


# Brief Announcement: Communication Patterns for Optimal Resilience

Hagit Attiya ✉ 

Technion – Israel Institute of Technology, Israel

Itay Flam ✉

Technion – Israel Institute of Technology, Israel

Jennifer L. Welch ✉ 

Texas A&M University, College Station, TX, USA

---

## Abstract

In response to the impossibility of solving the consensus problem in asynchronous systems subject to failures, various relaxations of the consensus problem have been proposed, including approximate agreement, crusader agreement, gather, and reliable broadcast. Some are interesting in their own right while others are useful building blocks for solving other problems. We focus on message-passing systems of  $n$  processes, up to  $f$  of which can experience malicious (Byzantine) failures. These problems all require that  $n > 3f$  and they frequently have fairly simple and efficient algorithms when  $n > 5f$ . Challenges arise when considering resilience between  $3f + 1$  and  $5f$ . For instance, nearly twenty years elapsed between the discovery of an approximate agreement algorithm for  $n > 5f$  [10] and one for  $n > 3f$  [1]. A stumbling block could be too much focus on looking for algorithms in a certain natural and intuitive form, which we call *canonical (asynchronous) rounds*. In such an algorithm, each process repeatedly sends a message containing its entire state tagged with a round number, then waits to receive  $n - f$  messages with that same round number, does some local computation and proceeds to the next round number. The  $n > 5f$  approximate agreement algorithm is in canonical round form *but the  $n > 3f$  one is not*.

For algorithms in canonical round form, an obvious way of measuring time is the number of canonical rounds until the algorithm completes. However, this approach does not apply to other algorithms, such as those in which processes wait to receive a certain number of messages that have other properties besides simply having a certain round number. Attempts to rewrite these latter algorithms in canonical round form can result in drastically increased round complexity. This blow-up in the round complexity is inherent, as we show in this paper that *for a wide set of problems, there is no algorithm in canonical round form that has a finite upper bound on the number of rounds if  $n \leq 5f$* . In contrast, the standard way of measuring time results in constant time complexity.

We first show the impossibility of a bounded number of canonical rounds for a generic problem that captures the key properties needed in the proof. The result then follows immediately for, most notably, crusader agreement and flavors of approximate agreement. We then show via reductions that the same result holds for reliable broadcast and gather, since there are crusader agreement algorithms that use reliable broadcast and gather with no round overhead.

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** canonical rounds, reliable broadcast, gather, crusader agreement, approximate agreement, time complexity

**Digital Object Identifier** 10.4230/LIPIcs.DISC.2025.64

**Category** Brief Announcement

**Funding** Hagit Attiya: Supported by the Israel Science Foundation (22/1425 and 25/1849)

Itay Flam: Supported by the Israel Science Foundation (22/1425 and 25/1849)



© Hagit Attiya, Itay Flam and Jennifer L. Welch;  
licensed under Creative Commons License CC-BY 4.0  
39th International Symposium on Distributed Computing (DISC 2025).  
Editor: Dariusz R. Kowalski; Article No. 64; pp. 64:1–64:7



Leibniz International Proceedings in Informatics  
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Modern geo-distributed systems are often modeled as a set of  $n$  *asynchronous* processes that communicate by sending and receiving messages; the possibility of arbitrary failures is captured by assuming that  $f$  of the processes might be *malicious* (*Byzantine*) and deviate from their algorithms or even collude. While traditional *consensus* cannot be solved in this model, even when  $f = 1$  [14], relaxations of it are solvable, e.g., crusader agreement [9], approximate agreement [10], reliable broadcast [5], and gather [1, 7]. Solutions for these problems play a key role in Byzantine fault tolerance in asynchronous systems.

