Optimistic Atomic Broadcast*

Fernando Pedone and André Schiper

Département d'Informatique
Ecole Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland

Abstract. This paper presents an Optimistic Atomic Broadcast algorithm (OPT-ABcast) that exploits the spontaneous total order message reception property experienced in local area networks, in order to allow fast delivery of messages. The OPT-ABcast algorithm is based on the Optimistic Consensus problem (OPT-Consensus) that allows processes to decide optimistically or conservatively. A process optimistically decides if it knows that the spontaneous total order message reception property holds, otherwise it decides conservatively. We evaluate the efficiency of the OPT-ABcast and the OPT-Consensus algorithms using the notion of latency degree.

1 Introduction

Atomic Broadcast is a useful abstraction for the development of fault tolerant distributed applications. Understanding the conditions under which Atomic Broadcast is solvable is an important theoretical issue that has been investigated extensively. Solving Atomic Broadcast efficiently is also an important and highly relevant pragmatic issue. We consider in this paper an *Optimistic Atomic Broadcast* algorithm (called hereafter OPT-ABcast), derived from the Chandra-Toueg Atomic Broadcast algorithm [4] (called hereafter CT-ABcast), which allows processes, in certain cases, to deliver messages *fast*. The idea of our OPT-ABcast algorithm stems from the observation that, with high probability, messages broadcast in a local area network are received totally ordered. We call this property *spontaneous total order message reception*.[1] Our algorithm exploits this observation: whenever the spontaneous total order reception property holds, the OPT-ABcast algorithm delivers messages fast.

Similarly to Chandra-Toueg Atomic Broadcast algorithm, our OPT-ABcast algorithm is also based on a reduction to the Consensus problem.

* Research supported by the EPFL-ETHZ DRAGON project and OFES under contract number 95.0830, as part of the ESPRIT BROADCAST-WG (number 22455).

[1] Spontaneous total order reception occurs for example with very high probability when network broadcast or IP-multicast are used.

However, the classical Consensus problem is not the right abstraction to use in the context of the OPT-ABcast algorithm. This led us to the specification of the *Optimistic Consensus* problem (called hereafter OPT-Consensus). In the OPT-Consensus problem a process can take up to two decisions: (1) an optimistic decision, and (2) a conservative solution. The two decisions are not necessarily the same. A process can decide only optimistically, or both optimistically and conservatively, or only conservatively. The details of the specification of the OPT-Consensus problem are given in Section 5. In our OPT-ABcast algorithm, the consensus decisions are optimistic whenever the spontaneous total order reception property holds, and the run is failure free and suspicion free. The efficiency of our OPT-ABcast algorithm is related to an optimistic decision of the underlying OPT-Consensus problem. We evaluate the *efficiency* of our OPT-ABcast algorithm using the notion of *latency degree* introduced in [13].

The rest of the paper is structured as follows. Section 2 describes some related work. Section 3 is devoted to the system model and to the definition of latency degree. In Section 4 we give an overview of the OPT-ABcast algorithm and of the OPT-Consensus problem. In Section 5 we specify the OPT-Consensus problem and give an algorithm that solves the problem. Section 6 describes the OPT-ABcast algorithm. We conclude in Section 7. Due to space limitations, all proofs have been omitted. They can be found in [12].

2 Related Work

The paper is at the intersection of two issues: (1) Atomic Broadcast algorithms, and (2) *optimistic* algorithms.

The literature on Atomic Broadcast algorithms is abundant (e.g., [1], [3], [4], [5], [6], [8], [11], [14]). However, the multitude of different models (synchronous, asynchronous, etc.) and assumptions needed to prove the correctness of the algorithms renders any fair comparison difficult. We base our solution on the Atomic Broadcast algorithm as presented in [4] because it provides a theoretical framework that permits to develop the correctness proofs under assumptions that are realistic in practical system (i.e., unreliable failure detectors).

