2011 30th IEEE International Symposium on Reliable Distributed Systems

On the Reduction of Atomic Broadcast to Consensus with Byzantine Faults

Zarko Milosevic, Martin Hutle, André Schiper
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Email: firstname.lastname@epfl.ch

***Abstract*—We investigate the reduction of atomic broadcast to consensus in systems with Byzantine faults. Among the several definitions of Byzantine consensus that differ only by their validity property, we identify those equivalent to atomic broadcast. Finally, we give the first deterministic atomic broadcast reduction with a constant time complexity with respect to consensus.**

***Keywords*-Atomic broadcast; Consensus; Byzantine faults; Reduction**

I. Introduction

The relation of consensus and atomic broadcast, including the reduction of atomic broadcast to consensus, is well understood in the case of crash faults [1]. On the contrary, little is known on the relation between atomic broadcast and the consensus problem in the context of Byzantine faults. Although there is a huge amount of literature on the implementation of atomic broadcast under Byzantine faults, and some of them even use some sort of agreement as a building block, most of these papers do not—strictly speaking—reduce atomic broadcast to consensus.[1] In fact, we are aware only of three *reductions* of atomic broadcast to consensus with Byzantine faults [2], [3], [4], and in our opinion this work does not fully clarify the relation of consensus and atomic broadcast.

It is easy to observe that the standard reduction of atomic broadcast to consensus with benign faults does not work with Byzantine faults. One can also observe that there exist several definitions of consensus with Byzantine faults (which differ in the validity property), and it is not clear at all which one should be considered for an atomic broadcast reduction. In addition, the relation between these definitions of consensus and atomic broadcast is only partially understood.

Contribution. The first question addressed in the paper is the relation between these various definitions of consensus for Byzantine faults and the atomic broadcast problem. We show that only some of the consensus validity properties in the literature lead to consensus problems that are equivalent to atomic broadcast. We also show that consensus with *weak unanimity* [5] is not sufficient to solve atomic broadcast, while with *strong validity* [6] consensus is harder than atomic broadcast.

After that, we give a reduction of atomic broadcast to *range validity* consensus [7].[2] The reduction has a constant time complexity with respect to consensus, which is not the case for the reduction in [2], [3], [4] [3]. As discussed in Section IV-D, this means that the time for adelivering a message is bounded by $k_1\tau^C + k_2\delta$, where τ^C is the maximum execution time of some consensus algorithm, δ the maximum communication delay, and k_1, k_2 two constants.

After the reduction of atomic broadcast to range validity consensus, we give a reduction of range validity consensus to binary consensus that has constant time complexity (with respect to binary consensus). This implies that atomic broadcast can be reduced to binary consensus with constant time complexity (with respect to binary consensus).

About constant time reductions. The key feature of range validity consensus that allows us to achieve constant time reduction with respect to consensus is the fact that it constrains the decision value even if not all initial values of correct processes are the same. This is not the case with the *strong unanimity* [5] consensus, which was used in the reduction in [3]. The reduction in [3] requires, as written by the authors, a "deterministic and fair rule" for choosing which messages to propose to consensus such that eventually all correct processes propose the same set of messages. However, [3] gives no examples of such a rule. Moreover, relying on such a mechanism clearly leads to a reduction algorithm that does not have constant time complexity with respect to consensus. The

[1] We discuss these papers in the related work section.

[2] We denote with *range validity* the validity property used in [7], which requires the decision value to be in the range of initial values of correct processes.

[3] [4] gives two reductions, one deterministic and the other randomized. The randomized reduction has a constant expected time complexity with respect to binary consensus.

1060-9857/11 $26.00 © 2011 IEEE
DOI 10.1109/SRDS.2011.36

reduction in [2] relies on *abortable consensus* [8], [2],[4] which restricts a decision value to the default value $\perp$ if not all initial values of correct processes are the same. The consequence is that the reduction algorithm only makes progress when the initial values of correct processes are the same. This leads to an algorithm that does not have constant time complexity with respect to consensus.

Structure of the paper: The system model and problem definitions are given in Section II. We then compare the different definitions of consensus and atomic broadcast in Section III. In Section IV, the reduction of atomic broadcast to range validity consensus is presented. Solving range validity consensus is addressed in Section V. The reduction of atomic broadcast to binary consensus with constant time complexity with respect to consensus is discussed in Section VI. Section VII is devoted to related work, and Section VIII concludes the paper.

II. Definitions

We consider a system of n processes $\Pi = \{1, \ldots, n\}$ that are connected with asynchronous reliable links. At most t processes may fail in an arbitrary way (Byzantine faults). A correct process is the one that is not Byzantine. Links satisfy the integrity property, i.e., a message that is received from a process was also sent by this process. We do not use signatures.

A. Reliable Unique Broadcast

In our reduction we use *reliable unique broadcast* [9],[5] a broadcast primitive slightly different from *reliable broadcast* [10]. As shown in [9], the primitive can be implemented in an asynchronous system with reliable links. Reliable unique broadcast is defined in terms of two primitives, rubcast and rubdeliver. A process p that wants to broadcast a message m invokes $\mathsf{rubcast}_p(k, m)$ with a tag k it has not used before. This message is then delivered by executing $\mathsf{rubdeliver}_q(k, m, p)$ at process q. Reliable unique broadcast fulfills the following properties:

- *Validity:* If a correct process p invokes $\mathsf{rubcast}_p(k, m)$, and this is the only invocation of p for index k, then p eventually executes $\mathsf{rubdeliver}_p(k, m, p)$.
- *Agreement:* For any two correct processes p and q, if p executes $\mathsf{rubdeliver}_p(k, m, s)$, then q eventually executes $\mathsf{rubdeliver}_q(k, m, s)$.
- *Integrity:* For any index k and every process q, every correct process p executes $\mathsf{rubdeliver}_p(k, m, q)$ at most once. Moreover, if q is correct, then q previously invoked $\mathsf{rubcast}_q(k, m)$.