The *resilience* needed to solve a task in such a system is the relation between the total number of processes,  $n$ , and the maximal number of Byzantine processes,  $f$ . It is known that a resilience of  $n > 3f$  is necessary and sufficient for solving many consensus-related problems. However, finding algorithms that have the optimal resilience can be challenging. For example, the seminal paper defining the approximate agreement problem [10] proved that  $n > 3f$  is a lower bound on the resilience but the algorithms in that paper only have resilience  $n > 5f$ . Finding an algorithm with the optimal resilience, that is, an algorithm that only assumes  $n > 3f$ , has taken about twenty years [1].

We believe that a stumbling block to obtaining these algorithms is the inclination to look for algorithms that work in *canonical rounds*, a notion first mentioned by Fekete [11]. These are *full-information* algorithms, in which a process always sends all the information it has received so far, that proceed in *asynchronous rounds*. In each round  $r$ , a process  $p$  sends an  $r$ -message with the information it has received so far. Then, process  $p$  waits to receive  $r$ -messages from any  $n - f$  distinct processes, changes state and proceeds to round  $r + 1$ . After some number of rounds  $S$ , process  $p$  decides on a value that is some function of the information it has received, and stops sending or receiving messages.

Assuming this communication pattern seems innocuous; indeed, what else can a process do? Fekete even claims that “*this form of protocol is completely general*” [12]. However, several widely-used algorithms employ variations on this communication pattern, by waiting to receive a certain number of messages *that satisfy a certain property*, not just any set of  $n - f$  messages. For instance, in Bracha’s reliable broadcast algorithm [5], the sender sends an *initial* message containing its input to all processes. Each process waits to receive an *initial* message, or  $(n + f)/2$  *echo* messages for a common value, or  $f + 1$  *ready* messages for a common value, and then sends an *echo* message for the value. Each process then waits to receive  $(n + f)/2$  *echo* messages for a common value or  $f + 1$  *ready* messages for a common value, and then sends a *ready* message for the value. Once a process has received  $2f + 1$  *ready* messages for the same value, it accepts that value.

Bracha’s algorithm can be restructured to use canonical rounds. Since processes must move to the next round after receiving  $n - f$  messages for the current round, null messages are sent as placeholders while waiting to receive enough messages of a required type. The resulting algorithm reveals an interesting property: *The number of canonical asynchronous rounds required is unbounded*. The reason is that, even with a correct sender, the sender’s *initial* message can be delayed arbitrarily long, while canonical rounds consisting largely of *null* messages continue to advance. In contrast, the canonical round version of Bracha’s algorithm, like the original, has *bounded* (in fact, *constant*) time complexity, measured as the maximum real time elapsed until termination assuming every message sent between correct processes is delivered within one time unit [3, 4]. The reason is that while Byzantine processes are “rushing” the correct processes through many canonical rounds, messages between correct processes are pending, and until they arrive less than one time unit can elapse.

This undesirable behavior is not caused simply by a poor conversion into canonical rounds. On the contrary, we prove in this paper that this behavior is inherent for a wide collection of fundamental problems. First, we show that any algorithm for a generic problem among  $n \leq 5f$  processes has *an execution with an unbounded number of canonical rounds*. We then show that crusader agreement, approximate agreement on the real numbers, and approximate agreement on graphs are all special cases of the generic problem, which implies the same impossibility result for these problems. In particular, this explains why the code for the  $n > 3f$  approximate agreement in [1] cannot be structured in canonical rounds. We observe that the approximate agreement algorithm for  $n > 5f$  in [10] and the crusader agreement algorithm for  $n > 5f$  in [4] both employ a bounded number of asynchronous canonical rounds, showing that requiring  $n$  to be at most  $5f$  is necessary for the impossibility result.

To derive the same impossibility result for reliable broadcast and gather, we rely on the fact that they can be used to solve crusader agreement with little to no overhead.

To summarize, our main contribution is proving that canonical round algorithms for certain problems require an unbounded number of rounds, for Byzantine fault-tolerance with resilience  $n \leq 5f$ . The impossibility result holds for several important building blocks, including crusader agreement, reliable broadcast and gather, as well as for approximate agreement on the real line and on graphs. This points to the limitations of round-based asynchronous algorithms, both as a design principle and as a way to measure time, at least for resilience in the range  $[3f + 1, 5f]$ .

