Science of Computer Programming 239 (2025) 103184

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

# Formal verification of timely knowledge propagation in airborne networks

Saswata Paul [a,*], Chris McCarthy [b], Stacy Patterson [b], Carlos Varela [b]

[a] GE Aerospace Research, 1 Research Circle, Niskayuna 12309, NY, USA
[b] Rensselaer Polytechnic Institute, 110, 8th Street, Troy 12180, NY, USA

ARTICLE INFO

*Keywords:*
Formal methods
Distributed systems
Autonomous systems
Probabilistic properties
Theorem proving
Proof library

ABSTRACT

Ensuring timely coordination between autonomous aircraft is a challenging problem in decentralized air traffic management (ATM) applications for urban air mobility (UAM) scenarios. This paper presents an approach for formally guaranteeing timely progress in a Two-Phase Acknowledge distributed knowledge propagation protocol by probabilistically modeling the delays using the theory of the Multicopy Two-Hop Relay protocol and the M/M/1 queue system. The guarantee states a probabilistic upper bound to the time for progress as a function of the probabilities of the total transmission and processing delays following two specific distributions. The proof uses a general library of formal theories, that can be used for the rigorous mechanical verification of autonomous aircraft coordination protocols using the Athena proof checker and assistant.

## 1. Introduction

The use of *uncrewed aircraft systems* (UAS) for *urban air mobility* (UAM) operations will pose significant air traffic management (ATM) challenges as the UAS will need to operate in highly congested urban airspaces while maintaining safe distance from one another. Centralized ATM techniques, that rely on human controllers or *ground stations*, are prone to human errors, do not scale, and are economically infeasible for UAM operations [1,2]. Decentralized *UAS traffic management* (UTM) protocols, on the other hand, can be used by the UAS to autonomously coordinate and operate safely.

One of the challenging aspects of decentralized coordination is ensuring that a fact $\phi$ is sufficiently propagated through a network of autonomous agents. In the context of UTM, $\phi$ may represent a set of autonomous aircraft that are authorized to use a four-dimensional airspace, the knowledge of which is important for any new aircraft that would want to operate in the same airspace. The *two-phase acknowledge* knowledge propagation protocol (TAP) [3] can be used to *sufficiently propagate* the knowledge of a fact $\phi$ in a network of aircraft to attain a *safe distributed state of knowledge* [4]. TAP satisfies two important properties—*consistency*, which implies that TAP will correctly propagate the knowledge; and *eventual progress*, which implies that eventually, the safe state will be attained. However, a guarantee of eventual progress *alone* is insufficient for time-critical UTM applications, as it does not state any bound on the time for successful propagation. Guarantees of *timely progress*, on the other hand, can provide some useful time bounds for successful propagation by considering message transmission and processing delays and the number of messages involved. In asynchronous networks, message delays are unbounded, making it impossible to provide deterministic bounds on the total time

* Corresponding author.
*E-mail address:* paulsaswata1@gmail.com (S. Paul).

https://doi.org/10.1016/j.scico.2024.103184
Received 27 May 2024; Received in revised form 5 August 2024; Accepted 9 August 2024

Available online 20 August 2024

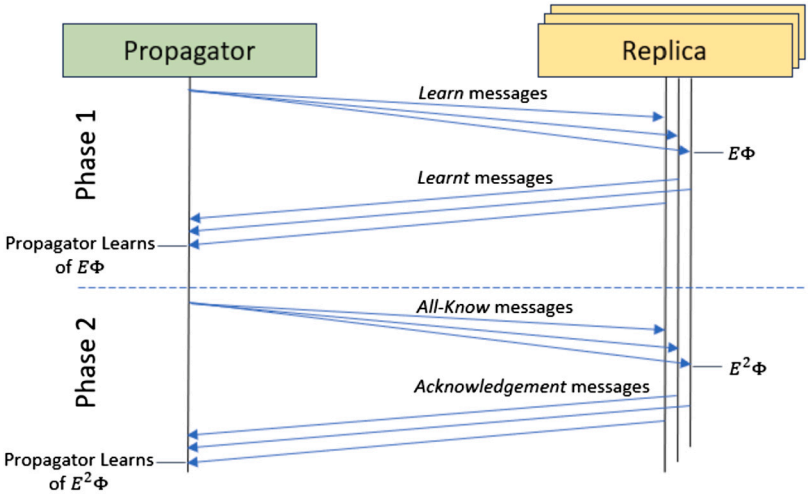


**Fig. 1.** Schematics of knowledge flow in TAP via messages (example uses 3 replicas).

that may be required for successful propagation. However, using appropriate theories and assumptions, it is possible to formalize some useful *probabilistic bounds* on progress that can be helpful.

In this paper, we formulate *probabilistic progress guarantees* for TAP (going beyond eventual progress) by using theories apposite to *low-altitude platforms* (LAP) [5] of airborne communication such as *vehicular ad hoc networks* (VANET) [6]. Particularly, we use the theory of the *Multicopy Two-Hop Relay* (MTR) protocol [7] to model message transmission delays in VANETs and the *M/M/1 queue system* [8] to model message processing delays in avionics systems. The probabilistic bounds on these delays are then used to provide sufficiently high-level guarantees of timely progress that can be useful for UAM applications such as collision avoidance. We also present a formal reasoning library in the Athena proof assistant [9,10] that is adequate to mechanically verify the timely progress guarantees of TAP. The library contains formalized theories from various domains, such as random variables, distributions, queues, and VANETS, that can be used for reasoning about and proving high-level properties of distributed coordination protocols.

The main contributions of this paper are—(1) formalization of the theory of the Multicopy Two-Hop Relay (MTR) protocol to model message transmission delays; (2) formalization of theory related to the probability of events; (3) formalization of properties of the Poisson distribution in queues; (4) formalization of the theory of the M/M/1 queue system to model message processing delays; (5) a high-level probabilistic guarantee of timely progress for TAP assuming the usage of MTR protocol and M/M/1 queues; and (6) a discussion on the development and use of formal proof libraries to verify complex system properties. This work extends upon an earlier version [11] by developing the high-level formal proofs of two important lemmas that were previously left as conjectures—*the probability of implied events* (Eq. (15)) and *the average number of messages in queue for a Poisson distribution* (Eq. (10))—and by providing a more insightful and structured discussion on our reasoning library, that has been extended with new theories to support the new proofs.

The paper is structured as follows: Section 2 describes TAP; Section 3 presents a timely progress guarantee for TAP; Section 4 discusses the proof library; Section 5 presents a discussion on our approach; Section 6 presents prior related work; and Section 7 concludes the paper with a summary and potential directions of future work.

## 2. The two-phase acknowledge protocol

TAP considers an asynchronous, non-Byzantine system model in which agents may fail temporarily and message delivery is reliable [12]. There is a non-empty set of *propagators*, and a logically separate non-empty set of *replicas*. All propagators know the set of replicas and the same fact $\phi$.

In *knowledge logic* [4], the expression $k_i\phi$ represents that *an agent i knows the fact* $\phi$, and $E\phi$ represents that *all agents in the system know* $\phi$. 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—$\phi$ and $E\phi$. $E^2\phi$, in the context of TAP, implies that *all replicas know that all other replicas know* $\phi$. The goal of every propagator is to propagate $\phi$ among all the replicas and eventually learn that $E^2\phi$ has been achieved. Propagators and replicas are logical abstractions and may be functionally implemented by the same aircraft simultaneously.

