Collaborative Situational Awareness for Conflict-Aware Flight Planning

Saswata Paul, Stacy Patterson, and Carlos A. Varela

Department of Computer Science
Rensselaer Polytechnic Institute, Troy, New York, 12180
pauls4@rpi.edu, {sep, cvarela}@cs.rpi.edu

Abstract—In autonomous air-traffic management scenarios of the future, manned and unmanned aircraft will be able to safely navigate through the National Airspace System, independent of centralized air-traffic controllers, by sharing critical data necessary for maintaining standard separation with each other. Under such conditions, every aircraft must have sufficient knowledge about other aircraft sharing the airspace to operate safely. In this paper, we specify such a state of knowledge and present a formally verified distributed knowledge propagation protocol, which guarantees that this state will eventually be attained, leading to heightened collaborative situational awareness among the aircraft. We use the TLA+ Specification Language to specify our protocol and some safety-critical correctness properties. We also provide mechanically-verified proofs of the correctness properties, under a set of suitable operating conditions of the system, by using the TLA+ Proof System.

I. INTRODUCTION

In the near future, civilian applications of unmanned aerial vehicles (UAVs) for purposes such as package delivery and scientific surveillance, and the use of micro-aircraft for urban transportation will lead to a significant increase in the density of aircraft in the National Airspace System (NAS). The current system of human-operated air-traffic control (ATC) is prone to human errors and is not scalable in the face of high-density air-traffic, rendering it ill-suited for use in Urban Air Mobility (UAM) scenarios. ATC errors can cause loss of standard separation between aircraft, leading to near mid-air collisions (NMACs) and wake-vortex induced rolls, which can be catastrophic. For this reason, it will be imperative for both manned and unmanned aircraft to have the ability to independently navigate through the NAS while maintaining standard separation from other aircraft. The future of aviation will see the advent of "smarter" air-traffic management (ATM) protocols that will be capable of gathering real-time data from multiple sources and using it to enhance the overall situational awareness of pilots and flight-control systems. Aircraft navigation systems of the future will, therefore, be capable of autonomously and safely navigating through the NAS without being dependent on human-operated ATC (free-flight). The advantages of such automated techniques over traditional human-operated ATC include that they are faster [1], can be formally verified for correctness [2], and can be efficiently scaled over significantly larger numbers of aircraft [3].

We envision a formally verified approach for achieving collaborative situational awareness among network-connected aircraft in the NAS. This capability will stem from the availability of a vast amount of real-time data that will be shared among them via a dedicated vehicle-to-vehicle (V2V) network which we term as the Internet-of-Planes (IoP). Heightened collaborative situational awareness will, in turn, allow the aircraft in the NAS to maintain standard separation among themselves by employing a cooperative conflict-aware flight planning approach. In conflict-aware flight planning [4], flight data from all traffic aircraft is used by an ownship to avoid possible NMACs in the flight planning stage itself (Fig. 1). This reduces the reliance on tactical collision avoidance approaches like TCAS [5] which often require instantaneous responses from pilots or flight-control systems and cannot be used in low-altitude terminal areas.

For any data to be used for cooperative flight planning, all participating aircraft first need to agree on the same data. Consensus protocols like the Synod Protocol [6] and the Paxos Protocol [6] allow participants in multi-agent distributed systems to reach agreement on data. However, simply reaching agreement on some data is not sufficient in decentralized safety1-critical air-traffic control applications. The next step in cooperative flight planning requires that every participating aircraft knows two facts – (1) the data φ that has been agreed upon, and (2) that every other participating aircraft

1 Safely here implies protection from harm to life, environment, or property.
has knowledge of $\phi$ [7]. This state of knowledge, which is represented by $E^2\phi$ in knowledge logic, is crucial in multi-agent systems because unless every agent is aware that all other agents know some rule, they cannot trust the other agents’ actions to be governed by that rule, even if there is an implicit trust that all agents follow any rule that they know. The absence of $E^2\phi$ would create a scenario in which an agent would not be able to trust that the system is "safe" to operate in, thus preventing it from performing its actions.

The failure of a safety-critical aerospace system can be catastrophic to life, environment, or property [8]. Therefore, for any software that is used in such systems, the guarantees provided by the software must be extensively verified to ensure correctness. Formal methods facilitate the verification of such software by providing tools and techniques for mechanically checking the formal proofs of these guarantees. The TLA+ [9] specification language is designed for specifying concurrent systems and their properties and verifying them using the TLA+ Proof System (TLAPS) [10]. TLAPS can be used to write structured hierarchical proofs [11] which are automatically converted to individual proof obligations and solved by automated theorem provers like Isabelle [12], Zenon [13], Yices [14], CVC3 [15], Z3 [16], and veriT [17].

