RZ 3202 (# 93248) 01/17/00
Computer Science/Mathematics 9 pages

Research Report

Optimistic Asynchronous Byzantine Agreement

Klaus Kursawe

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Optimistic Asynchronous Byzantine Agreement

Klaus Kursawe

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

Agreement problems are a fundamental building block for constructing reliable distributed systems. While robust and efficient protocols exist in the crash-failure setting, protocols resilient against a Byzantine adversary tend to have problems with either efficiency or security.

This paper proposes an optimistic approach to the Byzantine agreement problem, combining the efficiency of fully synchronous protocols with the robustness of asynchronous ones. If the system satisfies certain timing assumptions, and all parties are honest (the optimistic case), the protocol reaches optimal performance.
Otherwise, if the timing assumptions are violated or parties become corrupted the protocol invokes an asynchronous "fallback" protocol. Neither liveness nor security are violated in the transition, and the performance overhead is minimal.

Furthermore, no expensive cryptographic operations are needed in the optimistic case, unlike in most practical Byzantine agreement protocols. Thus, the optimistic approach gives a maximum of security while being more efficient than most (less secure) protocols.

Keywords: Asynchronous Consensus, Byzantine Faults, Optimism, Hybrid model.

1 Introduction

When modeling a reliable distributed system, one of the most important factors one has to consider lies in the underlying timing assumptions. On the one hand, a fully synchronous system allows for nice, simple and fast algorithms; on the other hand, timing assumptions are dangerous to rely on, especially in large scale, heterogeneous networks that might be influenced by an adversary. This is the case for the Internet, where the performance of the network and the connected computers has a high variety, and maliciously slowing down individual components is relatively easy. Asynchronous protocols are robust against timing misbehavior, and an adversary cannot prevent the protocol from succeeding by slowing down the network. This stability comes with a price, however; although most asynchronous agreement protocols do not suffer a significant slowdown in the presence of adversaries, they are slower than their synchronous counterparts if nothing goes wrong.

This paper presents an approach that combines the efficiency of the synchronous system with the reliability of the asynchronous one to solve the Byzantine agreement problem. In the optimistic case, i.e., if the protocol is not disturbed by adversaries and all timing assumptions hold, the protocol terminates as efficiently as any synchronous protocol; otherwise, the protocol is slowed down, but consensus is still reached.

The timing information required to distinguish between the optimistic and the pessimistic case is minimal and easily achievable in a practical system. Furthermore, the worst that can happen if all timing assumptions fail is that the protocol is permanently in the pessimistic case; neither liveness nor safety are endangered if the timing is completely controlled by a malicious adversary. This holds especially on the transition between the optimistic and the pessimistic case, i.e., if some parties act optimistically and others act pessimistically. not corrupt it, this synchronous protocols, as endangering security.

Background.
Byzantine agreement is the problem of several parties agreeing on a certain value, in spite of some of the participants being corrupt and actively trying to prevent the protocol from succeeding. We say a protocol achieves Byzantine agreement, if every party gets an input value and produces an output (decision) value such that the following properties hold:

Validity. If all honest parties start with input value ρ, then ρ is the only possible decision value.

Agreement. If one honest party decides ρ, then no honest party decides anything but ρ.

Termination. All honest parties eventually decide.

In their seminal paper, Fischer et al. prove that in an asynchronous environment, Byzantine agreement is impossible for a deterministic algorithm if even one party crashes [FLP85]. Two ways to solve it nevertheless have emerged:

Weakening the Model. Various models have been proposed for a system that behaves realistically, but that offers enough synchrony to solve the Byzantine agreement problem [DLS88, VA95, CF95]. For most recent implementations, the *failure detector* approach [CT91] has been chosen. Most practical protocols implemented in a failure-detector model deal only with crash failures, as failure detectors in this model are much easier to handle. Recently, several groups started moving the failure-detector approach into the Byzantine setting [Rei95, KMMS97, DS98].

Randomized Protocols. A randomized protocol can circumvent the impossibility proof by guaranteeing agreement with probability one [BO83, Rab83]. So far, most randomized protocols are either inefficient or require strong assumptions such as a periodically available dealer [CR93, BG93].
To the best of my knowledge, there are no implementations of a randomized Byzantine agreement protocol in the literature.