Optimistic algorithms have been widely studied in transaction concurrency control ([9, 2]). To our knowledge, there has been no attempt, prior to this paper, to introduce optimism in the context of agreement algorithms. The closest to the idea presented in the paper is [7], where the authors reduce the Atomic Commitment problem to Consensus and, in order to have a fast decision, exploit the following property of the Consensus problem: if every process starts Consensus with the same value v, then the decision is v. This paper presents a more general idea, and does not require that all the initial values be equal. Moreover, we have here the trade-off of typical optimistic algorithms: if the optimistic assumption is met, there is a benefit (in efficiency), but if the optimistic assumption is not met, there is a loss (in efficiency).

3 System Model and Definitions

3.1 System Model

We consider an asynchronous system composed of n processes $\Pi = \{p_1, \ldots, p_n\}$. Processes communicate by message passing. A process can only fail by crashing (i.e., we do not consider Byzantine failures). Processes are connected through reliable channels, defined by the two primitives *send*(m) and *receive*(m).

We assume causal order delivery for a subset of the messages.[2] We distinguish two types of messages, denoted by $\mathcal{M}$ and $\mathcal{M}_{CO}$. Causal order delivery is ensured only for messages of type $\mathcal{M}_{CO}$. Causal order is defined by the *send*(m) primitive and the *CO-deliver*(m) primitive. If m_1 and m_2 are two messages of type $\mathcal{M}_{CO}$ sent to the same destination process p_i, and *send*$(m_1) \rightarrow$ *send*(m_2), then *CO-deliver*$(m_1) \rightarrow$ *CO-deliver*(m_2), where $\rightarrow$ is the happened before relation [10]. In order to simplify the notation, we rename the *CO-deliver*(m) primitive to *receive*(m). So, if m_1 and m_2 are two messages of type $\mathcal{M}_{CO}$ sent to the same destination process p_i, and *send*$(m_1) \rightarrow$ *send*(m_2), we have *receive*$(m_1) \rightarrow$ *receive*(m_2).

Each process p has access to a local failure detector module that provides (possibly incorrect) information about the processes that are suspected to have crashed. A failure detector may make mistakes, that is, it may suspect a process that has not failed or never suspect a process that has failed. Failure detectors have been classified according to the mistakes they can make [4]. We consider in the paper the class of Eventually Strong failure detectors, denoted by $\diamond\mathcal{S}$.

[2] Causal order simplifies the presentation of the OPT-ABcast algorithm. However, OPT-ABcast can also be implemented with an algorithm that does not need causal order.

3.2 Reliable Broadcast and Atomic Broadcast

We assume the existence of a *Reliable Broadcast* primitive, defined by *R-broadcast*(m) and *R-deliver*(m). Reliable Broadcast satisfies the following properties [4]: (i) if a correct process R-broadcasts a message m, then it eventually R-delivers m *(validity)*, (ii) if a correct process R-delivers a message m, then all correct processes eventually R-deliver m *(agreement)*, and (iii) for every message m, every process R-delivers m at most once, and only if m was previously R-broadcast by *sender*(m) *(uniform integrity)*. We assume that the execution of *R-broadcast*(m) results in the execution of *send*(m) to every process p.

Atomic Broadcast is defined by *A-broadcast*(m) and *A-deliver*(m). In addition to the properties of Reliable Broadcast, Atomic Broadcast satisfies the *total order* property [4]: if two correct processes p and q A-deliver two messages m and m', then p A-delivers m before m' if and only if q A-delivers m before m'.

3.3 Latency Degree

The latency degree has been introduced in [13] as a measure of the efficiency of a distributed algorithm. It has been defined based on a slight variation of Lamport's clocks [10]:

- a *send* event and a *local* event on a process p_i do not modify p_i's local clock;
- let $ts(send(m))$ be the timestamp of the $send(m)$ event, and $ts(m)$ the timestamp carried by message m: $ts(m) \stackrel{\text{def}}{=} ts(send(m)) + 1$;
- the timestamp of $receive(m)$ on a process p_i is the maximum between $ts(m)$ and p_i's current clock value.