From the perspective of a propagator, TAP has two consecutive phases:

- **Phase 1**
  (a) The propagator sends a *learn* message with $\phi$ to each replica.
  (b) A replica learns $\phi$ if and only if it receives a *learn* message with $\phi$ from the propagator; it replies to the propagator with a *learnt* message if and only if it has learned $\phi$.
- **Phase 2**
  (a) The propagator sends an *all-know* message to each replica if and only if it has received *learnt* messages from all replicas.

(b) A replica learns that all replicas know the value $\phi$ if and only if it receives an *all-know* message from the propagator; it replies to the propagator with an *acknowledgement* message if and only if it has learned that all replicas know the value $\phi$.
(c) The propagator learns $E^2\phi$ has been achieved if and only if it receives *acknowledgement* messages from all replicas.

Under our system model, eventual progress has been guaranteed for TAP under some suitable conditions [3]. Assuming eventual progress, the total time taken for successful propagation is dependent on three factors—the message transmission delays between the aircraft; the message processing delays in each aircraft; and the total number of messages involved. From the perspective of a propagator, successful propagation involves a deterministic number of messages—for each replica, 4 messages are required (Fig. 1), so for $R$ replicas, the total number of messages required for successful propagation is $4 \times R$. However, in asynchronous settings, transmission and processing delays are unbounded. Therefore, even with a deterministic number of messages, it is impossible to provide a deterministic bound on the time for progress.

## 3. Formal guarantees of probabilistic timely progress

In the absence of deterministic properties, it is possible to employ a probabilistic approach for providing formal guarantees of timely progress. This can be done by reasoning about the probabilistic message delays using either theoretical models or data-driven models. In contrast to theoretical models that are based on analytical results about system characteristics, data-driven models are usually based on real-time or historical statistical observations about the system. This paper focuses on using theoretical models for reasoning about probabilistic timely progress.

For UAM applications, it is desirable to choose theoretical models that are appropriate for airborne networks. *Mobile ad hoc networks* (MANETS) have been shown to be viable options for inter-aircraft communication in areas where there are no dedicated communication networks [13]. Using such networks, aircraft are not only able to communicate their own information but also relay received information from other aircraft using a *multi-hop ad hoc* approach [14]. Moreover, an aircraft can be considered to be a single *server* where messages arrive, wait for some time until they are processed, and take some time to be processed. This makes it appropriate to use *queueing theory* [8] to reason about message processing delays. The M/M/1 system is an elementary queue system that consists of a single server that processes all messages which are received, making it a reasonable choice for modeling message processing with respect to a single aircraft. M/M/1 system also provides some useful analytical properties that make it possible to formally reason about the message processing delays. It has been shown that *two-hop relaying* and the M/M/1 queue are appropriate for message transmission and processing in MANETs [15,16]. Therefore, we use assumptions valid under these paradigms to develop the proof of timely progress for TAP.

To formally derive timely progress guarantees for TAP, we make the following assumptions—(1) all aircraft use the MTR protocol for message transmission between each other; (2) each aircraft independently implements an M/M/1 queue to process the messages that it receives; (3) the transmission delays of the messages are *independent and identically distributed* (i.i.d.); (4) the processing delays of the messages are i.i.d.; and (5) the transmission delays are independent of the processing delays.

From the perspective of a propagator, if eventual progress is guaranteed, then in the worst-case, when there is no concurrency in message transmission and processing, the total time ($T_S$) taken for successful propagation can be calculated by using the total number of messages that are involved ($N_M$), and the transmission delay ($T_{D_m}$) and processing delay ($T_{P_m}$) of each message $m$. Since we know that $N_M$ has a deterministic value ($4 \times R$) for TAP, the worst-case time can, therefore, be obtained using Eq. (1).

$$T_S = \sum_{m=1}^{N_M} T_{D_m} + \sum_{m=1}^{N_M} T_{P_m} \tag{1}$$

Now, to derive a probabilistic bound on $T_S$, we first need to probabilistically model $T_{D_m}$ and $T_{P_m}$ for every message $m$ involved.

### 3.1. Modeling the message transmission delays

*Two-hop relaying*, for data transmission between a *source* and a *destination* when the two nodes are not within *transmission range*, has been proposed to be an efficient mode of communication in MANETs [15]. Relaying has also been considered to be appropriate for airborne networks where two communicating aircraft may not always be within direct *transmission range* [14]. Therefore, to ensure that our assumptions about message transmission delays in airborne networks are consistent with prior work in the literature, we use the theory of the multicopy two-hop relay (MTR) protocol. For this, we base our assumptions on the description of the MTR protocol presented by Al Hanbali et al. [17].

In the basic two-hop relay protocol, there are a set of $M + 1$ mobile nodes whose trajectories are i.i.d. [15]. If a *source* node wants to transfer a message to a *destination* node, it can either transmit it directly to the destination, if the destination is within its transmission range, or, it can transmit copies of the message to one or more *relay* nodes. A relay node can transmit a copy of a message directly to the destination node if it is within transmission range. A relay node, however, cannot transmit a copy of a message to another relay node. Each message, therefore, makes a maximum of two hops—it is either transmitted directly from the source to the destination, or it is transmitted through one intermediate relay node.

In the MTR protocol [17], transmission delay $T_{D_m}$ is the time taken for the destination to receive a message $m$ or a copy of $m$ for the first time. Two nodes *meet* when they come within the transmission range of one another and *inter-meeting time* is the time interval between two consecutive meetings of a given pair of nodes. The inter-meeting times of all pairs of nodes are i.i.d. with the

common *cumulative distribution function* (CDF) $G(t)$. The source can only transmit a message to a relay that does not already hold a copy. Message transmission between two nodes within range is instantaneous. Each copy of a message has a *time-to-live* (TTL), which is the time after which a relay has to drop an untransmitted copy.

To derive a probabilistic bound on the time taken to transmit a message using MTR, we make the following assumptions:

- the TTLs for all messages are unrestricted, *i.e.*, a message can be held by a relay node until it can transmit it,
- since delivery is instantaneous when the nodes meet and TTLs are unrestricted, the inter-meeting time between the source and the destination ($T_{sd}$) or between a relay $i$ and the destination ($T_{r_i d}$) also represents the time taken by the source or the relay to directly deliver a message to the destination,
- the source transmits the copy of a message to all $M-1$ relay nodes. A message (or a copy) can, therefore, be delivered to the destination by the source or any of the relay nodes. Hence, the actual time taken to deliver the message will be the minimum of $\{T_{sd}, T_{r_1 d}, ..., T_{r_{M-1} d}\}$ (Eq. (2)),

$$T_{D_m} = \min\left(\{T_{sd}, T_{r_1 d}, ..., T_{r_{M-1} d}\}\right) \tag{2}$$

- the inter-meeting times of the mobile nodes are *exponentially distributed* with the *rate parameter* $\lambda_{MTR}$.

By using the above assumptions and necessary theories from probability, random variables, and exponential distributions, we can derive the following probabilistic bound on the time taken to deliver a message $m$ using MTR:

$$P\left(T_{D_m} \leq t\right) = 1 - e^{-\lambda_{MTR} M t} \tag{3}$$

Now, the derivative of the above expression conforms to the *probability density function* (PDF) of an exponential distribution with rate parameter $\lambda_{MTR} M$. Therefore, we can conclude that $T_{D_m}$ is exponentially distributed with a rate parameter $\lambda_{D_m} = \lambda_{MTR} M$.

