Optimality of Simultaneous Byzantine Agreement with Limited Information Exchange (Preliminary Report)

Kaya Alpturer
Princeton University, USA

Ron van der Meyden
UNSW, Australia

Sushmita Ruj
UNSW, Australia

Godfrey Wong
UNSW, Australia

Abstract

Work on the development of optimal Byzantine Agreement protocols using the logic of knowledge has concentrated on the "full information" approach to information exchange, which is costly with respect to message size. Alpturer, Halpern, and van der Meyden (PODC 2023) introduced the notion of optimality with respect to a limited information exchange, and studied the *Eventual* Byzantine Agreement problem in the sending omissions failure model. The present paper studies the *Simultaneous* Byzantine Agreement problem for the crash failures model, and a number of limited information exchanges from the literature. In particular, the paper considers information exchanges from a FloodSet protocol (Lynch, Distributed Algorithms 1996), a variant of this in which agents also count the number of failures (Castañeda et al, NETYS 2017), and a variant in which agents associate each agent with a value (Raynal, PRDC 2002). By determining implementations of a knowledge based program, protocols are derived that are optimal amongst protocols for each of these information exchanges.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Logic and verification; Theory of computation → Distributed algorithms

Keywords and phrases Distributed algorithms, Epistemic logic, Reasoning about knowledge, Byzantine Agreement, Consensus, Fault tolerance

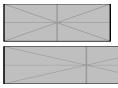
Digital Object Identifier 10.4230/LIPIcs...

Funding *Kaya Alpturer*: Research supported by AFOSR grant FA23862114029.
Ron van der Meyden: The Commonwealth of Australia (represented by the Defence Science and Technology Group) supported this research through a Defence Science Partnerships agreement.
Godfrey Wong: This research is supported by an Australian Government Research Training Program (RTP) Scholarship.

1 Introduction

Epistemic logic has been shown to play a useful role in the analysis of distributed algorithms [5]. By characterizing the knowledge that an agent needs to have in order to take an action, it is possible to obtain distributed algorithms that are optimal in the sense of terminating as soon as any other algorithm that solves the same problem.

In particular, variants of the Byzantine Agreement problem have been fruitfully studied using this methodology. In Byzantine Agreement, a group of agents is required to make common decision, in a setting where there could be faulty agents in the system that might be unreliable or even malicious. In the Simultaneous Byzantine Agreement (SBA) version of this problem, nonfaulty agents are required to make the decision at the same time (in the same round of computation).



Reaching simultaneous agreement was shown to be related to common knowledge [3], and this characterization has been used to derive optimal protocols for a variety of failure models, including crash [3] and omission failures [8].

In the present paper, we reconsider this type of optimality result in the particular case of crash failures and a synchronous message passing communications model. We assume synchronized clocks and a known upper bound on message delivery time, so that the computation can be structured into rounds. In a distributed system prone to crash failures, a faulty agent can crash during a round and send a subset of messages it was supposed to send. In the rounds following the crash, the crashed agent no longer send any messages.

In the quest for optimality, most existing literature on using epistemic logic to study distributed algorithms has worked with *full information protocols*, in which each agent broadcasts its complete local state in each round, and stores all messages received. This results in states that grow exponentially with time. Even when the state size can be optimized, there are still cases where the computation required to be performed in each round is intractable. For example, in the general omissions model, optimal implementations require P^{NP} computations [7]. This means that practical protocols will often exchange less information than does the full information protocol. This raises the following question: having made a compromise on the information that is exchanged by the agents, is the protocol making full use of *that* information. That is, is the protocol optimal amongst protocols that use the same pattern of information exchange, terminating no later than does any other such protocol?

There has only been one existing paper on optimality of Byzantine agreement protocols with respect to limited information exchange [1]. In that paper [1], the authors studied the *eventual* Byzantine agreement problem in synchronous message passing systems that are prone to sending omission failures. In the present paper, we will study the Simultaneous Byzantine Agreement (SBA) problem in synchronous message passing systems that are prone to crash failures.

The main contribution of this paper is to obtain the earliest possible early stopping conditions for four Simultaneous Byzantine Agreement protocols that use a limited information exchange. For FloodSet [6], we are able to terminate as soon as $m \geq \min\{t+1, n-1\}$, where m is time, t is maximum possible number of failures in a run, and n is the total number of agents. For Counting FloodSet [2], which is an extension of FloodSet that counts the number of messages an agent has received, the early stopping condition is the same as FloodSet, except for the case when an agent does not receive any message from other agents.

We have shown that even by extending Counting FloodSet to store a history of number of missing messages in each round, we are unable to obtain an early stopping condition that terminates earlier. In another variant of FloodSet [9] in which agents keep not just a record of initial values, but also know which agents have those initial values, and in which a nonfaulty agent sends a message only when it learns a new agent-value pair, we have obtained a nontrivial early stopping condition.

We begin by introducing our model of the problem in Section 2. Then, in Section 3 we will discuss what it means for a Simultaneous Byzantine Agreement protocol to be optimal relative to an information exchange protocol. We give an analysis of the FloodSet protocol in Section 4 and its variants in Section 5. Section 6 is an analysis of the protocol from [9]. Finally, we conclude this paper in Section 8.

2 Framework

We will mostly follow the framework from [1] and [10]. In the Byzantine agreement problem, agents from a set $Agt = \{1, \ldots, n\}$ communicate using a synchronous, round-based, message-passing network in order to decide on a value. Each agent starts with an initial value from the set

$Values = \{0, 1\}$. We assume that $t < n$ is the maximum number of failures that can occur in a run. We focus on *crash* failures, in which a faulty agent j acts correctly until a crash occurs in some round m in which an arbitrary subset of its messages in round m are delivered. Moreover, agent j never sends a message again in rounds $m' > m$.

The specification of Eventual Byzantine Agreement can be stated as follows:

- **Termination:** every agent eventually decides or fails,
- **Decision:** every agent decides at most once,
- **Agreement:** every nonfailed agent decides on the same value,
- **Validity:** if every agent has the same initial value, then every nonfailed agent decides on that value.

The specification of the Simultaneous Byzantine Agreement (SBA) problem involves all the rules of Eventual Byzantine Agreement problem together with simultaneity:

- **Simultaneity:** if a nonfailed agent decides in a round, then every nonfailed agent decides in the same round.

We decompose SBA protocols into two components $(P, \mathcal{E})$, comprised of a decision protocol P and an information exchange protocol $\mathcal{E}$. We represent the states of the environment in which agents operate by a set L_e, defined more precisely below. An information exchange protocol describes what information the agents record in their local states, and what messages an agent sends in a given local state. Formally, an information exchange protocol is a tuple $\langle \mathcal{E}_1, \ldots, \mathcal{E}_n \rangle$ containing a local information exchange protocol $\mathcal{E}_i$ for each agent i. A local information exchange protocol $\mathcal{E}_i$ of agent i is a tuple $\langle L_i, I_i, A_i, M_i, \mu_i, \delta_i \rangle$ that consists of:

- a set of local states L_i,
- a set of initial states $I_i \subseteq L_i$,
- a set of allowed actions A_i for agent i
- a set M_i of messages that are allowed to be sent by agent i,
- a message selection function $\mu_i : L_i \times A_i \to \prod_{j \in Agt}(M_i \cup \{\perp\})$,
- a transition function $\delta_i : L_i \times L_e \times A_i \times \prod_{j \in Agt}(M_j \cup \{\perp\}) \to L_i$.

For SBA protocols, we assume that there exist, for each agent i, a function $init_i : I_i \to Values$ that identifies an agent's initial value, and a function $time_i : L_i \to \mathbb{N}$ that identifies the time on the agent's clock. The message selection function μ_i takes a local state and an action performed by an agent in a round to a tuple of messages that the agent sends in that round. The transition function δ_i updates the local state depending on the state of the environment, the action an agent performs in the round and the set of messages it receives in that round.

In the full information exchange, the local initial states are the initial values of the agent. In every round, each agent sends their local state to every other agent, and records every message it receives in its local state.

A decision protocol describes what actions the agents perform at a given situation. For the SBA problem, the action set for each agent i is $A_i = \{\texttt{decide}_i(v) \mid v \in Values\} \cup \{\texttt{noop}\}$. A *local decision protocol* $P_i : L_i \to A_i$ for agent i maps a local state of agent i to an action that i can perform. A *decision protocol* P is a tuple $\langle P_1, \ldots, P_n \rangle$ of local decision protocols for all agents.

It will help when comparing two information exchanges below to assume that agent's decisions do not affect the information that they transmit. We say that an information exchange $\mathcal{E}$ *does not record information about actions* if, for all agents i, the values of both the functions μ_i and δ_i do not depend on the value of the action in their inputs. That is, changing the value of the input from A_i while holding the other inputs constant does not change the output of these functions. All the concrete information exchanges we consider in this paper have this property. Since this does mean that the protocol P_i cannot, in general, itself enforce that an agent decides as most one, we assume that an external process monitors the output of P_i in order to enforce this property. In the SBA

specification, we interpret "i decides value v at the point (r, m)" to mean that m is the least time m' such that $P_i(r_i(m')) \neq \bot$, and $P_i(r_i(m')) = \mathtt{decide}_i(v)$.

A failure model describes the failures that could occur in a run of a protocol. We use the hard crash failure model $Crash_t$, depending on a parameter t that is a number less than the number of agents. In this failure model, there may be up to t faulty agents, who can crash at any time. During the round in which an agent crashes, it sends a subset of the messages it is supposed to send. After an agent has crashed it does not send any message and no longer runs the protocol.