With this definition, the *latency* of a run $\mathcal{R}$ of an algorithm $\mathcal{A}_{\mathcal{P}}$ solving an agreement problem $\mathcal{P}$ is defined as the largest timestamp of all *decide* events (at most one per process) of run $\mathcal{R}$. The *latency degree* of algorithm $\mathcal{A}_{\mathcal{P}}$ is defined as the minimum latency over all the runs that can be generated by the algorithm $\mathcal{A}_{\mathcal{P}}$. The minimal latency is obtained in failure free and suspicion free runs.

We consider in the paper the Atomic Broadcast problem. To define the latency degree of Atomic Broadcast we assume that every process p_i A-broadcasts only one single message, and that A-broadcast is the first event of process p_i.

4 Overview of the Results

4.1 OPT-Consensus Problem

Similarly to the Chandra-Toueg Atomic Broadcast algorithm, our OPT-ABcast algorithm is based on a reduction to the Consensus problem [4]. However, the classical Consensus problem is not adequate here: we need a Consensus that, under certain (problem dependent) conditions, allows processes to decide *fast*, even if without the guarantee that all processes decide on the same value. We call this Consensus the OPT-Consensus problem. We formalise the OPT-Consensus problem by introducing the notion of *optimistic* and *conservative* decisions, where the optimistic decision is the fast one. A process p_i can decide optimistically if it knows that a certain condition is satisfied (evaluating this condition requires the knowledge about the initial values of all processes; however, it does not necessarily require that all the initial values be identical). If a process p_i cannot decide optimistically, then p_i decides conservatively. The optimistic and the conservative decisions can be different, but are related.

The details of the specification of the OPT-Consensus problem are given in Section 5.1.[3] In Section 5.2, we show that the Chandra-Toueg consensus algorithm using $\diamond\mathcal{S}$ can, with minor modifications, evaluate the condition that allows the fast decision and solve the OPT-Consensus problem.

4.2 OPT-ABcast Algorithm

The Chandra-Toueg Atomic Broadcast algorithm is based on a Reliable Broadcast and on a sequence of consensus problems, where the initial value of each consensus is a *set* of messages (i.e., the decision of each consensus is a *set* of messages). Our OPT-ABcast algorithm is based on a Reliable Broadcast and on a sequence of OPT-Consensus problems, where the initial value of each consensus is a *sequence* of messages (i.e., the decision of each OPT-Consensus problem is a *sequence* of messages). The initial value of process p_i for the OPT-Consensus depends on the order in which the messages that are Atomically Broadcast are received (more precisely, R-delivered) by p_i.

Consider the k-th OPT-Consensus problem, and let v_i^k be the initial value of p_i. The optimistic decision is possible whenever all the sequences

[3] The OPT-Consensus problem is defined with the optimistic Atomic Broadcast algorithm in mind, but we believe that it can be used to solve other problems, where assumptions about the values proposed by the processes can be made. For lack of space we do not further develop this statement in the paper.

v_i^k $(1 \leq i \leq n)$ have a non empty common prefix. If a process p_i decides optimistically, the optimistic decision is the longest common prefix. If a process p_j decides conservatively, the conservative decision is an initial value. So, an optimistic decision is a *prefix* of a conservative decision. In other words, the fact that an optimistic decision is different from a conservative decision does not lead to the violation of the properties of Atomic Broadcast.

The performance of our OPT-ABcast algorithm is directly related to an optimistic decision of the OPT-Consensus algorithm, which depends on the order of reception (more precisely, the order of R-delivery) of the messages that are Atomically Broadcast. We show in Section 6.4 that, in failure free and suspicion free runs, if messages are spontaneously R-delivered in the same order at all processes, then all the OPT-Consensus algorithms terminate with an optimistic decision. Furthermore, if the spontaneous ordering property does not hold for a while, then as soon as the property holds again, all OPT-Consensus problems again terminate with an optimistic decision.

4.3 Latency Degree of the OPT-ABcast Algorithm

The latency degree of the OPT-ABcast algorithm is given by $L_{OPT\_AB} = L_{RB} + L_{OPT\_C}$, where L_{RB} is the latency degree of the Reliable Broadcast algorithm and L_{OPT_C} is the latency degree of the OPT-Consensus algorithm.