## 2 Canonical Round Algorithms have Unbounded Round Complexity

We assume the standard asynchronous model for  $n$  processes, up to  $f$  of which can be faulty, in which processes communicate via reliable point-to-point messages (cf., e.g., [4]). We consider *malicious* (or *Byzantine*) failures, where a faulty process can change state arbitrarily and send messages with arbitrary content.

Informally, an *execution* is a sequence of alternating configurations (states of the processes and in-transit messages) and events (WakeUp's and message deliveries). If  $\alpha$  and  $\beta$  are executions and  $X$  is a set of processes, we say the executions are *indistinguishable* to  $X$ , denoted  $\alpha \stackrel{X}{\sim} \beta$ , if, for each process  $p$  in  $X$ ,  $p$  has the same initial state and experiences the same sequence of events in  $\alpha$  as in  $\beta$ .

An algorithm for  $n$  processes,  $f$  of which may be faulty, is in *canonical round* format and decides in  $S$  rounds if it operates as follows. Every message sent is labeled with the current round number of the sender, starting with 1. When a process wakes up, it sends a message containing its initial state to all the processes. Once a process receives  $n - f$  messages for the current round, it increments its round counter and sends a message containing its initial state and the sequence of messages it has received to all the processes. Once a process reaches round  $S$ , it decides based on all the messages it has received.

Our result is proved for a **nontrivial convergence** problem in which there are at least two possible input values  $x_0$  and  $x_1$  and at least two decision values  $d_0$  and  $d_1$ , such that if all correct processes have input  $x_i$ , then all correct processes must decide  $d_i$ , for  $i = 0, 1$  (*validity*), and if a correct process decides  $d_0$  in an execution, then no correct process can decide  $d_1$  in the same execution (*agreement*).

► **Theorem 1.** *For any canonical round algorithm that solves a nontrivial convergence problem with  $n \leq 5f$  and for any integer  $K \in \mathbb{N}$ , there exists an execution and a correct process that does not decide by round  $K$ .*

**Proof.** Assume towards contradiction that there exists a nontrivial convergence canonical round algorithm with  $n \leq 5f$  and some  $K \in \mathbb{N}$  such that in every execution, all correct processes decide by the end of round  $K$ .

For simplicity, we assume  $n = 5f$ , and divide the processes into five disjoint sets of  $f$  processes each:  $A, B, C, D, E$ . We consider the following initial configurations:

- Denote by  $C_0$  the initial configuration such that processes in groups  $B, C, D, E$  are correct and processes in group  $A$  are Byzantine. All correct processes begin with input 0.
- Denote by  $C_1$  the initial configuration such that processes in groups  $A, B, D, E$  are correct and processes in group  $C$  are Byzantine. All correct processes begin with input 1.
- Denote by  $C_2$  the initial configuration such that processes in groups  $A, B, C, D$  are correct and processes in group  $E$  are Byzantine. Processes in groups  $B, C$  begin with input 0, and processes in groups  $A, D$  begin with input 1.

We construct three executions  $\alpha_0, \alpha_1, \alpha_2$  starting at the initial configurations  $C_0, C_1, C_2$  respectively, such that  $\alpha_1 \stackrel{A,D}{\sim} \alpha_2 \stackrel{B,C}{\sim} \alpha_0$ . Each execution is constructed as follows:

