Flight Object Sharing Capability Using Blockchain

Duncan Thomson[1], Steven R. Bodie[2], David E. Bryson[3], Timothy S. Luc[4], and Joel G. Korb[5]
The MITRE Corporation, McLean, VA, 22102, USA

Abstract

This paper describes a concept and a prototype for sharing flight information using blockchain technology. Providing all stakeholders access to complete, consistent, and up-to-date information about each flight facilitates efficient aviation operations. Existing flight information exchange methods are limited; the concept of a "flight object" that provides a complete solution for all stakeholders has yet to be realized. A complete solution requires either a centrally administered data store, which is unsuitable for an international context, or a distributed ledger. Distributed ledgers are at the heart of blockchain technology, which makes this technology a good match for implementing the flight object. The authors have proven it is possible to provide a complete flight object solution using blockchain technology by demonstrating a prototype based on Tendermint: an open-source blockchain implementation. The demonstrated solution offers additional benefits like strong integrity guarantees and role-based update permissions enforcement. Prototype performance indicates a production implementation is likely to provide sufficient speed and capacity to support global aviation operations for flight planning negotiation, as well as possible future concepts such as trajectory-based operations and international flow management. The paper's conclusion discusses steps necessary for the flight object sharing capability to be adopted as the basis for a real-world international aviation solution

I. Introduction

The idea of a flight object providing stakeholders complete, consistent, and up-to-date information about every flight has been discussed for many years. In 2000, a MITRE paper [1] defined the flight object as "…a collection of common information elements describing an individual flight and available electronically for use by both the National Airspace System (NAS) users and the Air Traffic Management (ATM) service providers." And, a 2003 EUROCONTROL paper [2] described the concept as follows: "For each flight which is planned, is currently active, or has taken place, there will be a 'flight object' which contains the latest confirmed information about the flight…". Providing an interoperable flight object to stakeholders such as ATM service providers (ASPs), aircraft operators, air defense units, and airport authorities was described in Ref. [2] as a means to reduce "unnecessary workload, inefficient use of resources, and unnecessary delays." The flight object remains an important part of the International Civil Aviation Organization (ICAO)

[1] Senior Principal Engineer, T8C1 Aviation Communications
[2] Software Systems Engineering Lead, T8C1 Aviation Communications
[3] Blockchain Technology Lead, T851 Agile Engineering & Innovation
[4] Senior Electrical Engineer, T8C1 Aviation Communications
[5] Senior Software Systems Engineer, T831 Cognitive Science and Artificial Intelligence

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

Global Air Traffic Management Concept [3] to this day. However, despite its importance, the flight object concept has never been implemented. There are flight object implementations within specific ASP systems, but these are internal solutions not accessible by other ASPs or airspace users. There are solutions for digital exchange of flight information among international ASPs and airspace users, but the information exchanged is limited and these methods do not ensure that all participants have access to complete, consistent, and up-to-date information. The objective of this paper is to show how the flight object concept can be fully realized, through the application of blockchain technology.

Section II describes limitations of legacy flight information exchange methods, which are based on air traffic services (ATS) messages. Section II also describes improvements to ATS messaging being explored by the Federal Aviation Administration (FAA) and international aviation participants. These improvements, which include the introduction of XML message structures and message bus technology, will allow more flight information to be exchanged but still cannot guarantee that complete and consistent information is available to every concerned entity. At first, it might appear that a network accessible central database could be the solution. However, in Sec. II, we explain why a centralized solution is not suitable for international aviation. To make the flight object available internationally, to ASPs as well as airspace users, a distributed solution is needed.

In Sec. III, we describe the flight object sharing capability (FOSC) concept, which is a distributed solution using blockchain technology. In this concept, each stakeholder controls their own flight object instances, and consistency is maintained among these instances using a blockchain-based distributed consensus protocol. Section III describes the proof-of-concept FOSC prototype we built to test the feasibility of this concept.

Having implemented the FOSC, we also needed to ensure it can provide the necessary functionality to be used in international aviation operations. Section IV describes how this was accomplished. We created an end-system application to represent the automation systems used by ASPs and airline flight operation centers (FOCs). We developed an operational scenario consistent with international collaborative flight planning concepts being explored by the international aviation community. Then, we used the end-system application to demonstrate how airlines and ASPs could use the FOSC to conduct early flight planning for a flight traversing multiple national airspaces. This successful demonstration with our prototype provides confidence that an FOSC implemented in this way can provide the functionality necessary to support international flight planning operations.

The next question to be addressed is whether the FOSC can provide the necessary performance and scale. Section V describes how we addressed this question. We built a performance test framework and used it to measure end-to-end latency and query response times under varying conditions and at varying scale. The results show that even our initial rapid prototype, running on limited capacity virtual machines, could potentially scale up to support international aviation flight planning operations. A robust and optimized implementation should be able to support not only flight planning but even more dynamic scenarios like envisioned trajectory-based operations (TBO). [A trajectory defines where an aircraft will be at any given point in time. This is

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

sometimes referred to as a "four-dimensional (4-D) trajectory," referencing the four parameters: time, latitude, longitude, and altitude.]

Section VI discusses some of the steps that would be required to transition the FOSC from a proof-of-concept prototype to an operational system. We present a possible deployment model for FOSC blockchain nodes, end systems, and governance entities; and we discuss the need for an ICAO standardization path.

II. Problem Statement

A. Legacy Flight Information Sharing

The flight information life cycle has remained relatively unchanged for many years. The current process focuses on the international flight operator "filing" a flight plan by sending messages over the Aeronautical Fixed Telecommunications Network or the ATS message handling system (AMHS). The messages include filed flight plan (FPL), modification, delay, cancellation, and many others. Reflecting their teletype legacy, these messages are encoded in uppercase letters, numbers, and punctuation characters. Figure 1 shows an example FPL message. In this example, we can see the flight call sign (UAL43); aircraft type (B763 means this is a Boeing 767-300); and characters providing more details on the flight rules, aircraft equipage, and so on. Further down, we see that this flight is planned to depart Denver (KDEN) at 0030 hrs, with a cruising speed of 459 kt (N0459) and an altitude of 32,000 ft (F320), following a route from the airport direct (DCT) to the DBL waypoint, and so on.

```
(FPL-UAL43-IS
-B763/H-SDGHIJ7RWXYZ/SB1D1
-KDEN0030
-N0459F320 DCT DBL DC KROST DCT OAK
DCT BEBOP R464 BITTA MAGGI3
-PHNL0654 POGG
-PBN/A1D2L1 NAV/RNVD1E2A1 REG/N669UA
EET/KZLC0041 KZOA0131 KZAK0240 PHZH0608
CODE/A8D76B)
```

Fig. 1 Example Legacy Flight Plan (FPL) Message

As the flight progresses, information is transferred point to point. Each ASP controls and manages elements of flight information by receiving or transmitting flight plan amendments. In some cases, the information is transferred by voice communications or requires human interpretation. In other cases, the information is transferred point to point in ATS messages. Only limited information can be transmitted electronically due to the limited ATS message format. For example, in the FPL message, there is no way to provide a complete trajectory, nor is there a means to communicate constraints (for example, altitude limits, likelihood of congestion, or time constraints) that apply at points along the route. Another limitation is that the ATS messages are only sent to specific recipients, and not to all stakeholders. Each system will only have access to information that it has originated or that has been specifically sent to it by another system. And, since flight information is only communicated to neighbors when very specific events occur, stakeholders will frequently be unaware that the information has changed until one of these triggering events occurs. The net result of all these limitations is that there is no assurance that all stakeholders have complete, consistent, and up-to-date flight information.

