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. They will do this 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 a *safe state* of knowledge that is necessary for aircraft to operate safely in the absence of a centralized air-traffic controller and present a distributed knowledge propagation protocol to attain this safe state. This protocol can be used by network-connected aircraft to achieve collaborative situational awareness for cooperative flight planning. We identify certain system conditions necessary to guarantee two correctness properties for our protocol – safety and progress. We use the TLA$^+$ Specification Language to formally specify our protocol, the correctness properties, and the conditions necessary to guarantee the properties. Using the formal specifications, we also provide mechanically-verified proofs of the correctness properties in 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 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].

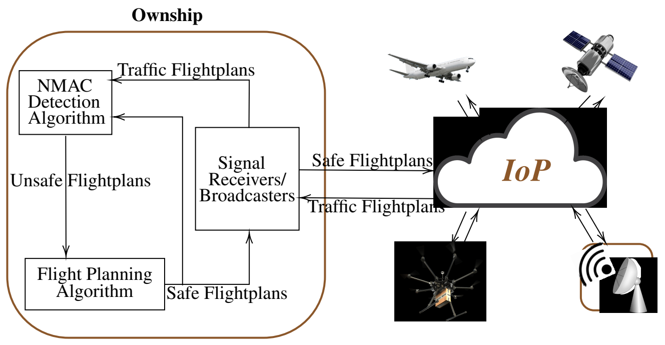

Fig. 1: Data-driven feedback loop for NMAC detection.

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.

In our vision of a decentralized air-traffic control system, if there are N aircraft already inside an airspace (*owners*), any new aircraft that wants to enter the airspace (*candidate*) has to obtain permission from the owners (Fig. 2). For this, a candidate has to first compute a provably conflict-free set of $N + 1$ flight-plans with the owners and then get them to agree on this set. A conflict-free set of flight-plans can be computed locally by a candidate by using a *conflict-aware flight planning algorithm* [4]. Then the candidate may employ a fault-tolerant *distributed consensus protocol* like *Paxos* [6]

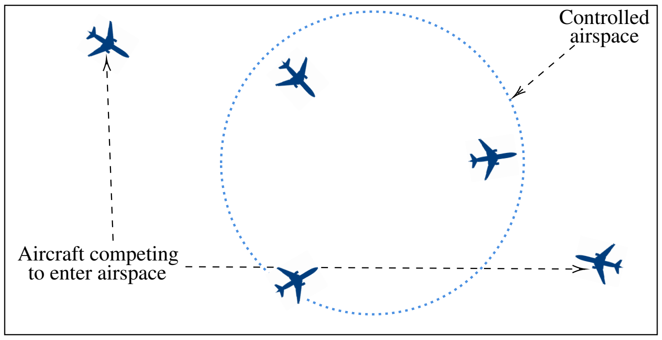

Fig. 2: Aircraft trying to enter a controlled airspace (top view).

to have the owners reach an agreement on the new set of flight-plans.

Using Paxos, multiple candidates may compete for entry into an airspace. However, a candidate will not consider other candidates while computing its conflict-free set of flight-plans with the owners, thereby generating a solution which is only valid if the set of owners does not change. This implies that the owners can allow entry to only one candidate by reaching consensus on only one of the proposed sets of flight-plans. This is necessary because when a candidate is allowed entry, the set of owners changes, invalidating the conflict-free sets computed by the other candidates. It has been formally proven that Paxos will guarantee that only one value is chosen [7], thus guaranteeing that only one candidate will be allowed entry by the owners.

When the set of owners changes, all aircraft "relevant" to an airspace must be informed so that they can update their set of owners. Aircraft relevant to an airspace comprises of the set of new owners and the set of aircraft expected to try to enter the airspace in the future (which also includes the candidates who may have been previously denied entry). Paxos only guarantees that the participants will reach consensus on a single value, which requires only a majority of the participants to chose that value. The successful completion of Paxos will not necessarily guarantee that all relevant aircraft will learn the set of new owners. It will, however, guarantee that a non-zero number of the relevant aircraft will learn the set of new owners. Therefore, a *knowledge propagation protocol* will be required to propagate the knowledge of the set of new owners (and their flight-plans) to all relevant aircraft.