In the synchronous model, a combination of randomized and deterministic protocols has been studied in [GP90]. However, the protocol does not make use of the power a randomized protocol can offer in the asynchronous model, and depends on timing assumptions.

More recently, a combination of randomization and failure detectors has been proposed [AT96], but they still rely on the correct working of the failure detectors and work only in the crash-failure model. Pedone and Schiper apply optimism to the consensus problem [PS98], but their approach also works only in the crash-failure model and requires a reliable broadcast as well as a failure detector in the pessimistic case. To the best of my knowledge, no protocol has combined the synchronous and the asynchronous model, maintaining the strength of the asynchronous model (i.e., no timing assumptions at all) and the high optimistic performance of a timed protocol.

Contribution of this paper.
This paper introduces an optimistic pre-protocol that can be used in conjunction with an asynchronous consensus protocol. Basically, any protocol in the literature will do; the exact requirements are stated in section 2.2.

In the optimistic case, i.e., network timing assumptions hold and no adversary exists, consensus is reached in optimal time. Furthermore, no expensive computation is needed. In a normal system, the probability that a transaction falls into this case is very high, and for efficiency considerations this case is the important one.

In the pessimistic case, the optimistic pre-protocol invokes the "normal" Byzantine agreement protocol (the *fallback protocol*) without endangering any liveness or security assumptions. Furthermore, if few failures occur (i.e., there is a small number of traitors and network failures), the fallback protocol is invoked with all honest parties having the same initial value. This causes most Byzantine agreement protocols to terminate quite fast.

2 Model

2.1 Optimistic Asynchrony

This section describes the optimistic asynchronous timing model. Note that all timing assumptions spelled out here could be incorporated into a failure detector (see section 5). To allow a precise efficiency analysis and stress that the protocol does not need any hidden synchrony assumptions for anything but efficiency improvements, the presentation of the protocol will use concrete timeouts instead of an abstract failure detector.

The system consists of n parties $P_1, \ldots, P_n$, up to $t, t < \frac{n}{3}$ of which might be controlled by an adversary. The interconnecting network is an asynchronous point-to-point network. In our model, the network is completely controlled by the adversary, which has all freedom to delay, reorder or duplicate messages. The only restriction we put upon the adversary is that every message send between two honest parties has to be delivered eventually.

However, we have an optimistic assumption. In the optimistic case, no active adversary is attacking the system. If this case occurs, two bounds on network and processing speed hold most of the time:

- The maximum time allowed between P_i receiving a message, P_i generating the response and and other parties receiving it (assuming P_i sends them something) is Δ.
- The minimum allowed time between P_i receiving a message, P_i generating the response and the other parties receiving it is δ.

If the network performance has a high variety, the *timing uncertainty* $\frac{\Delta}{\delta}$ might be rather large. Therefore we define another timing constant for each run of the protocol. This timing model is similar to the one used in [ADLS94].

- The maximum time between P_i receiving a message and other parties receiving the response during one execution of the protocol that occurs during one run of the protocol is d.;

The value Δ can be freely chosen and might be adapted to the network performance. Tighter bounds decrease the time needed to detect "bad" behavior, but increases the probability that the protocol falls out of the optimistic case.

2.2 The Fallback Algorithm ABA

Let us assume that we have access to a protocol *ABA* that solves Asynchronous Byzantine agreement. As ABA is invoked only in the pessimistic case, its efficiency plays a secondary role; however, the more efficient the fallback protocol is, the more aggressive timeouts can be chosen for the optimistic protocol. In a separate paper [CKS00], a practical protocol is introduced that is suitable for an implementation of our approach.

Each party P_i takes three input parameters to execute ABA:

- A *transaction identifier* α, and
- an input value v_i.
- a verification value Σ that is used to verify if v_i is a legal input value.

With these inputs, ABA must satisfy the following conditions:

Weak Validity. If the only legal input-value for α, i.e., an input value that satisfies a consistency check with the verification value Σ, is ρ, then all honest parties will decide ρ.

Agreement. If one honest party decides ρ for transaction α, then no honest party decides anything other than ρ for α.

Termination. If every honest party starts ABA on transaction α, then every honest party eventually decides for α.