The ICAO Global Air Traffic Management Operational Concept [3] summarizes the problem as

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

"limited facilities for real-time information exchange among ATM, aerodrome operators and aircraft operators, resulting in less than optimal responses to real-time events and changes in users' operational requirements." The ICAO plan for TBO [4] provides more detail on these limitations, citing "lack of information sharing," "disparate information across participants and automation systems," and "No single, consistent trajectory is maintained using the best-known information." And, in Ref. [5], the ICAO asserts:

*The Global ATM Operational Concept has greater data requirements than can be supported by the existing flight planning provisions. These include system-wide information sharing, providing early intent data, management by trajectory, CDM,** and high automation support requiring machine readability and unambiguous information.*

B. Improvements Currently Underway

Overcoming the limitations of the global ATM system is a goal of the international aviation community. The ICAO Manual on ATM System Requirements [6] states "It is expected that through dynamic renegotiation of agreed 4-D trajectory contracts … the ATM system will not experience 'chokepoints.' Potential ATM system chokepoints should be increasingly more predictable as 4-D trajectories become available together with automated tools for mitigation."

The initial steps being taken to improve international flight information sharing to make 4-D trajectories available include Flight and Flow–Information for a Collaborative Environment (FF-ICE) [5] and the Flight Information Exchange Model (FIXM) [7]. FF-ICE is a set of international flight management collaboration concepts that use the information contained in FIXM, which is an evolving structured model of flight information. FIXM provides a much richer and more complete structure for representing flight information than is possible with legacy ATS messages. In FIXM, a Flight is a complex data structure that is built from other data structures, including an Aircraft, a Departure, an Arrival, a Desired route, and so on. Each of these is in turn a complex data structure built up of other structures. A FIXM Flight structure can also contain trajectory information. A FIXM Route Trajectory is a complex type that contains climb and descent profiles as well as schedules, and an array of up to 2000 elements that specify predicted aircraft state and position along the route as a function of time. The Route Trajectory structure can also specify constraints that apply at any point along the route. This capability to represent complex structured flight information allows the airspace users and ASPs to exchange the additional information needed for the FF-ICE concepts. Initial FF-ICE concept demonstrations have been conducted in which flight information is exchanged using FIXM-based messages, formatted in Extensible Markup Language (XML).

While the information content of FIXM is a significant improvement over the legacy ATS message formats, there remains the question of how this content will be shared among different systems. Initially, the FF-ICE concepts will be implemented by ASP automation systems sending FIXM-based messages addressed individually to other automation systems, just as ATS messages are currently sent in a point-to-point manner. In the future, FF-ICE might be extended to use the publish/subscribe paradigm envisioned in global System Wide Information Management (SWIM) concepts [8]. This would allow

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

an ASP to notify all affected stakeholders of a change to flight information by publishing an appropriate FIXM message, which SWIM would disseminate to all subscribers. Regardless of whether point-to-point or publish/subscribe messaging is used, aviation standards bodies will need to define the rules that determine which entities should send which messages to which other entities, and when these messages should be sent. Currently, the rules for sending ATS messages for international flights are defined in ICAO Document 4444 [9], with specific rules for the United States defined in Ref. [10]. These rules are currently relatively simple, but as FF-ICE is extended to handle concepts such as TBO, the information exchanges will become more dynamic and complex, and the message transmission rules will need to be reexamined and may need to be revised or extended. The rules must accommodate not only the expected scenarios but also all the possible "error paths," including message transmission failures, systems crashing and restarting, and systems misbehaving in unpredictable or even malicious ways. As we attempt to handle increasingly dynamic and complex scenarios, it will become increasingly difficult to ensure that our rules are correctly handling all the various things that can go wrong.

Another, perhaps even more serious, problem with an approach based on ATS messaging (either legacy or FIXM-based) is the resulting software complexity. As illustrated on the left side of Fig. 2, with this approach, each ATM application uses a messaging application program interface (API) defined by a messaging service provider to send and receive ATS or FIXM messages. The ATM application must of course contain the business logic (e.g., route or trajectory negotiation and management) that makes changes to flight information and reacts to changes made by other systems. With a messaging approach, the ATM application must also contain the logic that uses the messaging API to generate messages when needed. While the underlying details of message transmission are encapsulated behind the API, the information exchange logic that determines which messages to send, and when to send them, remains within the same application as the business logic. This increases application complexity and will make it difficult to implement new operational concepts without affecting the message transmission logic.

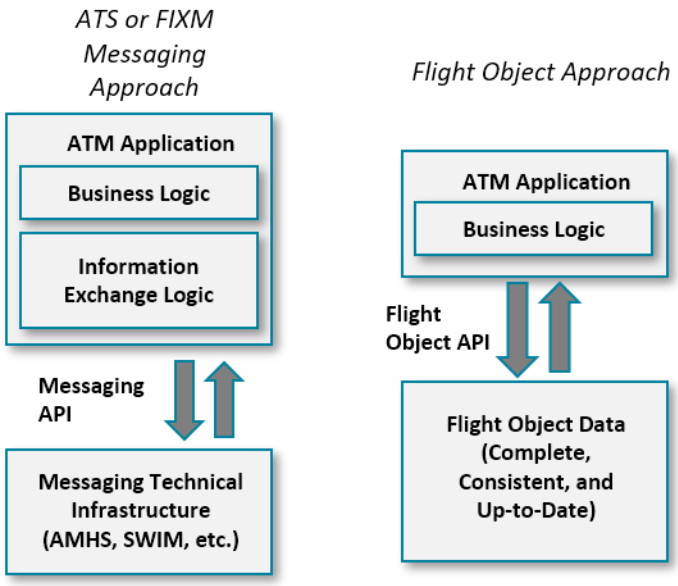

Fig 2. Messaging Approach vs. Flight Object Approach

The complexity of the messaging approach can be avoided by defining a flight object API that clearly separates the business logic from the information exchange logic. Such an approach is illustrated on the right side of Fig. 2. In this approach, the ATM application accesses a well-defined API that allows it to read and write the flight object, and the information exchange logic is encapsulated behind this API. The problem then is to find a way to

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

implement this flight object approach in a way that is suitable for aviation.

C. Distributed Consensus

To address the problems discussed earlier in this paper, we need "…a globally consistent mechanism and consistent interface for providing and receiving flight and flow (FF) information" [2]. This mechanism should clearly separate the application "business logic" software performing functions such as route planning and traffic management from the software that ensures all participants have complete and consistent information.

One might consider solving the problem by providing a centralized database that contains a complete set of flight information, with all participants having remote access to the database so they can update it and read it, as shown in Fig.3. However, this is not a viable solution in an international context because it requires a central entity that controls and operates the database. Such an entity would need the technical capability to run an operational system critical to the efficient operation of international aviation. It would need a source of funding, and it would need to be trusted and accepted globally by ASPs and airspace users. It does not seem possible for the international aviation community to select and fund a single service provider, either private or governmental, that could fill this role globally. Furthermore, the database administrators would have the ability to control international flight information content and could control which entities have access to the information. Because of the daunting political, economic, and technical challenges, we do not believe that a centralized flight object database is a feasible or desirable solution for international aviation.