### 3.2. Modeling the message processing delays

*Queueing theory* [18] has been extensively used in the literature for modeling throughput in MANETs [19–21,16]. An aircraft communicating using a MANET can be considered to be a single *server* where messages arrive, wait some time in a queue until they are processed, and take some time to be processed. Therefore, for modeling the message processing delays in the aircraft, we use theory of queues, particularly the *M/M/1* queue system [8]. The M/M/1 queue system is a special queueing system where there is a single server that processes all incoming messages, message arrivals are determined by a *Poisson process* [22], and message service times have an exponential distribution. This makes it suitable for modeling message transmission in aircraft. The M/M/1 queue system provides elegant analytical properties for computing the total time $T_{P_m}$ required for a message $m$ to be processed. This processing time includes the time spent in the queue and the time spent actually processing the message. An important property of $T_{P_m}$ in the M/M/1 queue system is that $T_{P_m}$ is exponentially distributed [18].

The M/M/1 queue system has the following properties [18]:

- the *interarrival times* of messages in the queue are exponentially distributed with a rate parameter $\lambda_a$,
- the *service time* of messages, which is the time spent actually processing a message, is exponentially distributed with a mean $1/\mu_s$,
- the queue is managed by a single server, and
- the number of message arrivals in an interval of length $\tau$ follows a *Poisson distribution* [23] with a parameter $\lambda_a \tau$.

To probabilistically model $T_{P_m}$, we use *Little's theorem* [24], which is an important fundamental result in queueing theory. If $N_q$ and $T_p$ represent the *average number of messages* and the *mean processing delay* (queueing delay + service delay) respectively,[1] then Little's Theorem states the useful relationship conveyed in Eq. (4)

$$N_q = \lambda_a T_p \tag{4}$$

We have formalized the proof of Little's Theorem as presented in [18]. The proof uses $N(\tau)$, $\alpha(\tau)$, and $T(i)$ to represent the number of messages in the system at time $\tau$, the number of messages that arrived in the interval $[0, \tau]$, and the average time spent in the system by message $i$ respectively. The average arrival rate over the time period $[0, t]$ is then given by:

$$\lambda_t = \frac{\alpha(t)}{t} \tag{5}$$

Similarly, the average processing delay of messages that arrived in the interval $[0, t]$ is given by:

$$T_t = \sum_{i=1}^{\alpha(t)} . \frac{T(i)}{\alpha(t)} \tag{6}$$

The average number of messages in the system in $[0, t]$ is given by:

[1] Note that $T_p = \mathbf{E}\left[T_{P_m}\right]$ is the *mean* or the *expected value* of $T_{P_m}$.

$$N_t = \frac{1}{t}\int_0^t N(\tau)d\tau \tag{7}$$

The graphical analysis of a FIFO system reveals the following [18]:

$$\frac{1}{t}\int_0^t N(\tau)d\tau = \frac{1}{t}\sum_{i=1}^{\alpha(t)} T(i) \tag{8}$$

Using Eq. (5) to Eq. (8), it can be shown that:

$$N_t = \lambda_t T_t \tag{9}$$

Assuming that the limits $N_q = \lim_{t\to\infty} N_t$, $\lambda_a = \lim_{t\to\infty} \lambda_t$, and $T_p = \lim_{t\to\infty} T_t$ exist, we can now obtain Eq. (4) to prove Little's Theorem.

Now, using the properties of the Poisson distribution, we can prove that:

$$N_q = \frac{\lambda_a}{\mu_s - \lambda_a} \tag{10}$$

Again, from Eq. (10) and Eq. (4), we can obtain the following relationship:

$$T_p = \frac{1}{\mu_s - \lambda_a} \tag{11}$$

Finally, since $T_{P_m}$ is exponentially distributed, its rate parameter $\lambda_{P_m}$ can be obtained as[2]:

$$\lambda_{P_m} = \frac{1}{\mathbf{E}\left[T_{P_m}\right]} = \frac{1}{T_p} = \mu_s - \lambda_a \tag{12}$$

### 3.3. Probabilistic bound on the worst-case time

Using the rate parameters $\lambda_{D_m}$ and $\lambda_{P_m}$ for the exponentially distributed message transmission and processing delays, we can now provide a probabilistic bound on the worst-case time for progress $T_S$. For that, let

$$T_D = \sum_{m=1}^{N_M} T_{D_m} \quad \& \quad T_P = \sum_{m=1}^{N_M} T_{P_m} \tag{13}$$

From Eq. (1) and Eq. (13), for any two real numbers $x$ and $y$, we can state:

$$((T_D \le x) \wedge (T_P \le y)) \implies (T_S \le (x+y)) \tag{14}$$

Now, for two events, $A$ and $B$, the following statement holds [25]:

$$(A \implies B) \implies (P(B) \ge P(A)) \tag{15}$$

Therefore, from Eq. (14) and Eq. (15), we get:

$$P(T_S \le (x+y)) \ge P((T_D \le x) \wedge (T_P \le y)) \tag{16}$$

Since the processing delays are independent of the transmission delays, by the *product rule* of independent events [26] we have[3]:

$$P((T_D \le x) \wedge (T_P \le y)) = P(T_D \le x) \times P(T_P \le y) \tag{17}$$

From Eq. (16) and Eq. (17) we have:

$$P(T_S \le (x+y)) \ge P(T_D \le x) \times P(T_P \le y) \tag{18}$$

Now, both $T_D$ and $T_P$ are sums of i.i.d. exponential random variables. Therefore, $T_D$ and $T_P$ follow two different *Erlang distributions* [23], each of which are parameterized by a *shape parameter* (which is $N_M$ in both cases) and a rate parameter ($\lambda_{D_m}$ for $T_D$ and $\lambda_{P_m}$ for $T_P$). The CDF of an Erlang distribution with shape parameter $k$ and rate parameter $\lambda$ is given by:

$$F_{\mathrm{Er}}(t,k,\lambda) = 1 - \sum_{n=0}^{k-1} \frac{1}{n!} e^{-\lambda t}(\lambda t)^n \tag{19}$$

[2] For an exponential random variable, the rate parameter is the reciprocal of the mean.

[3] The product rule states that for independent events $A$ and $B$, $P(A \wedge B) = P(A) \times P(B)$.

```
conclude THEOREM-KP-time-bound
pick-any r1
pick-any r2
assume  ((not (= r1 0.0)) and (not (= r2 0.0)))
let{
    A := (Prob.probE (Prob.consE Prob.<= (dpTotTim KP) (r1 + r2)));
    B :=
        (Prob.probE
            (Prob.cons2E
                (Prob.consE Prob.<=  (Random.SUM (msGetTd (dpGetMsgs KP)))
    r1)
                (Prob.consE Prob.<= (Random.SUM (msGetTp (dpGetMsgs KP))) r2
    )
            )
        );
    C :=
            ((Dist.erlangCDF
                    (4.0 * (KPNumReplicas KP))
                    (*(Dist.ratePar (Random.pdf (Random.rvSetIdElmnt (Netwrk
    .netNodeIMTmSet (dpNet KP)))))
                      (Netwrk.netNumSrcDst (dpNet KP)))
                    r1)
              * (Dist.erlangCDF
                    (4.0 * (KPNumReplicas KP))
                     (- (Dist.ratePar (Random.pdf (Queue.srvcTm (dpQ KP))))
                       (Dist.ratePar (Random.pdf (Queue.cstArRat (dpQ KP))))
    )
                    r2));
    r1-r2-not-zero := ((not (= r1 0.0)) and (not (= r2 0.0)));
    A>=B := (!uspec (!uspec (!uspec THEOREM-P-TS>=P-TD&TP KP) r1) r2);
    B=C :=  (!uspec (!uspec THEOREM-P-T_D-&-P-T_P-erlangCDF-ratepars r1) r2)
    ;
    B=C_conclusion := (!mp B=C r1-r2-not-zero);
    conn2->=-transitive2-axiom := (!uspec (!uspec (!uspec RealExt.>=
    -transitive2-axiom A) B) C);
    A>=B&B=C := (!both A>=B B=C_conclusion)
}
(!mp conn2->=-transitive2-axiom A>=B&B=C)
```