In this paper, we have formalized the definition of a "safe" state of knowledge and presented a protocol for distributed knowledge propagation in the IoP which enables network-connected aircraft to achieve $E^2\phi$ for a value $\phi$, thus giving them the ability to operate in the airspace with explicit confidence in the safety of the system. Our protocol allows an aircraft to propagate $\phi$ among all other aircraft and learn when $E^2\phi$ has been successfully achieved. We have also identified certain system conditions under which it can be guaranteed that our protocol satisfies two main correctness properties - safety\(^2\) and progress. We have used these conditions to provide formal proofs of the correctness properties. We have specified our protocol in TLA* and mechanically checked the proofs of the correctness properties in TLAPS.

The rest of the paper is arranged as follows: in Section II, we discuss related work on knowledge states in distributed systems and formal verification of knowledge propagation protocols; in Section III, we formalize the concept of a safe state of knowledge and present the problem statement; in Section IV, we present our knowledge propagation protocol and its correctness properties; in Section V, we identify the conditions required for proving the correctness properties for our protocol; in Section VI we discuss how our protocol may be used for cooperative flight planning; in Section VII, we formally specify our protocol; in Section VIII, we present the correctness proofs; and finally in Section IX, we end the paper with a conclusion and future directions of work.

II. Related Work

There exists prior work in the literature on reasoning about knowledge states in distributed multi-agent systems. Halpern et al. [18] present a formal model to capture the interaction between knowledge and action in distributed systems by modeling the distributed system as a set of runs. They define a run as a function from time to global states of the system. They define knowledge based protocols where a process’ actions depend on its knowledge and can be used to describe high-level descriptions of a process’ behavior depending on its local state. Kseihmkalani et al. [19] examine the feasibility of achieving $E^n\phi$ for $n > 1$ and use a restricted distributed messaging framework to explore the possibility of achieving $E^n\phi$ in asynchronous systems by using only logical clocks. They also explore the use of different types of logical clocks for attaining concurrent common knowledge by using their framework. Fagin et al. [20] provide a model for explicitly incorporating probability in knowledge logic formulas to reason about knowledge and probability. They introduce a probabilistic variant of common knowledge in multi-agent systems and provide fundamental axioms for reasoning about the same. Fagin et al. [21] investigate a model of distributed communication and provide a logical formalization of runs. They also analyze the logic of implicit knowledge, which is the knowledge that can be attained by combining the knowledge of a group. Their work explores the dependence of knowledge in a distributed system on the way processes communicate with one another. Guzmán et al. [22] introduce the theory of group space functions to reason about the information distributed among the members of a potentially infinite group. They develop the semantic foundations and algorithms to reason about distributed knowledge in multi-agent systems and analyze the properties of distributed spaces for reasoning about the distributed knowledge of such systems. Matteo [23] presents a formal model to capture the interaction between knowledge and action in distributed systems by using only logical clocks. Kuznets et al. [27] propose a framework for reasoning about knowledge in multi-agent asynchronous systems with Byzantine fault failures. Their framework combines epistemic and temporal logic for specifying distributed protocols and their behaviors. The modularity of
their approach allows modeling of any timing and synchrony properties of distributed systems. Knight et al. [28] propose a logic for reasoning about epistemic messages in asynchronous distributed systems where knowledge is true at the start of announcement and agents can imagine messages that have been sent, but not yet received. They extend the Public Announcement Logic [29] in which announcements from an external source can be used to model messages broadcast by agents within the system. Halpern [30] presents a survey of formalizations of distributed knowledge in asynchronous and unreliable distributed systems. They also provide examples to argue why this formalization of distributed knowledge is important for analyzing distributed systems. Dworket al. [31] present a general framework for formalizing and reasoning about knowledge in distributed systems. They formalize the relationship between common knowledge, global knowledge, and other states of distributed knowledge and show that in real-life distributed systems, common knowledge cannot be attained. They also investigate other weaker states of distributed knowledge that are practically attainable. Halpern et al. [32] present a categorization of epistemic and temporal logic along two dimensions: the language used and the assumptions about the underlying distributed systems. They use these categorizations to introduce a ninety-six logics, which they investigate by characterizing their complexity and their dependence on parameter combinations. Choi et al. [33] introduce a class of new consensus protocols for distributed networks using a directed acyclic graph-based structure called the OPERA chain. They prove eventual consensus for their protocols without using partial synchrony, leader-election, round-robin, or proof-of-work. They present an analysis of their approach using Lamport time-stamps and concurrent common knowledge. However, none of this work has presented any mechanically-verified knowledge propagation protocols.