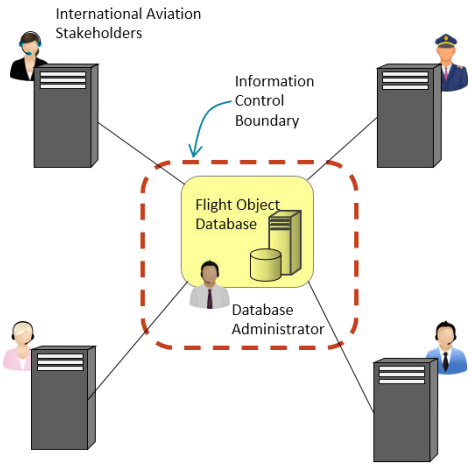

Fig. 3 Flight Object Database Solution

We believe that ASPs, airspace users, and other international aviation stakeholders are much more likely to adopt a solution in which they can operate autonomously. We seek a solution in which each participant controls their own instance of the flight object store, with some internationally standardized way to maintain consistency among all instances, as shown in Fig. 4. Such a solution would allow each stakeholder to operate their own system to maintain their instance of the flight object, or to select a service provider to perform this function on their behalf. It would not require stakeholders to agree on, and place their trust in, a central flight object database operator. The question then becomes: If we allow each participant to control their own instance of the flight object, how do we ensure that these instances are kept consistent?

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

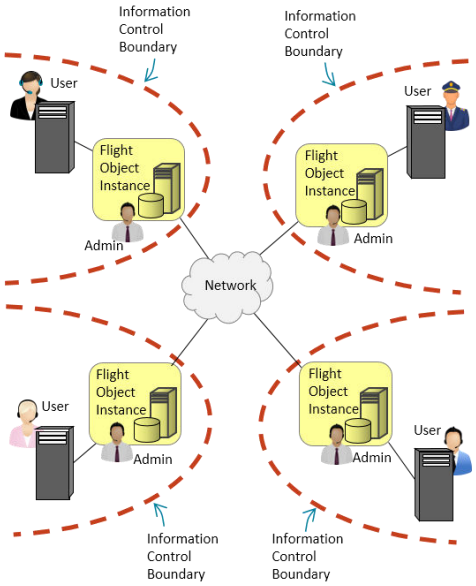

Fig. 4 Distributed Consensus Protocol Flight Object Solution

Maintaining information consistency among a set of independent instances distributed across a network is a well-studied problem in computer science, which is referred to as "distributed consensus." Distributed consensus is discussed in classic papers such as Refs. [11–13]. There are well-known algorithms, known as "distributed consensus protocols," that that solve distributed consensus problems with formally provable properties. A distributed consensus protocol solution provides each party with its own copy of the shared information, with consistency maintained by a protocol that operates over a communications network connecting the instances. If we can apply a distributed consensus protocol to the flight object problem, we can create the desired architecture shown in Fig. 4, and we can do this in a way that maintains a clear separation between the application business logic (within the automation system) and the information sharing logic (embodied in the distributed consensus protocol). The next section describes how this can be done using blockchain technology.

III. Proposed Flight Object Solution Based on Blockchain Technology

A. What is Blockchain?

A full explanation of blockchain concepts is beyond the scope of this paper; we provide only a brief introduction for readers who may not be familiar with the technology. It can be thought of as a technology for distributing information with strong guarantees of integrity. The term "blockchain" refers to the way that information is organized into blocks, each of which contains the secure hash of the previous block. The hashes link the blocks into a chain in such a way that any changes to any block can be easily detected and rejected. The other essential element of any blockchain solution is a distributed consensus protocol that allows a set of participants to come to agreement on the content of the blocks. The information contained in the blockchain is referred to as the "distributed ledger." In the original blockchain application, Bitcoin [14], the digital ledger tracks the transfer of virtual "coins" from one owner to another. Following the success of Bitcoin, other digital currencies and other applications were introduced, built on variations of the blockchain technology used in Bitcoin.

B. Why Blockchain for the Flight Object?

While distributed consensus algorithms are well studied, they are complex and difficult to implement correctly. Implementing these algorithms for any given application in a way that is robust and performs well in the real world has in the past been a daunting task. However, this situation has changed with the emergence of blockchain. As stated earlier in this paper, at the heart of every blockchain is a distributed consensus protocol, which maintains a

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

distributed ledger without requiring any central store. Multiple blockchain implementations are now available as open source software, and these implementations are designed to be extensible for different applications. This provides robust and well-tested solutions that can be applied wherever a distributed ledger is needed. For international flight information sharing, the distributed ledger can be a set of flight objects, and the blockchain can provide all stakeholders with a strong guarantee that they have complete and consistent access to this information. With blockchain, this can be done without the need for any central store or trusted administrator, and without the need for each stakeholder to trust that every other stakeholder will always act correctly. Blockchain also includes strong integrity guarantees, which provides assurance that critical flight object information has not been corrupted or tampered with. Thus, blockchain provides a base of technology that can be easily applied and is very well suited to solving the flight object problem.

C. Flight Object Sharing Capability

Figure 5 illustrates the FOSC concept. The FOSC is not a single entity; rather, it is a distributed capability achieved by multiple instances. Each instance contains a state machine that maintains a set of flight objects containing FIXM-structured information. The state machines process transactions that represent creation, deletion, and updates of the flight objects. The transactions are communicated among all instances using a blockchain-based distributed consensus protocol, which ensures that every state machine receives exactly the same set of transactions, in the same order. Note that the illustration shows a JavaScript object notation (JSON) representation of FIXM rather than the more usual XML representation. The choice of JSON versus XML will be discussed in the following.

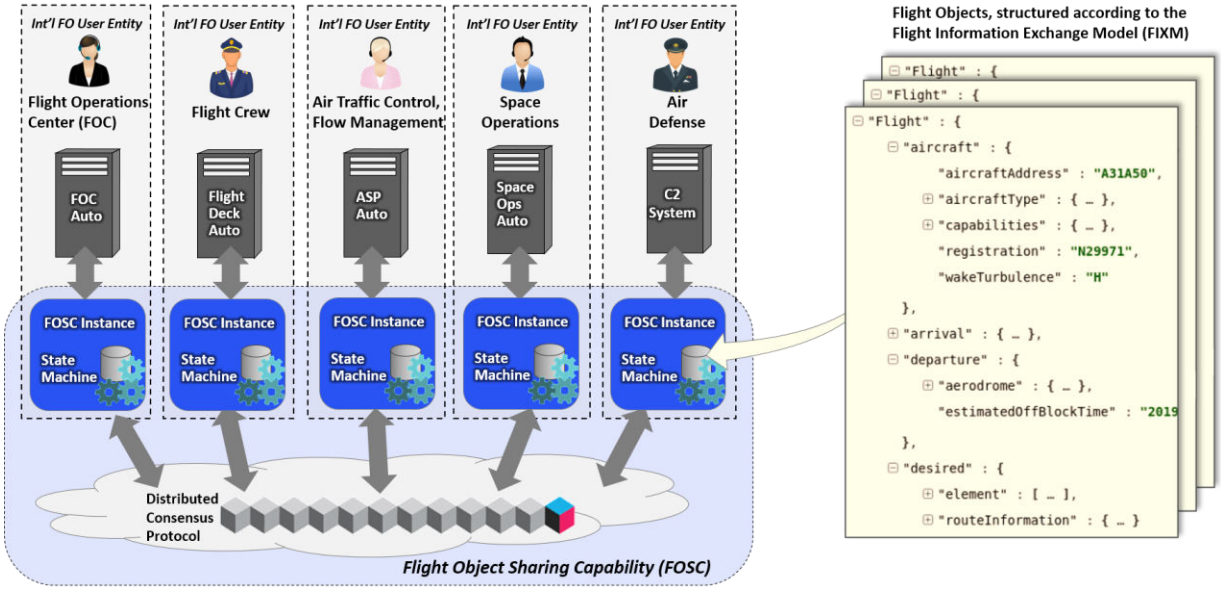

Fig. 5 Flight Object Sharing Capability (FOSC) Concept

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

In this concept, each instance of the FOSC provides an API that supports one or more stakeholder automation systems. The stakeholders might include flight operations centers; flight crew; ASPs providing air traffic control and flow management functions; entities performing space launch, monitoring, and recovery operations; and military entities performing air defense functions.