Any knowledge propagation protocol used in cooperative flight planning should ensure a *safe state of knowledge* in which all aircraft can "feel safe" to operate. In knowledge logic [8], the expression $k_i\phi$ represents that *an agent i knows the fact* ϕ and $E\phi$ represents that *all agents in the system know* ϕ. The expression $EE\phi$ (or $E^2\phi$) represents a higher state of knowledge than $E\phi$ and implies that every agent knows two facts – (1) ϕ, and (2) $E\phi$. In the context of cooperative flight-planning, ϕ represents the set of new owners (and their flight-plans). We believe that if all aircraft operating in the NAS are autonomous, then $E\phi$ is sufficient for safety since autonomous agents will have an implicit trust in the system. However, if there are human pilots involved, then the absence of $E^2\phi$ will create a scenario in which the pilots will not be able to trust the safety of the system. Fagin *et al.* [8] succinctly explain this concern with the following example – *even if all drivers in a society know the rules for following traffic lights and follow them, that is not enough to make a driver "feel safe". This is because unless a driver knows that everyone else knows the rules and will follow them, then the driver may consider it possible that some other driver, who does not know the rules, may run a red light.* Therefore, we define a safe state of knowledge as the state $E^2\phi$.

The failure of *safety-critical*[1] aerospace systems can be catastrophic [9]. 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 techniques and tools for mechanically checking the proofs of these guarantees. The TLA+ [10] specification language is designed for specifying concurrent systems and their properties, and verifying them using the TLA+ Proof System (TLAPS) [11]. TLAPS can be used to write structured hierarchical proofs [12] which are automatically converted to individual proof obligations and solved by automated theorem provers like Isabelle [13], Zenon [14], Yices [15], CVC3 [16], Z3 [17], and veriT [18].

In this paper, we formalize the definition of a safe state of knowledge $E^2\phi$ and present a protocol for distributed knowledge propagation that can be used to achieve $E^2\phi$ after the set of owners for an airspace changes to a new set ϕ. This will give aircraft pilots the ability to operate in the airspace with explicit confidence in the safety of the system (we assume that all aircraft are non-Byzantine and non-adversarial). Our protocol allows an aircraft to propagate ϕ among all other aircraft and learn when $E^2\phi$ has been successfully achieved. We also identify certain system conditions under which it can be guaranteed that our protocol satisfies two main correctness properties – *safety*[2] and *progress*. We use these conditions to provide formal proofs of the correctness properties. Furthermore, we specify our protocol in TLA$^+$ and mechanically verify 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 "relevant set" for an airspace 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 and informally analyze them; in Section VI we discuss how our protocol can be implemented for cooperative flight planning applications; in Section VII, we formally specify our protocol and introduce two theorems corresponding to the correctness properties; in Section VIII, we present some lemmas, informal

[1]"Safety" here implies *protection from harm to life, environment, or property*

[2]Not to be confused with "safety" in "safety-critical"