The OPT-Consensus algorithm given in Section 5.2 is such that the latency of an optimistic decision is equal to 2, and the latency of a conservative decision is at least equal to 4. Therefore, the latency degree of our OPT-ABcast algorithm is equal to $L_{RB}+2$. By comparison, the Chandra-Toueg Consensus algorithm, using the failure detector $\diamond\mathcal{S}$, has latency degree of 4, but a trivial optimisation leads to a latency degree of 3. This results in a latency degree of $L_{RB} + 3$ for the CT-ABcast algorithm.

The latency degree of the OPT-ABcast algorithm can even be reduced by considering another algorithm for solving OPT-Consensus. If, instead of deriving the OPT-Consensus algorithm from the Chandra-Toueg consensus algorithm, we derive it from the Early Consensus algorithm [13], we get an OPT-Consensus algorithm such that the latency of an optimistic decision is equal to 1, and the latency of a conservative decision is at least equal to 3. So the latency degree of the overall OPT-ABcast algorithm is equal to $L_{RB} + 1$.

5 The Optimistic Consensus Problem

5.1 Problem Definition

The Optimistic Consensus problem is defined in terms of the primitive *propose*(v, f_{opt}) and the primitives *decide*(OPT, v) and *decide*(CSV, v). The *propose* primitive has two parameters: an initial value v (the initial value of p_i is denoted by v_i), and a function $f_{opt} : \mathcal{V}^n \rightarrow \mathcal{V} \cup \perp$, where $v \in \mathcal{V}$ and $\perp \notin \mathcal{V}$ (the function f_{opt} is the same *for all processes*). The primitive *decide*(OPT, v) corresponds to an "optimistic" decision, and the primitive *decide*(CSV, v) corresponds to a "conservative" decision. A process can decide both optimistically *and* conservatively, and the two decisions can be different. The Optimistic Consensus problem is specified as follows.

- *Termination.* Every correct process eventually decides optimistically or conservatively. If a process conservatively decides, then all correct processes also eventually conservatively decide.
- *Uniform Integrity.* No process can optimistically decide more than once and no process can conservatively decide more than once. Moreover, no process can decide optimistically after having decided conservatively.
- *Uniform Validity.* If a process p conservatively decides v then v was proposed by some process. If a process optimistically decides v then $f_{opt}(v_1, \ldots, v_n) \neq \perp$ and $v = f_{opt}(v_1, \ldots, v_n)$.
- *Uniform Conservative Agreement.* No two processes conservatively decide differently.

The *Termination* condition requires that if some process decides conservatively, then every correct process also decides conservatively. This is not true for the optimistic decision: some processes might decide optimistically and others not. So, some processes may decide twice (once optimistically and once conservatively), while other processes decide only once (conservatively). The above specification allows, for example, the following runs: (1) all the correct processes decide optimistically, and no process decides conservatively, or (2) some processes decide optimistically and all correct processes decide conservatively, or (3) no process decides optimistically and all correct processes decide conservatively.

Furthermore, it follows from the *Uniform Validity* condition that no two processes optimistically decide differently. This is because an optimistic decision is computed by the function f_{opt} over *all* proposed values. Thus we have the following result.

Lemma 1. (Uniform Optimistic Agreement). *No two processes optimistically decide differently.*

5.2 The OPT-Consensus Algorithm

Algorithm 1 solves the OPT-Consensus problem using any Eventual Strong failure detector $\mathcal{D} \in \diamond\mathcal{S}$ and assuming a majority of correct processes. The algorithm is very similar to Chandra-Toueg's Consensus algorithm [4]: the mainly difference is in Phase 2 of the algorithm, and in the task responsible for the decision. However, the advantage of the OPT-Consensus algorithm is that it has a latency degree of 2, whereas the Chandra-Toueg Consensus algorithm has a latency degree of 3. A run with latency 2 happens whenever all processes can decide optimistically and no process decides conservatively. The OPT-Consensus algorithm works as follows:

1. In Phase 2 of the first round, the coordinator waits either (1) for estimates from a majority of participants if some process is suspected, or (2) for the estimates from all participants if no participant is suspected.
2. If all estimates are received, the coordinator applies the function f_{opt} over them. Let tmp_p be the value returned by f_{opt}. If $tmp_p \neq \bot$, then tmp_p is the optimistic decision, and the coordinator R-broadcasts the message $(OPT, -, -, tmp_p, decide)$.
3. A process that R-delivers the message $(OPT, -, -, estimate, decide)$ optimistically decides *estimate*.
4. If not all the estimates are received by the coordinator (item 2 above), the execution proceeds as in Chandra-Toueg's algorithm. In this case, the decision is conservative, and the latency is at least equal to 4.