Observe that $\mathsf{rubcast}_q(k, m)$ provides stronger guarantees than $\mathsf{rbcast}_q(m')$ (reliable broadcast) with $m' = \langle k, m \rangle$. With the latter, a correct process could deliver both $m' = \langle k, m \rangle$ and $m'' = \langle k, \overline{m} \rangle$. With the former, because of the integrity property, a correct process cannot deliver (k, m) and $(k, \overline{m})$.

B. Atomic Broadcast

Atomic broadcast is defined in terms of two primitives, abcast and an adeliver. A process p that wants to atomic broadcast a message m invokes $\mathsf{abcast}_p(m)$. This message m is then delivered by executing $\mathsf{adeliver}_q(m, p)$ at process q. We assume that all messages are unique. Atomic broadcast fulfills the following properties [10]:

- *Validity:* If a correct process p invokes $\mathsf{abcast}_p(m)$, then p eventually executes $\mathsf{adeliver}_p(m, p)$.
- *Agreement:* For any two correct processes p and q, and any process s, if p executes $\mathsf{adeliver}_p(m, s)$, then q eventually executes $\mathsf{adeliver}_q(m, s)$.
- *Integrity:* For any message m and any process s, every correct process p executes $\mathsf{adeliver}_p(m, s)$ at most once. Moreover, if s is correct, then it previously invoked $\mathsf{abcast}_s(m)$.
- *Total order:* If some correct process p executes $\mathsf{adeliver}_p(m, s)$ before $\mathsf{adeliver}_p(m', s')$ for any two processes s and s', then every correct process q executes $\mathsf{adeliver}_q(m', s')$ only after it has executed $\mathsf{adeliver}_q(m, s)$.

C. Consensus

In the consensus problem, every process has an initial value from a set $\mathcal{V}$ and has eventually to irrevocably decide on a value from $\mathcal{V}$. The problem is defined by an agreement, a termination, and a validity property. Validity is discussed in the next section.

- *Agreement:* No two correct processes decide differently.
- *Termination:* All correct processes eventually decide.

The interface to consensus uses two primitives, $\mathsf{propose}_p(i, v)$ to denote the start of a consensus instance i at p with the initial value v, and $\mathsf{decide}_p(i, d)$ to denote termination at p of consensus instance i with the decision value d.

[4] In [8] abortable validity is called *non-intrusion* validity, and the corresponding consensus *intrusion-tolerant Byzantine consensus*. In [2] it is called *multi-valued consensus validity*.

[5] In [9] the primitive is called *consistent unique broadcast*.

III. Atomic broadcast reduction and the validity property of consensus

Consensus is defined by three properties, *validity*, *agreement* and *termination*. With benign faults, agreement can be uniform or non uniform, termination requires that all correct processes decide, and the standard validity states that the decision must be the initial value (taken from a set $\mathcal{V}$) of some process. Other validity properties have also been considered [11]. However, it is well understood that the standard validity property above allows the reduction of atomic broadcast to consensus in an asynchronous system with benign faults.

With Byzantine faults, more definitions of consensus have been proposed, which differ only by the validity property (with Byzantine faults, only non uniform agreement makes sense). Depending on the validity property, the definition of consensus can be weaker, stronger or equivalent to atomic broadcast, as we now show. We consider the following definitions:

- *Strong validity* ([12], [6]): If a correct process decides on v, then v is the initial value of some *correct* process.[6]
- *Weak unanimity* [5]: If all correct processes have the same initial value, and no process is faulty, then this is only possible decision value.
- *Strong unanimity* [5]: If all correct processes have the same initial value, then this is the only possible decision value.
- *Abortable validity* ([2], [8]):[7]
  (a) If all correct processes have the same initial value, then this is the only possible decision value.
  (b) If a correct process decides v, then v is the initial value of some correct process, or $v = \perp$.

Abortable validity considers a special decision value $\perp$. This special value can be decided if not all correct processes have the same initial value.

The last validity property considered is introduced (without name) in [7]. We call it *range validity* since the decision value is in the range of initial values of correct processes. Range validity requires the domain $\mathcal{V}$ of initial values to be totally ordered:

- *Range validity* [7]: There exist two distinct correct processes p_1, p_2, so that every decision value d_q of a correct process q is between the initial values of p_1 and p_2 ($\mathcal{C}$ is the set of correct processes, v_p denotes the initial value of process p):

$$\exists p_1, p_2 \in \mathcal{C} : \ \forall q \in \mathcal{C} : \ v_{p_1} \leq d_q \leq v_{p_2}.$$

We have the following relations between these validity properties: strong validity implies range validity and abortable validity; both range validity and abortable validity imply strong unanimity; strong unanimity implies weak unanimity.

The special case $|\mathcal{V}| = 2$ is called *binary consensus*. For $|\mathcal{V}| = 2$, strong validity, range validity, strong unanimity and abortable validity are equivalent. In our proofs, when referring to binary consensus, we will assume the "strong unanimity" formulation.

In the rest of the section, we compare the difficulty of these various consensus problems and atomic broadcast.

A. Atomic broadcast is harder than weak unanimity consensus

We show that with $n > 3t$, atomic broadcast is harder than weak unanimity consensus.[8] To do so, we first show that whenever atomic broadcast is solvable, range validity consensus is also solvable.

Lemma 1. *If $n > 3t$, then range validity consensus can be reduced to atomic broadcast.*

Proof: The reduction is as follows. First, every process atomic broadcasts its initial value, and waits until $n - t$ messages from different processes are delivered. When a process has adelivered $n - t$ messages from different processes, it orders the values of these messages in ascending order and decides on the value at $(t + 1)$st position.

Termination. Since atomic broadcast guarantees the delivery of all messages from correct processes, and there are at least $n - t$ correct processes, all correct processes decide.

