, m > & \text{if received } m \text{ from at least} \\ & 4t+1 \text{ processors;} \\ < r, p, \perp > & \text{otherwise.} \end{cases}$$

Since in round $r+1$ at most t new processors get infected, all the remaining noninfected processors reconstruct m. □

Note that there is no guarantee that objects stored by infected processors will be univocally reconstructed, but this will be of no consequence to us. Lemma 4 can be generalized to reconstruct objects stored, say, k rounds back. Notice though that in MF, per the argument above, it is imperative that the copies be continuosly refreshed. We leave the complete formulation of the concept for the full version of the paper.

We now introduce *Mobile Phase King* (MPK), a protocol that is able to cope with mobile faults, for model $\texttt{MF}(\frac{1}{6}, 1)$. The protocol is shown in Figure 2. During the "universal exchange," all noninfected processors send their values (V) to all processors, and store the received values in vector MV. (We assume that the cured processors have the ability to receive messages. These would also include the protocol and the global state of execution—e.g., round number—which we

```
protocol Mpk;
        V := v_p;
        for r := 0 to ∞ do begin
                (* universal exchange *)
            send(V);
            receive(MV);
            C := the number of 1's in MV;
            V := C ≥ (1/2)n;
                (* reconstruction and king's broadcast *)
            send(MV)
            for each i do
                ECHO(i, ·) := MV received from i;
            if cured then
            Reconstruct(2r);
            k := (r mod n) + 1; (* k is the phase's king *)
            if k = p then
                send(V);
            let v be the value received from k;
(†)         if (v ∈ {0, 1} and C < n − 2t) then
                V := v;
        end;
```

Fig. 2. Protocol *Mobile Phase King* for $\mathtt{MF}(\frac{1}{6}, 1)$.

omit for clarity.) Each processor (both noninfected and cured) then computes the value that is most popular among the received values. In the "reconstruction and king's broadcast" exchange, processors send their MVs from the previous round, which they store in table *ECHO*. Processors that became cured in this round use that introspective ability to execute procedure Reconstruct, shown in Figure 3. [4] Finally, the phase's king broadcasts its value, which is adopted by both cured and noninfected processors if their previously computed popular value is not strong enough. Although the king's broadcast is presented separately, in practice these two send operations can be implemented as one, and we will thus consider a phase as composed of two communication rounds. Note that in Mpk the reconstruction procedure is only used in odd rounds.

We first make the following observation. Let s (for "silence") denote the number of cured processors in a given round, and i (for "infected") the number of newly infected processors.

Lemma 5. *If the roaming pace* $\rho = 1$*, then* $s + i \leq 2t$.

[4] The superscripts we use in procedure Reconstruct to indicate the time of the objects are not really necessary here since we are reconstructing objects from the previous round. We will continue to use them in order to remind us of our ability to reconstruct the past.

```
procedure RECONSTRUCT(r);
        for each j do
(‡)             if ∃ v ∈ {0,1} s.t. ECHO(·,j) = v
                at least n − 2t times then
                        V_j^r = v
                else
                        V_j^r = ⊥;
        C^r := the number of V_j^r = 1;
        V^r := C^r ≥ (1/2)n;
```

Fig. 3. Procedure RECONSTRUCT for $\mathtt{MF}(\frac{1}{6}, 1)$.

We now address the correctness of the protocol. We first define the following potential function:

$$\Phi(v) \stackrel{\text{def}}{=} \text{the number of correct processors } p \text{ for which } V_p = v.$$

Observe that the Consistency Maintenance condition of *MBA* enforces on functions Φ the property that whenever they reach the "ceiling" of $n-t$, they should irrevocably remain there. The next lemma shows that this is indeed the case for MPK.

Lemma 6 (Consistency Maintenance). *In* $\mathtt{MF}(\frac{1}{6}, 1)$*, if in a phase* $l \geq 0$ *of* MPK *there is a value* $v \in \{0,1\}$ *such that* $\Phi(v) \geq 5t+1$ *at the end of the phase, then* $\Phi(v) \geq 5t+1$ *at all phases* $r > l$.