Previous work on formal verification of atomic commit protocols has been limited to model checking. Popovic et al. [34] have presented a model checking based approach for analysis of distributed transaction management protocols using the SMV formal verification tool and have used their approach for the verification of the Two-Phase Commit protocol. Atif [35] has presented an analysis of Two-Phase Commit and its improved variant, the Three-Phase Commit protocol, by using the process algebra mCRL2 and modal μ-calculus. They have model checked both variants of the protocol under different communication settings to analyze their behavior in practical distributed networks. None of the above work has presented any mechanically-verified theorems that can guarantee that correctness properties will hold in infinite input states.

We improve upon prior work by presenting a formally-verified distributed knowledge propagation protocol that can be used to attain a safe state of knowledge in multi-agent distributed systems like the IoT and providing machine-checked formal proofs of some correctness guarantees necessary for safety-critical aerospace applications. In the next section, we elaborate on the concept of a safe state of knowledge for cooperative flight planning and present our problem statement.

III. COLLABORATIVE SITUATIONAL AWARENESS

In this section, we discuss how collaborative situational awareness can be achieved in a network of connected aircraft for cooperative flight planning.

![Fig. 2: Aircraft trying to enter an occupied airspace (top view).](image)

We envision a decentralized air-traffic control system in which, if there are $N \geq 1$ aircraft already inside an airspace, an $(N+1)^{th}$ aircraft which wants to enter the airspace (Fig. 2) has to do the following:

1) Produce a provably conflict-free set of flight-plans with the $N$ aircraft already inside the airspace.

2) Get the $N$ aircraft already inside the airspace to agree on the new set of $N + 1$ flight-plans.

3) Communicate the knowledge of this new set among all aircraft relevant to the airspace in question.

Previously, we have proposed a formally verified conflict-aware flight planning algorithm [4] which allows an aircraft to generate a safe set of flight-plans. We assume that if one or more aircraft propose one or more safe set of flight-plans, eventually the second step, where all aircraft inside the airspace agree on a unique safe set $\phi_i$ will be completed, possibly by using a consensus algorithm. Therefore, in this paper we focus on the third step for attaining collaborative situational awareness, i.e., propagating the knowledge of $\phi_i$ among all aircraft relevant to the airspace in question.

For cooperative flight planning, the following sets of aircraft are relevant to an airspace:

- **C1** The aircraft allowed entry by the agreement on $\phi_i$.
- **C2** The aircraft already inside the airspace.
- **C3** The aircraft expected to try to enter the airspace.

By definition, $\phi_i$ contains information about all aircraft in $C1$ and $C2$. We assume that there is some mechanism, perhaps one that uses ADS-B [36] based intent-broadcast [37], that provides the information of aircraft in $C3$. The set $S : S = C1 \cup C2 \cup C3$ represents all aircraft relevant to an airspace.

As mentioned in Section I, the goal of the knowledge propagation phase is to ensure a "safe" state of knowledge in which all aircraft can feel safe to operate. In knowledge logic [7], the expression $E_{i\geq} \phi$ represents that an agent $i$ knows the fact $\phi$ and $E \phi$ represents that all agents in the system know $\phi$. Therefore, we can now define a safe state of knowledge as the state $EE\phi$ or $E^2\phi$ which implies that every agent knows
that every agent knows $\phi$. Therefore, $E^2\phi$ implies that every aircraft knows two facts:

1) $\phi$ – The safe set of flight-plans.
2) $E\phi$ – Every agent knows $\phi$.

The problem of knowledge propagation for cooperative flight planning can now be precisely stated as – “Given a set of aircraft $K : K \subset S$ in which all aircraft know the same value $\phi$ but do not know the membership of the set $K$, each aircraft in the set $K$ should try to propagate the knowledge of $\phi$ in the set $S$ to attain $E^2\phi$ and at least one member of the set $K$ should eventually learn that $E^2\phi$ has been attained.”

The knowledge of $\phi$ can be propagated to all aircraft in $S$ by using atomic commit protocols [38] which can ensure that a value is committed at all nodes. In the next section, we will introduce a distributed knowledge propagation protocol that can be used for cooperative flight planning applications.

IV. OUR KNOWLEDGE PROPAGATION PROTOCOL

In this section, we propose a distributed knowledge propagation protocol that can be used for propagating a value $\phi$ to a group of distributed nodes to eventually achieve $E^2\phi$.

The System Model

We consider an asynchronous, non-Byzantine system model in which agents operate at arbitrary speed and may fail temporarily. We also assume reliable messaging [39] where message delivery is guaranteed. Messages can be duplicated and have arbitrary transmission times, but cannot be corrupted.