**Fig. 2.** Athena proof of Eq. (22).

This can be used to compute the delay probabilities as follows:

$$P(T_D \leq x) = 1 - \sum_{n=0}^{N_M-1} \frac{1}{n!} e^{-\lambda_{D_m} x} (\lambda_{D_m} x)^n = F_{\text{Er}}(x, N_M, \lambda_{D_m}) \tag{20}$$

$$P(T_P \leq y) = 1 - \sum_{n=0}^{N_M-1} \frac{1}{n!} e^{-\lambda_{P_m} y} (\lambda_{P_m} y)^n = F_{\text{Er}}(y, N_M, \lambda_{P_m}) \tag{21}$$

From Eq. (18), Eq. (20), and Eq. (21), we can state the required bound on $T_S$:

$$P(T_S \leq (x+y)) \geq F_{\text{Er}}(x, N_M, \lambda_{D_m}) \times F_{\text{Er}}(y, N_M, \lambda_{P_m}) \tag{22}$$

where $N_M = 4 \times R$, $\lambda_{D_m} = \lambda_{MTR} M$ and $\lambda_{P_m} = \mu_s - \lambda_a$.

We have mechanically verified all the equations described in this section in Athena by using theory of limits, distributions, real numbers, and other domains (the proof of Eq. (22) is shown in Fig. 2). However, currently the library does not support graphical analysis, and, therefore, we have taken Eq. (8) as a conjecture in the library.[4] In the next section, we will present a brief insight into how such complex proofs can be developed using interactive theorem provers such as Athena by using illustrative examples.

[4] The formal proof of this established property [18] is left for future work.

```
# A type to express events relevant to our proofs
datatype Event  :=
    (consE rel:Rel rv:Random.RandVar rl:Real) # wrt to a single rv
  | (consErvs rel:Rel rvS:Random.RandVarSet rl:Real) # wrt all in a set of
    rvs

# An event happens
declare happens : [Event] -> Boolean

# Probability of an event
declare probE : [Event] -> Real

# Get complementary event
declare get-comp : [Event] -> Event

# Two events are complementary
declare complement : [Event Event] -> Boolean

# Disjoint events
declare disjoint : [Event Event] -> Boolean

# Two events are independent
declare indEvnt : [Event Event] -> Boolean

# Conditional probability of A on B
declare condProbE : [Event Event] -> Real
```

**Fig. 3.** Snippet from our Athena formalism to express events over random variables.

```
# First of Kolmogorov's axioms of probability - every probability is at
    least 0
assert kolmogorov-1 := (forall E1 . (probE E1) >= 0.0)

# P(A|B) P(B) = P(A & B) - this is the definition of conditional probability
     (as an axiom in some cases)
assert conditional-def := (forall E1 E2 . (condProbE E1 E2) * (probE E2) = (
    probE (cons2E E1 E2)))

# A -> B means that P(B | A) = 1
define e1==>e2-1 := (forall E1 E2 . ((happens E1) ==> (happens E2)) ==> ((
    condProbE E2 E1) = 1.0))
(!force e1==>e2-1)
```

**Fig. 4.** Some background theory needed for the proof of Eq. (15).

## 4. Coordination protocol verification using athena

In this section, we will present some insight into our proof development process in Athena and present our proof library that is tailored for developing proofs of distributed communication in airborne networks.[5]

To develop formal proofs of mathematical equations and other statements, it is necessary to first develop a library of symbols and terminologies that can be used to express the equations and statements in a computer understandable manner. For example, consider Eq. (15) which states a property about the probabilities of events. To develop the formal proof of this equation, we introduced a datatype of events `Event` (Fig. 3) that allowed us to express the properties of events. As our proofs considered events which were either of the form "$x$ `<operator>` $X$" or "$\forall x \in \Phi : x$ `<operator>` $X$", where $x$ is a random variable, $\Phi$ is a set of random variables, `<operator>` $\in \{>, <, =, \geq, \leq\}$, and $X$ is a real value, we introduced two event constructors `consE` and `consErvs` to express such events. We also developed constructs to express statements like "*an event happens*", "*probability of an event*", and "*two events are complementary*" by creating appropriate functions over `Event`.

Once the terminologies have been defined, necessary theory must then be encoded for developing the proofs. In our case, after defining the data type, constructors, and function symbols necessary to express events, we introduced some theory that the proof of Eq. (15) relies on. This included two axioms—Kolmogorov's first axiom of probability and the fundamental definition of conditional

[5] Access the library at https://wcl.cs.rpi.edu/assure/.

```
# e1==>e2-prob: if A --> B, then P(B) >= P(A)
conclude e1==>e2-prob :=
(forall E1 E2 . ((happens E1) ==> (happens E2)) ==> ((probE E2) >= (probE E1
    )))
pick-any E1 E2
    assume hyp := ((happens E1) ==> (happens E2))
    conclude ((probE E2) >= (probE E1))
        let {
            condProb1 := (!mp (!uspec* e1==>e2-1 [E1 E2]) hyp);
            A&B<B := (!uspec* p-a&b [E2 E1])
            }
        (!chain-last [A&B<B
            ==> ((probE E1) Real.<= (probE E2)) [
                    (probE (cons2E E2 E1))
                = ((condProbE E2 E1) * (probE E1)) [conditional-def]
                = (1.0 * (probE E1)) [condProb1]
                = ((probE E1) * 1.0) [RealExt.prod-commutative-axiom]
                = (probE E1) [RealExt.prod-identity-axiom]]
            ==> ((probE E2) >= (probE E1)) [RealExt.<=-inverse->=-axiom]])
```

**Fig. 5.** Athena proof of Eq. (15).

probability—which we asserted; and a statement about conditional probability, which we forced into the assumption base as a part of our top-down proof development approach (Fig. 4).

Once the necessary theory has been specified, special methods called *proof tactics* must be used to develop the proofs. We developed the proof of Eq. (15) by using Athena's proof tactics (as shown in Fig. 5). Athena supports the use of *forward* and *backward* reasoning tactics for interactively guiding proofs using facts in the assumption base. This particular proof used the `chain` *tactic* for forward reasoning. Once proven, Eq. (15) was then simply instantiated for the appropriate variables in the proof of Eq. (16) (Fig. 6).

The proofs presented in Section 3 rely on theories from a variety of fundamental domains such as reals numbers, probability theory, network theory, etc. In order to fully mechanize the proofs, we needed to formalize in Athena the relevant low-level fundamental theories from these domains as applicable. One of these was the domain of real numbers, for which we created the `RealExt` module (Fig. 7). This module contains some fundamental theories (*e.g.,* the commutative and associative properties of operators) of Real numbers needed for the proofs. As a part of `RealExt`, we also had to develop the proofs of some simple properties such as the property that two variables representing real numbers, which are equal to each other, can be replaced with one another in an expression. Although such properties are very simple, designing generic proofs that work for all types of expressions was challenging since we had not created a domain of expressions over which generic proofs could be developed. So we had to develop custom proofs for each expression where we needed to use such properties. This satisfied our needs to mechanize the proofs of the equations presented in this paper, but it will be useful to generalize these proofs over all types of expressions in the future.