We represent $Crash_t$ as a set of *adversaries*, where an adversary is a function $\mathcal{F} : \mathbb{N} \times Agt \times Agt \to \{\top, \bot\}$. If agent i crashes in round m, then we require that $F(m, i, i) = \bot$. If the message that agent i was supposed to send to agent j is not sent in round m, we have $\mathcal{F}(m, i, j) = \bot$, otherwise $\mathcal{F}(m, i, j) = \top$. If $\mathcal{F}(m, i, i) = \bot$ then $\mathcal{F}(m', i, k) = \bot$ for all times $m' > m$ and every $k \in Agt$. There may be at most t agents i such that $\mathcal{F}(m, i, j) = \bot$ for some m and j. We take the set of states of the environment L_e to be the set of all such adversaries. Furthermore, for this model, we assume that the set of local states of agent i contains a special state $crashed$, representing that the agent has crashed. The state update function operates so that in a round m in which the adversary $s_e = \mathcal{F}$ and $\mathcal{F}(m, i, i) = \bot$, we have $\delta(s_i, s_e, a, v) = crashed$ for all local states s_i of agent i, actions a and vectors v of messages.

A global state is an element of $L_e \times \prod_{i \in Agt} L_i$ consisting of a tuple comprised of the state of the environment and a local state of each agent. A *run* is a mapping $r : \mathbb{N} \to L_e \times \prod_{j \in Agt} L_j$ from times to a global states. If $r(m) = \langle s_e, s_1, \ldots, s_n \rangle$, we write $r_e(m)$ for s_e and, for each agent i, $r_i(m)$ for s_i. The time between (r, m) and $(r, m + 1)$ is *round* $m + 1$. A pair (r, m) comprised of a run r and a time m is a *point*.

A *system* is a set $\mathcal{R}$ of runs. An *interpreted system* is a pair $\mathcal{I} = (\mathcal{R}, \pi)$, where $\mathcal{R}$ is a system and π is an *intepretation* mapping each point (r, m) of $\mathcal{I}$ to a function $\pi(r, m) : Prop \to \{true, false\}$ that determines whether or not each atomic proposition from a set $Prop$ is true at the point.

Given an information exchange $\mathcal{E}$ and decision protocol P for the crash failure model $Crash_t$, we construct an interpreted system $\mathcal{I}_{P,\mathcal{E},Crash_t} = (\mathcal{R}, \pi)$ as follows. (Generally, $Crash_t$ will be implicit, and we write $\mathcal{I}_{P,\mathcal{E}}$ for brevity.) The runs in $\mathcal{R}$ are generated by selecting an initial global state $r(0) = \langle s_e, s_1, \ldots, s_n \rangle$, where $s_e \in L_e = Crash_t$ is an environment state encoding an adversary from the crash failure model and $s_i \in I_i$ for all $i \in Agt$ are initial local states of agents. The remainder of the run is uniquely determined from the initial global state by the following induction. The state of the environment $r_e(m)$ will be the same for all times m. For each round $k + 1$, the local state $r_i(k + 1)$ of agent i is determined as follows. First, each agent i uses the decision protocol P_i to select its action $a_i = P_i(r_i(k))$, and attempts to send the messages $\mu_i(r_i(m), a_i)$. The adversary $r_e(k)$ determines which of these messages are delivered, so that each agent i receives an agent indexed vector v_i of messages (or $\bot$ in case of a message that was not sent or is not delivered because the agent sending it crashed). Agent i's local state $r_i(k + 1)$ is then equal to $\delta_i(r_i(k), r_e(k), a_i, v_i)$. We consider only *synchronous* protocols, for which $time_i(r_i(m)) = m$ for all points (r, m) of runs of the protocol.

Note that if $\mathcal{E}$ does not record information about actions, then the set of runs of $\mathcal{I}_{P,\mathcal{E}}$ does not depend on P. We may therefore write simply $\mathcal{I}_{\mathcal{E}}$.

The atomic propositions $Prop$ in the interpretation π consist of:

- $\exists v$, which is true if an agent in the current run has v as its initial value,
- $i \in \mathcal{N}$, which is true if agent i has not failed up to the current time, and
- $clean$, which is true if a clean round has occurred (defined below).

We work with a modal logic that uses the atomic propositions along with the standard boolean operators $\neg, \vee, \wedge$. For an agent i and an *indexical* set $\mathcal{N}$ of agents (which may depend on the point at which we evaluate the formula) we also have unary modal operators $K_i, E_{\mathcal{N}}, C_{\mathcal{N}}$ which we will

define later. The semantics of the logic in an interpreted system $\mathcal{I}$ is given by a relation $\models$, such that $(\mathcal{I}, r, m) \models \phi$, for a point (r, m) of $\mathcal{I}$ and a formula ϕ, represents that ϕ is true at the point (r, m) of $\mathcal{I}$. The definition of this relation is given by an induction on the construction of the formula ϕ. For an atomic proposition p, we have $(\mathcal{I}, r, m) \vDash p$, if $\pi(r, m)(p) = \textbf{True}$.

The semantics of the modal operators is given using an indistinguishability relation $\sim_i$ on points for each agent i. The points (r, m) and (r', m') are indistinguishable to agent i, written as $(r, m) \sim_i (r', m')$, if $r_i(m) = r'_i(m')$. Note that the indistinguishability relation is an equivalence relation. The intuition behind the definition of knowledge is that agent i knows φ if φ is true in all the points that are indistinguishable to i.

For a formula φ and point (r, m) of an interpreted system $\mathcal{I}$, we say:

- agent i knows φ at (r, m), written as $(\mathcal{I}, r, m) \vDash K_i\varphi$, if for all points (r', m') satisfying $(r, m) \sim_i (r', m')$, we have $(\mathcal{I}, r', m') \vDash \varphi$,
- everyone in $\mathcal{N}$ knows φ at (r, m), written as $(\mathcal{I}, r, m) \vDash E_{\mathcal{N}}\varphi$, if for every $i \in \mathcal{N}(r, m)$, we have $(\mathcal{I}, r, m) \vDash K_i(\varphi)$,
- φ is common knowledge among $\mathcal{N}$ at the point (r, m), written as $(\mathcal{I}, r, m) \vDash C_{\mathcal{N}}\varphi$, if for every $k \in \mathbb{Z}^+$, we have $(M, s) \vDash E^k_{\mathcal{N}}\varphi$.
- φ is distributed knowledge amongst group $\mathcal{N}$, written as $(\mathcal{I}, r, m) \vDash D_{\mathcal{N}}\varphi$, if $(\mathcal{I}, r', m) \vDash \varphi$ for all $(r', m) \in \bigcap_{i \in \mathcal{N}(r,m)} S_i(r, m)$, where $S_i(r, m)$ is the set of points that are indistinguishable to agent i from (r, m).

A formula φ is valid in $\mathcal{I}$, written as $\mathcal{I} \vDash \varphi$, if for every point $(r, m) \in \mathcal{I}$, we have $(\mathcal{I}, r, m) \vDash \varphi$.

In this paper, the indexical set $\mathcal{N}$ represents the set of nonfailed agents. Formally, $i \in \mathcal{N}(r, m)$ if $r_i(m) \neq \mathit{crashed}$.

We say a point (r', m') is $\mathcal{N}$-reachable from point (r, m) in k steps if there exist points $(r^0, m_0), \ldots, (r^k, m_k)$ such that $(r^0, m_0) = (r, m)$, $(r^k, m_k) = (r', m')$ and for all $0 \leq j < k$, there exists agent i_j that is nonfailed at both (r^j, m_j) and (r^{j+1}, m_{j+1}) such that $(r^j, m_j) \sim_{i_j} (r^{j+1}, m_{j+1})$. Furthermore, we say (r', m') is $\mathcal{N}$-reachable from (r, m), and write $(r', m') \sim_{\mathcal{N}} (r, m)$, if (r', m') is $\mathcal{N}$-reachable from (r, m) in k steps for some $k > 0$. An equivalent way to define common knowledge is $(\mathcal{I}, r, m) \vDash C_{\mathcal{N}}\varphi$ if and only if $(\mathcal{I}, r', m') \vDash \varphi$ for all points (r', m') that are $\mathcal{N}$-reachable from (r, m).

The semantics makes the following $S5_n$ axioms valid:

- Distribution axiom: $\vDash (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi$.
- Knowledge generalization rule: If $\mathcal{I} \vDash \varphi$ then $\mathcal{I} \vDash K_i\varphi$.
- Knowledge axiom: $\vDash K_i\varphi \Rightarrow \varphi$.
- Positive introspection axiom: $\vDash K_i\varphi \Rightarrow K_iK_i\varphi$.
- Negative introspection axiom: $\vDash \neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$.

We also have the following induction rule, which gives us a way of proving common knowledge.

▶ **Theorem 1** (Induction Rule). *If* $\mathcal{I} \vDash \varphi \Rightarrow E_G(\varphi \wedge \psi)$ *then* $\mathcal{I} \vDash \varphi \Rightarrow C_G\psi$.