intuitions behind the proofs of the lemmas and theorems, and detailed proof sketches for readers interested in replicating our results; and finally, in Section IX, we conclude the paper with a discussion about 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 and Fagin [19] 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. Ksehmkalyani [20] examines 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 and Halpern [21] 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 and Vardi [22] 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.* [23] 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 [24] analyzes the use of Datalog [25] to reason about the knowledge of a group of distributed nodes and develop Knowlog, which is a variant of Datalog that can express nodes' state of knowledge using a set of epistemic modal operators. Their approach abstracts the modes of communication exchange from states of knowledge so that reasoning is simplified. They use an implementation of the *Two-Phase Commit* protocol to analyze their approach. Fagin *et al.* [26] present a formal model for the analysis of attainable knowledge states in distributed systems under certain assumptions. Their model can be used to formalize assumptions about distributed systems, such as whether they are deterministic or non-deterministic and whether the knowledge is cumulative or non-cumulative. They also provide a complete axiomatization of knowledge for some important cases of interest. Van Der Mayden [27] establish the completeness of a logic of knowledge and time for all classes of systems that satisfy the perfect recall property which is true if, at all times, a processor's state includes a record of its previous states. They also provide an abstract characterization of systems with perfect recall that can be used to prove completeness. Kuznets *et al.* [28] 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.* [29] propose a logic for reasoning about epistemic messages in asynchronous distributed systems where knowledge is true at the start of announcement and agents can predict messages that have been sent, but not yet received. They extend the Public Announcement Logic [30] in which announcements from an external source can be used to model messages broadcast by agents within the system. Halpern [31] presents a survey of formalizations of distributed knowledge in asynchronous and unreliable distributed systems, and provides examples to argue why this formalization of distributed knowledge is important for analyzing distributed systems. Dwork *et al.* [32] 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.* [33] 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 ninety-six logics, which they investigate by characterizing their complexity and their dependence on parameter combinations. Choi *et al.* [34] 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.* [35] 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 [36] has presented an analysis of Two-Phase Commit protocol and its 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 IoP and providing machine-checked formal proofs of some correctness guarantees necessary for safety-critical aerospace applications.

III. Collaborative Situational Awareness

In this section, we present our problem statement for achieving collaborative situational awareness for cooperative flight planning.

As discussed in Section I, the following sets of aircraft are relevant to an airspace for cooperative flight planning:

C1 The aircraft allowed entry by the agreement on ϕ.
C2 The aircraft already inside the airspace.
C3 The aircraft expected to try to enter the airspace.

By definition, ϕ contains information about all aircraft in *C1* and *C2*. We assume that there is some mechanism, perhaps one that uses ADS-B [37] based intent-broadcast [38], that provides the information of aircraft in *C3*. The set $S : S = C1 \cup C2 \cup C3$ represents all aircraft relevant to an airspace.

Section I describes the mechanism by which only a subset of S is aware of the set of new owners (and their flight-plans) ϕ when a candidate aircraft is allowed entry into an airspace. The goal of the knowledge propagation phase is to ensure $E^2\phi$ in the set S. The problem of knowledge propagation for cooperative flight planning can, therefore, be precisely stated as – *"Given a set of aircraft $K : K \subset S$ in which all aircraft know the same value ϕ and the membership of S, but not the membership of K, each aircraft in K should try to propagate the knowledge of ϕ to all aircraft in S for attaining $E^2\phi$ and at least one aircraft in K should eventually learn that $E^2\phi$ has been attained."*

The knowledge of ϕ can be propagated to all aircraft in S by using atomic-commit protocols [39] which can ensure that a value is committed at all nodes. In the next section, we will introduce an atomic-commit-inspired distributed knowledge propagation protocol that can be used for attaining a safe state of knowledge for safety-critical applications.

IV. Our Knowledge Propagation Protocol

In this section, we propose a distributed knowledge propagation protocol that can be used for propagating a value ϕ 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 [40] 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*. Each coordinator has knowledge of the set of all replicas. A single value ϕ is known to all coordinators. $E^2\phi$, in the context of our protocol, implies that *all replicas know that all other replicas know* ϕ. The goal of every coordinator is to propagate ϕ 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 ϕ to all replicas.
  (b) A replica learns a value ϕ if and only if it receives a *learn* message with the value ϕ from a coordinator and it replies back to the coordinator with a *learnt ("1b")* message if and only if it has learnt a value ϕ.
- **Phase 2**
  (a) A coordinator sends an *all-know ("2a")* message to each replica if and only if it has received a *learnt* message from all replicas.
  (b) A replica learns that all replicas know the value ϕ if and only if it receives an *all-know* message from a coordinator and it replies back to the coordinator with an *acknowledgement ("2b")* message if and only if it has learnt that all replicas know the value ϕ.
  (c) A coordinator learns $E^2\phi$ if and only if 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 if and only if all replicas know $E\phi$ and ϕ. 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 [41] impossibility result states that in asynchronous systems, it is impossible to reach agreement among a set of agents even if only one of the agents fails. This is because, in asynchronous systems, there is no bound on message delivery and processing times, which makes 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. We identify a set of suitable conditions below for formally proving the correctness properties.