The messages in the OPT-Consensus algorithm that are issued using the *send* primitive are of type $\mathcal{M}$ (see Section 3.1). The messages in the OPT-Consensus algorithm that are issued using the *R-Broadcast* primitive (*R-Broadcast* of the optimistic or conservative decisions) are of type $\mathcal{M}_{CO}$. We come back to this issue in Section 6.2 when discussing the reduction of Atomic Broadcast to OPT-Consensus.

5.3 Proof of Correctness

Lemma 2. *No process remains blocked forever in the* ***wait*** *statement of Phase 2 in the OPT-Consensus algorithm.*

Theorem 1. *If $f < n/2$, the OPT-Consensus algorithm solves the OPT-Consensus problem using a failure detector of class $\diamond\mathcal{S}$.*

Algorithm 1 OPT-consensus algorithm

```
procedure propose(v_p, f_opt)
  estimate_p ← v_p
  OPTstate_p ← undecided
  CSVstate_p ← undecided
  r_p ← 0
  ts_p ← 0
  while OPTstate_p = undecided and CSVstate_p = undecided do
    r_p ← r_p + 1
    c_p ← (r_p mod n) + 1
    send (p, r_p, estimate_p, ts_p) to c_p                                    {Phase 1}
    if p = c_p then                                                           {Phase 2}
      wait until for { ⌈(n + 1)/2⌉ q: received(q, r_p, estimate_q, ts_q) from q and
                     { [ ∀q: received(q, r_p, estimate_q, ts_q) from q or D_p ≠ ∅ ]
      msgs_p[r_p] ← {(q, r_p, estimate_q, ts_q) | p received (q, r_p, estimate_q, ts_q) from q}
      if ∀q, (q, r_p, estimate_q, ts_q) ∈ msgs_p[r_p] and ts_q = 0 then
        tmp_p ← f(all estimates)
        if tmp_p ≠⊥ then
          R-broadcast(OPT, p, r_p, tmp_p, decide)
          return from procedure
      t ← largest ts_q such that (q, r_p, estimate_q, ts_q) ∈ msgs_p[r_p]
      estimate_p ← select one estimate_q such that (q, r_p, estimate_q, t) ∈ msgs_p[r_p]
      send (p, r_p, estimate_p) to all
    wait until [received (c_p, r_p, estimate_{c_p}) from c_p or c_p ∈ D_p]     {Phase 3}
    if [received (c_p, r_p, estimate_{c_p}) from c_p] then
      estimate_p ← estimate_{c_p}
      ts_p ← r_p
      send (p, r_p, ack) to c_p
    else
      send (p, r_p, nack) to c_p
    if p = c_p then                                                           {Phase 4}
      wait until [for ⌈(n+1)/2⌉ processes q : received (q, r_p, ack) or (q, r_p, nack)]
      if [for ⌈(n+1)/2⌉ processes q : received (q, r_p, ack)] then
        R-broadcast(CSV, p, r_p, estimate_p, decide)
  return from procedure

repeat                                                               {Decision Task}
  when R-deliver(decision_type, q, r_q, estimate_q, decide)
    if decision_type = OPT then
      if OPTstate_p = undecided and CSVstate_p = undecided then
        decide(decision_type, estimate_q)
        OPTstate_p ← decided
    else
      if CSVstate_p = undecided then
        decide(CSV, estimate_q)
        CSVstate_p ← decided
until decision = CSV
```

5.4 Latency Degree of the OPT-Consensus Algorithm

Proposition 1. *If all processes optimistically decide, the latency of the run of the OPT-Consensus algorithm is equal to 2. If the processes conservatively decide, the latency is greater than or equal to 4.*

Proposition 2. *The OPT-Consensus algorithm has latency degree equal to 2.*

6 The Optimistic Atomic Broadcast Algorithm

6.1 Additional Notation

Our OPT-ABcast algorithm handles sequences of messages, and not sets of messages as in the Chandra-Toueg algorithm [4]. We define some terminology that will be used in Section 6.2.