Agreement. Since atomic broadcast ensures that all messages are delivered in total order, the set of the $n-t$ first messages from different processes is the same at all correct processes, and thus all correct processes decide on the same value.

Range Validity. After values from the first $n - t$ messages from different processes are ordered, we have

[6]Strong validity can be seen as the counterpart, in the context of Byzantine faults, of the standard validity property with benign faults: *If a correct process decides v, then v is the initial value of some process.* Indeed, since it does not make sense to refer to the initial value of a Byzantine process, the definition becomes: *If a correct process decides v, then v is the initial value of some correct process*.

[7]In [2] this is called *multi-valued consensus validity*, and is defined by three validity properties MVC1 to MVC3. The MVC1 validity property corresponds to abortable validity (a). The MVC2 validity property can be rephrased as follows: *If a correct process decides v, then v was proposed by some correct process.* Together with MVC3, we get abortable validity (b).

[8]Consensus with validity property *X* will simply be called *"X consensus"* or *"X validity consensus"*.

two cases to consider: (1) the t first values are from faulty processes, (2) the t first values contain at least one value from a correct process. It is easy to see that in both cases range validity holds. ■

Since range validity implies strong unanimity validity, we have the following corollary:

Corollary 1. *If $n > 3t$, then strong unanimity consensus can be reduced to atomic broadcast.*

Now we show that for $n > 3t$, the solvability of weak unanimity consensus does not imply solvability of atomic broadcast. We start with a lemma.

Lemma 2. *Strong unanimity consensus cannot be reduced to weak unanimity consensus.*

Proof: Assume by contradiction that such a reduction T exists. Then T, together with weak unanimity consensus, solves strong unanimity consensus, and in particular in all runs where at least one process is faulty and weak unanimity consensus always decides a fixed value v_0 (independently of the initial values of correct processes). Now consider a modified algorithm T' that uses always v_0 as a fixed value instead of the output of weak unanimity consensus. Clearly, for all runs of T' there is a run of T that is indistinguishable to the correct processes, and thus they decide as in T. This means that T' is a deterministic algorithm that solves strong unanimity consensus in an asynchronous systems with at least one faulty process. A contradiction with the FLP impossibility result [13]. ■

We can now prove the above claim.

Proposition 1. *If $n > 3t$, atomic broadcast cannot be reduced to weak unanimity consensus.*

Proof: Assume by contradiction that atomic broadcast can be reduced to weak unanimity consensus. If $n > 3t$, by Corollary 1, consensus with strong unanimity can be reduced to atomic broadcast. Thus, strong unanimity consensus is reducible to consensus with weak unanimity. A contradiction with Lemma 2. ■

From Corollary 1 and Proposition 1 it follows that with $n > 3t$ atomic broadcast is harder than weak unanimity consensus.

B. Strong validity consensus is harder than atomic broadcast

We show that strong validity consensus with more than three values ($|\mathcal{V}| > 3$) is harder than atomic broadcast. To do so, we will use the result from [2] that atomic broadcast can be reduced to binary consensus.

Proposition 2. *If $n > 3t$, then atomic broadcast can be reduced to binary consensus.*

Proof: The result is proven in [2]. ■

Proposition 3. *If $n > 3t$, then atomic broadcast can be reduced to strong validity consensus.*

Proof: If $n > 3t$ then by Proposition 2 atomic broadcast can be reduced to binary consensus, that is strong validity consensus with $|V| = 2$. ■

We now show that the opposite does not hold in general.

Proposition 4. *For $|\mathcal{V}| > 3$, in general strong validity consensus cannot be reduced to atomic broadcast.*

Proof: The proof is by contradiction. Consider a synchronous system with $n = 4$ and $t = 1$. We first show that in this setting, atomic broadcast is solvable. Indeed, in this setting binary consensus is solvable. If binary consensus is solvable, by Proposition 2 atomic broadcast is also solvable.

Let us assume by contradiction that there is a reduction of strong validity consensus to atomic broadcast in an asynchronous system. With the result of the previous paragraph, this means strong validity consensus is solvable in a synchronous system with $n = 4$ and $t = 1$. This contradicts the results from [6], where it is shown that solving strong validity consensus in a synchronous system requires $n > \max(3, |\mathcal{V}|)t$. ■

C. Consensus problems equivalent to atomic broadcast

The other consensus problems introduced, namely strong unanimity consensus, abortable validity consensus and range validity consensus are all equivalent to atomic broadcast. This result follows from three propositions:

Proposition 5. *If $n > 3t$, atomic broadcast and range validity consensus are equivalent.*

Proof: From Proposition 2 it follows that if $n > 3t$ atomic broadcast can be reduced to binary consensus, that is range validity consensus with $|V| = 2$. On the other hand, by Lemma 1, if $n > 3t$ range validity consensus can be reduced to atomic broadcast. ■

Proposition 6. *If $n > 3t$, atomic broadcast and abortable validity consensus are equivalent.*

Proof: This result is proven in [2]. ■

Proposition 7. *If $n > 3t$, atomic broadcast and strong unanimity consensus are equivalent.*

Proof: From Corollary 1 it follows that if $n > 3t$ strong unanimity consensus can be reduced to atomic