- $\alpha_0$ : The execution begins with WakeUp events for all processes in  $A, B, C, E$ ; call this part of the execution  $\alpha_0^0$ . Next appear  $(n - f)^2$  receive events in which each of the  $n - f$  processes in  $A, B, C, E$  receives the  $n - f$  round 1 messages sent by the processes in  $A, B, C, E$ . Since  $|A \cup B \cup C \cup E| = 4f = n - f$ , the processes complete round 1 and send their round 2 messages. Call this part of the execution  $\alpha_0^1$ . Similarly, define  $\alpha_0^2$  through  $\alpha_0^K$ , so that processes receive round  $r$  messages and send round  $r + 1$  messages in  $\alpha_0^r$  with the caveat that in  $\alpha_0^K$ , processes decide instead of sending round  $K + 1$  messages. The processes in  $B, C, E$ , which are correct, send messages whose content is determined by the algorithm; the contents of the messages sent by the processes in  $A$ , which are Byzantine, are specified below. Note that processes in  $D$  take no steps in  $\alpha_0$  even though they are correct; consider them as starting late, after the other processes have completed  $K$  rounds.
- $\alpha_1$ : This execution and its partitioning into  $\alpha_1^0$  through  $\alpha_1^K$  is defined analogously to  $\alpha_0$ , but with processes in  $A, C, D, E$  exchanging messages, those in  $C$  being Byzantine, and those in  $B$  starting late.
- $\alpha_2$ : This execution and its partitioning into  $\alpha_2^0$  through  $\alpha_2^K$  are similar to the previous executions but with some key differences.  $\alpha_2^0$  consists of WakeUp events for *all* the processes.  $\alpha_2^1$  consists of  $(n - f)^2 + f$  receive events in which each of the  $n - f$  correct processes receives a carefully selected set of  $n - f$  round 1 messages and each faulty process takes a step in order to send a round 2 message. In particular, (correct) processes in  $A, D$  receive round 1 messages from processes in  $A, C, D, E$ , while (correct) processes in  $B, C$  receive round 1 messages from processes in  $A, B, C, E$ . Similarly, define  $\alpha_2^2$  through  $\alpha_2^K$ . The contents of the messages sent by the (Byzantine) processes in  $E$  are defined below; unlike in  $\alpha_0$  and  $\alpha_1$ , the round  $r$  messages sent to processes in  $A, D$  by a faulty process are not the same as those sent to processes in  $B, C$  by that process.

We now specify the messages sent by the faulty processes in each round  $r$ ,  $1 \leq r \leq K$ , of the three executions. In  $\alpha_0$ , a faulty process  $p_i \in A$  sends the round  $r$  message sent by the corresponding correct process  $p_i$  in  $\alpha_2$ . In  $\alpha_1$ , a faulty process  $p_i \in C$  sends the round  $r$  message sent by the corresponding correct process  $p_i$  in  $\alpha_2$ . Finally, in  $\alpha_2$ , a faulty process  $p_i \in E$  sends to the correct processes in  $B, C$  the round  $r$  message sent by the corresponding correct process  $p_i$  in  $\alpha_0$ , and to the correct processes in  $A, D$  the round  $r$  message sent by the corresponding correct process  $p_i$  in  $\alpha_1$ .

Recall that  $\alpha_i = \alpha_i^0 \dots \alpha_i^K$  for  $i = 0, 1, 2$ . Denote  $\alpha_i^0 \alpha_i^1 \dots \alpha_i^r$  by  $\alpha_i^{0:r}$  for  $i = 0, 1, 2$  and  $0 \leq r \leq K$ . Claim 2, implying that  $\alpha_0$  and  $\alpha_2$  are indistinguishable to processes in  $B, C$  through round  $K$ , and Claim 3, implying that  $\alpha_1$  and  $\alpha_2$  are indistinguishable to processes in  $A, D$  through round  $K$ , can be proved by induction.

▷ **Claim 2.** For each  $r, 0 \leq r \leq K$ , (a)  $\alpha_0^{0:r} \stackrel{B,C}{\sim} \alpha_2^{0:r}$  and (b) the same set of messages are in transit from  $A, B, C, E$  to  $B, C$  in the last configurations of  $\alpha_0^{0:r}$  and  $\alpha_2^{0:r}$ .

▷ **Claim 3.** For each  $r, 0 \leq r \leq K$ , (a)  $\alpha_1^{0:r} \stackrel{A,D}{\sim} \alpha_2^{0:r}$  and (b) the same set of messages are in transit from  $A, C, D, E$  to  $A, D$  in the last configurations of  $\alpha_1^{0:r}$  and  $\alpha_2^{0:r}$ .