One important aspect we kept in mind while developing the formalizations was to ensure that they are sufficiently abstract and reusable to facilitate their seamless integration into proofs of higher-level theories across different contexts. *E.g.,* Fig. 8 shows the usage of some of the theories from the `RealExt` module where they have been simply "plugged-in" appropriately in the proof of Eq. (10) to prove the sub-steps of the `chain` tactic-based proof.

When there is a considerable amount of theory in a library, it becomes necessary to partition and organize it for clarity and reusability. This is also important in order to support modular enhancements of theories when needed so that a minor update to a theory does not force major rewriting of the formalisms in the entire library. Athena allows the use of *modules* that can be used to organize, partition, and encapsulate theories into logically separate *namespaces*. Modules are dynamic in the sense that they can be extended at any time and additional content can be added to them using the `extend-module` command (Fig. 9). This feature of Athena allows theories, which are logically separate from one another, to be categorized and structured for brevity. It also makes it easy to generalize theories for use in different contexts by importing as required. Several important modules in our Athena library, such as `Prob` (probability), `Dist` (distributions), and `Event` (events), have been modularized and organized into smaller sections using the `extend-module` command.

Our Athena library contains theories from distributed systems, network theory, VANETS, and mathematical theories such as probability theory, logarithms, and calculus. Parts of the library have also been used for dynamic data-driven flight state awareness applications [27], thereby showing its applicability beyond distributed communication and for data-driven autonomous aerospace systems. Fig. 10 shows the different modules in the current version of the library and the relationships between them. The v0.3 of the library currently consists of 159 axioms, 9 forced conjectures, and 52 theorems.

The library does not yet contain an end-to-end proof for Eq. (22) that is supported only by fundamental axioms. There are conjectures that have been relied upon to design the high-level proofs of intermediate proof steps such as Eq. (10) and Eq. (15). Eq. (8) has been forced since we have not encoded the theory of graphical analysis that is needed for the proof [18]. However, designing a complete end-to-end proof is a matter of determining the correct formalisms and the theory to encode in Athena so that the conjectures can be proven from fundamental axioms. The technical challenges include designing new Athena constructs that can

```
# P(TS <= (x + y)) >= P((TD <= x) /\ (TP <= y))
define THEOREM-P-TS>=P-TD&TP :=
(forall DP r1 r2 .
    ((Prob.probE  (Prob.consE Prob.<= (dpTotTim DP) (r1 + r2)))
    >=
    (Prob.probE
        (Prob.cons2E
          (Prob.consE Prob.<= (Random.SUM (msGetTd (dpGetMsgs DP))) r1)
          (Prob.consE Prob.<= (Random.SUM (msGetTp (dpGetMsgs DP))) r2)))))
conclude THEOREM-P-TS>=P-TD&TP
pick-any DP
pick-any r1
pick-any r2
let{
    E1 := (Prob.consE Prob.<= (dpTotTim DP) (r1 + r2));
    E2 :=
        (Prob.cons2E
            (Prob.consE Prob.<= (Random.SUM (msGetTd (dpGetMsgs DP))) r1)
            (Prob.consE Prob.<= (Random.SUM (msGetTp (dpGetMsgs DP))) r2));
    conn2-e1==>e2-prob := (!uspec (!uspec Prob.e1==>e2-prob E2) E1);
    conn2-THEOREM-sum-implies-TS := (!uspec (!uspec (!uspec
    THEOREM-sum-implies-TS DP) r1) r2)
}
(!mp conn2-e1==>e2-prob conn2-THEOREM-sum-implies-TS)
```

**Fig. 6.** `e1==>e2-prob` as used in the Athena proof of Eq. (16).

```
# commutative property of addition
assert sum-commutative-axiom :=
(forall r1 r2 . r1 + r2 = r2 + r1)

# associative property of addition
assert sum-associative-axiom :=
(forall r1 r2 r3 . (r1 + r2) + r3 = r1 + (r2 + r3))

# commutative property of multiplication
assert prod-commutative-axiom :=
(forall r1 r2 . ((r1 * r2) = (r2 * r1)))

# associative property of multiplication
assert prod-associative-axiom :=
(forall r1 r2 r3 . r1 * (r2 * r3) = (r1 * r2) * r3)
```

**Fig. 7.** A snippet from the `RealExt` module.

be used to express the lower-level theory that the proofs of the conjectures rely on and then interactively designing the proofs using the tactics provided by Athena. Making sure that the lower-level constructs are as generic and reusable as possible is also a challenge.

## 5. Discussion

An aircraft can be considered to be a rational agent [28] that uses information (the guarantee of timely knowledge propagation) that it perceives from its operating environment (the network of connected aircraft) to take goal-oriented decisions, where the goal is to operate safely without conflicts with other aircraft. Probabilistic guarantees of timely progress for distributed coordination protocols will allow such autonomous aircraft to make educated operational decisions, *e.g.*, if aircraft know that it will take at most $x$ seconds to propagate the knowledge of their flight plans with 99.99% probability, then they can safely decide to compute plans that start after $x$ seconds, and be highly confident that the plans will not become stale. As the guarantees presented in Section 3 are based on well-established theoretical models of VANETs, it makes them suitable for safety-critical airborne applications where autonomous aircraft may need to coordinate over ad hoc networks in a decentralized manner. Two-hop relaying has been proposed to be an efficient mode of communication in VANETs [15] and since each aircraft will be independently responsible for processing all the messages sent to it, the proposed combination of the MTR protocol and the M/M/1 queue system is a reasonable approach for formally modeling communication between aircraft.

Although there exist technologies to generate formal proofs in a completely automated fashion [29,30], we believe that interactive theorem proving is a better choice for verifying complex aerospace systems for several reasons. First, interactive systems are usually

```
(!chain [(Dist.mean (Random.pdf (numCst Q)))
         = (RealExt.summation 0.0 RealExt.INFY n (n * (Prob.probE (Prob.consE
      Prob.== (numCst Q) n)))) [expected-numCst]
         = (RealExt.summation 0.0 RealExt.INFY n (n * (p0 * (RealExt.pow rho
    n)))) [MM1-prob-n-cst]
         ...
         = (rho * (p0 * (RealExt.partial rho (1.0 / (1.0 - rho))))) [RealExt.
    THEOREM-ab-paren-swap]
         = (rho * (p0 * ((1.0 / (1.0 - rho)) / (1.0 - rho)))) [RealExt.
    div-derivative-axiom]
         ...
         = (rho * ((1.0 - rho) * ((1.0 / (1.0 - rho)) * (1.0 / (1.0 - rho))))
    ) [RealExt.a-by-b-axiom]
         = (rho * (((1.0 - rho) * (1.0 / (1.0 - rho))) * (1.0 / (1.0 - rho)))
    ) [RealExt.prod-associative-axiom]
         = (rho * (((1.0 / (1.0 - rho)) * (1.0 - rho)) * (1.0 / (1.0 - rho)))
    ) [RealExt.prod-commutative-axiom]
         = (rho * (((1.0 - rho) / (1.0 - rho)) * (1.0 / (1.0 - rho)))) [
    RealExt.a-by-b-axiom]
         ...
         = (Lambda / (mu - Lambda)) [RealExt.a-by-d-x-b-by-a-axiom]
         ])
```

**Fig. 8.** A snippet from the proof of Eq. (10) showing the usage of the `RealExt` module.

```
# Module Prob being extended by module PrbIndRv
load "Athena_LibDDDAS/math/Prob/Prob.ath"
extend-module Prob {
  module PrbIndRv {
    # contents of module PrbIndRv
    ...
    }
}
```

**Fig. 9.** Extending the `Prob` module in our Athena library, which contains fundamental probability theory, with the `PrbIndRv` module, which contains theory of probabilities of independent random variables, using the `extend-module` command.

more transparent than automated systems since interactive proofs require all steps to be clearly specified, which is not the case for automated proofs. This also helps in increasing trust by promoting in-depth understanding of the properties, the proof steps, and the specifications that the proofs are based on [31]. Second, interactive proofs promote the modularization and reuse of theories which allows them to be easily modified when the design of a system changes and to be used across systems that share common characteristics. This makes them suitable for aerospace systems verification where legacy systems are often modified and partially reused in new systems. For these reasons, we chose to use interactive theorem proving techniques for this work.