As an example of the application of this rule that we will use below, define the formula $\mathcal{N} = \{i\}$ as $\bigwedge_{j \in Agt}(j \in \mathcal{N} \Leftrightarrow j = i)$. Then we have

▶ **Lemma 2.** *The formula* $K_i(\mathcal{N} = \{i\} \wedge \phi) \Rightarrow C_{\mathcal{N}}(\phi)$ *is valid.*

Proof. Using the Introspection and the Knowledge axioms, we have that $K_i(\mathcal{N} = \{i\} \wedge \phi)$ implies $\mathcal{N} = \{i\} \wedge K_iK_i(\mathcal{N} = \{i\} \wedge \phi)$ and hence $E_{\mathcal{N}}(K_i(\mathcal{N} = \{i\} \wedge \phi))$. This, in turn, implies $E_{\mathcal{N}}(K_i(\mathcal{N} = \{i\} \wedge \phi) \wedge \phi)$ using the Knowledge Axiom. Thus, $\psi \Rightarrow E_{\mathcal{N}}(\psi \wedge \phi)$ is valid, with $\psi = K_i(\mathcal{N} = \{i\} \wedge \phi)$. Using the Induction rule, we obtain that $K_i(\mathcal{N} = \{i\} \wedge \phi) \Rightarrow C_{\mathcal{N}}(\phi)$ is valid. ◀

3 Optimality and Knowledge-based Programs

In this section we study knowledge-based programs and what it means for a protocol to be optimal with respect to a given information exchange.

To compare two protocols, we consider their behaviour in runs in which the environment has the same behaviour. Two runs $r \in \mathcal{I}$ and $r' \in \mathcal{I}'$ *correspond* if they have identical initial values and the failure pattern is the same in both runs, that is, for all agents i, we have $init_i(r_i(0)) = init_i(r'_i(0))$ and $r_e(0) = r'_e(0)$.

To define optimality, we define a partial order on protocols. Let P and P' be two decision protocols with identical information exchange $\mathcal{E}$. We write $P \leq_{\mathcal{E}} P'$ if for every run r of $\mathcal{I}_{P,\mathcal{E}}$, and corresponding run r' of $\mathcal{I}_{P',\mathcal{E}}$, and for each agent i, if i decides at (r, m), then if agent i decides at (r', m') we have $m' \geq m$. Protocol P is an *optimal SBA protocol* with respect to information exchange $\mathcal{E}$ if it is an SBA protocol and for every SBA protocol P' that uses $\mathcal{E}$, we have $P' \leq_{\mathcal{E}} P$ implies $P \leq_{\mathcal{E}} P'$. Furthermore, protocol P is an *optimum SBA protocol* with respect to information exchange $\mathcal{E}$ if it is an SBA protocol and for every SBA protocol P' that uses $\mathcal{E}$, we have $P \leq_{\mathcal{E}} P'$.

The following lemma is taken from Moses and Tuttle [8]. Its proof does not assume a full information protocol. The proposition $\mathrm{deciding}_i(v)$ is defined to be true at a point (r, m) if agent i decides v in round $m + 1$ of run r, i.e., $P_i(r_i(m)) = \mathtt{decide}_i(v)$.

▶ **Lemma 3.** *[8] Let r be a run of an SBA protocol generating interpreted system $\mathcal{I}$ and let $i \in \mathcal{N}(r, m)$. If $(\mathcal{I}, r, m) \vDash \mathrm{deciding}_i(v)$ for $v \in \{0, 1\}$, then $(\mathcal{I}, r, m) \vDash C_{\mathcal{N}}(\exists v)$.*

It is useful to know that a failure-free run requires at least $\min\{t + 1, n - 1\}$ rounds to attain common knowledge of initial values. Therefore, when time $m < \min\{t + 1, n - 1\}$ and r is a run of any SBA protocol, if (r, m) is $\mathcal{N}$-reachable from a failure-free run then we do not have common knowledge of any initial value at (r, m).

▶ **Lemma 4.** *[3] Let r be a failure-free run of an optimal full-information SBA protocol generating system $\mathcal{I}$. When $m < \min\{t + 1, n - 1\}$ we have $(\mathcal{I}, r, m) \vDash \neg C_{\mathcal{N}}(\exists 0) \wedge \neg C_{\mathcal{N}}(\exists 1)$.*

Let $\mathcal{E}^1$ and $\mathcal{E}^2$ be two information exchange protocols that do not record information about decisions. We say $\mathcal{E}^1$ stores at least as much information as $\mathcal{E}^2$ if, for every pair of corresponding runs r^1 and r^2 of $\mathcal{I}_{\mathcal{E}_1}$ and $\mathcal{I}_{\mathcal{E}_2}$, respectively, and times m such that no agent has decided before time m in both r^1 and r^2, and every pair of corresponding runs r^3 and r^4 of $\mathcal{I}_{\mathcal{E}_1}$ and $\mathcal{I}_{\mathcal{E}_2}$, respectively, $(r^1, m) \sim_i (r^3, m)$ implies $(r^2, m) \sim_i (r^4, m)$. This relation is reflexive and transitive.

▶ **Lemma 5.** *Let $\mathcal{E}^1$ be an information exchange that stores at least as much information as information $\mathcal{E}^2$ and let r^1 and r^2 be corresponding runs of $\mathcal{I}_{\mathcal{E}^1}$ and $\mathcal{I}_{\mathcal{E}^2}$ respectively. If r^3 and r^4 are corresponding runs, of $\mathcal{I}_{\mathcal{E}^1}$ and $\mathcal{I}_{\mathcal{E}^2}$ respectively, such that (r^3, m) is $\mathcal{N}$-reachable from (r^1, m) in k steps, then (r^4, m) is $\mathcal{N}$-reachable from (r^2, m) in k steps.*

Proof. By induction on k. Suppose (r^3, m) is $\mathcal{N}$-reachable from (r^1, m) in one step. Because $\mathcal{E}^1$ stores at least as much information as $\mathcal{E}^2$, we have that (r^4, m) is $\mathcal{N}$-reachable from (r^2, m) in one step. For the inductive hypothesis, assume that (r^3, m) is $\mathcal{N}$-reachable from (r^1, m) in k steps implies (r^4, m) is $\mathcal{N}$-reachable from (r^2, m) in k steps. If (r^3, m) is $\mathcal{N}$-reachable from (r^1, m) in $k+1$ steps, let r' be a run of $\mathcal{I}_{\mathcal{E}^1}$ such that (r', m) is $\mathcal{N}$-reachable from (r^1, m) in k steps and (r^3, m) is $\mathcal{N}$-reachable from (r', m) in one step. Then there exists an agent i $i \in \mathcal{N}(r', m) \cap \mathcal{N}(r^3, m)$ such that $r'_i(m) = r^3_i(m)$. Let r'' be a run of $\mathcal{I}_{\mathcal{E}^2}$ corresponding to r'. Then, by the inductive hypothesis, (r'', m) is $\mathcal{N}$-reachable from (r^2, m) in k steps. Given they are corresponding runs, we have $\mathcal{N}(r'', m) = \mathcal{N}(r', m)$ and $\mathcal{N}(r^4, m) = \mathcal{N}(r^3, m)$. Because $\mathcal{E}^1$ stores at least as much information as $\mathcal{E}^2$, we have $(r'', m) \sim_i (r^4, m)$. Therefore, (r^4, m) is $\mathcal{N}$-reachable from (r^2, m) in $k + 1$ steps. ◀

A *proposition φ is about the environment* if its truth value in interpreted systems for the SBA problem, as defined above, depends only on the the adversary and the initial values of the agents. That is, for corresponding runs r and r' of systems $\mathcal{I}$ and $\mathcal{I}'$, respectively, and all times m, we have $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}', r', m) \models \varphi$.

▶ **Proposition 6.** *Suppose φ is a proposition about the environment. Suppose $\mathcal{E}^1$ stores at least as much information as protocol $\mathcal{E}^2$ and let r^1 and r^2 be corresponding runs of $\mathcal{I}_{\mathcal{E}^1}$ and $\mathcal{I}_{\mathcal{E}^2}$, respectively. Then $(\mathcal{I}_{\mathcal{E}^1}, r^1, m) \models \neg C_{\mathcal{N}}\varphi$ implies $(\mathcal{I}_{\mathcal{E}^2}, r^2, m) \models \neg C_{\mathcal{N}}\varphi$.*

Proof. Suppose we have $(\mathcal{I}_{\mathcal{E}^1}, r^1, m) \models \neg C_{\mathcal{N}}\varphi$. This implies there exists a point (r^3, m) that is $\mathcal{N}$-reachable from (r^1, m) in $\mathcal{I}_{\mathcal{E}^1}$ such that $(\mathcal{I}_{\mathcal{E}^1}, r^3, m) \models \neg\varphi$. Lemma 5 says we can find a run r^4 of P^1 corresponding to r^3 such that (r^4, m) is $\mathcal{N}$-reachable from (r^2, m) in ß$_{\mathcal{E}^2}$. Because φ is about the environment, $(\mathcal{I}_{\mathcal{E}^2}, r^4, m) \models \neg\varphi$. It follows that $(\mathcal{I}_{\mathcal{E}^2}, r^2, m) \models \neg C_{\mathcal{N}}\varphi$. ◀

We remark that Proposition 6 does not make assumptions about the failure model and hence would work even with arbitrary failures.

Knowledge based programs [4] can be understood as specifications, in a code-like form, that describe how each agent's actions relate to its knowledge. Syntactically, they are like standard programs for an agent, except that the formulas that occur in as the conditions of conditional statements may be a formula of the logic of knowledge stating a property of the agent's knowledge, rather than just a predicate of the agent's local state, as in standard programs. Program 1 gives an example, which will be the specific knowledge based program that we study in the present paper. This program is to be interpreted as the rule used to determine agent i's action in each round, until one of the actions $\texttt{decide}_i(v)$ is selected, after which the program terminates.

