A DDS-Based Middleware for Cooperation of Air Traffic Service Units

Erwin Mayer and Johannes Fröhlich

Offenburg University of Applied Sciences,
Faculty of Electrical Engineering & Information Technology, Offenburg, Germany
{erwin.mayer,johannes.froehlich}@hs-offenburg.de

Abstract. Air traffic is by nature crossing borders and organizations. The supporting infrastructure represents a federative distributed system of independent Air Traffic Service Units, typically each with its own proprietary system architecture. Interaction between the centers is taking place over dedicated protocols, often organized as a mesh of 1:1 bilateral data exchanges.

This contribution gives an overview of the ongoing efforts to standardize this data exchange. At the core is a data-centric view, using a shared virtual Flight Object as the IT counterpart of a real flight. It permits a uniform way to access and update a flight's static and dynamic attributes. A middleware is presented that implements this abstraction and maps it onto a physical level, employing DDS (Data Distribution Service) technology for the 1:N dissemination of flight data.

Keywords: Air Traffic Control, Data Distribution Service (DDS), distributed systems, data replication, middleware, standardization.

1 Introduction

An aircraft, between departure and landing, typically crosses a large number of different areas of responsibility, each implemented by an individual Air Traffic Service Unit (ATSU) such as airports, approach control centers or en-route centers. Each of these ATSUs is based on a complex proprietary technical infrastructure, maintaining the counterpart of the physical aircraft flight in the form of a proprietary IT data record that is being updated along the progress of the flight [1].

Once an aircraft leaves the air space of a given center, a transfer to the neighboring ATSU is initiated, often accompanied by a telephone transfer of the respective air traffic controllers or by the use of 1:1 coordination protocols like OLDI (Online Date Interchange) [2]. In the neighboring center, upon notification of the prospective arrival of an aircraft, another IT data record is constructed and maintained, comprising similar attributes for the given flight, however again in a proprietary representation and often with an equipment vendor-specific set of services to interact with this representation.

Though this federative approach of ATSU organization has in the past demonstrated its effectiveness in terms of safe operation around the world, it is not ideal.

A. Sikora et al. (Eds.): Nets4Cars/Nets4Trains/Nets4Aircraft 2014, LNCS 8435, pp. 94–102, 2014.

First, because there are multiple representations of the same physical object, there may be varying views of a flight in terms of its detailed attributes and the time that they are updated [1]. For example, while a neighboring ATSU may be informed about the exact arrival time and transfer altitude at the time of transfer, a ATSU further downstream, e.g. the arrival airport, may not be immediately aware of this data and possibly cannot anticipate implicit delays.

Second, a large number of 1:1 interactions between ATSUs may need to take place, in order to communicate and agree changes in a flight plan, like the re-routing of a flight due to conditions at the arrival airport.

The air traffic control community is for many years aware of this situation [1][3][4]. As part of large research programs like SESAR (Single European Sky ATM Research Programme) [3], the harmonization and standardization of air traffic control technology and procedures is ongoing.

This contribution describes standardization activities in the domain of center cooperation and presents a prototype middleware implementation, DDS-ATC (Data Distribution System for Air Traffic Control) taking into account results of this standardization. DDS-ATC provides a data-centric approach to the problem: Instead of the use of multiple 1:1 interactions between ATSUs operating on proprietary IT data records, it employs a single representation of a flight in the form of a shared virtual Flight Object (VFO). It is virtual because there exists no single physical storage location for it. It is shared, because all ATSUs, secured by access rights, can equally access it for reading and modification over a set of standardized services.

Chapter 2 gives an overview of the standardization efforts in this domain. Chapter 3 introduces the middleware's architecture and functional scope, while chapter 4 gives some details about the project environment, followed by a summary (chapter 5).

2 Ongoing Standardization Efforts

The main goal of ongoing standardization is to provide a common interface for all air traffic service participants and a data format in which the flight information is unambiguously described and exchanged.

Initial ideas in this area were discussed in [1]. Based on this and other general developments (e.g. [3]), during recent years several standard initiatives, driven by U.S. and European aviation authorities, have been in place. These include the Flight Object Interoperability Proposed Standard (FOIPS) [5], the Flight Object Interoperability Specification (ED-133) [5] which is itself based on FOIPS, and the Flight Information Exchange Model (FIXM) [6]. All three proposals are still under development with a large number of still open issues.