A sequence s of messages is denoted by $s = <m_1, m_2, \ldots>$. We define the operators $\oplus$ and $\ominus$ for concatenation and decomposition of sequences. Let s_i and s_j be two sequences of messages: $s_i \oplus s_j$ is the sequence of all the messages of s_i followed by the sequence of all the messages of s_j that are not in s_i. $s_i \ominus s_j$ is the sequence of all the messages in s_i that are not in s_j. The $prefix$ function applied to a set of sequences returns the longest common sequence that is a prefix of all the sequences, or the empty sequence denoted by ϵ.

For example, if $s_i = <m_1, m_2, m_3>$ and $s_j = <m_1, m_2, m_4>$, then $s_i \oplus s_j = <m_1, m_2, m_3, m_4>$, $s_i \ominus s_j = <m_3>$, and the function $prefix(s_i, s_j) = <m_1, m_2>$.

6.2 The OPT-ABcast Algorithm

We give now the reduction of Atomic Broadcast to OPT-Consensus (see Algorithm 2). The reduction has similarities with the reduction proposed by Chandra and Toueg [4], however with some additional complexity to cope with sequences of messages and optimistic and conservative decisions. When process p_i A-broadcasts some message m, then p_i executes *R-broadcast*(m). Process p_i starts the OPT-Consensus algorithm with its initial value s_i equal to the sequence of messages that it has R-delivered (but not yet A-delivered). An optimistic decision is a non-empty prefix over all the sequences s_i $(1 \leq i \leq n)$ of messages proposed by all the processes p_i. The function f_{opt} that defines an optimistic decision is as follows:

$$f_{opt}(s_1, \ldots, s_n) = \begin{cases} prefix(s_1, \ldots, s_n) & \text{if } prefix(s_1, \ldots, s_n) \neq \epsilon \\ \perp & \text{otherwise} \end{cases}$$

A conservative decision is a sequence s_i of messages proposed by some process p_i. Multiple OPT-Consensus executions are disambiguated by processes by tagging all the messages pertaining to the k-th OPT-Consensus execution with the counter k.

All tasks in Algorithm 2 execute concurrently. The algorithm works as follows for any process p.

1. When p wants to A-broadcast a message m, p executes $R\text{-}broadcast(m)$ (Task 1). Message m is R-delivered by Task 2 and included in the sequence $R\_delivered_p$.
2. If the sequence $R\_delivered_p \ominus A\_delivered_p$ is not empty, p executes $propose(k_p, seq, f_{opt})$ in Task 3 with $seq = R\_delivered_p \ominus A\_delivered_p$. Before proceeding to the next OPT-Consensus $k_p + 1$, p waits until a decision (optimistic and/or conservative) for OPT-Consensus k_p is known.
3. Task 4 waits for *decide*. However a process does not know whether there will be one or two decisions for any given execution of the OPT-Consensus algorithm. Therefore if a process executes an optimistic *decide* it also has to expect a conservative one. However, conservative decisions are not mandatory, that is, a process never knows until when it has to wait for a second decision.
   This is handled by Task 4 as follows. Whenever p decides optimistically for OPT-Consensus number k_p, the variable $prev_p^{CSV}$ is set to k_p. Task 4 then waits either a $decide(prev_p^{CSV}, -, -)$ (conservative decision of the previous OPT-Consensus $k_p - 1$) or a $decide(k_p, -, -)$ (optimistic or conservative decision of the current OPT-Consensus k_p). Causal order delivery of messages of type $\mathcal{M}_{CO}$ (see Section 3.1 and 5.2) ensures that a conservative decision of OPT-Consensus $k_p - 1$, if any, occurs before any decision of OPT-Consensus k_p.
   Once a sequence of messages $msgSeq_p^k$ is decided, process p A-delivers sequentially the messages in $msgSeq_p^k$ that it has not A-delivered yet.