D. Proof of Concept Prototype

As part of a mission-oriented research activity, a MITRE team set out to create a prototype to test the feasibility of the FOSC concept. The first step in implementing the prototype was to select blockchain software on which to build. We selected Tendermint [15] because it has the following characteristics:

1. It is based on a private, permissioned model, in which blockchain nodes have known identities and can enforce rules that determine which end users can update the flight object.
2. It contains a Byzantine fault tolerant (BFT) consensus protocol, which is robust in the face of network failures, node crashes or errors, or malicious attempts to subvert the network (no single point of failure).
3. It provides high performance and is scalable, making it suitable for a rapid rate of flight object transactions.
4. The software is open source and extensible, allowing flight object logic to be implemented separate from the internal blockchain logic.

Having selected a blockchain implementation, we extended it to allow end user applications to submit flight object transactions (creation, updates, and deletion) to the blockchain and to maintain the flight object state within the blockchain nodes.

Figure 6 illustrates the major components that form our prototype.

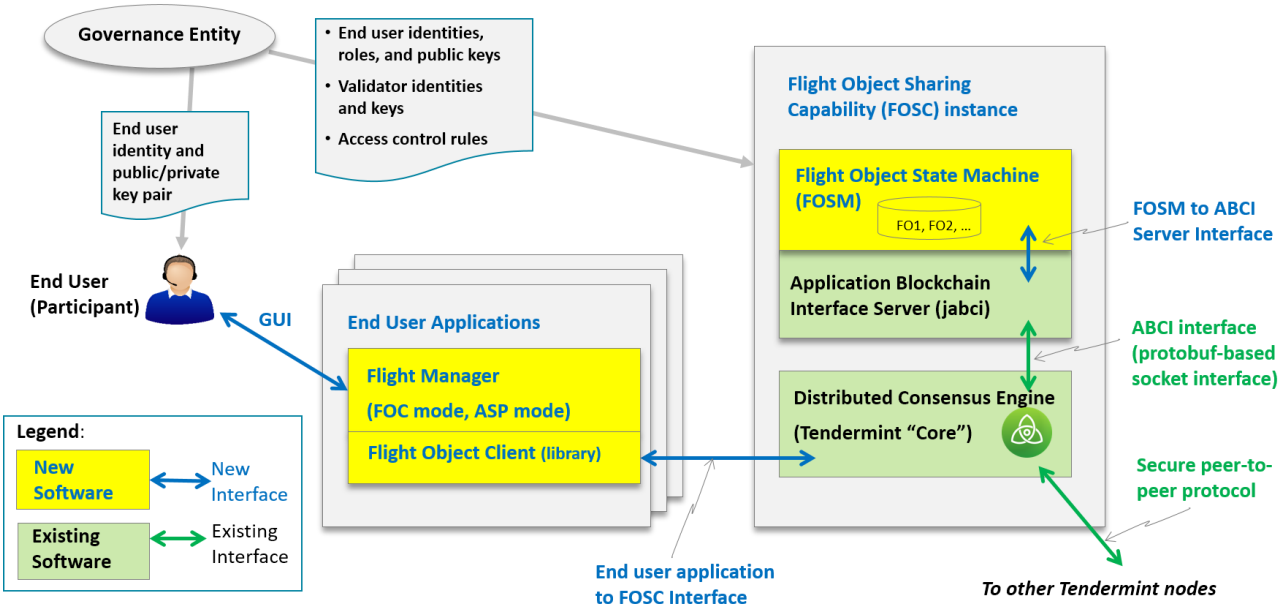

Fig. 6 Proof-of-Concept Implementation

The items and text in green indicate existing open source software and interfaces. The items and text in gold and blue indicate new software that we wrote and new interfaces that we designed. An instance of the FOSC includes the core blockchain functionality that implements the distributed consensus protocol provided by the Tendermint core. The application blockchain interface (ABCI) server is an open source component that allows Tendermint to be extended with application-specific functionality. In our case, the application-specific functionality is the flight object state machine (FOSM), which maintains the flight object state. End user applications use an API provided by the flight object client module to submit transactions to create, update, or delete flight objects. The API also supports a subscription filter that will cause the end system to be notified whenever flights of interest to the application change. This "notification push" model means end systems do not need to continually poll to determine when new information is available. Rather, the end systems can register their interest in flights using subscription filters specified in terms of FIXM elements like the operator of the flight, departure and destination airport, aircraft identifier, and so on. The API also allows the application to query the Tendermint node for flight object information.

The FOSM maintains the flight objects' state by processing transactions received from the end user applications via the Tendermint consensus protocol. In doing so, the FOSM enforces rules determining which end user transactions can make changes. The BFT consensus protocol ensures that every node makes an identical decision on each transaction, and thereby ensures state information consistency. For our prototype, the rules are relatively simple. We required that: every transaction must be signed by a known end system with an appropriate role (e.g., an airline or ASP); only the airline operating a flight can update the flight information; only an ASP can update the ASP's "agreement" status information within the flight object; and other rules as appropriate for testing.

For our prototype, we wrote the transaction processing rules into the state machine code. However, this approach is not what we would recommend for a production solution. A more extensible implementation would allow users to load rules into the state machine using a machine-processable access control language like the eXtensible Access Control Markup Language (XACML).††

Blockchain uses cryptographic hashes to provide strong integrity guarantees. A block of flight object transactions commits to the chain at a nominal rate of once per second. Then the FOSM computes a hash of the application state using a Merkle tree containing the serialized form of all flight objects in the state store. The application state hash becomes part of the block header, and the block header itself is then hashed and linked to the next block. The Tendermint-distributed consensus protocol ensures that all instances compute the same block header, which guarantees that all instances have the same application state. End systems also can check block hashes to validate the integrity of data received from an FOSM instance or from an archive. These strong integrity guarantees could be extremely valuable, for example, when conducting an incident analysis based on archived data.

One of the interesting design choices we faced in creating the prototype was the format to use for serializing the flight objects for storage, transmission, and encryption functions. FIXM is an

abstract model for the flight object information content, which is defined in universal markup language (UML). The FF-ICE prototypes and other implementations that have used the FIXM have used a standard XML representation of the FIXM UML model. Our prototype used a JSON representation of the FIXM. We wrote the end user application as a browser-based graphical user interface (GUI), which made JavaScript the natural choice of programming language for the GUI since it is supported within the browser. We also used JavaScript (running in Node.js) for the FOSM to maximize code reuse between the GUI and the FOSM. Having chosen JavaScript as a programming language, JSON became the natural choice for representing FIXM. We considered efficient serialization formats such as Apache Avro and Google protocol buffers (protobufs) (which is used internally within Tendermint) but chose a simple JSON serialization of our JavaScript flight object data structures for prototyping expediency. We did use an open source library that provides a deterministic JSON representation, which is important in order to ensure that the hash values used within the blockchain are deterministic. For a production environment, an XML representation would be more consistent with the way the FIXM is being used in FF-ICE. More research into an optimized serialization format for the flight object would be valuable.

IV. Demonstration and End User Applications

To demonstrate the FOSC functionality for international aviation, we created a demonstration based on FF-ICE concepts. The demonstration shows an airline planning an international flight (we assumed a flight from San Francisco to Singapore). The airline FOC uses the FOSC to provide early-intent flight route information to ASPs along the route, receives feedback from the ASPs regarding constraints that might impact the flight, and adjusts the route accordingly.