Algorithm 1 Atomic broadcast by reduction to range-validity consensus (code of process p)

```
1:  r ← 0
2:  adelivered ← ∅
3:  rubdelivered ← ∅
4:  lsn ← 0
5:  decided_sn[1...n] ← [0...0]

6:  upon abcast(m) do
7:      lsn ← lsn + 1
8:      rubcast(lsn, m)

9:  upon rubdeliver(i, m, q) do
10:     rubdelivered ← rubdelivered ∪ {⟨m, i, q⟩}

11: loop
12:     wait until rubdelivered − adelivered ≠ ∅
13:     r ← r + 1
14:     for π = 1 to n do
15:         rdel_sn[π] ← largest ℓ, so that
            ∀j, 1 ≤ j ≤ ℓ : ⟨−, j, π⟩ ∈ rubdelivered
16:         v[π] ← min(rdel_sn[π] − decided_sn[π], max(V))

17:         range-propose((r − 1) · n + π, v[π])
18:     for π = 1 to n do
19:         wait until
            range-decide((r − 1) · n + π, decision[π])
20:         if decision[π] > 0 then
21:             for ℓ = decided_sn[π] + 1 to
                decided_sn[π] + decision[π] do
22:                 wait until
                    ∃msg = ⟨m, ℓ, π⟩ : msg ∈ rubdelivered
23:                 if ⟨m, −, π⟩ ∉ adelivered then
24:                     adeliver(m, π)
25:                     adelivered ← adelivered ∪ msg
26:             decided_sn[π] ← decided_sn[π] + decision[π]
```

broadcast. On the other hand, by Proposition 2 if $n > 3t$ then atomic broadcast can be reduced to binary consensus, that is strong unanimity consensus with $|V| = 2$. This shows that atomic broadcast and strong unanimity consensus are equivalent. ■

IV. Reducing atomic broadcast to consensus with range validity

In this section we give a reduction of atomic broadcast to consensus with range validity, the first constant time reduction to consensus for Byzantine faults. For range validity consensus we assume that $\mathcal{V}$ is a countable set, and without loss of generalization, we assume that $\mathcal{V}$ is the set $\{0, \ldots, |\mathcal{V}| - 1\}$ if $\mathcal{V}$ is finite, or $\mathbb{N}_0$ if $\mathcal{V}$ is infinite.

A. Reduction algorithm

The reduction is given as Algorithm 1. It uses *reliable unique broadcast*. In order to abcast a message, the message is first rubcast together with a local sequence number lsn (line 8). When this message is rubdelivered it is stored in the variable $rubdelivered$ for further processing (line 10).

The main loop (lines 14-26) is executed when there are messages that are rubdelivered but not yet adelivered. The code between the wait statements and the "upon" blocks are executed atomically. We call each iteration of the main loop a "round". In the sequel, we denote with x_p the value of a variable x at process p, and for any round $r > 0$ we denote with x_p^r the value of x_p at the beginning of round r (a round begins at line 14). In every round r, Algorithm 1 executes n instances of range validity consensus in parallel, see lines 14 and 18. Consensus instance $(r-1)\cdot n+\pi$ is used to agree which messages abcast by process π are adelivered in round r. A process p proposes in consensus instance $(r-1)\cdot n+\pi$ the difference between $rdel\_sn_p[\pi]$ and $decided\_sn_p[\pi]$, where

- $rdel\_sn_p[\pi]$ contains the highest sequence number, such that all messages from π with a smaller sequence number are rubdelivered at p, and
- $decided\_sn_p[\pi]$ contains the sequence number of the last message from π that is adelivered by p.

The decision value of consensus instance $(r-1)\cdot n+\pi$ determines the number of messages from π that will be adelivered in round r. Finally, all messages from π are adelivered by all correct processes in the order of their sequence numbers lsn.

Remark: Executing range validity consensus n times in parallel, as done in Algorithm 1, has a high message complexity. This cost can easily be reduced to the message complexity of one instance of range validity, with larger messages. The solution consists of executing *one* instance of consensus on a "vector of n elements" instead of "n instances of range consensus" in parallel. The consensus then operates simultaneously on each entry of the vector, providing an independent element-wise range consensus semantics for each of the vector's elements.

B. Proof of Algorithm 1

Lemma 3. *For all correct processes p and q, and all $r > 0$, $decided\_sn_p^r = decided\_sn_q^r$.*

Proof: The proof is by induction on r. For $r = 1$, at all processes $decided\_sn^r = [0 \ldots 0]$, and thus the lemma holds. Assume now that the lemma is true for $r - 1$. When the correct processes decide consensus instances $(r-2)n+1$ to $(r-2)n+n$ in round $r-1$, because of the agreement property of range validity consensus they all decide the same values. Thus, at the end of round $r-1$ (at line 26), all processes update $decided\_sn$

consistently. Therefore, $decided\_sn_p^r = decided\_sn_q^r$ and the lemma holds also for r. ■

Lemma 4. *For all correct processes p and q, and all $r > 0$, $adelivered_p^r = adelivered_q^r$.*

Proof: The proof is by induction on r. For $r = 1$, at all processes $adelivered^r = \emptyset$, and thus the lemma holds. Assume now that the lemma is true for $r - 1$. By the agreement property of consensus, all correct processes terminate consensus instance $(r-2)n + \pi$ for $\pi = 1..n$ in round $r - 1$ with the same decision $decision(\pi)$ ($\star$). The value of $decided\_sn$ in round $r - 1$ at line 21 is equal to $decided\_sn^{r-1}$, therefore by Lemma 3, $decided\_sn$ has the same value at line 21 in round $r - 1$ at all correct processes ($\star\star$). By ($\star$) and ($\star\star$), for every instance $\pi = 1..n$ of loop at line 18 all correct processes will execute the same number of iterations at line 21. For each iteration they will consider the same message due to the agreement property of reliable unique broadcast. By induction assumption we have $adelivered_p^{r-1} = adelivered_q^{r-1}$, and since the value of $adelivered$ at line 23 at round $r - 1$ is equal to $adelivered^{r-1}$, the condition at p and q at line 23 of round $r - 1$ will evaluate to the same value. Therefore, p and q will update $adelivered$ in round $r - 1$ at line 25 to the same value, so the lemma holds also for r. ■

Lemma 5. *If $\mathsf{adeliver}_p(m, s)$ occurs at some correct process p in round r, then $\mathsf{adeliver}_q(m, s)$ also occurs in round r at any correct process q.*