The Protocol

There is a non-empty set of coordinators, and a logically separate non-empty set of replicas. A unique value $\phi$ is known to all coordinators. $E^2\phi$, in the context of our protocol, implies that all replicas know that all other replicas know $\phi$. The goal of every coordinator is to propagate $\phi$ and eventually learn that $E^2\phi$ has been achieved. Coordinators and replicas are logical abstractions and may be functionally implemented by the same physical node (e.g., an aircraft) simultaneously.

The protocol operates in the following phases:

- **Phase 1**
  (a) A coordinator sends a learn (“1a”) message with a value $v$ to all replicas.
  (b) A replica learns a value $v$ iff it receives a learn message with the value $v$ from a coordinator and it replies back to the coordinator with a learnt (“1b”) message iff it has learnt a value $v$.

- **Phase 2**
  (a) A coordinator sends an all-know (“2a”) message to each replica iff it has received a learnt message from all replicas.
  (b) A replica learns that all replicas know the value $v$ iff it receives an all-know message from a coordinator and it replies back to the coordinator with an acknowledgement (“2b”) message iff it has learnt that all replicas know the value $v$.

(c) A coordinator learns of $E^2\phi$ iff it has received acknowledgement message from all replicas.

Required Correctness Properties

For use in safety-critical systems, the protocol must satisfy the following correctness properties:

- **Safety** - This property implies that a coordinator will learn that $E^2\phi$ has been attained iff all replicas know $E\phi$ and $\phi$. This property is important because it ensures the safe state of knowledge that is described in Section III.
- **Progress** - The protocol should ensure that eventually, $E^2\phi$ is attained. This property is important for applications where an eventual outcome is necessary.

Informal Analysis of the Protocol

The system model described earlier in this section is not sufficient to guarantee that the required correctness properties will be satisfied – e.g., if even one replica fails, the protocol will never be able to make progress. Therefore, to guarantee the correctness properties, we need to make certain additional assumptions about the system. We list some important observations that will be helpful for identifying such assumptions.

- All the coordinators try to commit the same value, removing the competition for votes. Therefore, a replica can respond to ”1b” and ”2b” messages from multiple coordinators, even if they have already committed a value.
- Since successful propagation requires a value to be replicated in all replicas, the protocol cannot tolerate the failure of even one replica.
- The non-Byzantine behavior of available agents directs that some sets of operations are atomic in nature – e.g., a coordinator has to send ”2a” messages if it has received ”1b” messages from all replicas and it has to receive ”1b” messages from all replicas to send ”2a” messages.
- If an agent receives messages that had not been sent, then its non-Byzantine behavior cannot ensure correctness – e.g., if a replica receives ”2a” messages that had not been sent, it will incorrectly learn that all replicas know a value and respond with ”2b” messages, affecting safety.

From the above observations, we can see that a replica can handle requests from multiple coordinators. However, the protocol cannot tolerate the failure of even one replica. Therefore, in the next section, we will identify a set of conditions that will allow us to formally prove the correctness properties and mechanically verify the proofs.

V. IDENTIFYING THE CONDITIONS FOR FORMALLY PROVING CORRECTNESS

The famous Fischer, Lynch, and Patterson [40] impossibility result states that in asynchronous systems, it is impossible to reach agreement among a set of agents even if one of the agents fail. This is because, in asynchronous systems, there is no bound on message delivery and processing times, which make it difficult to distinguish agent failures from processing delays. This suggests that to formally prove progress in a knowledge propagation protocol, we need some conservative
assumptions. In this section, we identify some conditions that will allow us to formally prove the correctness properties.

The Conditions

It is easy to identify the following general condition about availability of agents that can be useful for proving correctness for most distributed protocols in an asynchronous setting:

\( \text{G1} \) All agents are always available.

However, our protocol only needs one coordinator to be always available. Therefore, we can further weaken \( \text{G1} \):

\( \text{G1a} \) At least one coordinator is always available.

\( \text{G1b} \) All replicas are always available.

In Section VII, we will formally specify these conditions as some assertions for use in the formal proofs.

Informal Analysis of the Conditions and the System Model

Below, we informally analyze the conditions and our system model with respect to the correctness properties:

- For the protocol to make progress, there must be some coordinator to send "1a" and "2b" messages and to eventually learn of \( E^2 \phi \). \( \text{G1a} \) ensures that at least one coordinator is always available.
- The protocol cannot make progress in each phase unless a coordinator receives "1b" and "2b" messages from all replicas. \( \text{G1b} \) ensures that all replicas are always available.
- If a particular message is always lost, this will prevent the protocol from making progress. Reliable message delivery ensures that all messages are eventually delivered.
- If an agent delivers any message that has not been sent, that will cause it to incorrectly perform some action, thereby affecting safety. Since messages cannot be corrupted and agents are non-Byzantine, this ensures that only a message that has been sent may be delivered.