Figure 7 illustrates the demonstration scenario. It begins with the FOC creating a flight object containing the desired route. The ASP, having registered a subscription indicating interest in flights that match a certain filter (e.g., flights that are destined for its international airports), is notified of the new flight; and it then queries the API for the flight object data. The ASP then uses the API to update the flight object to indicate that the route is acceptable. However, as the scenario develops, a constraint emerges (e.g., congestion, severe weather, military activity, etc.) that would affect the flight. The ASP updates the flight object with this information, triggering the FOC to adjust the desired route. This scenario ends with the AOC deciding to file the flight plan. However, we could easily extend the scenario to demonstrate the FOSC being used to share flight information throughout the life of the flight, enabling some of the more complex scenarios envisioned for future FF-ICE versions.

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

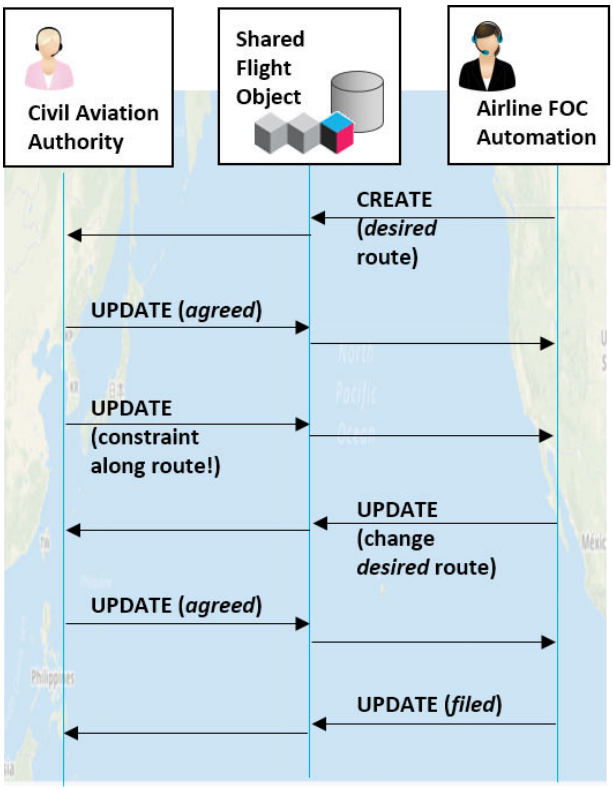

Fig. 7 Demonstration Scenario

Note that, while Fig. 7 shows only one (logical) flight object being shared by the FOC and the ASP, our demonstration has multiple instances, which are kept consistent by the Tendermint blockchain distributed consensus protocol.

To conduct the demonstration, we created an application called the Flight Manager with different modes allowing it to represent the automation systems used by ASPs and FOCs. The Flight Manager is not intended to realistically emulate the functionality of FOC or ASP automation. It is a simply a demonstration tool to provide a graphical user interface and enough functionality to allow an operator to perform some notional flight management functions using the flight object client API to access the FOSC. The screen capture in Fig. 8 shows the Flight Manager GUI at the point in the scenario at which the FOC has selected a new route to avoid the constraint, and the Singapore ASP has indicated its agreement with the route.

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

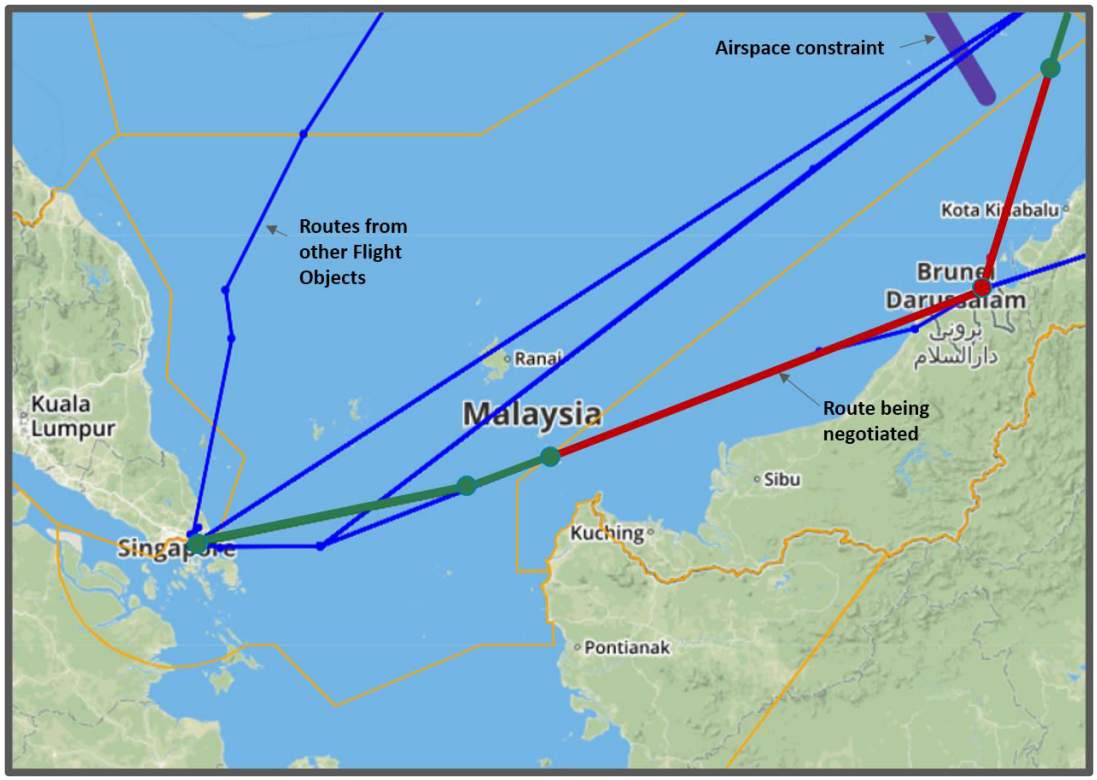

Fig. 8 Flight Manager GUI

In Fig. 8, the blue lines indicate flight objects for other flights that have been filed. The green and red line shows the route of the flight being planned. (The route is green in Singapore airspace and red elsewhere because, at this point in the demonstration scenario, Singapore has agreed to the route, but other ASPs have not.)

V. Performance

A. Performance Testing Overview

Published test performance on the underlying Tendermint software [15] indicates that the core software can support high update rates (thousands of transactions per second) on dozens of nodes. However, blockchain application performance depends on the application specifics, and so we ran tests to measure the performance of our prototype FOSC.

Our performance testing measured flight object update latency as a function of the number of blockchain nodes and the rate of flight object transactions. Update latency is the elapsed time from when one node initiates a flight object transaction (create, update, or delete) to the time that another node receives a notification that the flight has been updated and retrieves the new data.

We made the measurements using the test framework illustrated in Fig. 9. The test framework includes a test control application and a variable number of test nodes. The test nodes receive commands and configuration data from the test control application via a message broker, and they report test results (measured flight object update latency) via the same broker. Each test node generates flight object transactions, and each

transaction is disseminated via Tendermint and received by every other node.