One major advantage of using theorem provers is the strong confidence provided by mechanically verified proofs. If all the asserted statements and the final theorems are understandable with respect to their mathematical or English counterparts, then the trust on the proofs gets conveniently abstracted to the trust on the verification engine. However, this is also reliant on developing formalisms that can not only be used to express understandable statements but are also precise and robust enough for proof development. Athena is based on *many sorted first order logic* [32] and uses *natural deduction*-style of proofs [33], which follows natural human reasoning strategies such as *proof by induction*, *by cases*, *by contradiction*, etc. This makes the Athena proofs easy to understand with respect to their less-formal counterparts written in mathematical notation for human consumption. Moreover, Athena provides a strong *soundness* guarantee that a proven theorem is a logical consequence of the statements in its assumption base.[6] It also prevents ill-sorted statements by performing sort checking automatically.

Autonomous aircraft operations are complex in nature. They involve many interlinked aspects such as the mobility of the agents, the characteristics of the communication network, the internal computational capabilities of the agents, and the interaction between the agents. Therefore, formal reasoning about such operations must holistically consider all these aspects. Achieving this in a machine-verifiable manner requires access to formal constructs, that are sufficiently expressive to correctly and completely specify such aspects, in a machine-readable formal language. Developing such expressive formal constructs in a machine-readable language is a challenging task since it requires domain knowledge of all aspects of the system that need to be specified, strong knowledge of formal logic and reasoning techniques, experience and familiarity with the language in which the constructs are to be specified, and a significant

[6] The assumption base is the set of all asserted, forced, or verified statements in Athena.

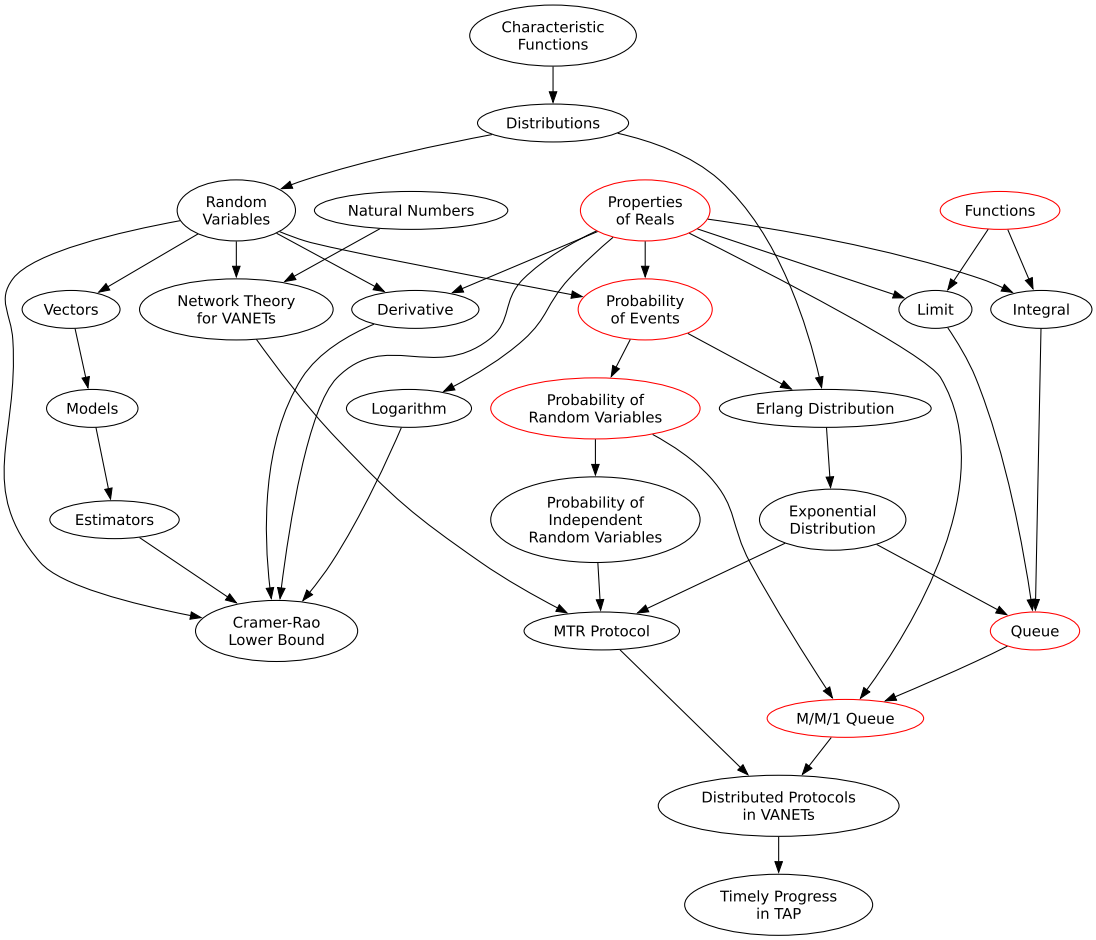


**Fig. 10.** Hierarchy of v0.3 of our Athena library for reasoning about autonomous aerospace systems. Modules in red have been created/updated as part of this paper. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

amount of time and effort. In the absence of a pre-existing formal library that provides the necessary formal constructs to express a system, every time high-level properties of the system need to be verified, engineers have to develop the required formalizations from the ground up, potentially making the task of verification intractable. This creates an incentive for developing formal proof libraries that can be reused as building blocks for verifying different higher-level proofs.

We have tried to make the specifications in our Athena library reusable to different contexts so that further expansion of the library can be aided by using the existing specifications as building blocks rather than requiring to redevelop them for use in other contexts. In our experience, efficiently designing reusable formal structures to express interconnected theories requires several iterations where the specifications need to be improved to be more general as new contexts for application are identified. Another challenge is the meaningful modularization of the developed theories into appropriate categories to make it easy to import them independently in different contexts. This is important because when theories are imported during proof development, they are automatically added to Athena's assumption base.