Because of the weak validity condition, we require a consistency check. This check guarantees that if one honest party could have optimistically decided ρ, then ρ is the only possible initial value for the fallback protocol. It can be implemented easily using digital signatures [RSA78] , as has been done in the protocol below. In this case, Σ is a collection of signatures. Using a reliable Broadcast primitive [Bra84], the signatures can be omitted at the cost of more communication, and Σ is not needed anymore. In this case, a party P_i only accepts P_j's input v_i after receiving all the votes that led to this input value.

Note that the consistency check becomes obsolete if we assume a stronger validity condition, i.e., that a value ρ is the only possible decision value if all *honest* parties have it as an initial value. Many protocols in the literature do not meet this validity condition, however.

Another problem is caused by the termination condition. If cryptographic assumptions are used (e.g., in the digital signature scheme), the adversary has to be computationally restricted. This model is incompatible with the notion of *eventual* decision, where the algorithm can take an arbitrarily long time to terminate.

Owing to space limitations, let us skip the discussion of a more consistent treatment of timing, which can be found in [CKS00].

Optimistic-ABA works on the same set of input values as ABA. Especially, if ABA is multi-valued, then so is Optimistic-ABA. The efficiency of ABA is only relevant in the presence of an adversary or network misbehavior.

The optimistic protocol does not need to put any further restrictions on the adversary, e.g., bounded computation power or static corruption, as long as the adversary cannot forge the consistency check. However, the Asynchronous Byzantine agreement protocol might require a more restricted adversary model.

2.3 Brief Discussion of the Validity Condition

In a hybrid synchronous-asynchronous model, the precise definition of the validity condition and the origin of the input values play an important role.

The main condition we want to put on validity to make it useful for practical purposes is the following:

Robustness. Suppose all parties are honest and the input values are generated in a way that ρ is the only possible decision according to the validity condition. Then, if up to t parties are corrupted, ρ still should be the only possible decision.

Non-triviality. Suppose all parties are honest and the only input value generated is ρ. Then ρ is the only possible decision value.

In the multivalued setting, one might want to add another condition, namely, if no party initially started with some value ρ, then ρ cannot be the decision value. This property is not straightforward; most multivalued Byzantine agreement protocols can decide on a value no party initially wanted [Rab83]. The current protocol satisfies the stronger validity condition, as long as ABA does.

Several models are conceivable that satisfy these conditions, all having their advantages and drawbacks.

Unauthenticated Source. If the values are generated from outside of the system, for example by a higher level application, the straightforward validity condition would read:

Validity. If all uncorrupted parties have initial value ρ, then ρ is the only possible decision.

This definition causes some problems with the timing, as we need a clean definition as to when a party receives its input value and therefore starts the protocol. The assumption that all parties start simultaneously is nice, but not very realistic. Otherwise, the difference in the starting time has to be specified precisely, as the synchronous part of the protocol may not do anything unless $t + 1$ parties received their input values. This draws additional, quite ugly timing assumptions into the protocol.

Byzantine Generals Agreement. In the Byzantine Generals agreement, the input values are issued by some source inside the system. The validity condition then reads:

Validity. If the source issues only input value ρ, then ρ is the only possible decision.

This is almost the same as the unauthenticated source, with two differences. First, as the source is a part of the system, its timing behavior is cleanly defined and therefore we know when to expect every party to start the protocol. Second, the source might be corrupt as well and, for example, send out only t inputs at all. Termination of the protocol is impossible, as no decision can be made on the basis of t input values, and the corruption of the source cannot always be detected in an asynchronous system.

Signing source. In the signing source model, the input values are generated and signed by some (external) source S. The validity condition then reads:

Validity. If any uncorrupted party decides ρ, then S issued a signature on ρ.

This is particularly nice to work with as no timing assumptions about the generation of the input values are necessary. Any party that starts the protocol can simply broadcast its input (or send it to some leader, who will then broadcast his decision), and all other parties can verify the signature and use this value as their own input without needing to be contacted by the source.

For the presentation of the protocol, we assume the easiest model. A source exists that starts all parties simultaneously with their initial values. Thus, we do not need to assume authentication on the input or synchronize the system first. The modifications for different models are straightforward.

3 Optimistic ABA

This section presents the optimistic protocol. Each party P_i, $1 \leq i \leq n$, gets an input value v_i and a corresponding transaction identifier α. The protocol outputs some decision value ρ or invokes the fallback protocol ABA.