Proof. Assume $\Phi(v) \geq 5t+1$ at the end of phase l. In the universal exchange of phase $l+1$, it follows from Lemma 5 that at least $4t+1$ processors send v, which is received by at least $5t+1$ correct (cured or noninfected) processors. All of these processors assign v to V. During the reconstruction and king's broadcast exchange, at least $4t+1$ processors remain uninfected, and therefore ignore the king's broadcast. Of the remaining t correct processors from the previous round, note that for each new infected processor there is a cured processor. Each cured processor p executes RECONSTRUCT, and it follows from Lemma 4 that each processor computes $V_p = v$ and $C_p \geq 4t+1$, thus also ignoring the king's broadcast. As a consequence, $\Phi(v) \geq 5t+1$ during phase $l+1$. Now the situation repeats itself, and we are done. □

Theorem 7. *In* $\mathtt{MF}(\frac{1}{6}, 1)$*, protocol* MPK *achieves Mobile Byzantine Agreement in* $O(n)$ *rounds of communication.*

Proof. We first consider Validity. The configuration is unanimous at the start with at least $5t+1$ noninfected processors holding value $v \in \{0,1\}$. We now argue that at the end of phase 0, $\Phi(v) \geq 5t+1$. First note that this trivially holds at the end of the universal exchange of phase 0. During the reconstruction and king's broadcast, at least $4t + 1$ processors remain uninfected, ignore the king's broadcast and stick to their initial values. Of the remaining t noninfected processors from the first round, for each new infected processor there is a cured processor. Each cured processor p executes RECONSTRUCT, and it follows from Lemma 4 that each cured processor computes $V_p = v$ and $C_p \geq 4t + 1$, thus also ignoring the king's broadcast. As a consequence, $\Phi(v) \geq 5t + 1$ at the end of phase 0, and Lemma 6 applies.

We now address Agreement. Let phase I be `l'Inc`'s turn. `l'Inc` cannot be touched by the faults, so it will be broadcasting a value $v_{\mathtt{l'Inc}} \in \{0, 1\}$ in phase I. From Lemma 5 it follows that at least $4t + 1$ noninfected values get sent in the universal exchange of phase I, all of which get reconstructed by the cured processors (if any) in the second exchange. As a consequence, all of these values are taken into account in the computation of C for at least $5t + 1$ correct (noninfected or cured) processors at the time they get to the last test of phase I. Two cases are now possible:

1. $C_p < n - 2t$ for every correct (cured or noninfected) p. Then all processors assign v, the value sent by `l'Inc`, to V_p at the end of the phase.
2. $C_p \geq n - 2t$ for at least some correct (cured or noninfected) p. This means that p received (or reconstructed) a value v_p at least $n - 2t$ times. Of those values, less than t originated from infected processors. As a consequence, `l'Inc` also received v_p at least $n - 3t > \frac{1}{2}n$ times, and therefore $v_{\mathtt{l'Inc}} = v_p$.

In either case, all correct processors have $V = v_{\mathtt{l'Inc}}$ at the end of phase I, and Lemma 6 applies.

Regarding the round complexity, observe that `l'Inc`'s turn becomes at round $2n$ at the latest. □

We now briefly consider model $\mathtt{MF}(\frac{1}{4}, 2)$. The following counterpart of Lemma 5 captures the main property of this model:

Lemma 8. *If the roaming pace $\rho = 2$, then $s + i \leq t$.*

The lemma captures the limitation of the faults when operating at half speed: Whenever noninfected processor p perceives silence from another processor q, p knows that it must be case that either q is faulty, or q is cured, and the agent that q was hosting is on the move to a new host, but not there yet. This higher price the mobile faults have to pay in this model can be utilized to design a more fault-tolerant network, as demonstrated by the following

Theorem 9. *In $\mathtt{MF}(\frac{1}{4}, 2)$, Mobile Byzantine Agreement is achievable in systems satisfying $\mathcal{I}$ in $O(n)$ rounds of communication.*

The proof uses a version of protocol MPK that is similar to the one presented in Figure 2, except for replacing lines (†) in Figure 2 and (‡) in Figure 3 by:

(†')   **if** ($v \in \{0,1\}$ **and** $C < n - t$) **then**

(‡')   **if** $\exists\ v \in \{0,1\}$ s.t. $ECHO(\cdot, j) = v$
at least $n - t$ times **then**

Albeit a bit more subtle than the analysis of Theorem 7, the proof of Theorem 9 is similar in flavor and we defer it to the full version of the paper.