We can see that the conditions identified are necessary for proving correctness under our system model. It is difficult to prove that these conditions are the weakest conditions required because there may be multiple equally-weak sets of conditions under which the proofs can be completed. Proving the weakest set, therefore, requires formalizing the meaning of "weakness" in the context of these conditions. Moreover, the strong guarantees provided by mechanically-verified proofs justify the possibly-conservative conditions in the context of safety-critical applications. Therefore, for now, we are satisfied with the informal analysis of the necessity of the conditions and will consider the set \( \{ \text{G1a, G1b} \} \) to be one of the weakest sets of conditions required for proving correctness under our system model. In Section VII, we will formalize a set of assertions from these conditions and the system model that we will use for the formal proofs of correctness.

VI. Application of the Protocol in Cooperative Flight Planning

The protocol defined in Section IV is a general-purpose distributed knowledge propagation protocol that can be used in asynchronous systems. In line with their definitions, coordinators and replicas are logical abstractions and a particular physical node may functionally implement both abstractions simultaneously. If the protocol is used in a scenario where the physical nodes exclusively implement either a coordinator or a replica, but not both, then the system can withstand the failure of multiple coordinators as long as \( \text{G1a} \) holds.

Since the IoP is a distributed network where the physical nodes are aircraft, our knowledge propagation protocol can be used for applications like cooperative flight planning. However, in the particular use case of cooperative flight planning, the set of aircraft \( K \), which are responsible for propagating a value \( \phi \), do not have any knowledge about the membership of the set \( K \), making it difficult to physically separate the set of coordinators and replicas. Therefore, this application necessitates the pessimistic implementation of the protocol where all aircraft in the set \( S \) functionally implement a replica and all aircraft in the set \( K \) functionally implement an additional coordinator. As a consequence, the set of physical nodes implementing coordinators are a subset of the physical nodes implementing replicas. Under such circumstances, the condition that \( \text{all replicas are always available} \) will imply that \( \text{all coordinators are always available} \). Since the set of coordinators and replicas are non-empty, \( \text{G1b} \) will imply \( \text{G1a} \) under such circumstances. Nevertheless, we will still use the weaker condition \( \text{G1a} \) for our proofs because this will allow the proofs to be pertinent, without loss of generality, even for applications where the implementation may not necessarily warrant \( \text{G1b} \) to imply \( \text{G1a} \).

In the next section, we will formally specify a distributed system implementing our protocol and some assertions that will be useful for the formal correctness proofs.

VII. The Formal Specification

In this section, we will formally specify a system implementing our protocol and introduce some assertions by formalizing the conditions and the system model.

The Network of Aircraft as a Distributed State Machine

We represent a distributed system implementing our protocol as a distributed state machine [41] that has a current state at any given time and changes its state by performing some action. There are the following sets:

- \( C \) – The set of all coordinators in the system.
- \( R \) – The set of all replicas in the system.
- \( V \) – The set of all values in the system.
- \( T \) – The set of discrete logical times.
- \( M \) – The set of all possible messages in the system.

Time is represented by natural numbers and every message in the system is represented by a tuple \( (\rho, \delta, \nu, \gamma) \) such that \( \rho \in C, \delta \in R, \nu \in V, \gamma \in \Gamma \) where \( \Gamma = \{ “1a”, “1b”, “2a”, “2b” \} \).

Let us assume that the value chosen for propagation after consensus is \( \phi \). The following global variables represent the current state of the system at any time:

- \( \kappa = \kappa[r] \) is True for a replica \( r \) iff it knows \( \phi \).

...
Following predicates:

- \(\dot{\epsilon} - \dot{\epsilon}[r]\) is True for a replica \(r\) iff it knows \(E\Phi\).
- \(\dot{\epsilon} - \dot{\epsilon}[c]\) is True for a coordinator \(c\) iff it knows \(E^2\Phi\).
- \(\mu\) - The set of all messages in the current state.
- \(\tau\) - The time at which the current state was recorded.

**Important Notations and Predicates**

We introduce the predicates below to detect various types of messages in the system state:

- \(\Phi_{1a}(c, v)\) is True if there is a message \(m\) of type "1a", value \(v\), and coordinator ID \(c\).

\[\Phi_{1a}(c, v) \equiv \exists m \in \mu : m.\gamma = "1a" \land m.\rho = c \land m.\nu = v\]

- \(\Phi_{1b}(r, c)\) is True if there is a message \(m\) of type "1b", replica ID \(r\), and coordinator ID \(c\).

\[\Phi_{1b}(r, c) \equiv \exists m \in \mu : m.\gamma = "1b" \land m.\delta = r \land m.\rho = c\]