A party P_i that decides optimistically may not entirely terminate the protocol, however. It is still possible that some other party could not decide and invokes the fallback protocol. Thus, even though the decision is final, the participation of P_i might be required to assist other parties to reach the same decision. It is important to note that if an honest party decides ρ in the optimistic part of the protocol, then the decision in ABA can only be ρ, as will be shown in the validity paragraph in the proof.

Basically, *Optimistic ABA* works in three steps:
Synchronize (not done here) (Init-Vote). A simple non-Byzantine agreement is invoked by every party broadcasting its preference and waiting for all other parties to answer.
Step 2 (Main-Vote and Check for decision). A simple vote suffices to find out if agreement exists; if this is the case, the protocol decides. It can not terminate completely, however, as it is still possible that another party requests a fallback into the pessimistic case.
Step 3 (Fall back to pessimistic protocol ABA). A party that gets either inconsistent or not enough answers broadcasts a complaint; on receiving such a complaint, an honest party starts the "pessimistic part" of the protocol, i.e., enters the fallback protocol ABA.
To legally enter ABA with an initial value ρ, a consistency check has to be made. This check proves that no party could have decided for $\rho' \neq \rho$ in step 2. This is done by redoing to last vote using digital signatures and supplying enough signatures that are not ρ'; if computation is expensive and communication is cheap, the signatures can be replaced by a reliable broadcast [Bra84].

We assume a signing function *sign(x)* that takes one parameter and creates a digital signature on it. The term $sign(x||y)$ denotes a signature on the concatenation of the messages x and y. Furthermore, Σ denotes a collection of signatures or a threshold signature.

Recall Δ is an upper bound on the message delivery time when the network behaves normally, i.e., in the optimistic case all messages are delivered in time Δ.

Protocol Optimistic ABA for party P_i asynchronous start, so a starting party must inform all others; away slightly cheaper if we assume everybody knows when to start

```
1. INIT-VOTE
   On getting an input value $v_i$
      send($\alpha$, init, $v_i$) to all parties;
      wait $\Delta$ time units or until $n$ votes are received;
      if $n$ votes were received
         $v_i \leftarrow$ simple majority of the received votes;

   /* Now, if no traitors exist and the timing conditions hold, on receiving this broadcast all parties
   are agreed. We have to find out whether that was the case. */

2. MAIN-VOTE
   let $w$ be the time passed since the init broadcast was sent;
   send ($\alpha$, main, $v_i$,) to all parties;
   wait $2\Delta - w$ time units or until $n$ votes are received;

   /* If less than n votes were received, we now waited exactly $2\Delta$ after issuing the first broadcast.
   This is enough time for $P_i$ to have received all main-votes. This value has to be increased in
   the case of an asynchronous start */

3. CHECK FOR DECISION
   if $n$ main-votes for some $\rho$ were received
      decide $\rho$;
```

```
else
    send ($\alpha$, pessimism, $v_i$, sign($\alpha||$main$||v_i$)) to all parties;

4. FALL BACK TO PESSIMISTIC PROTOCOL
On receiving a message ($\alpha$, pessimism, $v_i$, ...)
    if $P_i$ did not broadcast a pessimism message yet,
        send($\alpha$, pessimism, $v_i$, sign($\alpha||$main$||v_i$) ) to all parties as soon as $v_i$ is known;
    wait for $n-t$ signed main-votes;

    $v_i \leftarrow$ The value that got the simple majority from the received signed main-votes;
    if one value $\rho$ got at least $t+1$ votes
        $\Sigma_i \leftarrow t+1$ signatures on main-votes on $\rho$.;
    else
        $\Sigma_i \leftarrow n-t$ signatures, no $t+1$ of which sign the same vote;
/* The else case only occurs in multivalued settings. */
    enter ABA with parameters $(\alpha, v_i, \Sigma_i)$
/* The decided parties run ABA only for the others. Therefore, they will simply ignore the
decision made in ABA. */
```

3.1 Proof of Correctness

Theorem 1. *Given an asynchronous Byzantine agreement protocol ABA as defined above, Optimistic ABA solves Byzantine agreement.*

Proof. To prove the correctness of our protocol, we have to show that the protocol reaches validity, agreement and termination.

Validity. Suppose all honest parties start with ρ. All honest parties will main-vote either the majority of the received init-votes or their own initial value, both being ρ.