6.3 Proof of Correctness

Lemma 3. *For all $l > k$, causal order delivery of the messages of type $\mathcal{M}_{CO}$ ensures that no process executes $decide(l, -, -)$ before executing $decide(k, -, -)$.*

For the following Lemmata we define $A\_delivered_p^k$ as the sequence of messages A-delivered by process p and decided in OPT-Consensus k.

$A\_delivered_p^k$ may contain messages included in optimistic and/or conservative decisions. The sequence $A\_delivered_p^1 \oplus \ldots \oplus A\_delivered_p^k$ contains all the messages delivered by process p until, and including, the messages decided in OPT-Consensus k.

Algorithm 2 OPT-ABcast algorithm

Initialisation:
  $R\_delivered_p \leftarrow \epsilon$
  $A\_delivered_p \leftarrow \epsilon$
  $k_p \leftarrow 0$
  $prev_p^{CSV} \leftarrow 0$

To execute *A-broadcast*(m): {***Task 1***}

  R-broadcast(m)

A-deliver$(-)$ occurs as follows:

  when *R-deliver*(m) {***Task 2***}
    $R\_delivered_p \leftarrow R\_delivered_p \oplus \{m\}$

  when $R\_delivered_p \ominus A\_delivered_p \neq \epsilon$ {***Task 3***}
    $k_p \leftarrow k_p + 1$
    $A\_undelivered_p \leftarrow R\_delivered_p \ominus A\_delivered_p$
    $decision_p^{k_p} \leftarrow unknown$
    $propose(k_p, A\_undelivered_p, f_{opt})$
    wait until $decision_p^{k_p} \neq unknown$

  when $decide(k_p, decision\_type, msgSeq_p^{k_p})$ **or** {***Task 4***}
      $decide(prev_p^{CSV}, decision\_type, msgSeq_p^{prev_p^{CSV}})$
    let $\#c$ be equal to k_p or $prev_p^{CSV}$, according to the decide executed above
    $A\_deliver_p^{\#c} \leftarrow msgSeq_p^{\#c} \ominus A\_delivered_p$
    atomically deliver messages in $A\_deliver_p^{\#c}$ following the order in $A\_deliver_p^{\#c}$
    $A\_delivered_p \leftarrow A\_delivered_p \oplus A\_deliver_p^{\#c}$
    if $decision\_type = OPT$ **then** $prev_p^{CSV} \leftarrow \#c$ {*prepare a possible csv decision*}
    $decision_p^{\#c} \leftarrow known$

Lemma 4. *For any two processes p and q, and all $k \geq 1$, if*

(a) p and q decide only optimistically for OPT-Consensus k, or
(b) p and q decide only conservatively for OPT-Consensus k, or

(c) p decides optimistically and conservatively, and q decides only conservatively for OPT-Consensus k, or
(d) p and q decide optimistically and conservatively for OPT-Consensus k

then $A\_delivered_q^k = A\_delivered_p^k$.

Lemma 5. *For any two correct processes p and q, and all* $k \geq 1$, *if p A-delivers the sequence of messages* $A\_delivered_p^k$, *then q eventually A-delivers the sequence of messages* $A\_delivered_q^k$, *and* $A\_delivered_q^k = A\_delivered_p^k$.

Theorem 2. *Algorithm 2 reduces Atomic Broadcast to OPT-Consensus.*

6.4 Latency Degree of the OPT-ABcast Algorithm

Proposition 3. *Let* L_{RB} *be the latency degree of Reliable Broadcast. If every process decides only optimistically, the run of the OPT-ABcast algorithm based on our OPT-Consensus algorithm has latency equal to* $L_{RB}+2$. *If the decision is conservative the latency of the run of the OPT-ABcast algorithm is greater than or equal to* $L_{RB}+4$.

Proposition 4. *The OPT-ABcast algorithm based on OPT-Consensus has latency degree equal to* $L_{RB}+2$.