Program 1 $\mathbf{P}_i^{\text{OPT}}$

if $K_i(C_{\mathcal{N}}(\exists 0))$ **then** $\texttt{decide}_i(0)$
else if $K_i(C_{\mathcal{N}}(\exists 1))$ **then** $\texttt{decide}_i(1)$
else `noop`

The semantics of knowledge based programs is given by relating concrete protocols to a knowledge based program. Since we only consider one specific program in this paper, we do not give this definition in general, and just explain this relation for the specific program Program 1. Intuitively, a decision protocol P implements the knowledge based program with respect to an information exchange $\mathcal{E}$, if, in each round until termination, it selects for each agent i the same action as would be obtained for agent i from the rule in the knowledge based program. "If-then-else" statements are interpreted according to the usual semantics, but to interpret the knowledge formulas, we need an interpreted system. For this, we use the interpreted system obtained from running P with respect to $\mathcal{E}$.

Formally, given an interpreted system $\mathcal{I}$, for each agent i we define the decision protocol $(\mathbf{P}_i^{\text{OPT}})^{\mathcal{I}}$ on a local state $s \in L_i$ by taking $(\mathbf{P}_i^{\text{OPT}})^{\mathcal{I}}(s)$ to be the action a selected the rule $\mathbf{P}_i^{\text{OPT}}$, with the truth values of the knowledge formulas evaluated at any point (r, m) of $\mathcal{I}$ with $r_i(m) = s$. Note that the truth values of these formulas are independent of the choice of such an (r, m), since they depend only on the local state of agent i at that point. Using this, we define a decision protocol P *to be an implementation of Program* $\mathbf{P}^{OPT}$ *with respect to* $\mathcal{E}$ *in the crash failure context* $Crash_t$ if for all points (r, m) of $\mathcal{I}_{\mathcal{E}, Crash_t}$ and all agents $i \in \mathcal{N}(r, m)$, we have $(\mathbf{P}_I^{\text{OPT}})^{\mathcal{I}}(r_i(m)) = P_i(r_i(m))$.

The following result shows that implementations of the knowledge based program $\mathbf{P}^{\text{OPT}}$ with respect to an information exchange yield SBA protocols that are optimum with respect to that information exchange. In the following sections, we apply this result by considering a number of information exchanges and deriving protocols that are implementations of $\mathbf{P}^{\text{OPT}}$ with respect to

these information exchanges. The resulting protocols will therefore be optimum SBA protocols with respect to their information exchanges.

▶ **Theorem 7.** *Let P be an implementation of $\mathbf{P}^{opt}$ with respect to information exchange $\mathcal{E}$ in the crash failure context $Crash_t$. Then, P is an optimum simultaneous Byzantine agreement protocol with respect to $\mathcal{E}$.*

Proof. We omit the argument that an implementation P of $\mathbf{P}^{\text{OPT}}$ is an SBA protocol, which is the same as in [3]. It is shown in [10] that provided that the information exchange $\mathcal{E}$ "does not transmit information about actions", an implementation P of $\mathbf{P}^{\text{OPT}}$ with respect to an information exchange $\mathcal{E}$ is an optimum SBA protocol with respect to $\mathcal{E}$. Since all actions up to the decision time are `noop`, and in the crash failure context, an SBA protocol decides for all agents at the same time it decides for any agent, so this result follows., ◀

4 FloodSet

Lynch introduced the FloodSet protocol [6] as an example of a simple protocol that solves Simultaneous Byzantine Agreement under crash failures. In this section, we present a refined analysis of the Floodset protocol in terms of the agents' knowledge, showing that after a minor modification, the Floodset protocol is *optimal* given the amount of information the agents are exchanging.

We now define the FloodSet protocol in two parts using our framework, the information-exchange protocol $\mathcal{E}^{flood}$ and the decision protocol P^{flood}. The local states of agent i in $\mathcal{E}^{flood}$ will be either the special state $crashed_i$, or a tuple $\langle W_i, m, v_i\rangle$, where W_i is a set of distinct values agent i has seen so far, m is the current time, and v_i is the initial value of i. The information-exchange protocol $\mathcal{E}^{flood}$ has the set of initial local states I_i for each agent i consisting of all states of the form $\langle\{v_i\}, 0, v_i\rangle$ for some $v_i \in Values$. In each round, every agent i sends W_i to all other agents. That is, $\mu_i((W_i, m, v_i), a_i) = (W_i, \ldots, W_i)$ for all $(W_i, m, v_i) \in L_i$ and actions a_i, and $\mu_i(crashed_i, a_i) = (\bot, \ldots, \bot)$. At the end of a round $m+1$, v_i stays the same, and the local state of each agent is updated to have a new W_i which is the union of all messages received in the current round. That is, the function δ_i is defined by

$$\delta_i((W_i, m, v_i), \mathcal{F}, a_i, (W'_1, \ldots, W'_n)) = (W_i \cup \bigcup_j W'_j, m+1, v_i)$$

provided the environment does not crash agent i, i.e., $\mathcal{F}(m, i, i) \neq \bot$, otherwise

$$\delta_i((W_i, m, v_i), \mathcal{F}, a_i, (W'_1, \ldots, W'_n)) = crashed_i$$

and $\delta_i(crashed_i, \mathcal{F}, a_i, (W'_1, \ldots, W'_n)) = crashed_i$. Throughout this section, we fix $t \leq n$ and write $\mathcal{I}$ for $\mathcal{I}_{\mathcal{E}_{flood}, Crash_t}$. When agent i's local state at point (r, m) of $\mathcal{I}$ is $r_i(m) = (W, m, v)$, we define $W_i(r, m)$ to be W, and $time_i(r, m)$ to be m.

The standard program representing the FloodSet decision protocol for an agent i such that $P^{flood} = (P_1^{flood}, \ldots, P_n^{flood})$ is:

Program 2 P_i^{flood} from [6]

if $m = t + 1$ **then** $\texttt{decide}_i(\min W_i)$
else `noop`

A crucial definition used in the analysis is the notion of a *clean round*, a round in which every nonfailed agent receives the same messages.

▶ **Definition 8** (Clean Round). *A round is clean if for any nonfailed agents i and j, agent i receiving a message from agent k implies that agent j also received a message from agent k.*

We now show the importance of clean rounds.

▶ **Lemma 9.** *[6, Lemma 6.1] If round m of run r of FloodSet is clean, then $W_i(r,m) = W_j(r,m)$ for all nonfailed agents i, j in run r.*

▶ **Lemma 10.** *[6, Lemma 6.2] Suppose $W_i(r,m) = W_j(r,m)$ for any nonfailed agents i, j in run r of FloodSet. Then, $W_i(r,m') = W_j(r,m')$ for all $m \leq m' \leq t+1$.*

Therefore, since we can only have up to t crashes, by round $t+1$ we are guaranteed to have a clean round. Moreover, all agents $i \in Agt$ have the same $W_i(r, t+1)$ in any run r in $\mathcal{I}$. Note that however, when $t = n-1$, either we have a clean round by time t, or exactly one agent failed in every round until round t such that only one active agent is remaining. In both cases, we should be able to decide without waiting until $t+1$. This observation leads to the following refinement to the Floodset decision protocol. Let $P^{flood+} = (P_1^{flood+}, \ldots, P_n^{flood+})$ be the decision protocol defined by the following standard program:

Program 3 P_i^{flood+}

if $m = \min\{t+1, n-1\}$ **then** $\texttt{decide}_i(\min W_i)$
else `noop`

In the following theorem, we show that we do not have common knowledge of any initial value when $m < \min\{t+1, n-1\}$.

▶ **Theorem 11.** *Let (r,m) be a point in $\mathcal{I}$. If $m < \min\{t+1, n-1\}$, then $(\mathcal{I}, r, m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$.*

Proof. Suppose (r,m) is a point in $\mathcal{I}$ with $m < \min\{t+1, n-1\}$ and suppose the local state of nonfailed agent i at (r,m) is $r_i(m) = \langle W_i, m, v_i\rangle$. We now construct an $\mathcal{N}$-reachable path to a point (r^0, m) where r^0 is failure-free. Let run r^0 be the same as run r except that we change the initial value of every agent that does not have their initial value in W_i to v_i, and no agent fails in r^0. Note that we have $v_i \in W$. Then, the local state of agent i at (r^0, m) is $r_i^0(m) = \langle W, m, v_i\rangle = r_i(m)$. Hence, we have $(r,m) \sim_i (r^0, m)$, where r^0 is failure-free. Moreover $i \in \mathcal{N}$ in both points. Since a limited information exchange protocol cannot attain common knowledge earlier than a full-information protocol, applying Lemma 4 gives the desired result. ◀

In the following theorem, we show that we have common knowledge of initial values when $m \geq \min\{t+1, n-1\}$.

▶ **Theorem 12.** *Let (r,m) be a point in $\mathcal{I}$. If $m \geq \min\{t+1, n-1\}$, then for some $v \in Values$ we have that $v = \min\{W_i\}$ and $(\mathcal{I}, r, m) \models K_i(C_{\mathcal{N}}(\exists v))$ for all $i \in \mathcal{N}(r,m)$.*

Proof. Suppose $m \geq \min\{t+1, n-1\}$ for some point (r,m) in $\mathcal{I}$. We consider two cases.