As no party can see more than t main-votes on something different from ρ, all honest parties will either immediately decide ρ (if the timing conditions hold and every party is honest), or enter ABA with initial value ρ as the only possible legal value (if something went wrong). By the validity condition of ABA, the decision will be ρ in either case.

Agreement. Suppose one honest party decides ρ before entering ABA. Then this party received n main-votes for ρ, $n-t$ of which have been issued by honest parties.

If all honest parties decide before entering ABA, all decide ρ and the protocol is finished. It still might happen that a corrupted party sends a pessimism message to some parties. This might cause them to continue and enter ABA, but the result is irrelevant and does not change their decision.

Suppose now that some honest party did not decide. This party will broadcast the pessimism message. All honest parties will, on receiving this message, send out a signed main-vote. As they initially voted ρ, this signed vote will also be ρ.

Every set of $n-t$ signed main-votes contains at least $t+1$ votes issued by honest parties. Thus if all honest parties voted ρ, the simple majority of any set of $n-t$ votes must be ρ. Therefore, no party can enter ABA with a legal vote for anything other than ρ, and by the validity condition of ABA all honest parties can only decide ρ.

In the case that no honest parties decides before entering ABA, the agreement condition in ABA enforces agreement of the Optimistic ABA protocol.

Termination. Either every honest party decides optimistically (and thus the protocol terminates without entering ABA), or every honest party enters ABA, in which case ABA's termination condition holds.

□

4 Efficiency

The efficiency of the protocol has to be examined both in the optimistic and in the pessimistic case, although the optimistic case is more important.

The Optimistic case.
Suppose all parties are honest and all timing assumptions hold. Recall that d denotes the maximum message delivery time that occurred during the run of the protocol, while δ and Δ are the lower and upper bound that are considered "normal" in the implementation of the protocol. The time $\min\{d, \Delta\}$ is also called a *synchronous round* for the corresponding run of the protocol (note that d might be larger than Δ, but this implies the system is in the pessimistic case). After time $\leq 2d$, every party received n main-votes and will decide. This is the theoretic optimum for synchronous Byzantine Agreement.

Besides authenticating the links (which is unavoidable), no cryptography is needed, so the computing effort for all parties is negligible.
The message complexity is n messages per party and round, resulting in $2n^2$ messages altogether. All messages have constant (short) size, so the bit-complexity is $O(n^2)$ as well.

So far, no Asynchronous Byzantine agreement protocol can reach agreement within three rounds of communication. The best failure-detector implementations [Rei95, KMMS97, DS98] need at least four rounds. Furthermore, they require every message to be signed, which might be a relatively expensive operation.

The Pessimistic case.
In the pessimistic case, the protocol basically performs like the fallback protocol ABA, with an overhead created by the optimistic part. The time needed to detect the pessimistic case and start ABA is two synchronous plus one asynchronous round. The number of messages is $3n^2$.
Besides, if one honest party decides, then all parties will enter ABA with the same initial value, thus speeding it up. If no party decides before entering ABA, then all honest parties will broadcast the "pessimism" message simultaneously, saving one asynchronous round.

5 Variations

Asynchronous Start.
The model assumed so far is convenient to work with, but a synchronous start of all parties is not realistic. We now describe the changes needed for an asynchronous start, assuming an authenticated source.
To make the algorithm work, the *init-vote* has to be modified; any party that is aware that the protocol is starting has to inform the other ones, and the timeouts have to be increased:

```
1. INIT-VOTE
      On getting an authenticated input value $v_j$ for the first time, broadcast it, including the
   authentication.
      wait $2\Delta$ time units **or** until receiving $n$ legal votes;
      **if** $n$ votes were received
         $v_i \leftarrow$ simple majority of the received votes;
```

We have to wait 2Δ time units, as some of the parties might only be aware that the protocol started after time Δ. This also implies that the time-out in the main-vote has to be increased to $3\Delta - w$. Besides this, we have a new validity condition; however, the modifications to the proof are straightforward.

If the parties do not start synchronously, we need up to three synchronous rounds to reach agreement. Furthermore, as an authenticated source is assumed, at least one (transferable) authentication has to be verified.