- The FOIPS Standard [4] prepared by EUROCAE (European Organisation for Civil Aviation Equipment) and introduced in 2005 is one of the first standardization efforts in this domain. It includes both a data model defining flight objects and additional rules for exchanging this flight object data. The "analysis model" uses UML to define the data structure and states of flight objects. The "usability model" describes textually how the service participants should interact under various roles. A

Flight Object Server (FOS) is the system instance used for providing access to the network and exchanging the flight objects. FOIPS does only take care of the service interface between the FOS and the application layer and not between the FOSes themselves. Therefore there are no architectural limitations for further specifications.

- Based on FOIPS another standard initiative, the ED-133 [5], was raised by EUROCAE in 2009. The ED-133 replaces the FOIPS specification and delivers a more comprehensive requirement analysis and specification for the exchange of flight data. Unlike the FOIPS specification, the ED-133 covers only civilian aviation and focuses on the FOS interface to ensure interoperability for implementations of different vendors. On communication level DDS [7] is suggested for distributing the Flight Object data among the participants. Web services enable a parallel request/response-based communication scheme between single nodes. XML is proposed as encoding for the data payload.
- On the other side of the Atlantic, the FAA went a slightly different approach introducing the FIXM [6] model in 2012. FIXM focuses on the flight object data format and is by intent not defining a protocol. Based on the ISO19100 guidelines FIXM enables compatibility to existing EUROCONTROL standards like the Aeronautical Information Exchange Model (AIXM) and the Weather Information Exchange Model (WXXM). FIXM comprises a "Foundations Package" comprising basic data types, a "Core Package" defining flight attributes, and a "Message Package", for meta-data how to encapsulate Core Data for exchange (however no corresponding protocol). All data is foreseen to be XML-encoded. Another useful feature of FIXM is the core/extension structure which allows a dynamic adaption of the data model.

Currently it is not clear in which direction the standardization is further going. Therefore it is important to gather experience through implementations like DDS-ATC, which prototype this type of flight data exchange.

3 DDS-ATC Middleware

3.1 Architectural Model

The initial goal of the DDS-ATC prototype development was to provide a universal ATSU exchange service based on the concept of a virtual flight object on the basis of DDS transport. As part of the DDS-ATC requirement analysis the available standards described in chapter 2 were evaluated and compared. It was decided to take an approach that would not block going for either of the proposals. Figure 1 describes the architectural model for the DDS-ATC, which is in line with the FIXM [6], FOIPS [4] and ED133 [5] proposals.

The chosen architecture is such that it can support both the FIXM and the FOIPS/ED133 type of flight data. On the protocol level it takes onboard suggestions from the ED133 [5] proposal.

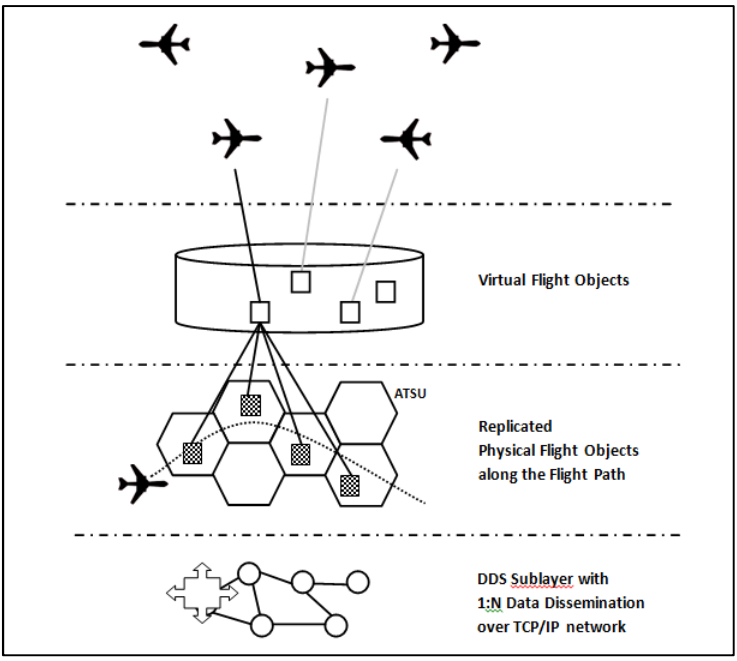

Fig. 1. Architectural Model

On top of the hierarchy in Fig. 1 are the physical aircraft. For each aircraft there exists a corresponding virtual flight object (VFO), composed of a large number of data attributes describing the details of the flight. Updates are triggered by air traffic controllers or automatic processes in the Flight Data Management Processes (FDMP) of the ATSUs. The VFO is created ahead of departure and has a lifetime at least until the aircraft has landed or departed the supervision area.