- \(\Phi_{2a}(c)\) is True if there is a message \(m\) of type "2a" and coordinator ID \(c\).

\[\Phi_{2a}(c) \equiv \exists m \in \mu : m.\gamma = "2a" \land m.\rho = c\]

- \(\Phi_{2b}(r, c)\) is True if there is a message \(m\) of type "2b", replica ID \(r\), and coordinator ID \(c\).

\[\Phi_{2b}(r, c) \equiv \exists m \in \mu : m.\gamma = "2b" \land m.\delta = r \land m.\rho = c\]

When the initial state holds, there is no message in \(\mu\) and all state variables have their default initial values. As the protocol progresses, the agents take actions depending upon the receipt of certain messages. The protocol makes progress by reaching some distinct intermediate states which are specified by the following predicates:

- \(\Delta_0(t)\) - This is the initial state when the system state contains no messages.

\[\Delta_0(t) \equiv \tau = t \land (\mu = \{\})\]

- \(\Delta_{1a}(c, v, t)\) - This is true when the system state contains "1a" messages from a coordinator \(c\).

\[\Delta_{1a}(c, v, t) \equiv \tau = t \land (\exists v \in \mathcal{V} : \Phi_{1a}(c, v))\]

- \(\Delta_{1b}(P, t)\) - This is true when the system state contains "1b" messages for a coordinator \(c\) from all replicas.

\[\Delta_{1b}(P, t) \equiv \tau = t \land \forall r \in \mathcal{R} : \Phi_{1b}(r, c)\]

- \(\Delta_{2a}(c, t)\) - This is true when the system state contains "2a" messages from a coordinator \(c\).

\[\Delta_{2a}(c, t) \equiv \tau = t \land (\exists v \in \mathcal{V} : \Phi_{2a}(c))\]

- \(\Delta_{2b}(P, t)\) - This is true when the system state contains "1b" messages for a coordinator \(c\) from all replicas.

\[\Delta_{2b}(P, t) \equiv \tau = t \land \forall r \in \mathcal{R} : \Phi_{2b}(r, c)\]

The following predicates are used for specifying message transmission, agent availability, and state of knowledge:

- \(\$\!(m, t)\) denotes sending of message \(m\) at time \(t\).
- \(\$\!(m, t)\) denotes delivery of message \(m\) at time \(t\).
- \(\alpha(x, t)\) denotes that agent \(x\) is available at time \(t\).
- \(\mathcal{E}(x, t)\) denotes that agent \(x\) knows \(\phi\) at time \(t\).
- \(\mathcal{E}(x, t)\) denotes that agent \(x\) knows \(E\Phi\) at time \(t\).

**Assertions Implied by the Conditions and the System Model**

**A1** By \(G\!\!1\alpha\) there is a coordinator which is always available.

\[\exists c \in \mathcal{C} : \forall t \in \mathcal{T} : \alpha(c, t)\]

**A2** By \(G\!\!1\beta\) all replicas are always available.

\[\forall r \in \mathcal{R} : \forall t \in \mathcal{T} : \alpha(r, t)\]

**A3** By reliable message delivery, all messages which are sent must be eventually delivered at a later time.

\[\forall t_s \in \mathcal{T} : \forall m \in \mathcal{M} : (\$\!(m, t_s) \implies \exists t_d \in \mathcal{T} : (t_d > t_s) \land \$\!(m, t_d))\]

**A4** Since messages cannot be corrupted, by non-Byzantine nature of the system, if at any time a message is delivered by an agent, there must have been a time when it was sent.

\[\forall t_d \in \mathcal{T} : \forall m \in \mathcal{M} : (\$\!(m, t_d) \implies \exists t_s \in \mathcal{T} : (t_d > t_s) \land \$\!(m, t_s))\]

We can split the assertions into two sets \(\mathcal{A}_s = \{A2, A4\}\) and \(\mathcal{A}_p = \{A1, A2, A3\}\) as per their use in proving safety and progress respectively.

**The Formal Correctness Properties**

Now that we have formally specified the system and the assertions, we can formally specify the correctness properties.

**Safety:** Given \(\mathcal{A}_s\), if an available coordinator knows about \(E^2\Phi\), then all replicas know about \(\phi\) and \(E\Phi\).

\[\mathcal{A}_s \implies (\forall t \in \mathcal{T} : \forall c \in \mathcal{C} : ((\tau = t \land \alpha(c, t) \land \mathcal{E}(c, t)) \implies (\forall r \in \mathcal{R} : \mathcal{E}(r, t) \land \mathcal{E}(r, t))))\]

**Progress:** Given \(\mathcal{A}_p\), eventually, a coordinator will know about \(E^2\Phi\).

\[\mathcal{A}_p \implies \exists t \in \mathcal{T} : \exists c \in \mathcal{C} : \mathcal{E}(c, t)\]
In the next section, we provide the formal proof sketches of these theorems by using the formal specifications.