Using Failure Detectors.
Instead of assuming fixed timeouts, we can also implement the optimistic part of the protocol using failure detectors [CT91]. In this case, all timeouts are removed from the protocol; a party broadcasts the pessimism message as soon as a failure detector suspects some other party of being faulty prior to decision. In this model, nothing changes for the optimistic case; the efficiency of the pessimistic case becomes impossible to analyze however, as its performance depends on the speed of failure detection. Note that we need a very weak failure detector; no accuracy is needed at all (if all parties are suspected, we simply start the pessimistic protocol), and for completeness it suffices if one faulty party is once suspected by at least one honest party for a short time, thus not even weak completeness is required.

6 Conclusions

We have presented a protocol that can be combined with any asynchronous Byzantine agreement protocol to increase efficiency under optimistic conditions.

The protocol is optimal in the optimistic case, needing three synchronous rounds with synchronization and two rounds without; furthermore, the protocol adapts to the real network speed, so a varying network performance is compensated.

In the pessimistic case, the optimistic part of the protocol generates a small overhead; this is a small price to pay for a significant speedup in the optimistic case.

This protocol is an important step towards implementations of efficient, fully asynchronous Byzantine agreement primitives; especially, the optimistic protocol for Byzantine agreement is no less efficient than protocols that solve consensus in a crash-failure setting with failure detectors only; thus we can provide a high degree of additional security at no extra cost.

References

[ADLS94] H. Attiya, C. Dwork, N. Lynch, and L. Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. *Journal of the ACM*, 41(1):122–152, January 1994.

[AT96] M. K. Aguilera and S. Toueg. Randomization and failure detection: A hybrid approach to solve consensus. In Ö. Babaoglu and K. Marzullo, editors, *Distributed Algorithms, 10th International Workshop, WDAG '96*, volume 1151 of *Lecture Notes in Computer Science*, pages 29–39, Bologna, Italy, 9–11 October 1996. Springer.

[BG93] P. Berman and J. A. Garay. Randomized distributed agreement revisited. *23rd Int. Conf. on Fault-Tolerant Computing (FTCS-23)*, pages 412–419, 1993.

[BO83] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 27–30, Montreal, Quebec, Canada, 17–19 August 1983.

[Bra84] G. Bracha. An asynchronous $[(n-1)/3]$-resilient consensus protocol. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, pages 154–162, Vancouver, B.C., Canada, 27–29 August 1984.

[CF95] F. Cristian and C. Fetzer. Timed asynchronous systems: A formal model. Technical Report CSE95-454, UCSD, 1995.

[CKS00] C. Cachin, K. Kursawe, and V. Shoup. Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement using Cryptography. In *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2000. To appear.

[CR93] R. Canetti and T. Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *STOC93*, pages 42 – 51, 1993.

[CT91] T. D. Chandra and S. Toueg. Unreliable failure detectors for asynchronous systems (preliminary version). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 325–340, Montreal, Quebec, Canada, 19–21 August 1991.

[DLS88] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, April 1988.

[DS98] A. Doudou and A. Schiper. Muteness Detectors for Consensus with Byzantine Processes, 1998.

[FLP85] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[fST91] N. I. for Standards and Technology. Digital Signature Standard (DSS). *Federal Register*, 56(169), August 30 1991.

[GP90] O. Goldreich and E. Petrank. The best of both worlds: Guaranteeing termination in fast Byzantine Agreement protocols. *Information Proceeding Letters*, 36:45–49, October 1990.

[KMMS97] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. Solving consensus in a Byzantine environment using an unreliable fault detector. In *Proceedings of the International Conference on Principles of Distributed Systems (OPODIS)*, pages 61–75, December 1997.

[PS98] F. Pedone and A. Schiper. Optimistic atomic broadcast. In *Proceedings of the 12th International Symposium on Distributed Computing (DISC'98, formerly WDAG)*, September 1998.

[Rab83] M. O. Rabin. Randomized Byzantine generals. In *24th Annual Symposium on Foundations of Computer Science*, pages 403–409, Tucson, Arizona, 7–9 November 1983. IEEE.

[Rei95] M. K. Reiter. The Rampart Toolkit for Building High-Integrity Services. *Theory and Practice in Distributed Systems*, pages 99 – 110, 1995.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[VA95] P. Verissimo and C. Almeida. Quasi-synchronism: a step away from the traditional fault-tolerant real-time system models. *Winter 1995 Bulletin of the Technical Committee on Operating Systems and Application Environments (TCOS)*, 7 (4):35 – 39, 1995.