While it would be ideal to physically implement the VFO on a central server, this forbids itself, mainly for safety reasons. Instead, a replicated approach is used, where each ATSU interested in a given flight (these are at least those centers that control the airspace along the flight path) store a copy of the VFO in form of a PFO (Physical Flight Object). In order to keep the PFOs consistent, a replication control protocol is required (see below).

The update of a VFO must include the update of all related PFOs (i.e. physical replica). This is effected by using OMG's DDS (Data Distribution Service) [7], as suggested in [5], over a well secured TCP/IP provider network.

3.2 Virtual Flight Objects

A VFO is composed of a number of static and dynamic attributes. Example of static attributes may be the flight identification, departure airport or available onboard equipment. Dynamic attributes include the flight's SSR code (dynamic radar-based identification), its position, trajectory, flight path or arrival time. A VFO's attributes

are specified and stored in a platform independent, standardized manner as discussed in chapter 2.

While FOIPS [4] and FIXM [6] rely on the use of a trajectory (i.e. a tracker-derived sequence of recent and future positions that a flight passes), ED133[5] proposes the additional use of a shared flight script, which comprises all control instructions (e.g. a climb command) and constraints that determine the current and future path of a flight. By additionally sharing this information each ATSU can compute its own trajectory of the flight and is not dependent on the validity and accuracy of the trajectory computation of a previous center [5].

It may be the case that an individual ATSU may not be capable to decide upon a change to a VFO by itself. For example, if a re-routing of an aircraft is to take place, the centers along the new path have to agree. While already today there exist procedures and technical infrastructure to cope with this, the concept of VFOs lends itself to support this requirement. As proposed by [5], negotiation shall be supported on VFO level.

Negotiation is implemented by means of so-called What-If Flight Objects (WIFOs). WIFOs have the same quality as normal VFOs, i.e. are stored in a replicated way on all affected ATSUs. However, they exist in one or multiple forms in parallel to the original VFO and include the effects of proposed changes (e.g. a new route).

The benefit of this approach is the immediate availability of the complete attribute set of a new air traffic situation at all affected centers. Based on WIFOs, controllers can decide, whether they agree with the change, make counterproposals (in form of updates to the WIFOs) or may reject a proposal (deletion of WIFO), leaving the original VFO unchanged. In case of acceptance by all related ATSUs the WFO is transformed to a VFO.

3.3 Replication Approach

While [6] only provides a data model without a concrete protocol, [5] proposes a replication scheme on top of an underlying transport. This chapter describes the role of replication and its implementation in the DDS-ATC middleware.

While the use of replication in distributed systems has a long history and is well understood (e.g. summary [10]), an adaptation to the specific air traffic control scenario is required.

As sketched in Fig. 1, the physical flight objects implementing a VFO are not stored at all participating ATSUs, but only on a subset of "interested" nodes. Typically this includes the centers along the flight path of an aircraft, and possibly some individual neighboring centers. Thus replication is partial and dynamic.

Fig. 2 rehashes the principle problem of updates in such an environment. Due to delays of the underlying network inconsistent views of the stored data and optionally an inconsistent state of the replica may result. In the example, two ATSUs controlling adjacent airspace of a specific aircraft update the flight's VFO by sending 1:N messages to all nodes hosting a PFO. Due to differing network latency, the updates arrive in a different order (A→B, B→A) at some of the other ATSUs, resulting in potentially different end states.