VIII. THE PROOFS

In this section, we will use the formal specification defined in Section VII to introduce some important lemmas and prove the theorems for safety and progress.

Some Important Lemmas

Lemma 1. Given $\mathfrak{A}_p$, for all coordinators, if there exists a time such that the coordinator $c$ is available it has not received "1b" messages from all replicas, then "1a" messages from $c$ will eventually be delivered.

$$
\mathfrak{A}_p \implies (\forall c \in C : (\exists t \in T : \tau = t \land (\exists v \in V : \Phi_{1a}(c,v)))) \implies (\exists t_2 \in T : \tau = t_2 \land (\exists v \in V : \Phi_{1a}(c,v)))
$$

Proof Sketch :-

(1) By non-Byzantine behavior of coordinators, an available coordinator will eventually send "1a" messages.

Proof Sketch :-

Lemma 2. Given $\mathfrak{A}_p$, for all coordinators, if there exists a time such that "1a" messages from a coordinator $c$ have been delivered, then "1b" messages from all replicas will eventually be delivered.

$$
\mathfrak{A}_p \implies (\forall c \in C : (\exists t \in T : \tau = t \land (\exists v \in V : \Phi_{1a}(c,v)))) \implies (\exists t_2 \in T : \tau = t_2 \land (\exists v \in V : \Phi_{1b}(c,v)))
$$

Proof Sketch :-

(1) By $A3$, all replicas are always available.

(2) By (1) and $A3$, eventually "1b" messages from a coordinator will be delivered.

Lemma 3. Given $\mathfrak{A}_p$, for all coordinators, if there exists a time such that the coordinator $c$ is available and it has received "1b" messages from all replicas, then "2a" messages from $c$ will eventually be delivered.

$$
\mathfrak{A}_p \implies (\forall c \in C : (\exists t \in T : \tau = t \land (\exists v \in V : \Phi_{1b}(c,v)))) \implies (\exists t_2 \in T : \tau = t_2 \land (\exists v \in V : \Phi_{2a}(c,v)))
$$

Proof Sketch :-

(1) By non-Byzantine behavior of coordinators, an available coordinator will eventually send "2a" messages.

Lemma 4. Given $\mathfrak{A}_p$, for all coordinators, if there exists a time such that "2a" messages from a coordinator $c$ have been delivered, then "2b" messages from all replicas will eventually be delivered.

$$
\mathfrak{A}_p \implies (\forall c \in C : (\exists t \in T : \tau = t \land (\exists v \in V : \Phi_{2a}(c)))) \implies (\exists t_2 \in T : \tau = t_2 \land (\forall r \in R : \Phi_{2b}(r,c)))
$$

Proof Sketch :-

(1) By $A2$, all replicas are always available.

(2) By non-Byzantine behavior of replicas, an available replica will eventually send "2b" messages.

(3) By (1), (2), and $A4$, eventually "2b" messages from all replicas will be delivered.

Lemma 5. Given $\mathfrak{A}_p$, there will be a coordinator $c$ such that $\Delta_{1a}(c,t)$ will eventually be true.

$$
\mathfrak{A}_p \implies (\exists c \in C : \exists t \in T : \Delta_{1a}(c,t))
$$

Proof Sketch :-

(1) By $A1$, a coordinator is always available.

(2) $\exists \in T : \Delta_0(t)$ since $\mu = \{\}$ in the initial state.

(3) By (1), (2), and Lemma 1, eventually "1a" messages from a coordinator will be delivered.

Lemma 6. Given $\mathfrak{A}_p$, there will be a coordinator $c$ such that $\Delta_{1b}(c,t)$ will eventually be true.

$$
\mathfrak{A}_p \implies (\exists c \in C : \exists t \in T : \Delta_{1b}(c,t))
$$

Proof Sketch :-

(1) By 5, eventually "1a" messages from an available coordinator will be delivered.

(2) By (1) and Lemma 2, eventually "1b" messages from all replicas will be delivered.

Lemma 7. Given $\mathfrak{A}_p$, there will be a coordinator $c$ such that $\Delta_{2a}(c,t)$ will eventually be true.

$$
\mathfrak{A}_p \implies (\exists c \in C : \exists t \in T : \Delta_{2a}(c,t))
$$

Proof Sketch :-

(1) By $A1$, a coordinator is always available.

(2) By Lemma 6, eventually "1b" messages from all replicas will be delivered to the coordinator.

(3) By (1), (2), and Lemma 3, eventually "2a" messages from the coordinator will be delivered.