Proof: Proof is in [14]. ■

Lemma 6 (Validity)**.** *If a correct process p invokes $\mathsf{abcast}_p(m)$, then p eventually executes $\mathsf{adeliver}_p(m, p)$.*

Proof: Assume by contradiction that message m abcast by p is the first message abcast by p such that (m, p) is not adelivered by some correct process q. When process p invokes $\mathsf{abcast}_p(m)$, the local sequence number lsn is incremented and the message $\langle lsn+1, m\rangle$ is rubcast. Messages that are abcast by p before m are rubcast with sequence number smaller than $lsn + 1$ and by assumption all these messages are adelivered. By line 26 (update of $decided\_sn_q[p]$), $\exists r$ s.t. eventually $decided\_sn_q^r[p] = lsn$. From Lemma 3 follows that at all correct processes we have $decided\_sn[p]^r = lsn$.

By the termination property of reliable unique broadcast, all messages $\langle i, -\rangle$ with $i = 1..lsn + 1$ are eventually rubdelivered at all correct processes. Because $\langle lsn + 1, m\rangle$ was rubcast by a correct process, by the agreement and integrity property of reliable unique broadcast, all correct processes eventually rubdeliver $\langle lsn + 1, m\rangle$, and have $\langle m, lsn + 1, p\rangle$ in $rubdelivered$. Therefore, in round r all correct processes propose in the consensus instance $(r-1)\cdot n+p$ a value larger or equal to 1 (see lines 16-17).[9] By the range validity property of consensus, $decision_q[p]$ is therefore larger or equal to 1 and p executes $\mathsf{adeliver}(m, p)$. A contradiction. ■

Lemma 7 (Agreement)**.** *For any two correct processes p and q, and any process s, if p executes $\mathsf{adeliver}_p(m, s)$, then q eventually executes $\mathsf{adeliver}_q(m, s)$.*

Proof: Follows directly from Lemma 5. ■

Lemma 8 (Integrity)**.** *For any message m and any process s, every correct process p executes $\mathsf{adeliver}_p(m, s)$ at most once. Moreover, if s is correct, then it previously invoked $\mathsf{abcast}_s(m)$.*

Proof: The first part of the claim follows directly from line 23. For the second part, a message is adelivered at a correct process p only if p rubdelivered $\langle i, m, q\rangle$ before. If the sender q is correct, by the integrity property of reliable unique broadcast, q previously invoked $\mathsf{rubcast}_q(-, m)$. By lines 6 and 8, q previously invoked $\mathsf{abcast}_q(m)$. ■

Lemma 9 (Total order)**.** *If some correct process p executes $\mathsf{adeliver}_p(m, s)$ before $\mathsf{adeliver}_p(m', s')$ for any two processes s and s', then every correct process q executes $\mathsf{adeliver}_q(m', s')$ only after it has executed $\mathsf{adeliver}_q(m, s)$.*

Proof: Let p and q be two correct processes such that $\mathsf{adeliver}_p(m, s)$, $\mathsf{adeliver}_p(m', s')$, $\mathsf{adeliver}_q(m, s)$, and $\mathsf{adeliver}_q(m', s')$ are executed. Let further r_x, resp. r'_x, denote the value of r when (m, s), resp. (m', s'), is delivered at process $x \in \{p, q\}$.

By Lemma 5, (m, s) and (m', s') are each delivered in the same rounds by all correct processes, i.e., $r_p = r_q$ and $r'_p = r'_q$. If (m, s) and (m', s') are delivered in different rounds, then either $r_p < r'_p$ and $r_q < r'_q$, or $r_p > r'_p$ and $r_q > r'_q$, and the order property holds. If (m, s) and (m', s') are delivered in the same round r, then they are delivered in the order of the process IDs and lsn, and the order property holds. ■

From Lemmas 6–9 and the fact that reliable unique broadcast can be implemented in an asynchronous system with reliable links when $n > 3t$ [9], we get the following theorem:

Theorem 1. *If $n > 3t$, then Algorithm 1 is a reduction of atomic broadcast to range validity consensus in an asynchronous system with reliable links.*

[9]For simplicity we assume that $\mathcal{V}$ is infinite: otherwise the same reasoning will apply for some round $r' > r$.

C. *Why* reliable unique broadcast*?*

We illustrate now on an example why *reliable unique broadcast* [9] (rather than reliable broadcast [10]) is needed in Algorithm 1. Consider $n = 4, t = 1$, processes denoted by $i \in [1, 4]$, and process 4 being the faulty process. If *reliable broadcast* is used in Algorithm 1, then processes 1 and 2 may rdeliver, from process 4 at line 9, $m' = \langle 1, m\rangle$ resp. $m'' = \langle 1, \overline{m}\rangle$. By the *Agreement* property of *reliable broadcast*, all correct processes eventually *rdeliver* $m' = \langle 1, m\rangle$ and $m'' = \langle 1, \overline{m}\rangle$ from process 4. Therefore, all correct processes will eventually start round k proposing 1 for consensus instance $(k-1)\cdot n+4$. By the range validity property of consensus, all processes will decide 1 in the consensus instance $(k-1)\cdot n+4$. Therefore, process 1 will *adeliver* $(m, 4)$ and process 2 will *adeliver* $(\overline{m}, 4)$, violating the total order property of atomic broadcast.