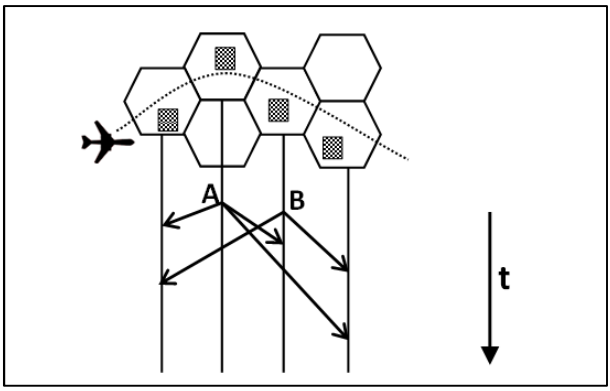

Fig. 2. Inconsistent Update Order as part of 1:N Distribution

In the case of node failures and recovery, further inconsistencies may result: For example, a recovering node may receive a mixture of (replayed) recovery messages from neighboring nodes and live messages for an active flight. Again a proper order of updates has to be ensured. Finally, a recovering node must be prevented itself from contributing to other nodes, while its state is not yet complete.

At the core of the replica update issue lies the uncontrollable latency of the network and the fact that updates can be triggered independently by different ATSUs. The approach taken in DDS-ATC for solving this problem is to use a variant of a Primary Copy Replication Control (PCRC) algorithm. The approach follows the ED133 [5] requirement analysis and specification.

General PCRC algorithms are well-understood (see e.g. [10]). In their core, they assume that one of the replicas assumes a distinguished role: Updaters always direct their request to the primary copy, which then synchronizes the updates to the other (secondary) copies. For the given context the standard PCRC is adapted in the following ways:

1. Instead of having a single node in the network acting as primary copy holder (which would be not acceptable from the safety point of view), the granularity of the primary-copy-ship is chosen to be a single flight. At a given point in time, for two distinct flights typically two differing ATSUs host the respective primary copy. This caters for a strong independence and resilience against failures.

2. The primary-copy-ship is not static but transferred between ATSUs dynamically: Each time the operational responsibility for a flight changes (e.g. an air traffic controller transfers a flight to the neighboring sector), implicitly on the level of the DDS-ATC middleware a transfer of primary-copy-ship is effected. The middleware receives notification of such a transfer over its API from the FDMP, and triggers a token transfer message, that is broadcast (1:N) to interested ATSUs. From now on the new token holder coordinates the update.

As long as the underlying 1:N communication protocol guarantees the ordered delivery of messages from the same sender to all of the receivers (see next chapter), a single primary copy holder can guarantee consistent updates. However the transfer of the primary-copy-ship itself is critical: If an update from Node A, immediately before the token transfer interleaves with another update from Node B after the transfer, situations like in Fig. 2 may again arise.

The selected solution to maintain consistency for these cases is version numbering for the PFOs. Each update comprises the previous version number of a PFO, upon which the update is based. After arrival of the update request at a node, it will only be implemented if the version number is consecutive. If there is a gap, the receiver must wait for the missing update, which may be delayed because of a longer network route. (A timeout mechanism has to ensure that lost updates do not produce deadlocks.)

The critical case of transfer of primary-copy-ship can also be secured by this mechanism: the current version number of PFO is itself conveyed as part of the token transfer message. Because the new primary holder continues to use this value, the synchronization at the reception side is guaranteed.

3.4 DDS Transport

In order to physically distribute PFO updates around the network an underlying transport protocol is required. OMG's DDS (Data Distribution Service) middleware [7] is used for this purpose. It offers a natural data-centric approach, permitting data types of the exchanged application objects to be defined as part of an IDL (Interface Definition Language). This follows the approach taken in CORBA (Common Object Request Broker Architecture) [9] and caters for flexibility and extensibility requirements of the given application. Flight objects are mapped onto DDS topics, which are maintained using a general Publish/Subscribe paradigm [11].

The following are rationales for the use of DDS as part of the project's middleware:

1. DDS implements a 1:N type of data transport and makes use of underlying Layer 3 multicast functionality, if available. This caters for a high efficiency, required in particular to cater for physical data replication.
2. DDS offers fine-grained QoS (Quality of Services) attributes for its services. Both reliable and best effort data transfer are possible, and are employed for different types of user data. Deadline monitoring (both towards the network and towards the service consumer) is used for peer- and self-supervision.
3. The DDS provider system can be configured to work in an entirely peer-to-peer manner, i.e. without central storage or broker, mandatory for the given environment, with its independent ATSUs.
4. Further benefit of using DDS as a lower layer relate to its realtime performance characteristics: The DDS stack is optimized for realtime transfer of data with a low computational overhead and a lean message exchange service. [14]

Next to the standardization of the API (application programming interface) [7], DDS also provides a "wire protocol", i.e. a unique specification of how data is exchanged, such that different nodes can use different DDS supplier stacks [8].