Lemma 8. Given $\mathfrak{A}_p$, there will be a coordinator $c$ such that $\Delta_{2b}(c,t)$ will eventually be true.
Proof Sketch :-

(1) By 7, eventually "2a" messages from an available coordinator will be delivered.
(2) By (1) and Lemma 4, eventually "2b" messages from all replicas will be delivered.

Lemma 9. Given $\mathfrak{A}_s$, always, if an available coordinator $c$ knows $E^2\phi$, then all replicas will know $E\phi$.

\[ \mathfrak{A}_s \implies (\forall t \in T : \forall c \in C : ((\tau = t \land \alpha(c,t) \land E(c,t)) \implies (\forall r \in R : E(r,t))) \]

Proof Sketch :-

(1) By non-Byzantine behavior of coordinators, an available coordinator can only learn $E^2\phi$ if it has received "2b" messages from all replicas.
(2) By A4, if "2b" messages from all replicas have been received, they must have been sent by the replicas.
(3) By non-Byzantine behavior of available replicas, a replica can only send "2b" messages if it knows $E\phi$.
(4) By A2, all replicas are always available.
(5) By (1), (2), (3), and (4), all replicas know $E\phi$.

Lemma 10. Given $\mathfrak{A}_s$, always, if an available replica $r$ knows $E\phi$, then all replicas will know $\phi$.

\[ \mathfrak{A}_s \implies (\forall t \in T : \forall r \in R : ((\tau = t \land \alpha(r,t) \land E(r,t)) \implies (\forall r \in R : E(r,t))) \]

Proof Sketch :-

(1) By non-Byzantine behavior of replicas, an available replica can only learn $E\phi$ if it has received "2a" messages from a coordinator.
(2) By A4, if "2a" messages from a coordinator have been received, they must have been sent by the coordinator.
(3) By non-Byzantine behavior of available coordinator, a coordinator can only send "2a" messages if it knows "1b" messages from all replicas.
(4) By non-Byzantine behavior of available replicas, a replica can only send "1b" messages if it knows $\phi$.
(5) By A2, all replicas are always available.
(6) By (1), (2), (3), (4), and (5), all replicas know $\phi$.

Proof of Safety

Theorem 1. Given $\mathfrak{A}_s$, if an available coordinator knows about $E^2\phi$, then all replicas know about $\phi$ and $E\phi$.

\[ \mathfrak{A}_s \implies (\forall t \in T : \forall c \in C : ((\tau = t \land \alpha(c,t) \land E(c,t)) \implies (\forall r \in R : E(r,t) \land E(r,t))) \]

Proof Sketch :-

(1) Trivially by Lemma 9 and Lemma 10.

Proof of Progress

Theorem 2. Given $\mathfrak{A}_p$, eventually, a coordinator will know about $E^2\phi$.

\[ \mathfrak{A}_p \implies \exists t \in T : \exists c \in C : E(c,t) \]

Proof Sketch :-

(1) By non-Byzantine behavior of available coordinator, a coordinator will learn about $E^2\phi$ if it has received "2b" messages from all replicas.
(2) By Lemma 8, an available coordinator will eventually receive "2b" messages from all replicas.
(3) By (1) and (2) an available coordinator will eventually know $E^2\phi$.

We have specified our protocol in TLA$^+$ and mechanically verified all our proofs using TLAPS. Since TLAPS does not support first-order temporal logic, we incorporated time explicitly in our specification using first-order temporal logic. The TLA$^+$ specification includes some additional temporal axioms required for verifying the proofs in TLAPS. The proofs and specifications are about 1500 lines of TLA code$^3$.

IX. Conclusion

In this paper, we have proposed a knowledge propagation protocol for attaining a safe state of knowledge in a network of aircraft for cooperative flight planning. We have specified two correctness properties and identified a set of conditions under which our protocol can guarantee correctness. Since there may be multiple weak sets of conditions under which correctness can be guaranteed, we informally argue that the set we have identified is one of the weakest sets. We have also provided mechanically-verified proofs of our guarantees that make our protocol suitable for safety-critical aerospace applications.

Since aircraft have limited fuel capacity, we realize that guarantees of eventual progress for knowledge propagation are not enough for aerospace systems. Therefore, a potential future direction of work would be to investigate the design and development of formal proofs for stochastic properties about progress by using data-driven statistical results. This will allow us to provide guarantees about stochastic progress properties by using statistical observations about message delivery and processing times.

ACKNOWLEDGMENT

This research was partially supported by the National Science Foundation (NSF), Grant Nos. – CNS-1816307 and CNS-1553340, and the Air Force Office of Scientific Research (AFOSR), Grant No. – FA9550-19-1-0054.

$^3$Complete proof available at http://wcl.cs.rpi.edu/pilots/fvcafp