The Conditions

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

G1 *All agents are always eventually available.*

In asynchronous systems with reliable message delivery, message delays can be arbitrarily long and processing can also take arbitrarily long. In such systems, if an agent is *"always eventually available"*, it is equivalent to the agent being *"always available"* since there is no way to differentiate temporary failures from processing delays. Furthermore, our protocol only needs one coordinator to be always eventually available. Therefore, we can identify the following availability requirements:

G1a *At least one coordinator is always available.*
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 $E^2\phi$. ***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. ***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 corrupt 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 $\{\boldsymbol{G1a}, \boldsymbol{G1b}\}$ to be one of the weakest sets of conditions required for proving correctness under our system model.

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 ***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 ϕ, 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. This implies that an implementation of our protocol for cooperative flight planning will require all aircraft relevant to an airspace to implement a replica and all aircraft which are aware of ϕ (the set of new owners and their flight-plans) to implement both a replica and a coordinator.

As a consequence of the above implementation, the set of physical nodes implementing coordinators will be a subset of the physical nodes implementing replicas. Under such circumstances, the condition that *all replicas are always available* will imply that *all coordinators are always available*. Since the set of coordinators and replicas are non-empty, ***G1b*** will imply ***G1a*** under such circumstances. Nevertheless, we will still use the weaker condition ***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 ***G1b*** to imply ***G1a***.

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 [42] that has a current state at any given time and changes its state by performing some action. There are the following sets:

- $\mathcal{C}$ – The set of all coordinators in the system.
- $\mathcal{R}$ – The set of all replicas in the system.
- $\mathcal{V}$ – The set of all values in the system.
- $\mathcal{T}$ – The set of discrete logical times.
- $\mathcal{M}$ – 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 $\rho \in \mathcal{C}$, $\delta \in \mathcal{R}$, $\nu \in \mathcal{V}$, and $\gamma \in \Gamma$ where $\Gamma = \{\text{“}1a\text{”}, \text{“}1b\text{”}, \text{“}2a\text{”}, \text{“}2b\text{”}\}$.

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

- κ – $\kappa[r]$ is `True` for a replica r if and only if it knows ϕ.
- $\dot{\varepsilon}$ – $\dot{\varepsilon}[r]$ is `True` for a replica r if and only if it knows $E\phi$.
- $\ddot{\varepsilon}$ – $\ddot{\varepsilon}[c]$ is `True` for a coordinator c if and only if it knows $E^2\phi$.
- μ – The set of all messages in the current state.
- τ – 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.

$$\Phi_{1a}(c, v) \equiv \exists m \in \mu : m.\gamma = \text{“}1a\text{”} \wedge m.\rho = c \wedge m.\nu = v$$

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

$$\Phi_{1b}(r, c) \equiv \exists m \in \mu : m.\gamma = \text{“}1b\text{”} \wedge m.\delta = r \wedge m.\rho = c$$

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

$$\Phi_{2a}(c) \equiv \exists m \in \mu : m.\gamma = \text{“}2a\text{”} \wedge m.\rho = c$$

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

$$\Phi_{2b}(r, c) \equiv \exists m \in \mu : m.\gamma = \text{“}2b\text{”} \wedge m.\delta = r \wedge m.\rho = c$$

When the initial state holds, there is no message in μ 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.

$$\Delta_0(t) \equiv (\tau = t) \wedge (\mu = \{\})$$

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

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

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

$$\Delta_{1b}(c, t) \equiv (\tau = t) \wedge (\forall r \in \mathcal{R} : \Phi_{1b}(r, c))$$

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

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

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

$$\Delta_{2b}(c, t) \equiv (\tau = t) \wedge (\forall r \in \mathcal{R} : \Phi_{2b}(r, c))$$

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

- $\S(m, t)$ denotes sending of message m at time t.
- $\P(m, t)$ denotes delivery of message m at time t.
- $\alpha(x, t)$ denotes that agent x is available at time t.