There is a fairly large amount of user experience in mission critical software available for DDS ([12],[13]), promising successful application to the DDS-ATC middleware.

4 Project Environment

The DDS-ATC middleware project is funded by a federal grant from the German ministry of commerce (BMWI) and is executed together with an industry partner from the air traffic control technology sector.

Target operating system for the system is LINUX (CentOS). Development environment is Eclipse using C++ as programming language. PostgreSQL is used as the database management system for storing the PFOs.

On the networking side four DDS candidate implementations have been evaluated as part of separate work ([12]): OpenDDS, RTI DDS, OpenSplice and CoreDX. Based on its combination of provided functionality, standard compatibility and licensing conditions, the community version of OpenSplice [13] was selected as the platform for the DDS-ATC middleware. It offers core functionality needed for VFO implementation on an open source basis.

5 Conclusion and Future Work

This paper presented a middleware approach for the cooperation of a regional federation of air traffic control centers taking into account newer standards of the air traffic control community ([4],[5],[6]). It could be shown that the abstraction of one Virtual Flight Object per aircraft, implemented on top of a replication-based distributed physical storage, caters for both: a simple, uniform access to the data by all players, as well as an efficient standardized exchange of data over a wide area network. For the latter, it could be shown, that the DDS transport vehicle can be tailored in a favorable way to the specific project requirements.

The DDS-ATC project is still ongoing. A base system is operational and shows good initial results concerning performance and stability. As future work of the project a simulation environment for DDS-ATC will be developed that in particular allows to carry out performance tests and to inject errors on networking or application level for system testing of the final application.

References

1. Hill, A.: The Flying Object: A Flight Data Management Concept. In: IEEE A&E Systems Magazine, pp. 11–18 (2004)
2. Eurocontrol: On-Line Data Interchange (OLDI), EUROCONTROL-SPEC-106, Ed. 4.2 (2010), ISBN 978-2-87497-061-0, http://www.eurocontrol.int/publications
3. Sesar (Single European Sky ATM Research) Programme, Joint Undertaking Site (2013), http://www.sesarju.eu

4. EUROCONTROL: Flight Object Interoperability Proposed Standard (FOIPS), Study D7 Revision 1.04 (2006), http://www.eurocontrol.int/services/foips
5. EUROCAE, EuropeanOrganisation for Civil Aviation Equipment: Flight Object Interoperability Specification, Standard ED-133 (2009), http://www.eurocae.net
6. Federal Aviation Authority (FAA): Flight Information Exchange Model (FIXM), Rev. 2.0 (2013), http://www.fixm.aero/content/fixm-core-releases
7. Open Management Group: Data Distribution Service for Real-time Systems, Standards specification document V 1.2 (2007), http://www.omg.org/spec/DDS/1.2/
8. Open Management Group: The Real-time Publish-Subscribe Wire Protocol, DDS Interoperability Wire Protocol. DDS-RTPS standard specification V2.1 (2010), http://www.omg.org/spec/DDS-RTPS/2.1/
9. Open Management Group: CORBA (Common Object Request Broker) Architecture (2012), http://www.omg.org/spec/CORBA
10. Tanenbaum, A., Van Steen, M.: Distributed Systems, Principles and Paradigms, pp. 273–320. Prentice Hall International (2007)
11. Eugster, et al.: The Many Faces of Publish/Subscribe. ACM Computing Surveys 35(2), 114–131 (2003)
12. Mayer, V.: Evaluation of the DDS Publish/Subscribe Standard and application to an Air Traffic Control Scenario. Master Thesis, Hochschule Offenburg (2013) (in German)
13. Schmidt, D.C., van't Hag, H.: Addressing the Challenges of Mission-Critical Information Management in Next-Generation Net-Centric Pub/Sub Systems with OpenSplice DDS. In: Proceedings of the 2008 IEEE International Parallel & Distributed Processing Symposium, IPDSP 2008, Miami, pp. 1–8 (2008)
14. Bliesner, S., Comitz, P., Sweet, D.: Enhancing the Distribution of Radar Surveillance Data. In: Integrated Communications, Navigation and Surveillance Conference, ICNS 2008, pp. 1–8 (2008)