D. Time complexity of Algorithm 1

We express the time complexity τ_m^{AB} of our reduction algorithm as the upper bound of the duration between the atomic broadcast of some message m by a correct process and the delivery of m at all correct processes.[10] We express τ_m^{AB} in terms of the maximum execution time of consensus — denoted by τ^C — and the maximum communication delay δ. For τ^C, we consider the starting time of an instance of consensus to be the time at which the last input event of a correct process occurs, and the ending time to be the time at which the last output event (decision value) of a correct process occurs. We define δ as the maximum transmission delay on messages exchanged among correct processes. We have the following result (proof in [14]):[11]

Theorem 2. *[Time complexity] If $|\mathcal{V}| = \infty$, for Algorithm 1 we have $\tau_m^{AB} \leq 2\tau^C + 3\delta$.*

V. Solving range validity consensus

In order to complete the contribution of the previous section, we discuss now the solution of range validity consensus. Range validity consensus is easy to solve in the partially synchronous model [5]. For example, this is achievable by modifying one single line of the CL consensus algorithm in [15], which itself is based on PBFT [16]. Instead of giving this algorithm, we find it more interesting to show a different solution, namely that range validity consensus is reducible to binary consensus with constant time complexity. The reduction is given by Algorithm 2.

[10]Since communication delays are unbounded in an asynchronous system, the analysis is done for finite runs, where a maximum value exists for every run. To simplify the notation, we drop the reference to runs.

[11]In the proof we assume that reliable unique broadcast is implemented as in [9] and therefore has time complexity of 3δ.

Algorithm 2 Reduction of range validity consensus to binary consensus (code of process p)

```
1:  r ← 0
2:  rubdelivered ← ∅

3:  upon range-propose(v) do
4:    rubcast(1, v)        /* 1 is the local index number, see
      Sect.II-A; each process rubcasts only once. */

5:  upon rubdeliver(1, v, q) do
6:    rubdelivered ← rubdelivered ∪ {⟨1, v, q⟩}

7:  wait until |rubdelivered| ≥ n − t
8:  loop
9:    r ← r + 1
10:   for π = 1 to n do
11:     v[π] ← 1 if ⟨−, π⟩ ∈ rubdelivered otherwise 0
12:     bin-propose((r − 1) · n + π, v[π])
13:   for π = 1 to n do
14:     wait until bin-decide((r − 1) · n + π, decision[π])
15:   Π1 ← {q ∈ Π : decision[q] = 1}
16:   if |Π1| ≥ n − t then
17:     for all i ∈ Π1 do
18:       wait until
            ∃m : msg = ⟨1, m, i⟩ ∈ rubdelivered
19:     dec ← max v s.t.
          ∃q_v ∈ Π1 : (1, v, q_v) ∈ rubdelivered ∧ |{q ∈ Π1 :
          (1, v', q) ∈ rubdelivered s.t. v' ≥ v}| ≥ t + 1
20:     range-decide(dec)
```

Algorithm 2 starts by having processes *rubcast* their initial value (line 4). After delivery of $n-t$ initial values (line 7), processes execute a sequence of rounds: in each round r, n binary consensus instances, namely instances $(r-1)\cdot n+i$, $i = 1..n$, are executed in parallel. A correct process proposes 1 in consensus instance $(r-1)\cdot n+i$ if it has *rubdelivered* the initial value of process i; otherwise it proposes 0 (line 11). The algorithm terminates in a round in which at least $n-t$ binary consensus instances decide 1 (line 16). Line 19 ensures that the decision value is $\geq$ than the initial value of at least one correct process. Moreover, since $n > 3t$, line 19 also ensures that the decision value is $\leq$ than the initial value of at least one correct process.

The next theorem establishes the correctness of the reduction:

Theorem 3. *If $n > 3t$ then Algorithm 2 reduces range validity consensus to binary consensus.*

Proof: To avoid ambiguity we use the prefix range resp. bin to distinguish the proposal and decision events of the two consensus specifications.

Agreement. Because of the agreement property of

binary consensus, for any round r, the set Π_1 is the same at all correct processes p. By the agreement and integrity property of reliable unique broadcast, if two correct processes rubdeliver a message with the same tag from some process, they rubdeliver the same message. Since the deterministic rules at lines 15-19 are based only on the set Π_1 and messages rubbcast by processes from the set Π_1, it follows that all correct processes decide the same value.

Termination. By the validity and agreement properties of reliable unique broadcast all correct processes eventually rubdeliver messages from all correct processes ($\star$). Since there are at most t Byzantine processes, no correct process waits forever at line 7. By the termination property of binary consensus, no correct process waits forever at line 14. Furthermore, by ($\star$), eventually all correct processes have received messages from all correct processes, so they propose 1 for all consensus instances $(r-1)\cdot n+\pi$ that correspond to correct processes π. By the agreement property of binary consensus, these instances of binary consensus decide 1. Therefore, the set Π_1 eventually has $n-t$ elements and the condition at line 16 evaluates to $true$.

If a process waits at line 18 to rubdeliver $(1, m, q)$, this means that the decision value of binary consensus instance $(r-1)\cdot n+q$ was 1. By the validity property of binary consensus, at least one correct process proposed 1 for this binary consensus instance. Therefore, by line 11, this process rubdelivered $(1, m, q)$. By the agreement property of reliable unique broadcast, all correct processes eventually rubdeliver $(1, m, q)$, so no process waits forever at line 18.

Range Validity. A value v is decided if (i) there are at least $t+1$ values v' such that $v' \geq v$, and (ii) if v is the maximum value among the values that satisfies condition (i) (see line 19). Condition (i) ensures that there is at least one initial value of a correct process that is greater or equal than the decision value ($\star$). Further, a value is decided only if the set Π_1 contains at least $n-t$ values. Since we assume that $n > 3t$, the number of values that satisfies condition (i) is $n-2t > t$, so at least one among these values is an initial value of a correct process. Therefore, condition (ii) ensures that selected value is greater or equal to at least one initial value of correct process ($\star\star$). From ($\star$) and ($\star\star$) it follows that the decision value is between the initial values of correct processes. ■

As in in Section IV-D, the time complexity of Algorithm 2 is expressed in terms of maximum execution time of binary consensus τ^C_{bin} and the maximum communication delay δ. We have the following result (proof in [14]):

Theorem 4. *[Time complexity] For the reduction given by Algorithm 2, we have* $\tau^{RV}_k \leq 2\tau^C_{bin} + 3\delta$.