From the validity property of the nontrivial convergence problem, by the end of  $\alpha_1$ , correct processes in groups  $A, D$  must decide 1. Since  $\alpha_1 \stackrel{A,D}{\sim} \alpha_2$ , the corresponding correct processes in these groups must decide 1 by the end of  $\alpha_2$ . Similarly from validity, by the end of  $\alpha_0$  the correct processes in groups  $B, C$  must decide 0. Since  $\alpha_0 \stackrel{B,C}{\sim} \alpha_2$ , processes in groups  $B, C$  must decide 0 by the end of  $\alpha_2$  as well. This is in contradiction to the agreement property of the nontrivial convergence problem for execution  $\alpha_2$ . ◀

### 3 Applications of Theorem 1

**Crusader agreement** [9] with input set  $V$  ensures that if all correct processes start with the same value  $v \in V$ , they must decide on this value, and otherwise, they may pick an *undecided* value, denoted  $\perp$ . If all correct processes start with  $v \in \{0, 1\}$  they must decide  $v$ , and if a correct process decides  $v \in \{0, 1\}$ , the other correct processes decide either  $v$  or  $\perp$ . Therefore, crusader agreement is a nontrivial convergence problem with 0 and 1 being the two distinguished inputs and the two distinguished decisions, which implies:

► **Corollary 4.** Any canonical round algorithm that solves crusader agreement with  $n \leq 5f$  has an execution in which a correct process does not decide by round  $K$ , for any integer  $K \in \mathbb{N}$ .

Crusader agreement is a special case of **connected consensus** [4], with parameter  $R = 1$ . It is easy to see that connected consensus for any  $R \geq 1$  is a nontrivial convergence problem, and as a special case, so is **gradecast** [13].

**Approximate agreement on the real numbers** with parameter  $\epsilon > 0$  [10] is defined as follows. Processes start with arbitrary real numbers and correct processes must decide on real numbers that are at most  $\epsilon$  apart from each other. Decisions must also be *valid*, i.e., contained in the interval of the inputs of correct processes. To show approximate agreement is a nontrivial convergence problem, choose any two real numbers whose difference is greater than  $\epsilon$  as the two distinguished inputs and two distinguished decisions. This implies:

► **Corollary 5.** Consider a canonical round algorithm that solves  $\epsilon$ -approximate agreement with  $n \leq 5f$ . If the input range includes  $v_0$  and  $v_1$  such that  $|v_1 - v_0| > \epsilon$ , then there is an execution in which some correct process does not decide by round  $K$ , for any integer  $K \in \mathbb{N}$ .

A similar result can be shown for **approximate agreement on graphs** [8], provided that the graph has two vertices that are at distance 2 apart.

**Reliable broadcast** [5] and its weaker variant **consistent broadcast** [6] are defined with one designated *sender* process  $s$ . The sender has an input value  $v$  that it wants to broadcast. Processes may terminate by accepting a message from the sender. Consistent broadcast ensures that if the sender  $s$  is correct then eventually all correct processes accept

$s$ 's input (*validity*), and all correct processes that accept a value from sender  $s$ , accept the same value (*agreement*). Reliable broadcast further ensures that if some correct process accepts a value from  $s$  then eventually all correct processes accept a value from  $s$  (*relay*).

Consider the following algorithm for crusader agreement, assuming  $n > 4f$ , which uses  $n$  concurrent instantiations of consistent broadcast. Each process consistently broadcasts its input and then waits to accept values from  $n - f$  consistent broadcast instances. If it accepts  $n - 2f$  copies of some value  $v$ , then it decides  $v$ , otherwise it decides  $\perp$ . *Validity* for crusader agreement is immediate since when all correct processes start with  $v$ , they all accept at least  $n - 2f$  copies of  $v$  and thus decide  $v$ . To argue *agreement*, assume for contradiction that one correct process accepts  $n - 2f$  copies of  $v$  and another correct process accepts  $n - 2f$  copies of  $w \neq v$ . By the agreement property of consistent broadcast, these messages are broadcast by different processes, and so  $n \geq 2(n - 2f)$ , which implies  $n \leq 4f$ , a contradiction.