- If $t < n-1$, then $m \geq t+1$ which implies by the pigeonhole principle that we must have had a clean round at some (r,m') for $m' \leq m$. Now, by Lemma 9 and 10, we conclude that at (r,m), there exists a set W such that all nonfailed agents j $W_j(r,m) = W$, and therefore there is a unique $v = \min\{W_j(r,m)\}$. Note that Lemma 9 and 10 apply since they only depend on the information-exchange. Moreover, as all $\mathcal{N}$-reachable points from (r,m) have time m, all $\mathcal{N}$-reachable points (r',m) will also have had a clean round and hence the same set $W = W_j(r',m)$ for all nonfailed agents $j \in \mathcal{N}(r',m)$. Hence, $(\mathcal{I}, r, m) \models K_i(C_{\mathcal{N}}(\exists v))$.
- If $t = n-1$, then we have $m \geq n-1$. We can then observe that either we have had a clean round in a run r before m in which case we can conclude the claim as in the previous case, or we had one agent crash in each prior run such that $t = n-1$ agents have crashed before and no clean round occured. In that case, there is a unique agent i that is nonfailed and all other agents have crashed. Then, trivially we get $(\mathcal{I}, r, m) \models K_i(C_{\mathcal{N}}(\exists v))$ for $v = \min\{W_i(r,m)\}$.

◀

▶ **Corollary 13.** *P^{flood+} implements $\mathbf{P}^{\text{OPT}}$ with respect to information-exchange $\mathcal{E}^{flood}$.*

Proof. We show that for all points (r, m) in $\mathcal{I}$, and agents $i \in \mathcal{N}(r, m)$, we have $P^{flood+}(r_i(m)) = (\mathbf{P}_i^{\text{OPT}})^{\mathcal{I}}(r_i(m))$. There are two cases.

- If $m < \min\{t + 1, n - 1\}$, then $P^{flood+}(r_i(m)) = \texttt{noop}$. By Theorem 11, we must have $(\mathcal{I}, r, m) \models \neg C_{\mathcal{N}}(\exists v)$ for all $v \in \mathit{Values}$. Hence, $(\mathbf{P}_i^{\text{OPT}})^{\mathcal{I}}(r_i(m)) = \texttt{noop}$ also.
- If $m = \min\{t + 1, n - 1\}$, then $P^{flood+}(r_i(m)) = \texttt{decide}_i(\min\{W_i\})$. By Theorem 12, $W = W_i(r, m)$ is the same for all active i and we must have $(\mathcal{I}, r, m) \models C_{\mathcal{N}}(\exists v)$ where $v = \min\{W\}$. Moreover, for all $v' < v$, we have $v' \notin W$, and therefore $(\mathcal{I}, r, m) \models \neg C_{\mathcal{N}}(\exists v')$ since all $i \in \mathcal{N}$ considers a point where no agent has initial value v' possible (namely, the point (r, m) itself). This shows that $(\mathbf{P}_i^{\text{OPT}})^{\mathcal{I}}(r_i(m)) = \texttt{decide}_i(v)$.

It follows that P^{flood+} implements $\mathbf{P}^{\text{OPT}}$ with respect to information-exchange $\mathcal{E}^{flood}$. ◀

By Theorem 7, this result implies that the protocol P^{flood+} is optimal with respect to the information-exchange $\mathcal{E}^{flood}$.

5 Counting Extensions

The aim of this section is to explore what happens if an agent stores the number of missing messages during a round. We aim to find out whether we can achieve a better early stopping condition by making the above change to the FloodSet information-exchange protocol $\mathcal{E}^{flood}$. This modification was studied in [2] and was shown to have early stopping condition for eventual Byzantine agreement. However, there is no literature on this modification for simultaneous Byzantine agreement. Furthermore, we will extend Counting FloodSet to store the entire history of number of missing messages to investigate if we are able to obtain an earlier stopping condition. We consider this variant because [2] shows that a predicate based on the *difference* between two successive counts of missing messages leads to an EBA protocol that stops earlier than the protocol based just on a single count. Again, we are interested in whether this information is helpful in the setting of SBA.

The local states of agent i in the Counting FloodSet information-exchange protocol $\mathcal{E}^{count}$ will be either the state $crashed_i$ or a tuple $\langle W_i, h_i, m, v_i\rangle$, where W_i is the set of distinct values agent i has seen so far, h_i is the number of missing messages during round m, m is the current time, and v_i is the initial value of i. The initial state of an agent i which starts with initial value of v_i is $\langle\{v_i\}, 0, 0, v_i\rangle$. As in the FloodSet protocol, in each round, each nonfailed agent i sends a message comprised of the set W_i to all other agents. The state update function adds any values in a message W_j received from another agent to the set W_i, but additionally sets h_i to the number of agents from which i did not receive a message in the current round. The time m is incremented in each round, but the value v_i is unchanged.

The local states of agent i in the Counting FloodSet with perfect recall information-exchange protocol $\mathcal{E}^{count(pr)}$ will be a tuple $\langle W_i, h_i, m, v_i\rangle$, where W_i is the set of distinct values agent i has seen so far, h_i is a vector in which $h_i[k]$ is the number of missing messages during round k, m is the current time, and v_i is the initial value of i. The initial state of an agent i which starts with initial value of v_i is $\langle\{v_i\}, [0], 0, v_i\rangle$. Messages and state updates are as in the Counting FloodSet protocol, except that the count of missing messages is appended to h_i.

The decision rules used in the early stopping condition in both information-exchanges can be found in the following programs:

Program 4 Counting FloodSet decision rule

if $m = \min\{t+1, n-1\}$ **then** decide$(\min W_i)$
else if $h_i \geq n-1$ **then** decide$(\min W_i)$
else noop

Program 5 Counting FloodSet (perfect recall) decision rule

if $m = \min\{t+1, n-1\}$ **then** $\texttt{decide}_i(\min W_i)$
else if $\exists k \leq m(h_i[k] \geq n-1)$ **then** $\texttt{decide}_i(\min W_i)$
else noop

▶ **Lemma 14.** *The following relations hold amongst the versions of the FloodSet information exchange:*
(a) $\mathcal{E}^{count(pr)}$ *stores at least as much information as* $\mathcal{E}^{count}$.
(b) $\mathcal{E}^{count}$ *stores at least as much information as* $\mathcal{E}^{flood}$.

Proof. Suppose r^7 and r^8 are runs of the Counting FloodSet with perfect recall such that $(r^7, m) \sim_i (r^8, m)$ for agent i that is nonfailed at both (r^7, m) and (r^8, m). Because the points are indistinguishable to agent i, $r_i^7(m) = \langle W, h, m, v\rangle = r_i^8(m)$. Let r^5 and r^6 be runs of Counting FloodSet corresponding to r^7 and r^8 respectively. Note that the only difference between the two information exchanges is how they store the number of messages. $r_i^7 \sim_i r_i^8$ implies agent i receives the same number of messages every round in r^7 and r^8. This means agent i receives the same message during round m. Because the runs are corresponding, $\mathcal{N}(r^7, m) = \mathcal{N}(r^5, m)$ and $\mathcal{N}(r^8, m) = \mathcal{N}(r^6, m)$. We have $r_i^5(m) = \langle W, h[m], m, v\rangle = r_i^6(m)$ and $i \in \mathcal{N}(r^5, m) \cap \mathcal{N}(r^6, m)$.

Suppose r^3 and r^4 are runs of the Counting FloodSet such that $(r^3, m) \sim_j (r^4, m)$ for agent j that is nonfailed at both (r^3, m) and (r^4, m). Because the points are indistinguishable to agent j, $r_j^3(m) = \langle W, h, m, v\rangle = r_j^j(m)$. Let r^1 and r^2 be runs of FloodSet corresponding to r^3 and r^4 respectively. The only difference between the two information exchanges is that FloodSet does not store number of missing messages while Counting FloodSet stores the number of missing messages. Because the runs are corresponding, $\mathcal{N}(r^3, m) = \mathcal{N}(r^1, m)$ and $\mathcal{N}(r^4, m) = \mathcal{N}(r^2, m)$. Therefore, $r_j^1(m) = \langle W, m, v\rangle = r_j^2(m)$ and $j \in \mathcal{N}(r^1, m) \cap \mathcal{N}(r^2, m)$. ◀

The following results characterize the situations in which agents obtain common knowledge when using the counting variants of FloodSet. We first identify situations where agents do attain common knowledge.

▶ **Theorem 15.** *Let r be a run of the Counting FloodSet protocol, let $\mathcal{I} = \mathcal{I}_{\mathcal{E}^{count}, Crash_t}$, and let m be a time and $v \in Values$. When the local state of agent i at point (r, m) is $r_i(m) = \langle W, h, m, v_i\rangle$, where $v \in W$ and either $m \geq \min\{t+1, n-1\}$ or $h \geq n-1$, we have $(\mathcal{I}, r, m) \vDash C_{\mathcal{N}}(\exists v)$.*

Proof. By applying Lemma 14, then Proposition 6, and finally Theorem 12 we obtain $(\mathcal{I}, r, m) \vDash \bigvee_{v \in Values} C_{\mathcal{N}}(\exists v)$ when $m \geq \min\{t+1, n-1\}$. When $h \geq n-1$, this means agent i received no message from other agents during the last round. This implies i knows itself to be the only nonfailed agent, so $(\mathcal{I}, r, m) \vDash K_i(\mathcal{N} = \{i\} \wedge \exists v)$. The result now follows using Lemma 2. ◀

▶ **Theorem 16.** *Let r be a run of Counting FloodSet with perfect recall, let $\mathcal{I} = \mathcal{I}_{\mathcal{E}^{count(pr)}, Crash_t}$, and let m be a time and $v \in Values$. When the local state of agent i at point (r, m) is $r_i(m) = \langle W, h, m, v_i\rangle$, where $v \in W$ and either $m \geq \min\{t+1, n-1\}$ or there exists $1 \leq k \leq m$ such that $h[k] \geq n-1$, we have $(\mathcal{I}, r, m) \vDash C_{\mathcal{N}}(\exists v)$.*

Proof. By applying Lemma 14, then Proposition 6, and finally Theorem 15 we obtain $(\mathcal{I}, r, m) \vDash \bigvee_{v \in Values} C_{\mathcal{N}}(\exists v)$ when $m \geq \min\{t+1, n-1\}$. When $h[k] \geq n-1$, this means agent i received no message from other agents during round k. This implies i knows itself to be the only nonfailed agent, so $(\mathcal{I}, r, m) \vDash K_i(\mathcal{N} = \{i\} \wedge \exists v)$. The result now follows using Lemma 2. ◀

We now consider the converse, and show that if the complement of the conditions in the above results does not hold, then agents do *not* have common knowledge.