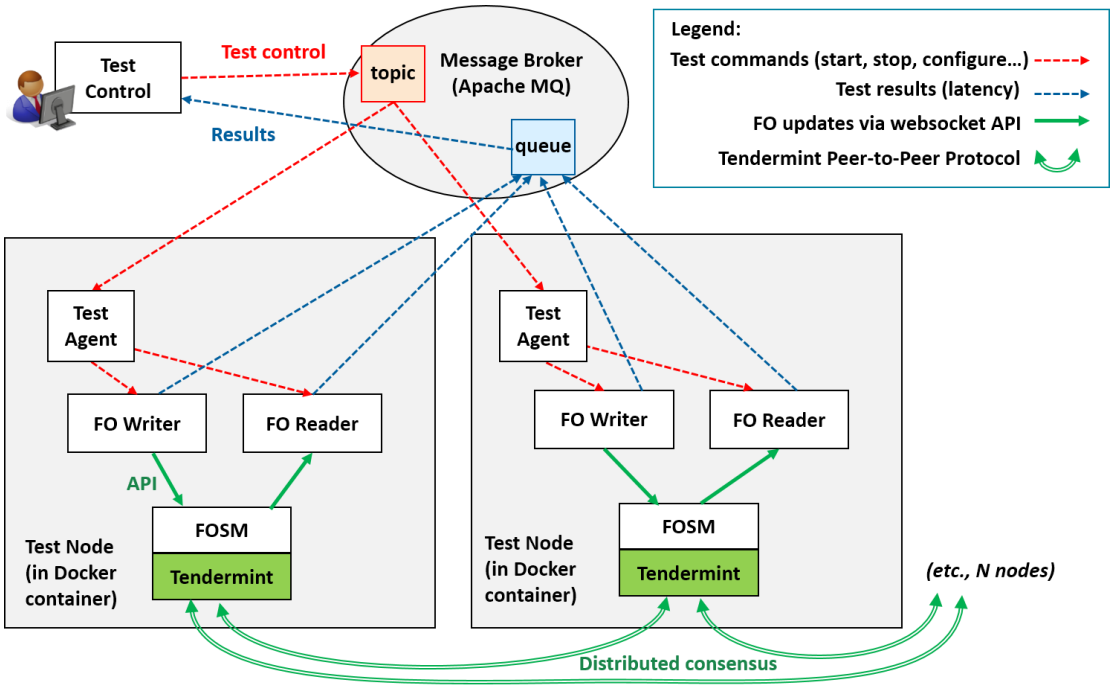

Fig. 9 Performance Test Framework

Each test node includes: 1) a flight object writer application that creates, updates, and deletes flight objects at a configurable rate‡‡; 2) a flight object reader application that receives notifications that flight objects have been updated, retrieves the updated flight object data, and computes and reports the end-to-end latency§§; 3) a test control agent that configures and controls the node; and 4) a FOSM blockchain instance, built on the Tendermint core, that disseminates the transactions via the blockchain.

Each test node runs in a Docker container, allowing easy instantiation of many nodes, limited by the computing power of the underlying infrastructure available.

B. Performance as a Function of the Number of Blockchain Nodes

Figure 10 shows the average latency measured as the number of nodes was varied.

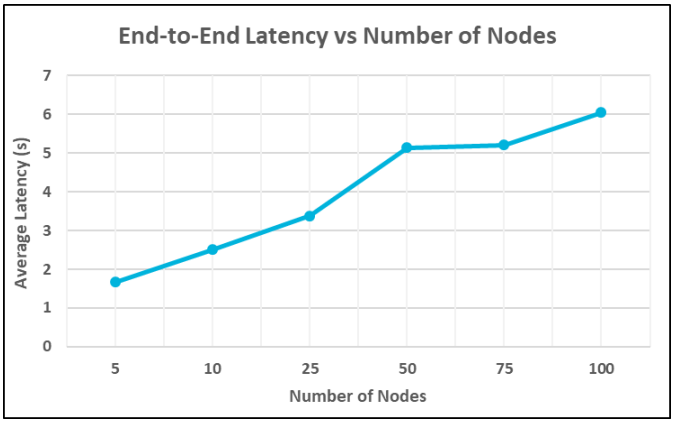

Fig. 10 Latency vs. Number of Nodes

These tests were run with flight object transactions being generated at a rate of 26 per

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

minute, including 12 flight object creates, two updates, and 12 deletes each minute. These parameters were chosen as follows:

1. Statistics from the International Air Transport Association (IATA) and the U.S. Department of Transportation provide a rough estimate of 16,295 international flights per day, or roughly 12 flights per minute.
2. Each flight that occurs causes a flight object to be created, and ultimately deleted, resulting in 12 creates/minute and 12 deletes/minute.
3. The number of updates for each flight in operation will depend on how the flight object is being used and what types of changes would require an update to be distributed. Since about one-sixth of the flights in the United States are delayed, we used two updates per minute for this test to represent the FOSC being used for preflight planning, with an update being generated each time a flight was delayed. Higher update transaction rates would occur if the flight object was being used for in-flight negotiations for more complex trajectory-based operations. (Testing with higher update rates is described in the following.)

The results in Fig. 10 show that the latency grows roughly linearly as the number of nodes increase, with good performance (6 s latency) at 100 nodes.

There are currently just under 200 ICAO member states. Since each node can support many end users, it is possible that 100 nodes (operated regionally by the more technologically advanced ICAO states or by regional service providers) could support the entire international aviation community. On the other hand, every ICAO state may wish to operate their own node, and air carriers and other entities may also want to operate nodes. This could result in a need to support many hundreds, or even thousands, of nodes worldwide. We stopped our testing at 100 blockchain nodes, simply due to the limitations of our test environment. Large-scale performance testing (for example, by leveraging the resources of a public cloud environment) would be needed to determine the upper limit, if one exists.

C. Performance as a Function of the Transaction Rate

Figure 11 shows latency measured as the flight object update rate was increased for a fixed number of nodes (50 nodes).

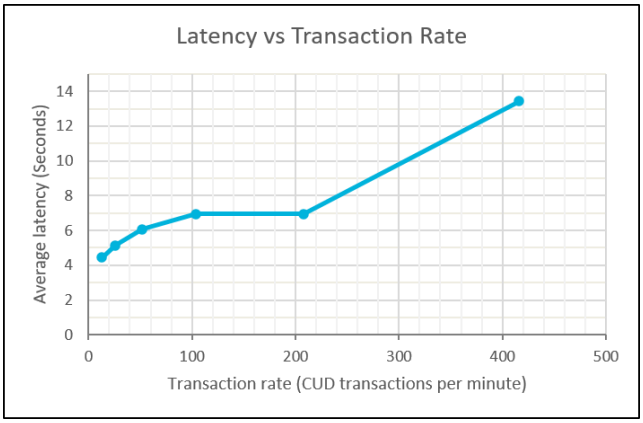

Fig.11 Latency vs. Transaction Rate

Latency increases smoothly as the create, update, and delete (CUD) transaction rate increases until a point somewhere above 400 transactions per minute. Above this point, latency increases dramatically, indicating that the software or network is becoming overloaded. This is likely due to limitations in our prototype implementation since other experiments with Tendermint [15] have demonstrated much higher transaction rates, with an upper limit around 10,000 transactions per second.

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

More work would be needed to identify and remove the bottleneck in our prototype. However, even 400 transactions per minute may be adequate to support international flight object sharing. If we estimate 1600 international flights airborne at a time, 400 transactions per minute would allow each flight object to be updated, on average, once every 4 min. Since updates would only occur when a trajectory needs to be renegotiated, this rate would appear to be more than adequate to support global TBO concepts.