VI. Reducing atomic broadcast to binary consensus

If we use Algorithm 1 with $|\mathcal{V}| = 2$, we get a reduction of atomic broadcast to binary consensus. Unfortunately, this reduction does not have constant time complexity with respect to binary consensus. For a finite domain, in particular for $\mathcal{V} = \{0, 1\}$, the reduction shown by Algorithm 1 has a constant time complexity with respect to binary consensus only if the number of abcast invocation by a correct process is bounded: Algorithm 1 can order at most $|\mathcal{V}|$ messages per round and process. Since every round takes at least the execution time of consensus, ordering $n|\mathcal{V}|$ requests requires $O(n\tau^C_{bin})$ time.

Constant time complexity with respect to binary consensus can be obtained by combining Algorithm 1 with Algorithm 2 (to implement range validity consensus). This leads to a reduction of atomic broadcast to binary consensus with constant time complexity with respect to binary consensus. From Theorem 1 and Theorem 2 follow:

Corollary 2. *For the atomic broadcast reduction to binary consensus, obtained by combining Algorithm 1 (with* $|\mathcal{V}| = \infty$*) with Algorithm 2, we have* $\tau^{AB}_m \leq 2(2\tau^C_{bin} + 3\delta) + 3\delta$.

VII. Related Work

Atomic broadcast: Algorithms for atomic broadcast have attracted a lot of attention, both for the benign and for the Byzantine fault model. In this paper, we focus on algorithms that solve atomic broadcast by reduction to consensus.

In [1], Chandra and Toueg solve atomic broadcast by reduction to consensus for benign faults. The reduction has constant time complexity with respect to consensus. The authors also mention that atomic broadcast is reducible to consensus in the Byzantine fault model, but no reduction is given.

In [2], [17], Correia *et al.* give reductions of atomic broadcast to consensus in the Byzantine fault model. They give two reductions, the first to abortable consensus and the second to binary consensus. The reductions do not have constant time complexity with respect to consensus, a property of our reduction. Indeed, the reduction to abortable consensus has a time complexity $\tau^{AB}_m \leq (f+1)\tau^C + 6\delta$, while the time complexity of the reduction to binary consensus is $\tau^{AB}_m \leq (f+1)(\tau^C_{bin} + 6\delta) + 6\delta$.

In [4], Cachin *et al.* give reductions of atomic broadcast to binary consensus for Byzantine faults with authentication. They give two reductions: the first is deterministic and the second randomized. As an intermediate module in these algorithms, they use multi-valued Byzantine agreement with *external validity*. External validity allows an application that requests agreement to specify the decision values that are acceptable. Note that, strictly speaking, multi-valued Byzantine agreement with *external validity* is not consensus since it depends on the application by the external validity property. [4] gives two implementations of multi-valued Byzantine agreement with *external validity* by reduction to binary consensus, the first being deterministic and the second randomized. Depending on the implementation considered, the atomic broadcast reduction to binary consensus is deterministic or randomized. The deterministic reduction does not have constant time complexity with respect to binary consensus (we have $\tau_m^{AB} \leq (f+1)(\tau_{bin}^C+\delta)+4\delta$). The randomized reduction to binary consensus has a constant expected time complexity with respect to binary consensus. Note that this is different from deterministic constant time reduction to binary consensus obtained in this paper. The approach highly relies on the use of cryptographic tools, which is not the case for our reduction. The approach was later followed in several papers [18], [19], [20].

Atomic broadcast is part of several Byzantine-tolerant group communication systems, where it is implemented either monolithically [21] or using *view synchrony* [22], [23], [24]. All these protocols rely on the use of cryptographic tools, which is not the case for our reduction.

Although atomic broadcast can be used to implement state machine replication, several algorithms directly implement Byzantine state machine replication, *e.g.*, [16], [25], [26]. The algorithm in [26] proceeds in two phases, pre-agreement and agreement phase, and in the agreement phase, similarly to our Algorithm 1, processes agree on message IDs instead on full messages. This idea of agreeing on message IDs was discussed by Ekwall and Schiper in [27], where the authors give a reduction of atomic broadcast to consensus in the benign fault model.

Validity property of consensus: Besides the validity properties discussed in the paper, other validity properties have been proposed. In [6], *differential consensus* is defined by introducing δ-differential validity, which requires that the decision value is of a certain plurality among the correct processes. It is not clear that such a validity property helps in the reduction of atomic broadcast.

Several papers have considered the *vector consensus* problem [28] as an intermediate step in solving atomic broadcast [2], [28]. Contrary to the consensus problem, in vector consensus the type of the decision differs from the type of the initial values (initial values of type T, decision of type vector of T). As noted in [2], vector consensus is an adaptation for asynchronous systems of *interactive consistency*, defined for synchronous systems [29].

An interesting observation is that range validity can be seen as a special case of the validity condition in approximate agreement [30]. We can think about range validity consensus as a "perfect" approximate agreement problem with $\epsilon = 0$ where ϵ defines allowed difference among decision values. Interestingly in SIFT, a fault tolerant system for aircraft control [31], range validity consensus would naturally fit in the algorithms for clock synchronization, stabilization of input from sensors, and agreement on results of diagnostic tests, where *interactive consistency* was used.

Consensus Reductions: In [32], Turpin and Coan give an algorithm that reduces abortable multi-valued consensus to binary consensus in a synchronous system. Reductions have also been described in an asynchronous system. [33] gives a reduction that uses signatures, and in [2] a similar algorithm without signatures is shown. Both reductions have constant time complexity with respect to binary consensus. However, in all these reductions, processes decide on a "default" value if correct processes do not have the same initial value. Therefore, they cannot be used to reduce (multi-valued) range validity consensus to binary consensus. Note that algorithm 2 relies on mechanisms similar to those found in other reduction to binary consensus, e.g., [34]. However, contrary to Algorithm 2, they do not consider the reduction of the standard consensus problem.