- $\text{Ⅎ}(x,t)$ denotes that agent x knows ϕ at time t.
- $\dot{\text{Ⅎ}}(x,t)$ denotes that agent x knows $E\phi$ at time t.
- $\ddot{\text{Ⅎ}}(x,t)$ denotes that agent x knows $E^2\phi$ at time t.

Assertions Implied by the Conditions and the System Model

A1 By ***G1a*** there is a coordinator which is always available.

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

A2 By ***G1b*** 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.

$$\forall t_s \in \mathcal{T} : \forall m \in \mathcal{M} : (\S(m,t_s) \implies \exists t_d \in \mathcal{T} : (t_d > t_s) \wedge \P(m,t_d))$$

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.

$$\forall t_d \in \mathcal{T} : \forall m \in \mathcal{M} : (\P(m,t_d) \implies \exists t_s \in \mathcal{T} : (t_d > t_s) \wedge \S(m,t_s))$$

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

The Formal Correctness Properties

The safety property implies that a coordinator will know $E^2\phi$ if and only if all replicas know both $E\phi$ and ϕ. Theorem 1 formally specifies this property in terms of our formal specification.

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

$$\mathfrak{A}_s \implies (\forall t \in \mathcal{T} : \forall c \in \mathcal{C} : ((\tau = t \wedge \alpha(c,t) \wedge \ddot{\text{Ⅎ}}(c,t)) \implies (\forall r \in \mathcal{R} : \text{Ⅎ}(r,t) \wedge \dot{\text{Ⅎ}}(r,t))))$$

The progress property implies that eventually $E^2\phi$ will be attained. Since there is no external agent that can monitor the state of the system, it will be necessary and sufficient to show that eventually at least one coordinator will learn $E^2\phi$. Theorem 2 formally specifies this property in terms of our formal specification.

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

$$\mathfrak{A}_p \implies \exists t \in \mathcal{T} : \exists c \in \mathcal{C} : \ddot{\text{Ⅎ}}(c,t)$$

VIII. The Proofs

In this section, we will introduce some important lemmas, present some informal intuitions about how the lemmas can be used to prove the safety and progress properties, and provide comprehensive proof sketches for interested readers.

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.*

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

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.*

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

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.*

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

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.*

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

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

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

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

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

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

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

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

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

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

$$\mathfrak{A}_s \implies (\forall t \in \mathcal{T} : \forall c \in \mathcal{C} : ((\tau = t \wedge \alpha(c,t) \wedge \ddot{\mathcal{E}}(c,t)) \implies (\forall r \in \mathcal{R} : \dot{\mathcal{E}}(r,t))))$$

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

$$\mathfrak{A}_s \implies (\forall t \in \mathcal{T} : \forall r \in \mathcal{R} : ((\tau = t \wedge \alpha(r,t) \wedge \dot{\mathcal{E}}(r,t)) \implies (\forall r \in \mathcal{R} : \mathcal{E}(r,t))))$$

Informal Intuitions Behind our Formal Proofs

We present some informal intuitions behind our proofs of safety and progress to give readers a clear understanding of our approach behind developing the formal proofs.

Proof of Safety: In order to prove safety, it is important to show that a coordinator can know $E^2\phi$ if and only if all replicas know both $E\phi$ and ϕ. Lemma 9 and Lemma 10 can be used to prove that the safety property, represented by Theorem 1, is satisfied by our protocol at all times. The lemmas can be proven directly by using the system model and the set of assertions $\mathfrak{A}_s$.