D. Network Considerations

Our performance tests were run using virtualized networks in a local area network environment. We did not attempt to assess the impact of the available underlying network bandwidth and delay on the results. However, we did measure the bandwidth being consumed by each node. For the tests shown in Fig. 10, at 26 transactions/min, the total receive and transmit bandwidth consumed at each node ranged from approximately 205 kb/s for five nodes to approximately 532 kb/s for 100 nodes. For the tests shown in Fig. 11, with 50 nodes, the total receive and transmit bandwidth consumed at each node ranged from 205 kb/s at 13 transactions/min to 1228 kb/s at 416 transactions/min.

E. Performance Results Conclusion

These performance results were achieved with a rapid proof-of-concept prototype code that was written in JavaScript and running in Node.js, using JSON serialization for hashing and transactions, running on nodes with very moderate CPU and memory resources, and not profiled or optimized for performance. Significantly better performance could certainly be achieved by an optimized production implementation written in a compiled language, with a more efficient and compact serialization, and running on higher performance servers.

The performance measurement results allow us to begin to evaluate the ability of a blockchain-based solution to scale up to support real-world global aviation operations. Further study and more detailed operational concept and scenario development would be needed to determine exactly what scale and performance levels are required. Nevertheless, the results obtained with the rapid prototype provide a preliminary indication that suitable performance can be obtained at the scale needed for global aviation operations using a production quality implementation.

VI. Path to Real-World Operations

A. Model for Operations and Governance

Figure 12 provides one model for how a blockchain-based flight object solution might be deployed for operational use. This model envisions FOSC blockchain nodes being deployed and operated by stakeholders or service providers (which could be commercial entities) and used by various user application end systems. In the figure, ellipsoids depict FOSC blockchain nodes, and rectangles depict applications. The blockchain nodes and applications could be deployed on cloud computing environments or run on dedicated systems: any combination is possible. The only requirement is the components can communicate over an internet protocol network.

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

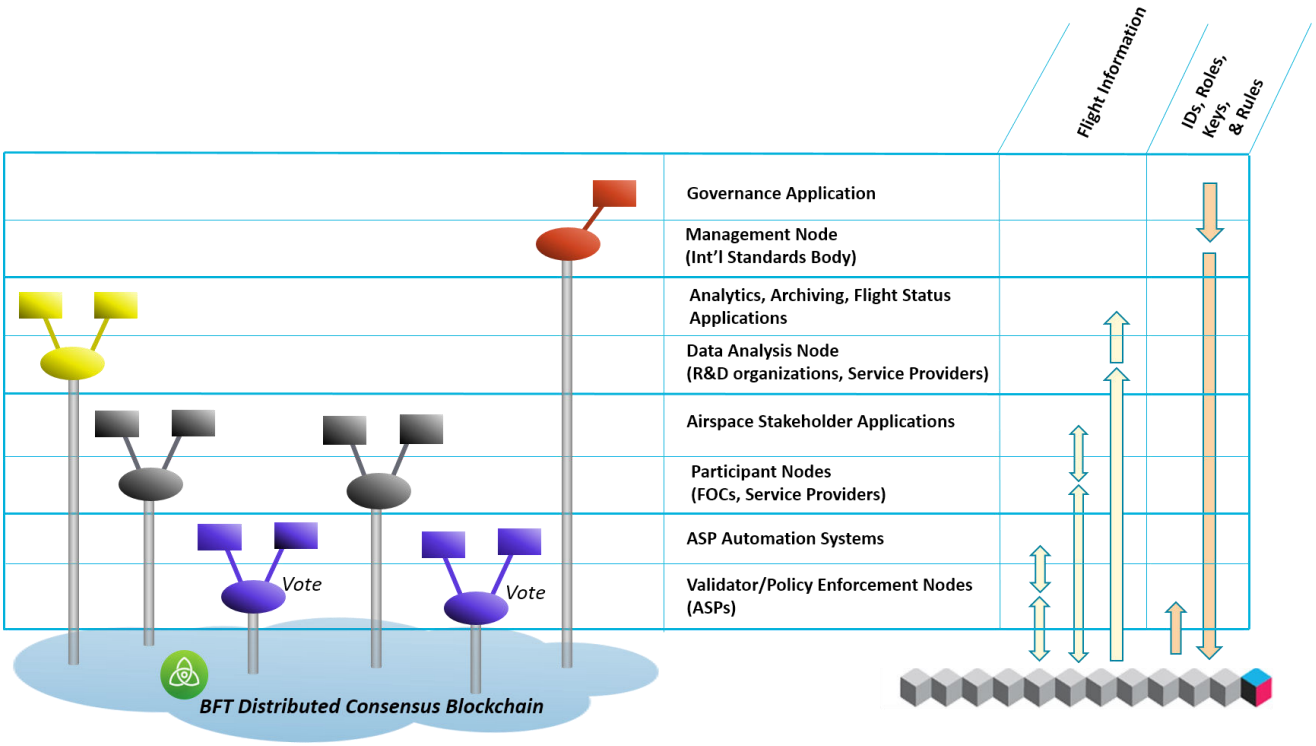

Fig. 12 One Possible Deployment Model

The deployment model depicted in Fig. 12 envisions different types of participants. At the bottom of the figure, in blue, are blockchain nodes controlled by the ASPs. These could be run by the ASPs themselves or by service providers acting on behalf of the ASPs. Since ASPs are recognized as authoritative entities in international aviation, it makes sense to designate the ASP-controlled nodes as the "validator nodes" that have voting power in the consensus algorithm. Restricting voting nodes to ASPs means that Tendermint's BFT consensus algorithm ensures that only transactions accepted as valid by a two-thirds majority of ASPs are used to update flight objects. ASP automation systems (the blue blocks) would connect to the ASP blockchain nodes to access the information in the flight objects, using the flight object client API. At the next level, in black, are the blockchain nodes serving active FOSC participants, which could include FOCs, general aviation users, military and law enforcement, and other airspace users. A sophisticated user like a large airline that is willing to invest in FOC automation could choose to operate their own blockchain node. Other users may choose to access the flight objects via blockchain nodes operated by service providers, perhaps in the form of cloud-based flight object services. These active flight object users would have known identities and roles that allow them to submit appropriate transactions to create, update, and delete flight objects, subject to the FOSM access control rules. At the next level, shown in yellow, are passive participants such as research and development (R&D) organizations and flight information service providers that do not need to create or update flight information but need to receive it for safety or performance analysis, or to provide functions such as

flight status and tracking. These passive nodes could be supported by "light" blockchain nodes.

In addition to different active and passive participants, the entire ecosystem needs one more type of element: a governance entity, depicted in red in Fig. 12. This entity's job would be to issue or authenticate "governance material," comprising end system and FOSC blockchain node identities (IDs), roles, keys, and rules. This would allow the validator nodes to ensure that the flight object is modified only in authorized ways by authorized users. To allow the flight object blockchain to grow and evolve over time, adding new users and even new information and FOSM processing, we propose that the governance material should be disseminated in the form of transactions submitted by a governance application to the blockchain. This also solves the problem of coordinating the change across all nodes. Tendermint will ensure all nodes receive these governance update transactions in the same order. The governance update results should also be included in the application state hash to guarantee that all nodes process the governance update in the same way.

B. Standardization

Stakeholders could create a FOSC with limited participants. For example, a single ASP and airspace users could create and operate a FOSC and use it to support collaborative planning among that ASP and its users. This might make sense as an initial trial to test the technology in real-world operations or as an operational solution for a single country or region. The long-term goal would be a single global solution to allow flight information to be available wherever it is needed. Standardization would prevent stakeholders from having to support multiple non-interoperable implementations in different regions.