▶ **Theorem 17.** *Let r be a run of Counting FloodSet with perfect recall, and let $\mathcal{I} = \mathcal{I}_{\mathcal{E}^{count(pr)}, Crash_t}$. For any time $m < \min\{t+1, n-1\}$, if $r_i(m) = \langle W, h, m, v_i\rangle$ where $h[\ell] < n-1$ for all $1 \leq \ell \leq m$, we have $(\mathcal{I}, r, m) \vDash \bigwedge_{v \in Values} (\neg C_{\mathcal{N}}(\exists v))$.*

Proof. Our strategy is to define a sequence of indistinguishable runs that connect r with a failure-free run. The sequence begins with an indistinguishability relation with a run in which a nonfailed agent at (r, m) receives messages from every agent during round 1. We will define a sequence of runs r^k such that an agent that is nonfailed at (r, m) receives messages from every agent during round k.

When $t < n-1$, we always have $h[\ell] < n-1$ for all $1 \leq \ell \leq m$ and $v_i \in W$ is valid. Since $h[\ell] < n-1$, we can choose an arbitrary agent $j \neq i$ that is nonfailed at (r, m). We define the run r^1 to be run r except we change the initial value of every agent that does not have their initial value in W to v_i and during round 1 of run r^1, every agent that does not send to i during round 1 of run r sends to agent j before crashing. Then, the local state of agent i at (r^1, m) is $r_i^1(m) = \langle W, h, m, v_i\rangle = r_i(m)$. Hence, we have $(r, m) \sim_i (r^1, m)$. The local state of agent j at (r^1, m) is $r_j^1(m) = \langle W, h_j(r^1, m), m, v_j\rangle$, where $h_j(r^1, m)[1] = 0$.

We define the run r^{k+1} to be the run r^k except when k is odd, every noncrashed agent that does not send to j during round $k+1$ of run r^k sends to agent i before crashing and when k is even, every noncrashed agent that does not send to i during round $k+1$ of run r^k sends to agent j before crashing. Agent i and j are nonfailed at (r^k, m) for all r^k.

For the run r^k, where k is odd, agent j receives exactly the same messages at all times in both run r^{k+1} and r^k. The local state of agent j at (r^{k+1}, m) is $r_j^{k+1}(m) = \langle W, h_j(r^k, m), m, v_j\rangle = r_i^k(m)$. Hence, we have $(r^k, m) \sim_j (r^{k+1}, m)$. The local state of agent i at (r^{k+1}, m) is $r_i^{k+1}(m) = \langle W, h_i(r^{k+1}, m), m, v_i\rangle$, where $h_i(r^{k+1}, m)[k+1] = 0$. Similarly, when k is even, we can prove $(r^k, m) \sim_i (r^{k+1}, m)$ and $r_j^{k+1}(m) = \langle W, h_j(r^{k+1}, m), m, v_j\rangle$, where $h_j(r^{k+1}, m)[k+1] = 0$.

Therefore, we have proven (r^m, m) is $\mathcal{N}$-reachable from (r, m). When m is even, we have $r_i^m(m) = \langle W, h_i(r^m, m), m, v_i\rangle$, where $h_i(r^m, m)[m] = 0$. For the other case when m is odd, we have $r_j^m(m) = \langle W, h_j(r^m, m), m, v_j\rangle$, where $h_j(r^m, m)[m] = 0$. We define the run r^0 to be r^m with no failure. We can see $(r^m, m) \sim_i (r^0, m)$ when m is even and $(r^m, m) \sim_j (r^0, m)$ when m is odd.

Since a limited information exchange protocol cannot attain common knowledge earlier than a full-information protocol, this gives us the result. ◀

▶ **Theorem 18.** *Let r be a run of the Counting FloodSet protocol. For any time $m < \min\{t+1, n-1\}$, if $r_i(m) = \langle W, h, m, v_i\rangle$ where $h < n-1$, we have $(\mathcal{I}, r, m) \vDash \bigwedge_{v \in Values} (\neg C_{\mathcal{N}}(\exists v))$.*

Proof. If $r_i(m) = \langle W, h, m, v_i\rangle$, where $h < n-1$, then its corresponding run r' in Counting FloodSet with perfect recall would have $r_i(m) = \langle W, h', m, v_i\rangle$, where $h'[m-1] < n-1$. By applying Lemma 14, then Proposition 6, and finally Theorem 17 we obtain our result. ◀

▶ **Corollary 19.** *P^{count} ($P^{count(pr)}$) implements $\mathbf{P}^{\text{OPT}}$ with respect to information-exchange $\mathcal{E}^{count}$ (resp. $\mathcal{E}^{count(pr)}$).*

Proof. Follows from a case analysis using Theorems 15, 16, Lemma 14, and Theorem 17, analogous to the proof of Corollary 13. ◀

We have proven that Counting FloodSet and Counting FloodSet with perfect recall are optimum with respect to their information exchange due to Theorem 7.

6 Raynal's Protocol

Raynal [9] describes a protocol that differs from the FloodSet protocol by recording not just the initial values received but also which agents had these initial values. Instead of sending a message every round, an agent only sends newly received information, immediately after the round in which the agent receives the new information. As in the Floodset protocol, agents do not learn if any other agent has detected a failure. Agents also do not record the round number in which a message is received. The local states of agent i at a point (r, m) have the form $\langle V_i(r,m), New_i(r,m), time_i(r,m)\rangle$ where $V_i(r,m)$ is an agent-indexed array of initial values (or $\perp$, representing an unknown value), $New_i(r,m)$ is a set of tuples of type $Values \times Agt$ recording new information that agent i received in round m, and $time_i(r,m) = m$ is the current time. Intuitively, a tuple $(v,j) \in Values \times Agents$ represents the information that agent j has initial preference v. This information is new to agent i in round k if at the start of the round (time $k-1$), the agent has $V_i[k](r,k-1) = \perp$ and the agent receives a message $New_{i'}$ in round k containing the tuple (v,j).

Code for Raynal's protocol is given in the following figure. From this, the the information exchange protocol $\mathcal{E}^R$ for this protocol can be easily constructed. (We leave the details for the reader to complete.) For the remainder of this section, we write $\mathcal{I}$ for $\mathcal{I}_{\mathcal{E}^R, Crash_t}$.

Algorithm 6 Raynal's protocol for crash failures for agent i

```
V_i ← [⊥, ..., v_i, ..., ⊥]
New_i ← {(v_i, i)}
for r = 1, 2, ..., t + 1 do
    begin round
    if New_i ≠ ∅ then
        for each agent j ≠ i do
            send New_i to agent j
    New_i ← ∅
    W_j ← message received from agent j or ∅ if none
    for each agent j ≠ i do
        for each (v, k) ∈ W_j do
            if V_i[k] = ⊥ then
                V_i[k] ← v
                New_i ← New_i ∪ {(v, k)}
    W_j ← ∅
    end round
if ∃0 ∈ V_i then decide 0
else decide 1
```

In Raynal's presentation, the protocol waits until the end of round $t+1$ before making a decision. We show that an optimal protocol using the same information exchange can make earlier decisions. Specifically, define $\beta_(r,m)$ be the number of $\perp$ values in the array $V_i(r,m)$. We show that the implementation of the knowledge based program uses the following predicate of agent i's local state to determine the time m at which a decision can be made: $m > \min\{t+1, n-1\} - \max\{1, \beta_i(r,m)\}$.

The proof considers a number of cases depending on whether $m \leq 1$, $New_i(r,m) = \emptyset$, $\beta_i(r,m) = 0$ and $m \leq \min\{t+1, n-1\} - \beta_i(r,m)$. We first note a property of the protocol after a clean round.

▶ **Lemma 20.** *If round m is a clean round during a run r of Raynal's protocol, then $V_i(r,m) =$*

$V_j(r,m)$ for nonfailed agents i and j at (r,m).

Proof. Suppose we have two nonfailed agents i and j at (r,m). Since we have crash failures only, if $V_i(r,m) \neq V_j(r,m)$, then there exists an agent k such that, without loss of generality, $V_i[k](r,m) \neq \perp$ and $V_j(r,m)[k] = \perp$. If $V_i[k](r,m) = v \neq \perp$, then agent i must have have had $(v,k) \in New_i(r,m')$ for some $m' \leq m$ Since agent i has not failed yet, if $(v,k) \in New_i(r,m')$ for $m' < m$, it would have sent (v,k) to agent j in round $m'+1$ and we would have $V_j[k](r,m) = V_i[k](r,m)$. If agent i receives (v,k) during round m, then also $V_j[k](r,m) = V_i[k](r,m)$ because round m is clean so agent i and j receive the same messages. ◀

▶ **Lemma 21.** *Let round m be clean during a run r of Raynal's protocol. For any time $m' > m$ and nonfailed agent i, we have $New_i(r,m') = \emptyset$.*

Proof. From Lemma 20, we know that $V_i(r,m) = V_j(r,m)$ for any agents i and j that are nonfailed at time m. Note that if agent i sends a tuple (v,k) in round $m+1$ then $V_i[k](r,m) = v = V_j[k](r,m)$. Hence $New_j(r,m+1) = \emptyset$ for all nonfailed agents j. It follows that no messages are sent after round $m+1$, and we have $New_j(r,m') = \emptyset$ for all $m' > m$. ◀

Moses and Tuttle [8] have proven the following lemma.