Proof of Progress: In order to prove progress, we use the set of assertions $\mathfrak{A}_p$ and the system model to show that eventually, at least one coordinator will learn $E^2\phi$. Lemma 1 to Lemma 4 state some temporal guarantees about how the protocol will proceed under some system conditions. By ***A1***, there is at least one coordinator which is always available. Using ***A1***, Lemma 5 to Lemma 8 can be used to show that at least one coordinator will eventually receive *"2b"* messages from all replicas, thereby learning $E^2\phi$ by non-Byzantine behavior. Lemma 5 to Lemma 8 are proven with the help of Lemma 1 to Lemma 4. Lemma 8 is then used to prove Theorem 2.

Proof Sketches of the Lemmas and Theorems

We present detailed proof sketches for the lemmas and the theorems that may be useful as a reference for the open-source TLAPS proofs[3] to readers interested in replicating our results.

Proof Sketch of Lemma 1 :-

(1) By non-Byzantine behavior of coordinators, an available coordinator will eventually send *"1a"* messages.
(2) By (1) and ***A3***, eventually *"1a"* messages from the coordinator will be delivered.
□

Proof Sketch of Lemma 2 :-

(1) By ***A2***, all replicas are always available.
(2) By non-Byzantine behavior of replicas, an available replica will eventually send *"1b"* messages.
(3) By (1), (2), and ***A3***, eventually *"1b"* messages from all replicas will be delivered.
□

Proof Sketch of Lemma 3 :-

(1) By non-Byzantine behavior of coordinators, an available coordinator will eventually send *"2a"* messages.
(2) By (1) and ***A3***, eventually *"2a"* messages from the coordinator will be delivered.
□

Proof Sketch of Lemma 4 :-

(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 ***A3***, eventually *"2b"* messages from all replicas will be delivered.
□

Proof Sketch of Lemma 5 :-

(1) By ***A1***, a coordinator is always available.
(2) $\exists t \in \mathcal{T} : \Delta_0(t)$ since $\mu = \{\}$ in the initial state.
(3) By (1), (2), and Lemma 1, eventually *"1a"* messages from a coordinator will be delivered.
□

Proof Sketch of Lemma 6 :-

(1) By Lemma 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.
□

Proof Sketch of Lemma 7 :-

(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.
□

Proof Sketch of Lemma 8 :-

(1) By Lemma 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.
□

Proof Sketch of Lemma 9 :-

(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 $E\phi$.
□

[3]Complete TLAPS proofs available at http://wcl.cs.rpi.edu/pilots/fvcafp

Proof Sketch of Lemma 10 :-

(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 has received *"1b"* messages from all replicas.
(4) By non-Byzantine behavior of available replicas, a replica can only send *"1b"* messages if it knows ϕ.
(5) By ***A2***, all replicas are always available.
(6) By (1), (2), (3), (4), and (5), all replicas know ϕ.
$\square$

Proof Sketch of Theorem 1 :-

(1) Trivially by Lemma 9 and Lemma 10.
$\square$

Proof Sketch of Theorem 2 :-

(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 $E^2\phi$.
$\square$

We have specified our protocol in TLA$^+$ and mechanically verified all our proofs using TLAPS. Since TLAPS does not currently support first-order temporal logic we have incorporated time explicitly in our specification using first-order temporal logic[4]. 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.

IX. Conclusion

In this paper, we have specified a safe state of knowledge for safety-critical aerospace applications and presented a knowledge propagation protocol for attaining the safe state of knowledge. This protocol can be used in a network of aircraft to achieve collaborative situational awareness for cooperative flight planning applications. We have specified two correctness properties (safety and progress) and identified a set of system conditions under which our protocol can guarantee these correctness properties. Since there may be multiple weak sets of conditions under which the properties 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 in the TLA$^+$ Proof System 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 sufficient for practical purposes. A potential future direction of work would be to investigate the 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. Our current protocol also relies on the assumption that all aircraft are non-Byzantine, requiring them to be cooperative and "perfectly behaved". This is a strong assumption and does not consider pilot errors and other uncertainties presented by real-world applications. Therefore, another potential direction of work would be to expand our system model to accommodate the Byzantine behavior of aircraft.

[4]TLAPS 1.4.5. as of June 2020 only supports propositional temporal logic

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.