The ICAO is aware of the limitations of existing flight infor-mation exchange methods (point-to-point text-based messages) and is promoting FF-ICE and FIXM as improvements. ICAO global SWIM concepts envision replacing legacy point-to-point message transmissions and supporting publish/subscribe message exchange patterns. These improvements will allow more flight information to be exchanged, but they do not fully realize the flight object concept and cannot guarantee that complete and consistent flight information is available to every concerned entity. ICAO may wish to consider a blockchain-based FOSC like the one demonstrated here, in addition to global SWIM, as a means of providing the information technology infrastructure to support future FF-ICE versions.

The MITRE team developed its FOSC prototype using Tendermint, an open source software package available under the Apache license, as well as other open source packages available under Apache or similar licenses. Subject to corporate approval, the MITRE FOSC prototype software could be made available under license or as open source as a basis for further concept exploration or as a starting point for a production service.

VII. Conclusions

The proof-of-concept prototype has shown that FF-ICE scenarios can be conducted by having each participant post updates to a shared flight object and receive notifications when the shared flight object changes, rather than by passing messages. The prototype's flight object structure is based on FIXM, and the demonstration's scenarios use FF-ICE concepts. Therefore, the prototype's success provides

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

confidence that a blockchain-based FOSC could be a viable basis for future FF-ICE versions. This would provide a more robust, secure, and general solution than the solutions currently being pursued. Note that, in the current design, the end user application is completely isolated from all the message-passing and consensus logic; it simply uses the flight object client to update the flight object and receive notifications when the flight object has changed. The separation of operational flight negotiation business logic from the flight information sharing mechanisms is another key advantage making a blockchain-based FOSC a good platform upon which to build future aviation applications.

M. J. KochenderferAssociate Editor

** CDM refers to collaborative decision making.

†† *eXtensible Access Control Markup Language (XACML)*, Version 3.0, OASIS Standard, 22 January 2013, **http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html**.

‡‡ Intervals between updates are randomized to result in the configured average rate.

§§ Transactions are time tagged when they are initiated, and clocks are synchronized in our test environment, allowing end-to-end latency to be computed upon receipt.

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

Acknowledgements

This work was sponsored by the Federal Aviation Administration. This work was produced for the U.S. Government under contract DTFAWA-10-C-00080 and is subject to Federal Aviation Administration Acquisition Management System clause 3.5-13, rights in data-general, Alternate III and Alternate IV (October 1996). The contents of this paper reflect the views of the authors and the MITRE Corporation and do not necessarily reflect the views of the Federal Aviation Administration (FAA) or the U.S. Department of Transportation (DOT). Neither the FAA nor the DOT makes any warranty or guarantee, expressed or implied, concerning the content or accuracy of these views. The authors would like to acknowledge our MITRE colleagues Stephane Mondoloni, Joyce Forman, Dan Greenbaum, and Patty Chavez for their contributions to the original flight object sharing capability concept. We would also like to acknowledge Valerie Gawron for her guidance and Matt Warnock for his editorial assistance.

Sponsor

This work was sponsored by the FAA.

References

1. Viets, K. J., and Taber, N. J., "*An Overview of a Flight Object Concept for the National Airspace System (NAS)*," MITRE Corp. TR00W0000085, McLean, VA, 2000, https://www.mitrecaasd.org/foc_archive/related/concept.pdf [retrieved 4 April 2020].
2. Hill, A., "The Flight Object-a New Flight Data Management Concept for Europe," *Digital Avionics Systems Conference*, Vol. 2, DASC-03, 2003, pp. 6.B.1–6.1-8. https://doi.org/10.1109/dasc.2003.1245879
3. "Global Air Traffic Management Operational Concept," International Civil Aviation Organization Doc. 9854, 1st ed., Montreal, 2005, https://www.icao.int/Meetings/anconf12/Document%20Archive/9854_cons_en%5B1%5D.pdf [retrieved 4 April 2020].
4. "Global Trajectory Based Operations (TBO) Concept," Ver. 0.11, International Civil Aviation Organization, Montreal, Aug. 2019, https://www.icao.int/airnavigation/tbo/PublishingImages/Pages/Why-Global-TBO-Concept/Global%20TBO%20Concept_V0.11.pdf [retrieved 4 April 2020].
5. "Manual on Flight and Flow—Information for a Collaborative Environment (FF-ICE)," International Civil Aviation Organization Doc. 9965, 1st ed., Montreal, 2012, https://www.icao.int/Meetings/anconf12/Documents/9965_cons_en.pdf [retrieved 4 April 2020].
6. "Manual on Air Traffic Management System Requirements," International Civil Aviation Organization Doc. 9882, 1st ed., Montreal, 2008, https://www.icao.int/airnavigation/IMP/Documents/Doc%209882%20-%20Manual%20on%20ATM%20Requirements.pdf [retrieved 4 April 2020].
7. "FIXM: Flight Information Exchange Model Primer," Ver. 4.2.0, FIXM Configuration Control Board, Feb. 2020, https://www.fixm.aero/releases/FIXM-

4.2.0/FIXM_Core_v4.2.0_Primer.pdf [retrieved 4 April 2020].

8. "Manual on System Wide Information Management (SWIM) Concept," International Civil Aviation Organization Doc. 10039, Interim Advance ed., Montreal, 2019, https://www.icao.int/airnavigation/IMP/Documents/SWIM%20Concept%20V2%20Draft%20with%20DISCLAIMER.pdf [retrieved 4 April 2020].
9. "Procedures forAir Navigation Services (PANS): Air Traffic Management," International Civil Aviation Organization Doc. 4444, 16th ed., Montreal, 2016, https://store.icao.int/products/procedures-for-air-navigation-servicesair-traffic-management-doc-4444 [retrieved 4 April 2020].
10. "FAA ICAO Flight Planning Interface Reference Guide," Ver. 3.0, U.S. Federal Aviation Administration, Washington, D.C., Nov. 2012, https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/air_traffic_services/flight_plan_filing/guidance/reference_guide/media/web%20Reference%20Guide%20v1.3.pdf [retrieved 4 April 2020].
11. Lamport, L., "The Part-Time Parliament," *ACM Transactions on Computer Systems*, Vol. 16, No. 2, May 1998, pp. 133–169. https://doi.org/10.1145/279227.279229
12. Lamport, L., Shostak, R., and Pease, M., "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 3, 1982, pp. 382–401. https://doi.org/10.1145/357172.357176
13. Castro, M., and Liskov, B., "Practical Byzantine Fault Tolerance," *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI 99)*,Association for Computing Machinery, New York, Feb. 1999, pp. 173–186, http://pmg.csail.mit.edu/papers/osdi99.pdf [retrieved 4 April 2020].
14. Nakamoto, S., "Bitcoin:APeer-to-Peer Electronic Cash System," 2009, https://bitcoin.org/bitcoin.pdf [retrieved 4 April 2020].
15. Buchman, E., "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," M.S. Thesis, School of Engineering, Univ. of Guelph, Guelph, ON, Canada, 2016, https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf [retrieved 4 April 2020].

Approved for Public Release. Public Release Case Number 19-2615. Distribution Unlimited.
©2020 The MITRE Corporation. All Rights Reserved.

NOTICE

This work was produced for the U.S. Government under Contract DTFAWA-10-C-00080 and is subject to Federal Aviation Administration Acquisition Management System Clause 3.5-13, Rights In Data-General, Alt. III and Alt. IV (Oct. 1996).

The contents of this document reflect the views of the author and The MITRE Corporation and do not necessarily reflect the views of the Federal Aviation Administration (FAA) or the Department of Transportation (DOT). Neither the FAA nor the DOT makes any warranty or guarantee, expressed or implied, concerning the content or accuracy of these views.

© 2020 The MITRE Corporation. All Rights Reserved.