5 Final Remarks

In this paper we have looked at the power of disruption of malicious mobile agents as a function of the speed with which they can traverse the network. We have defined models indexed by their number and traveling speed, and showed situations in which reaching agreement among the noninfected processors is possible in such models. We have mentioned in Section 3 that the bound of $n > 3t$ of [13] applies to these models as well. There are, however, strong signals that this bound can be improved.

We have insisted in this paper on performing computation deterministically. We note that if randomization is allowed, then the $\mathcal{I}$ condition can be dropped, and the shared coin tossing techniques of Feldman and Micali [9] can also be used in $\mathtt{MF}(\frac{1}{6}, 1)$ to achieve MBA in $O(1)$ expected time (modulo the fact that the coin tossing procedure itself takes several rounds of communication, but in any case...). Can this be improved?

Acknowledgements

The author would like to thank Amir Herzberg, Andrea Lenschow and Ken Perry for many helpful suggestions. Special thanks go to Amotz Bar-Noy, Yoram Moses and Moti Yung for much valuable input.

References

1. A. Bar-Noy, D. Dolev, C. Dwork and H.R. Strong, "Shifting gears: changing algorithms on the fly to expedite Byzantine Agreement," *Proc. 6th PODC*, pp. 42-51, August 1987.
2. P. Berman and J.A. Garay, "Asymptotically Optimal Distributed Consensus," *Proc. ICALP 89*, LNCS Vol. 372, pp. 80-94, July 1989.
3. R. Canetti and A. Herzberg, "Maintaining Security in the Presence of Transient Faults," *Proc. Advances in Cryptology—Crypto '94*, pp. 425-438, LNCS (839), Springer Verlag, August 1994.
4. T.D. Chandra and S. Toueg, "Unreliable Failure Detectors for Asynchronous Systems," *Proc. 10th PODC*, pp. 325-340, August 1991.
5. E.W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, 17(11), pp. 643-644, 1974.

6. C. Dwork and Y. Moses, "Knowledge and Common Knowledge in a in Byzantine Environment: Crash Failures," *Information and Computation*, Vol. 88, No. 2 (1990), pp. 156–186.
7. D. Dolev and H.R. Strong, "Polynomial Algorithms for Multiple Processor Agreement," *Proc. 14th STOC*, pp. 401-407, May 1982.
8. M. J. Fischer and N. A. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency," *Information Processing Letters*, Vol. 14, No. 4 (1982), pp. 183-186.
9. P. Feldman and S. Micali, "Optimal Algorithms for Byzantine Agreement," *Proc. 20th STOC*, pp. 148-161, May 1988.
10. M. Franklin and M. Yung, "Eavesdropping Games: A Graph-Theoretic Approach to Privacy in Distributed Systems," *Proc. 34th FOCS*, pp. 670-679, November 1993.
11. L. Lamport, R.E. Shostak and M. Pease, "The Byzantine Generals Problem," *ACM ToPLaS*, Vol. 4, No. 3 (1982), pp. 382-401.
12. R. Ostrovsky and M. Yung, "How to withstand mobile virus attacks," *Proc. 10th PODC*, pp. 51-59, 1991.
13. M. Pease, R. Shostak and L. Lamport, "Reaching Agreement in the Presence of Faults," *JACM*, Vol. 27, No. 2 (1980), pp. 121-169.
14. R. Reischuk, "A New Solution for the Byzantine Generals Problem," *Information and Control*, Vol. 64 (1985), pp. 23-42.