References

[1] S. Paul, F. Hole, A. Zytek, and C. Varela, “Wind-aware trajectory planning for fixed-wing aircraft in loss of thrust emergencies,” in *Proc. 37th AIAA/IEEE Digit. Avionics Syst. Conf.*, London, England, UK, Sep. 2018, pp. 558–567.
[2] J. Souyris, V. Wiels, D. Delmas, and H. Delseny, “Formal verification of avionics software products,” in *International symposium on formal methods*. Springer, 2009, pp. 532–546.
[3] M. Prandini, L. Piroddi, S. Puechmorel, and S. L. Brázdilová, “Toward air traffic complexity assessment in new generation air traffic management systems,” *IEEE transactions on intelligent transportation systems*, vol. 12, no. 3, pp. 809–818, 2011.
[4] S. Paul, S. Patterson, and C. A. Varela, “Conflict-Aware Flight Planning for Avoiding Near Mid-Air Collisions,” in *The 38th AIAA/IEEE Digital Avionics Systems Conference (DASC 2019)*, San Diego, CA, Sep. 2019.
[5] Federal Aviation Administration, “Introduction to TCAS-II Version 7.1,” 2011.
[6] L. Lamport, “Paxos Made Simple,” *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.
[7] L. Lamport, S. Merz, and D. Doligez, “A TLA+ specification of the Paxos Consensus algorithm described in Paxos Made Simple and a TLAPS-checked proof of its correctness,” https://github.com/tlaplus/v1-tlapm/blob/master/examples/paxos/Paxos.tla, last modified Fri Nov 28 10:39:17 PST 2014 by Lamport, Accessed: May 1, 2019.
[8] R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi, *Reasoning about knowledge*. MIT press, 2004.
[9] I. Sommerville, *Software engineering*. Addison-Wesley/Pearson, 2011.
[10] L. Lamport, *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Longman Publishing Co., Inc., 2002.
[11] K. Chaudhuri, D. Doligez, L. Lamport, and S. Merz, “Verifying Safety Properties with the TLA+ Proof System,” in *International Joint Conference on Automated Reasoning*. Springer, 2010, pp. 142–148.
[12] L. Lamport, “How to Write a Proof,” *The American mathematical monthly*, vol. 102, no. 7, pp. 600–608, 1995.
[13] L. C. Paulson, *Isabelle: A Generic Theorem Prover*. Springer Science & Business Media, 1994, vol. 828.
[14] R. Bonichon, D. Delahaye, and D. Doligez, “Zenon: An Extensible Automated Theorem Prover Producing Checkable Proofs,” in *International Conference on Logic for Programming Artificial Intelligence and Reasoning*. Springer, 2007, pp. 151–165.
[15] B. Dutertre and L. De Moura, “The Yices SMT Solver,” *Tool paper at http://yices. csl. sri. com/tool-paper. pdf*, vol. 2, no. 2, pp. 1–2, 2006.
[16] C. Barrett and C. Tinelli, “Cvc3,” in *International Conference on Computer Aided Verification*. Springer, 2007, pp. 298–302.
[17] L. De Moura and N. Bjørner, “Z3: An Efficient SMT Solver,” in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[18] T. Bouton, D. C. B. de Oliveira, D. Déharbe, and P. Fontaine, "veriT: An Open, Trustable and Efficient SMT-Solver," in *International Conference on Automated Deduction*. Springer, 2009, pp. 151–156.

[19] J. Y. Halpern and R. Fagin, "Modelling knowledge and action in distributed systems," *Distributed computing*, vol. 3, no. 4, pp. 159–177, 1989.

[20] A. Kshemkalyani, "On continuously attaining levels of concurrent knowledge without control messages," Technical Report UIC-EECS-98-6, University of Illinois at Chicago, Tech. Rep., 1998.

[21] R. Fagin and J. Y. Halpern, "Reasoning about knowledge and probability," *Journal of the ACM (JACM)*, vol. 41, no. 2, pp. 340–367, 1994.