## 6. Related work

There is existing work in the literature on the informal[7] analysis of timely progress of consensus. Attiya et al. [34] provided lower-bounds on the time for progress in *round-based* [35] consensus protocols. Attiya et al. [36] analyzed the time complexity of solving distributed decision problems. Berman et al. [37] studied the number of message rounds involved in various consensus protocols. Machine-verified guarantees have mainly been about eventual progress. McMillan *et al.* [38] have proven eventual progress of *Stoppable Paxos* [39] in Ivy [40]. Dragoi et al. [41] and Debrat et al. [42] have proven eventual progress in *LastVoting* [43]. Hawblitzel *et al.* [44,45] have proven eventual progress for a *Multi-Paxos* implementation using Dafny [46]. In [47], we have verified eventual progress of the Synod consensus protocol using Athena. Formalization of mathematical theories has also been presented

[7] We refer to mathematical proofs written for purely human consumption in a non-machine-readable and non-machine-verifiable language as *informal* proofs.

in the literature. Hasan [48] presented formalizations of statistical properties of discrete random variables in the *HOL Theorem Prover* [49]. Hasan *et al.* have formalized properties of the standard uniform random variable [50], discrete and continuous random variables [51,52], tail distribution bounds [53], and conditional probability [54]. Mhamdi *et al.* [55,56] have extended Hasan et al.'s work by formalizing *measure theory* and the *Lebesque integral* in HOL. HOL formalizations of the Poisson process, *continuous chain Markov process*, and M/M/1 queue have been presented in [57]. Qasim [58] has used Mhamdi et al.'s work to formalize the *standard normal variable*.

In contrast, we presented a formal proof library that unifies suitable theories from various domains to allow for the comprehensive verification of complex distributed protocols for autonomous airborne systems.

## 7. Conclusion and future work

We presented a formal guarantee of timely progress for a knowledge propagation protocol that can be used for coordination of autonomous aircraft. The guarantee was formally proven using theories from the Multi-copy Two-Hop Relay (MTR) protocol and the M/M/1 queue system to reason about the non-deterministic message delays in a probabilistic manner. We also showcased a proof library in Athena that is tailored towards the verification of autonomous aircraft coordination and provided insights into the reusable theories that were developed for the proof of timely knowledge propagation.

A potential direction of future work would be to expand the library with more low-level theories so that all high-level proofs in the library can be traced back to fundamental axioms. Another potential contribution would be to add the necessary support for reasoning about distributed protocols that involve a non-deterministic number of messages, such as the Synod consensus protocol. This will require probabilistic reasoning over computation sequences and trees [31,59,60]. It will also be interesting to model other routing protocols suitable for VANETs in addition to MTR.

Based on our experiences using Athena, we have also identified a few potential new Athena features that will be useful for proof engineers. One major feature would be a dedicated Interactive Development Environment (IDE) with syntax highlighting and checking capabilities. Another feature would be insightful error messages for proof failures that can make it easier for proof engineers to detect shortcomings in their proof logic and identify correct ways to overcome them. The availability of a *pretty printer*, that can print theories in Athena constructs using appropriate corresponding mathematical notations will also be advantageous since that can make it easy to compare the asserted facts to their mathematical counterparts to detect any inconsistencies. To make the theories in our Athena library more accessible to new users, it will also be important to create documentation tools that can be used to annotate helpful insights into the theories in the library.

## CRediT authorship contribution statement

**Saswata Paul:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Chris McCarthy:** Formal analysis, Data curation. **Stacy Patterson:** Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization. **Carlos Varela:** Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

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

## References

[1] D. Wing, W. Cotton, For spacious skies: self-separation with "autonomous flight rules" in US domestic airspace, https://doi.org/10.2514/6.2011-6865, 2011.
[2] S. Balachandran, C. Manderino, C. Muñoz, M. Consiglio, A decentralized framework to support UAS merging and spacing operations in urban canyons, in: 2020 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2020, pp. 204–210.
[3] S. Paul, S. Patterson, C.A. Varela, Collaborative situational awareness for conflict-aware flight planning, in: The 39th IEEE/AIAA Digital Avionics Systems Conference, 2020, pp. 1–10.
[4] R. Fagin, J.Y. Halpern, Y. Moses, M. Vardi, Reasoning About Knowledge, MIT Press, 2004.
[5] X. Cao, P. Yang, M. Alzenad, X. Xi, D. Wu, H. Yanikomeroglu, Airborne communication networks: a survey, IEEE J. Sel. Areas Commun. 36 (2018) 1907–1926, https://doi.org/10.1109/jsac.2018.2864423.
[6] M.M. Hamdi, L. Audah, S.A. Rashid, A.H. Mohammed, S. Alani, A.S. Mustafa, A Review of Applications, Characteristics and Challenges in Vehicular Ad-Hoc Networks (VANETs), in: 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), IEEE, 2020, pp. 1–7.
[7] J. Liu, X. Jiang, H. Nishiyama, N. Kato, On the delivery probability of two-hop relay MANETs with erasure coding, IEEE Trans. Commun. 61 (2013) 1314–1326, https://doi.org/10.1109/tcomm.2013.020413.120198.
[8] L. Lipsky, M/M/1 queue, in: Queueing Theory: A Linear Algebraic Approach, 2009, pp. 33–75.
[9] K. Arkoudas, Athena, http://proofcentral.org/athena, 2012.
[10] K. Arkoudas, D. Musser, Fundamental Proof Methods in Computer Science: A Computer-Based Approach, MIT Press, 2017.