Note that the algorithm simply waits for the termination of  $n - f$  out of  $n$  concurrent invocations of consistent broadcast. Thus, a consistent broadcast algorithm that terminates within  $K$  canonical rounds would yield a crusader agreement algorithm that terminates within  $K$  canonical rounds, contradicting Corollary 4. This implies:

► **Corollary 6.** *Any canonical round algorithm for consistent or reliable broadcast, with  $n \leq 5f$ , has an execution in which a correct process does not terminate by round  $K$ , for any integer  $K \geq 1$ .*

**Gather** [1,2,7,13] is an extension of consistent broadcast in which *all processes* broadcast their value, and accept values from a large set of processes. Beyond properties inherited from consistent broadcast, most notably, that if two correct processes accept a value from another process, it is the same value, gather also ensures that there is a *common core*  $S^C$  of  $n - f$  values that are accepted by all correct processes.

A very simple transformation shows that a gather algorithm can be used to solve crusader agreement, with no extra cost: Process  $p_i$  gathers the input values in a set  $S_i$ , and if some value  $v$  appears at least  $|S_i| - f$  times in  $S_i$ , then it decides on  $v$ ; otherwise, it decides on  $\perp$ . A counting argument shows that a value appearing  $|S_i| - f$  times in a correct process  $p_i$ 's variable  $S_i$  must appear at least  $n - 2f$  times in the common core  $S^C$ . Since at most one value can appear  $n - 2f$  times in  $S^C$ , all correct processes that decide on a non- $\perp$  value decide on the same value. Note that the transformation does not add any communication on top of the gather submodule. Thus, a gather algorithm that terminates within  $K$  canonical rounds would yield a crusader agreement algorithm that terminates within  $K$  canonical rounds, contradicting Corollary 4. This implies:

► **Corollary 7.** *Any canonical round algorithm for gather with  $n \leq 5f$  has an execution in which some correct process does not terminate by round  $K$ , for any integer  $K \geq 1$ .*

## 4 Discussion

We have shown that for many important building blocks for Byzantine fault-tolerance with maximal resilience, canonical round algorithms require an unbounded number of rounds. Given that every round requires all-to-all communication, this also means an unbounded number of messages, although many of them are expected to be empty. For crash failures, there is an approximate agreement algorithm [10] that works in a logarithmic number of canonical rounds when  $n > 2f$  (which is the optimal resilience). Thus the anomaly of requiring an unbounded number of canonical rounds when resilience is optimal does not (always) occur with crash failures.



---

References

---

- 1 Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In *OPODIS*, pages 229–239. Springer, 2004.
- 2 Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, page 363–373. Association for Computing Machinery, 2021.
- 3 Hagit Attiya and Jennifer L. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004.
- 4 Hagit Attiya and Jennifer L. Welch. Multi-valued connected consensus: a new perspective on crusader agreement and adopt-commit. In *27th International Conference on Principles of Distributed Systems (OPODIS)*, pages 6:1–6:23, 2023.
- 5 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.
- 6 Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 524–541. Springer, 2001.
- 7 Ran Canetti and Tal Rabin. Fast asynchronous Byzantine agreement with optimal resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, page 42–51, 1993.
- 8 Armando Castañeda, Sergio Rajsbaum, and Matthieu Roy. Convergence and covering on graphs for wait-free robots. *Journal of the Brazilian Computer Society*, 24:1–15, 2018.
- 9 Danny Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.
- 10 Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, 1986.
- 11 Alan D. Fekete. Asynchronous approximate agreement. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 64–76, 1987.
- 12 Alan D. Fekete. Asynchronous approximate agreement. *Information and Computation*, 115(1):95–124, 1994.
- 13 Peaseh Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.
- 14 Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.