[22] R. Fagin and M. Y. Vardi, "Knowledge and implicit knowledge in a distributed environment: Preliminary report," in *Theoretical Aspects of Reasoning About Knowledge*. Elsevier, 1986, pp. 187–206.

[23] M. Guzmán, S. R. Knight, S. Quintero, S. Ramírez, C. Rueda, and F. Valencia, "Reasoning about distributed knowledge of groups with infinitely many agents," in *30th International Conference on Concurrency Theory (CONCUR)*, 2019.

[24] M. Interlandi, "Reasoning about knowledge in distributed systems using datalog," in *International Datalog 2.0 Workshop*. Springer, 2012, pp. 99–110.

[25] T. Eiter, G. Gottlob, and H. Mannila, "Disjunctive datalog," *ACM Transactions on Database Systems (TODS)*, vol. 22, no. 3, pp. 364–418, 1997.

[26] R. Fagin, J. Y. Halpern, and M. Y. Vardi, "What can machines know? on the properties of knowledge in distributed systems," *Journal of the ACM (JACM)*, vol. 39, no. 2, pp. 328–376, 1992.

[27] R. Van Der Meyden, "Axioms for knowledge and time in distributed systems with perfect recall," in *Proceedings of the Ninth Annual IEEE Symposium on Logic in Computer Science*, 1994, pp. 448–449.

[28] R. Kuznets, L. Prosperi, U. Schmid, and K. Fruzsa, "Epistemic reasoning with byzantine-faulty agents," in *International Symposium on Frontiers of Combining Systems*. Springer, 2019, pp. 259–276.

[29] S. Knight, B. Maubert, and F. Schwarzentruber, "Reasoning about knowledge and messages in asynchronous multi-agent systems," *Mathematical Structures in Computer Science*, vol. 29, no. 1, pp. 127–168, 2019.

[30] J. Plaza, "Logics of public announcements," in *Proceedings 4th International Symposium on Methodologies for Intelligent Systems*, 1989.

[31] J. Y. Halpern, "Using reasoning about knowledge to analyze distributed systems," *Annual review of computer science*, vol. 2, no. 1, pp. 37–68, 1987.

[32] C. Dwork and Y. Moses, "Knowledge and common knowledge in a byzantine environment: crash failures," in *Theoretical Aspects of Reasoning about Knowledge*. Elsevier, 1986, pp. 149–169.

[33] J. Y. Halpern and M. Y. Vardi, "The complexity of reasoning about knowledge and Time. I. Lower Bounds," *Journal of Computer and System Sciences*, vol. 38, no. 1, pp. 195–237, 1989.

[34] S.-M. Choi, J. Park, Q. Nguyen, A. Cronje, K. Jang, H. Cheon, Y.-S. Han, and B.-I. Ahn, "Opera: Reasoning about continuous common knowledge in asynchronous distributed systems," *arXiv preprint arXiv:1810.02186*, 2018.

[35] I. Popovic, V. Vrtunski, and M. Popovic, "Formal verification of distributed transaction management in a SOA based control system," in *2011 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*. IEEE, 2011, pp. 206–215.

[36] M. Atif, "Analysis and verification of two-phase commit & three-phase commit protocols," in *2009 International Conference on Emerging Technologies*. IEEE, 2009, pp. 326–331.

[37] E. A. Lester, "Benefits and incentives for ADS-B equipage in the national airspace system," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[38] R. Barhydt and A. Warren, "Newly enacted intent changes to ADS-B MASPS: Emphasis on operations, compatibility, and integrity," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4932.

[39] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency control and recovery in database systems*. Addison-wesley New York, 1987, vol. 370.

[40] L. Gönczy, M. Kovács, and D. Varró, "Modeling and verification of reliable messaging by graph transformation systems," *Electronic Notes in Theoretical Computer Science*, vol. 175, no. 4, pp. 37–50, 2007.

[41] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.

[42] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys (CSUR)*, vol. 22, no. 4, pp. 299–319, 1990.