▶ **Lemma 22.** *[8, Corollary 6] Let φ be a proposition about the global state. Let r and r' be corresponding runs of an arbitrary protocol $(P,\mathcal{E})$ and a full-information protocol $(P',\mathcal{E}')$ respectively. If $(\mathcal{I}_{P,\mathcal{E}}, r, m) \vDash C_{\mathcal{N}}\varphi$, then $(\mathcal{I}_{P',\mathcal{E}'}, r', m) \vDash C_{\mathcal{N}}\varphi$.*

First, we deal with the case $m = 1$. We define $\beta_i(r,m)$ to be the number of $\perp$ in the array $V_i(r,m)$ of agent i at (r,m).

▶ **Theorem 23.** *Let r be a run of Raynal's protocol and agent i be a nonfailed agent at $(r,1)$. When $t < n-1$, we have $(\mathcal{I},r,1) \vDash \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$ and when $\beta_i(r,1) = t = n-1$, we have $(\mathcal{I},r,1) \vDash C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$.*

Proof. For the case $t < n-1$, we know that a full-information protocol cannot decide at time 1. Hence, Lemma 22 and the fact that we must have $C_{\mathcal{N}}(\exists v)$ when deciding v gives us the result.

For the case $\beta_i(r,1) = t = n-1$, agent i knows $n-1$ agents crashed and hence it would know it is the only nonfailed agent. ◀

The following lemma is about the case where $\beta_i(r,m) = 0$ and $New_i(r,m) = \emptyset$. Note that since $n \geq 2$, we cannot have this condition true at time $m \leq 1$.

▶ **Lemma 24.** *Let r be a run in Raynal's protocol such that a nonfailed agent i has $\beta_i(r,m) = 0$ and $New_i(r,m) = \emptyset$, and $m < \min\{t+1, n-1\}$. Then we have $(\mathcal{I},r,m) \vDash \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$.*

Proof. Let $v_1,\ldots,v_n$ be the initial values of the agents in r. Let r' be a failure-free run with the same initial values as r. We can see $r'_i(m) = \langle [v_1,\ldots,v_n], \emptyset, m\rangle = r_i(m)$. From Lemma 4, we know that a failure-free run requires $\min\{t+1, n-1\}$ rounds to reach common knowledge, and the converse of Lemma 22 gives the result. ◀

The following lemma deals with the case where agent i has $\beta_i(r,m) > 0$ and $New_i(r,m) = \emptyset$.

▶ **Lemma 25.** *Let r be a run in Raynal's protocol such that agent $i \in \mathcal{N}(r,m)$ has $\beta_i(r,m) > 0$ and $New_i(r,m) = \emptyset$. For any time m with $1 < m \leq \min\{t+1, n-1\} - \beta_i(r,m)$ we have $(\mathcal{I},r,m) \vDash \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$.*

Proof. Let the set of agents $j \neq i$ with $V_i[j](r,m) \neq \bot$ be $R = \{i_1, i_2 \ldots\}$. The number of i_k is $n - \beta_i(r,m) - 1$. By the inequality, this number is at least 2. Let $B = Agt \setminus (\{i\} \cup R)$ be the agents $j \neq i$ for which $V_i[j](r,m) = \bot$.

We construct a run r' in which agents have the same initial values as in r, but in which the faulty agents are $B \cup \{i_1 \ldots i_{m-1}\}$. Note that the number of these agents is $\beta_i(r,m) + m - 1$ which is at most t, by the assumed inequality. The failure pattern of r' is as follows. Every agent in B only sends a message to agent i_1 before crashing during round 1. At round k for $2 \leq k \leq m$, agent i_{k-1} sends only to agent i_k before it crashes. Agent i receives all values from agents R in round 1, but none of the values from agents in B reach i by time m. Hence, the local state of agent i at time m is $r'_i(m) = \langle V_i(r,m), \emptyset, m\rangle$. Hence, $(r',m) \sim_i (r,m)$.

In the run r', agent i_m receives all values of agents in $R \cup \{i\} \setminus \{i_m\}$ in round 1, and the values of all agents in B in round m. Thus $V_{i_m}(r',m)$ is the vector of all initial values, $\beta_{i_m}(r',m) = 0$ and $New_{i_m}(r',m) = \{(v_j, j) \mid j \in B\}$.

We define r'' to be r' except that agent i_{m-1} is nonfaulty at (r'',m). This run has one fewer failure than r', so also has at most t failures. The local state of agent i_m at (r'',m) is the same as the local state of i_m at (r',m). Hence, $(r'',m) \sim_{i_m} (r',m)$. In run r'', agent i_{m-1} has received all values by time $m-1$, so $\beta_{i_{m-1}}(r'',m) = 0$ and $New_{i_{m-1}}(r'',m) = \emptyset$. By the inequality and the fact that $\beta_i(r,m) > 0$, we have $m < \min(t+1, n-1)$. We can now apply Lemma 24 to conclude $(\mathcal{I}, r'', m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$. Since also $(r,m) \sim_{\mathcal{N}} (r'',m)$, it follows that $(\mathcal{I}, r, m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$. ◀

The following lemma deals with the case where the local state of agent i has $\beta_i(r,m) > 0$ and $New_i(r,m) \neq \emptyset$. We denote the number of agents that crashed between $(r, m-1)$ and (r,m) by $f(r,m)$.

▶ **Lemma 26.** *Suppose r is a run in Raynal's protocol and let m be a time such that $1 < m \leq \min\{t+1, n-1\} - \beta_i(r,m)$, agent i is active at (r,m), and we have $\beta_i(r,m) > 0$ and $New_i(r,m) \neq \emptyset$. Then $(\mathcal{I}, r, m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$.*

Proof. In Raynal's protocol, agent i would know that if $(v,j) \in New_i(r,m)$ then j has crashed during round 1 (else i would have received (v,j) in round 1 and this tuple would not be new by time $m > 1$). Moreover, $V_i[j](r,m) \neq \bot$, so j is not counted in $\beta_i(r,m)$. From Lemma 21, if $New_i(r,m) \neq \emptyset$ for $m > 1$, then it implies every round between round 1 and round $m-1$ is dirty. Hence $\beta_i(r,m) + |New_i(r,m)| \leq f(r,1)$ and $1 \leq f(r,k)$ for $1 \leq k < m$. Hence, this would mean $\beta(r,m) + |New_i(r,m)| + m - 2 \leq \sum_{k=1}^{m-1} f(r,k) \leq t$. Therefore, we must have $m \leq t + 2 - \beta(r,m) - |New_i(r,m)|$ for any run r of Raynal's protocol when $New_i(r,m) \neq \emptyset$ and $\beta_i(r,m) > 0$.

Let B be the set of agents k with $V_i[k](r,m) = \bot$, let R be the agents with $(v_j, i_j) \in New_i(r,m)$ for some value v_j, and let $i_1 \ldots i_k$ be the remaining agents, excluding i. We define the run r' be the run in which all agents have the same initial values as in r, in which all the agents in B crash in round 1 before sending any messages, and every agent in R also crashes in round 1, and sends a message only to agent i_1. In round k, for $2 \leq k \leq m-1$ agent i_k crashes and sends a message only to agent i_{k+1}. All other agents are nonfailed. There are $\beta_i(r,m) + |New_i(r,m)| + m - 2$ failures in r'. By the above reasoning, this is consistent with the failure model of at most t failures. Note that since agent i_m is nonfailed , in round r', agent i first receives the values of agents in R from i_m in round m, and we have $New_i(r',m) = New_i(r,m)$ and $V_i[j](r',m) = V_i[j](r',m)$ for $j \in R$. For agents $j$in B, we have $V_i[j](r',m) = \bot = V_i[j](r',m)$. Since the agents $i_1, \ldots$ do not crash in round 1, we have $V_i[i_j](r',m) = V_i[i_j](r,m)$ for each such agent i_j. Thus, we have $V_i(r',m) = V_i(r,m)$. The local state of agent i at (r',m) is $r'_i(m) = \langle V_i(r',m), New_i(r',m), m\rangle = \langle V_i(r,m), New_i(r,m), m\rangle = r_i(m)$. Hence, $(r',m) \sim_i (r,m)$. The local state of nonfailed agent i_m at (r',m) has $New_{i_m}(r'm) =$

$\emptyset$ and $V_{i_m}(r', m) = V_i(r, m)$, hence $\beta_{i_m}(r', m) = \beta_i(r, m) > 0$. By assumption, we have $1 < m \leq \min\{t + 1, n - 1\} - \beta_i(r, m)$ and $\beta_i(r, m) > 0$, so $1 < m < \min\{t + 1, n - 1\}$. Therefore, we may apply Lemma 25 to obtain $(\mathcal{I}, r', m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$. Since agent i is active at both (r, m) and (r', m), we have $(\mathcal{I}, r, m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$. ◀

Now we prove the result for $\beta_i(r, m) = 0$ and $New_i(r, m) \neq \emptyset$, where $m > 1$.