[11] S. Paul, S. Patterson, C.A. Varela, Formal guarantees of timely progress for distributed knowledge propagation, in: Formal Methods for Autonomous Systems (FMAS), in: Electronic Proceedings in Theoretical Computer Science, vol. 348, Open Publishing Association, The Hague, Netherlands, 2021, pp. 73–91.
[12] L. Gönczy, M. Kovács, D. Varró, Modeling and verification of reliable messaging by graph transformation systems, Electron. Notes Theor. Comput. Sci. 175 (2007) 37–50, https://doi.org/10.1016/j.entcs.2007.04.015.
[13] D. Medina, F. Hoffmann, S. Ayaz, C.-H. Rokitansky, Feasibility of an aeronautical mobile ad hoc network over the North Atlantic corridor, in: 2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, IEEE, 2008, pp. 109–116.
[14] Y. Wang, Y.J. Zhao, Fundamental issues in systematic design of airborne networks for aviation, in: 2006 IEEE Aerospace Conference, IEEE, 2006, p. 8.
[15] M. Grossglauser, D.N. Tse, Mobility increases the capacity of ad hoc wireless networks, IEEE/ACM Trans. Netw. 10 (2002) 477–486, https://doi.org/10.1109/tnet.2002.801403.
[16] X. Wen-jie, Y. Han, L. Feng-yu, The analysis of M/M/1 queue model with N policy for damaged nodes in MANET, in: 2011 IEEE International Conference on Computer Science and Automation Engineering, vol. 1, IEEE, 2011, pp. 289–294.
[17] A. Al Hanbali, A.A. Kherani, P. Nain, Simple models for the performance evaluation of a class of two-hop relay protocols, in: International Conference on Research in Networking, Springer, 2007, pp. 191–202.
[18] D.P. Bertsekas, R.G. Gallager, P. Humblet, Data Networks, vol. 2, Prentice-Hall International, New Jersey, 1992.
[19] R. Kushwah, S. Tapaswi, A. Kumar, Multipath delay analysis using queuing theory for gateway selection in hybrid MANET, Wirel. Pers. Commun. 111 (2020) 9–32, https://doi.org/10.1007/s11277-019-06842-9.
[20] X. Wang, Q. Peng, Y. Li, Cooperation achieves optimal multicast capacity-delay scaling in MANET, IEEE Trans. Commun. 60 (2012) 3023–3031, https://doi.org/10.1109/tcomm.2012.081512.110535.
[21] S. Yin, X. Lin, MALB: MANET adaptive load balancing, in: IEEE 60th Vehicular Technology Conference, vol. 4, IEEE, 2004, pp. 2843–2847.
[22] G. Last, M. Penrose, Lectures on the Poisson Process, vol. 7, Cambridge University Press, 2017.
[23] A. Leon-Garcia, Probability and Random Processes for Electrical Engineering, 2nd ed., Addison-Wesley, Reading, MA, 1994.
[24] J.D. Little, A proof for the queuing formula: L= $\lambda$ W, Oper. Res. 9 (1961) 383–387, https://doi.org/10.1287/opre.9.3.383.
[25] R. Meester, K. Slooten, Some Philosophy of Probability, Statistics, and Forensic Science, Cambridge University Press, 2021, pp. 1–29.
[26] R.G. Gallager, Stochastic Processes: Theory for Applications, Cambridge University Press, 2013.
[27] P. Zhou, S. Paul, A. Dutta, C. Varela, F. Kopsaftopoulos, On formal verification of data-driven flight awareness: leveraging the Cramér-Rao lower bound of stochastic functional time series models, in: International Conference on Dynamic Data Driven Applications Systems, Springer, 2022, pp. 44–52.
[28] M. Rabin, Communication between rational agents, J. Econ. Theory 51 (1990) 144–170.
[29] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, P. Wischnewski, SPASS version 3.5, in: International Conference on Automated Deduction, Springer, 2009, pp. 140–145.
[30] K. Korovin, IProver–an instantiation-based theorem prover for first-order logic (system description), in: International Joint Conference on Automated Reasoning, Springer, 2008, pp. 292–298.
[31] S. Paul, G. Agha, S. Patterson, C. Varela, Eventual consensus in synod: verification using a failure-aware actor model, Innov. Syst. Softw. Eng. 19 (2023) 395–410.
[32] M. Manzano, Extensions of First-Order Logic, Cambridge University Press, 1996.
[33] K. Arkoudas, Simplifying proofs in fitch-style natural deduction systems, J. Automat. Reason. 34 (2005) 239–294, https://doi.org/10.1007/s10817-005-9000-3.
[34] H. Attiya, C. Dwork, N. Lynch, L. Stockmeyer, Bounds on the time to reach agreement in the presence of timing uncertainty, J. ACM 41 (1994) 122–152, https://doi.org/10.21236/ada229766.
[35] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, B. Pochon, The perfectly synchronized round-based model of distributed computing, Inf. Comput. 205 (2007) 783–815, https://doi.org/10.1016/j.ic.2006.11.003.
[36] H. Attiya, T. Djerassi-Shintel, Time bounds for decision problems in the presence of timing uncertainty and failures, J. Parallel Distrib. Comput. 61 (2001) 1096–1109, https://doi.org/10.1006/jpdc.2001.1730.
[37] P. Berman, J.A. Garay, K.J. Perry, et al., Towards optimal distributed consensus, in: FOCS, vol. 89, 1989, pp. 410–415.
[38] K.L. McMillan, O. Padon, Deductive verification in decidable fragments with ivy, in: Static Analysis, Springer, 2018, pp. 43–55.
[39] D. Malkhi, L. Lamport, L. Zhou, Stoppable Paxos, Technical Report, Microsoft Research, 2008.
[40] O. Padon, K.L. McMillan, A. Panda, M. Sagiv, S. Shoham, Ivy: safety verification by interactive generalization, ACM SIGPLAN Not. 51 (2016) 614–630, https://doi.org/10.1145/2980983.2908118.
[41] C. Drăgoi, T.A. Henzinger, D. Zufferey, PSync: a partially synchronous language for fault-tolerant distributed algorithms, in: ACM SIGPLAN Notices, vol. 51, ACM, 2016, pp. 400–415.
[42] H. Debrat, S. Merz, Verifying fault-tolerant distributed algorithms in the heard-of model, Arch. Formal Proofs 2012 (2012).
[43] B. Charron-Bost, A. Schiper, The heard-of model: computing in distributed systems with benign faults, Distrib. Comput. 22 (2009) 49–71, https://doi.org/10.1007/s00446-009-0084-6.
[44] C. Hawblitzel, J. Howell, M. Kapritsos, J.R. Lorch, B. Parno, M.L. Roberts, S. Setty, B. Zill, IronFleet: proving practical distributed systems correct, in: Proceedings of the 25th Symposium on Operating Systems Principles, ACM, 2015, pp. 1–17.
[45] C. Hawblitzel, J. Howell, M. Kapritsos, J.R. Lorch, B. Parno, M.L. Roberts, S. Setty, B. Zill, IronFleet: proving safety and liveness of practical distributed systems, Commun. ACM 60 (2017) 83–92, https://doi.org/10.1145/3068608.
[46] K.R.M. Leino, Dafny: an automatic program verifier for functional correctness, in: International Conference on Logic for Programming Artificial Intelligence and Reasoning, Springer, 2010, pp. 348–370.
[47] S. Paul, G.A. Agha, S. Patterson, C.A. Varela, Verification of eventual consensus in synod using a failure-aware actor model, in: Proceedings of the 13th NASA Formal Methods Symposium (NFM 2021), 2021, pp. 249–267.
[48] O. Hasan, Probabilistic analysis using theorem proving, in: 21 St International Conference on Theorem Proving in Higher Order Logics, Citeseer, 2008, p. 21.
[49] M.J. Gordon, Mechanizing programming logics in higher order logic, in: Current Trends in Hardware Verification and Automated Theorem Proving, Springer, 1989, pp. 387–439.
[50] O. Hasan, S. Tahar, Formalization of the standard uniform random variable, Theor. Comput. Sci. 382 (2007) 71–83, https://doi.org/10.1016/j.tcs.2007.05.009.
[51] O. Hasan, S. Tahar, Using theorem proving to verify expectation and variance for discrete random variables, J. Automat. Reason. 41 (2008) 295–323, https://doi.org/10.1007/s10817-008-9113-6.
[52] O. Hasan, S. Tahar, Formal probabilistic analysis: a higher-order logic based approach, in: International Conference on Abstract State Machines, Alloy, B and Z, Springer, 2010, pp. 2–19.
[53] O. Hasan, S. Tahar, Formal verification of tail distribution bounds in the HOL theorem prover, Math. Methods Appl. Sci. 32 (2009) 480–504, https://doi.org/10.1002/mma.1055.
[54] O. Hasan, S. Tahar, Reasoning about conditional probabilities in a higher-order-logic theorem prover, J. Appl. Log. 9 (2011) 23–40, https://doi.org/10.1016/j.jal.2011.01.001.
[55] T. Mhamdi, O. Hasan, S. Tahar, On the formalization of the Lebesgue integration theory in HOL, in: International Conference on Interactive Theorem Proving, Springer, 2010, pp. 387–402.
[56] T. Mhamdi, O. Hasan, S. Tahar, Formalization of measure theory and Lebesgue integration for probabilistic analysis in HOL, ACM Trans. Embed. Comput. Syst. 12 (2013), https://doi.org/10.1145/2406336.2406349.

[57] D. Chaouch, Formalization of Continuous Time Markov Chains with Applications in Queueing Theory, Master's thesis, Concordia University, 2015.
[58] M. Qasim, Formalization of Normal Random Variables, Master's thesis, Concordia University, 2016.
[59] C.A. Varela, Programming Distributed Computing Systems: A Foundational Approach, MIT Press, 2013, http://wcl.cs.rpi.edu/pdcs.
[60] G. Agha, Actors: A Model of Concurrent Computation in Distributed Systems, The MIT Press, 1986.