VIII. Conclusion

The paper has discussed the relation between atomic broadcast and different variants of consensus in systems with Byzantine faults. It has shown that consensus with *weak unanimity* is not sufficient to solve atomic broadcast, while consensus with *strong validity* is harder than atomic broadcast. Furthermore, the paper has shown that atomic broadcast is equivalent to consensus with *strong unanimity*, consensus with *abortable validity*, and consensus with *range validity*.

The paper has also given a reduction of atomic broadcast to range validity consensus with constant time complexity with respect to consensus. Range validity consensus has been then reduced to binary consensus, also with constant time complexity. Together, this leads to a reduction of atomic broadcast to binary consensus,

with constant time complexity with respect to binary consensus. To the best of our knowledge, these are the first atomic broadcast reductions to consensus with the constant time complexity with respect to consensus in the Byzantine fault model.

Acknowledgement: We would like to thank Sam Toueg for having shared with us, a long time ago, a reduction of atomic broadcast to binary consensus in the context of Byzantine faults. We would like to thank also Martin Biely, Omid Shahmirzadi, the anonymous reviewers and Flavio Junqueira.

REFERENCES

[1] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM*, 1996.

[2] M. Correia, N. F. Neves, and P. Veríssimo, "From consensus to atomic broadcast: Time-free byzantine-resistant protocols without signatures," *Comput. J.*, 2006.

[3] V. Drabkin, R. Friedman, and A. Kama, "Practical byzantine group communication," *Distributed Computing Systems, International Conference on*, 2006.

[4] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols (extended abstract)," in *in Advances in Cryptology: CRYPTO*, 2001.

[5] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, 1988.

[6] M. Fitzi and J. A. Garay, "Efficient player-optimal protocols for strong and differential consensus," in *PODC*, 2003.

[7] D. Dolev and E. N. Hoch, "Constant-space localized byzantine consensus," ser. DISC, 2008.

[8] A. Mostéfaoui and M. Raynal, "Signature-free broadcast-based intrusion tolerance: never decide a byzantine value," ser. OPODIS, 2010.

[9] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg, "Consensus with byzantine failures and little system synchrony," in *DSN*, 2006.

[10] V. Hadzilacos and S. Toueg, "A modular approach to fault-tolerant broadcasts and related problems," Tech. Rep., 1994.

[11] N. A. Lynch, *Distributed Algorithms*, 1996.

[12] G. Neiger, "Distributed consensus revisited," *Inf. Process. Lett.*, 1994.

[13] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, 1985.

[14] Z. Milosevic, M. Hutle, and A. Schiper, "On the reduction of atomic broadcast to consensus with byzantine faults," no. EPFL-REPORT-165021, 2011.

[15] ——, "Unifying byzantine consensus algorithms with weak interactive consistency," in *OPODIS*, 2009.

[16] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, 2002.

[17] H. Moniz, N. F. Neves, M. Correia, and P. Verissimo, "Ritas: Services for randomized intrusion tolerance," *IEEE Transactions on Dependable and Secure Computing*, 2011.

[18] K. Kursawe and V. Shoup, "Optimistic asynchronous atomic broadcast," in *in the Proceedings of International Colloqium on Automata, Languages and Programming (ICALP05) (L Caires, G.F. Italiano, L. Monteiro, Eds.) LNCS 3580*, 2001.

[19] H. V. Ramasamy and C. Cachin, "Parsimonious asynchronous byzantine-fault-tolerant atomic broadcast," in *OPODIS*, 2005.

[20] D. Dobre, H. V. Ramasamy, and N. Suri, "On the latency efficiency of message-parsimonious asynchronous atomic broadcast," in *SRDS*, 2007.

[21] L. E. Moser and P. M. Melliar-Smith, "Byzantine-resistant total ordering algorithms," *Inf. Comput.*, 1999.

[22] M. K. Reiter, "Secure agreement protocols: Reliable and atomic group multicast in rampart," in *In Proceedings of the 2nd ACM Conference on Computer and Communications Security*, 1994.

[23] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith, "The securering group communication system," *ACM Trans. Inf. Syst. Secur.*, 2001.

[24] H. V. Ramasamy, P. Pandey, M. Cukier, and W. H. Sanders, "Experiences with building an intrusion-tolerant group communication system," *Softw. Pract. Exper.*, 2008.

[25] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in *SOSP*, 2007.

[26] Y. Amir, B. A. Coan, J. Kirsch, and J. Lane, "Byzantine replication under attack," in *DSN*, 2008.

[27] R. Ekwall and A. Schiper, "Solving atomic broadcast with indirect consensus," in *DSN*, 2006.

[28] A. Doudou and A. Schiper, "Muteness detectors for consensus with byzantine processes," Tech. Rep., 1997.

[29] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, 1980.

[30] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching approximate agreement in the presence of faults," *Journal of the ACM*, 1986.

[31] J. H. Wensley, L. Lamport, J. Goldberg, S. Member, M. W. Green, K. N. Levi'it, P. M. Melliar-smith, R. E. Shostak, Charles, and B. Weinstock, "Sift: Design and analysis of a fault-tolerant computer for aircraft control," in *Proceedings of the IEEE*, 1978, pp. 1240–1255.

[32] R. Turpin and B. A. Coan, "Extending binary byzantine agreement to multivalued byzantine agreement," *Information Processing Letters*, 1984.

[33] S. Toueg, "Randomized byzantine agreements," ser. PODC '84, 1984.

[34] M. Ben-Or and R. El-Yaniv, "Resilient-optimal interactive consistency in constant time," *Distrib. Comput.*, 2003.