It remains to discuss under what conditions the latency degree of $L_{RB}+2$ is obtained. Proposition 5 states that, if every message is R-delivered in the same order at every process, then every execution of the OPT-Consensus algorithm has only an optimistic decision (i.e., the latency degree is $L_{RB}+2$). This holds even if processes do not start each OPT-Consensus with the same sequence of messages as initial value (e.g., p starts OPT-Consensus k with the sequence $<m_1, m_2, m_3>$ and q starts OPT-Consensus k with the $<m_1, m_2>$).

Proposition 5. *Consider a failure free and suspicion free run, and assume that all the processes R-deliver the messages that are A-Broadcast in the same order. Then for each execution of the OPT-Consensus algorithm, every process decides only optimistically.*

Proposition 6 states that temporary violation of the spontaneous total ordered R-delivery of messages does not prevent future optimistic decisions: after the messages are again spontaneously R-delivered in total order, the decisions are again optimistic.

Proposition 6. *Consider a failure free and suspicion free run, and assume that after R-delivering the k-th message, all processes R-deliver all messages in total order. Then, after all the first k messages are A-delivered, for each execution of the OPT-Consensus every process decides only optimistically.*

7 Conclusion

This work originated from the pragmatic observation that messages broadcast in a local area network are, with high probability, spontaneously totally ordered. Exploiting this observation to develop a *fast* Atomic Broadcast algorithm turned out to be technically more difficult then we initially expected. The difficulty has been overcome by the introduction of the OPT-Consensus problem, with two (possibly different) decisions: an optimistic decision and a conservative decision. The OPT-Consensus problem abstracts the property exploited by our OPT-ABcast algorithm.

The efficiency of the OPT-ABcast algorithm has been quantified using the notion of latency degree, introduced in [13]. The latency degree of the OPT-ABcast algorithm has been shown to be $L_{RB}+2$, where L_{RB} is the latency degree of Reliable Broadcast. This result has to be compared with the latency of the Chandra-Toueg Atomic Broadcast algorithm, which is equal to $L_{RB}+3$.

Finally, to the best of our knowledge, the OPT-ABcast algorithm is the first agreement algorithm to exploit an *optimistic* condition: if the conditions are met the efficiency of the algorithm is improved, if the conditions are not met the efficiency of the algorithm deteriorates. We believe that this opens interesting perspectives for revisiting or improving other agreement algorithms.

References

1. Y. Amir, L. Moser, P. Melliar-Smith, D. Agarwal, and P. Ciarfella. Fast Message Ordering and Membership Using a Logical Token-Passing Ring. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 551–560, May 1993.
2. P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987.
3. K. Birman, A. Schiper, and P. Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.
4. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.
5. J. M. Chang and N. Maxemchuck. Reliable Broadcast Protocols. *ACM Transactions on Computer Systems*, 2(3):251–273, August 1984.

6. H. Garcia-Molina and A. Spauster. Ordered and Reliable Multicast Communication. *ACM Transactions on Computer Systems*, 9(3):242–271, August 1991.
7. R. Guerraoui, M. Larrea, and A. Schiper. Reducing the cost for non-blocking in atomic commitment. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 692–697, May 1996.
8. P. Jalote. Efficient ordered broadcasting in reliable csma/cd networks. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, pages 112–119, May 1998.
9. H. T. Kung and J. T. Robinson. On optimistic methods for concurrency control. *ACM Transactions on Database Systems*, 6(2):213–226, June 1981.
10. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.
11. S. W. Luan and V. D. Gligor. A Fault-Tolerant Protocol for Atomic Broadcast. *IEEE Trans. Parallel & Distributed Syst.*, 1(3):271–285, July 90.
12. F. Pedone and A. Schiper. Optimistic atomic broadcast. Technical Report TR-98/280, EPFL, Computer Science Department, 1998.
13. A. Schiper. Early consensus in an asynchronous system with a weak failure detector. *Distributed Computing*, 10(3):149–157, 1997.
14. U. Wilhelm and A. Schiper. A Hierarchy of Totally Ordered Multicasts. In *Proceedings of the 14th IEEE Symp. on Reliable Distributed Systems*, pages 106–115, September 1995.