▶ **Lemma 27.** *Suppose r is a run in Raynal's protocol and let m be a time such that $1 < m \leq \min\{t + 1, n - 1\} - \beta_i(r, m)$, agent i is active at (r, m), and we have $\beta_i(r, m) = 0$ and $New_i(r, m) \neq \emptyset$. Then $(\mathcal{I}, r, m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$.*

Proof. For $1 \leq k < m$, we let agent i_k be the k-th agent that has non-$\perp$ values in $V_i(r, m)$ such that $i_k \neq i$. We set $i_m = i$. We define run r' such that all agents in $New_i(r, m)$ crashes during round 1 and sends only to agent i_1. There are no other crash during round 1. During round $2 \leq k < m - 1$, agent i_{k-1} crashes and sends only to agent i_k. No failure occur during round m of r'. Then, we have $r_i(m) = \langle [v_1, \ldots, v_n], New_i(r, m), m\rangle = r'_i(m)$. Hence, $(r, m) \sim_i (r', m)$. The local state of nonfailed agent i_{m-1} in (r', m) is $r'_{m-1}(m) = \langle [v_1, \ldots, v_n], \emptyset, m\rangle$. This shows (r', m) is indistinguishable to a failure-free run. We can now apply Lemma 4. ◀

The following theorem concludes when do we not have common knowledge in Raynal's protocol at times greater than 1.

▶ **Theorem 28.** *Let r be any run in Raynal's protocol. For any time m with $1 < m \leq \min\{t + 1, n - 1\} - \max\{1, \beta_i(r, m)\}$ we have $(\mathcal{I}, r, m) \models \neg(C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1))$.*

Proof. Note that when $\beta_i(r, m) = 0$, the inequality is the same as the inequality from Lemma 24. When $\beta_i(r, m) \neq 0$, the inequality is the same as the inequality in Lemma 25 to Lemma 26. Let i be a nonfailed agent at (r, m). When $New_i(r, m) \neq \emptyset$, we use Lemma 26. When $New_i(r, m) = \emptyset$ and $\beta_i(r, m) > 0$, we use Lemma 25. When $New_i(r, m) = \emptyset$ and $\beta_i(r, m) = 0$, we use Lemma 24. When $New_i(r, m) \neq \emptyset$ and $\beta_i(r, m) = 0$, we use Lemma 27. ◀

▶ **Lemma 29.** *Let r be a run of Raynal's protocol and i be a nonfailed agent at (r, m). If $(\mathcal{I}, r, m) \models time > \min\{t + 1, n - 1\} - \max\{1, \beta_i\}$, then $(\mathcal{I}, r, m) \models K_i(clean \wedge time > \min\{t + 1, n - 1\} - \max\{1, \beta_i\})$.*

Proof. Let $(r, m) \sim_i (r', m)$. Since the points are indistinguishable, we have $V_i(r, m) = V_i(r', m)$ and $\beta_i(r, m) = \beta_i(r', m)$. We observe that $\beta_i(r', m)$ is less than or equal to the number of failures that happen during round 1, this implies $\beta_i(r', m) \leq f(r', 1)$. We must have $t \geq \sum_{k=1}^{m} f(r', k) = f(r', 1) + \sum_{k=2}^{m} f(r', k)$. This means there are at most $t - \beta_i(r', m) \geq t - f(r', 1) \geq \sum_{k=2}^{m} f(r', k)$ failures between $(r', 1)$ and (r', m). There are $m - 1$ rounds between $(r', 1)$ and (r', m). In each of the cases $\beta_i(r', m) = 0$ and $\beta_i(r', m) > 0$, by the pigeonhole principle, a clean round must have occured if $t - \max\{1, \beta_i(r', m)\} < m - 1$ for $t < n - 1$. Since t, n are common knowledge in the system and in synchronous systems the time $time$ is common knowledge, we have $(\mathcal{I}, r', m) \models clean \wedge time > \min\{t + 1, n - 1\} - \max\{1, \beta_i\}$. ◀

The following theorem concludes when do we have common knowledge in Raynal's protocol.

▶ **Theorem 30.** *Let i be a nonfailed agent of run r of Raynal's protocol, we have $(\mathcal{I}, r, m) \models C_{\mathcal{N}}(\exists 0) \vee C_{\mathcal{N}}(\exists 1)$ when $m > \min\{t + 1, n - 1\} - \max\{1, \beta_i(r, m)\}$.*

Proof. Write I_j for the formula $time > \min\{t + 1, n - 1\} - \max\{1, \beta_j\}$. In the crash failures model, with $\mathcal{N}(r, m)$ interpreted as the set of nonfailed agents at the point (r, m), we have that

$i \in \mathcal{N} \Rightarrow K_i(i \in \mathcal{N})$ is valid. Given $(\mathcal{I}, r, m) \models i \in \mathcal{N} \wedge I_i$, Lemma 29 implies $(\mathcal{I}, r, m) \models K_i(\mathit{clean} \wedge i \in \mathcal{N} \wedge I_i)$. Hence $(\mathcal{I}, r, m) \models K_i(\mathit{clean} \wedge \bigvee_{j \in \mathcal{N}} I_j)$.

By the knowledge axiom, $(\mathcal{I}, r, m) \models \mathit{clean}$. This means a clean round has occurred by (r, m). By Lemma 20 and the definition of β_i, we get $\beta_i(r, m) = \beta_j(r, m)$ for all nonfailed agents j at (r, m). Therefore, by the previous paragraph, we have $(\mathcal{I}, r, m) \models E_{\mathcal{N}}(\mathit{clean} \wedge \bigvee_{j \in \mathcal{N}} I_j)$. This shows that $\bigvee_{j \in \mathcal{N}} I_j \Rightarrow E_{\mathcal{N}}(\mathit{clean} \wedge \bigvee_{j \in \mathcal{N}} I_j)$ By the induction rule, we obtain $(\mathcal{I}, r, m) \models C_{\mathcal{N}} \mathit{clean}$.

Let p_v be the proposition $\bigvee_{j \in \mathcal{N}} \bigvee_k V_j[k] = v$. That is, p_v says that some nonfailed agent sees the value v. After a clean round, we have $V_i(r, m) = V_j(r, m)$ for all $j \in \mathcal{N}(r, m)$. This means we have $\models (p_v \wedge C_{\mathcal{N}} \mathit{clean}) \Rightarrow E_{\mathcal{N}}(p_v \wedge C_{\mathcal{N}} \mathit{clean})$. By induction, we obtain $\models (p_v \wedge C_{\mathcal{N}} \mathit{clean}) \Rightarrow C_{\mathcal{N}}(p_v)$. Since we have $(\mathcal{I}, r, m) \models C_{\mathcal{N}} \mathit{clean}$, it follows, taking v to be a value seen by some nonfaulty agent at (r, m), that $(\mathcal{I}, r, m) \models C_{\mathcal{N}} p_v$. This implies $(\mathcal{I}, r, m) \models C_{\mathcal{N}} \exists v$. ◀

We have found that Raynal's protocol is not optimal with respect to its information exchange. An optimum early stopping condition is when $m > \min\{t + 1, n - 1\} - \max\{1, \beta_i(r, m)\}$.

7 Conclusion

We have studied the FloodSet, Counting FloodSet, Counting FloodSet with perfect recall and Raynal's protocol for simultaneous Byzantine agreement in synchronous message-passing systems prone to crash failures. We all of the above protocol, we have determined the earlier possible time that common knowledge about the initial values is attained.

For FloodSet and its variants, the early stopping condition is trivial. On the other hand, Raynal's protocol has an interesting nontrivial early stopping condition. It would be interesting to see if we can achieve nontrivial early stopping condition if we not only store the count of the number of missing messages but also send the count to other agents.

Protocol	Computation	Message size	Space
FloodSet [6]	$O(n)$	$O(1)$	$O(\log t)$
Counting FloodSet	$O(n)$	$O(1)$	$O(\log t)$
Raynal's protocol [9]	$O(n^2)$	$O(n)$	$O(n)$
Dwork and Moses [3]	$O(nt)$	$O(t)$	$O(t)$

Table 1 Complexity and message size for each protocol we studied

In this paper, we considered the problem only under crash failures. It would be more interesting to study the problem under general omission failures because an optimal protocol for simultaneous Byzantine agreement in synchronous systems prone to general omission failures require NP-hard computations. We would like to obtain a protocol for general omissions with a nontrivial early stopping condition requiring only polynomial time computations.

References

1 Kaya Alpturer, Joseph Y. Halpern, and Ron van der Meyden. Optimal eventual byzantine agreement protocols with omission failures. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, PODC '23, page 244–252, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3583668.3594573.

2 Armando Castañeda, Yoram Moses, Michel Raynal, and Matthieu Roy. Early decision and stopping in synchronous consensus: A predicate-based guided tour. In Amr El Abbadi and Benoît Garbinato, editors, *Networked Systems*, pages 206–221, Cham, 2017. Springer International Publishing.

3 Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, 1990. URL: https://www.sciencedirect.com/science/article/pii/0890540190900149, doi:https://doi.org/10.1016/0890-5401(90)90014-9.

4 R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.

5 Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *J. ACM*, 37(3):549–587, jul 1990. doi:10.1145/79147.79161.

6 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

7 Yoram Moses. Optimum simultaneous consensus for general omissions is equivalent to an NP oracle. In Idit Keidar, editor, *Proc. Distributed Computing, 23rd International Symposium, DISC 2009*, volume 5805 of *Lecture Notes in Computer Science*, pages 436–448. Springer, 2009. doi:10.1007/978-3-642-04355-0_45.

8 Yoram Moses and Mark R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988. doi:10.1007/BF01762112.

9 M. Raynal. Consensus in synchronous systems: a concise guided tour. In *2002 Pacific Rim International Symposium on Dependable Computing, 2002. Proceedings.*, pages 221–228, 2002. doi:10.1109/PRDC.2002.1185641.

10 Ron van der Meyden. Optimal simultaneous byzantine agreement, common knowledge and limited information exchange. 2024.