SECTION 1.2

Refinement for Fault-Tolerance: An Aircraft Hand-off Protocol

Keith Marzullo\(^1\), Fred B. Schneider\(^2\), and Jon Dehn\(^3\)

Abstract

Part of the Advanced Automation System (AAS) for air-traffic control is a protocol to permit flight hand-off from one air-traffic controller to another. The protocol must be fault-tolerant and, therefore, is subtle—an ideal candidate for the application of formal methods. This paper describes a formal method for deriving fault-tolerant protocols that is based on refinement and proof outlines. The AAS hand-off protocol was actually derived using this method; that derivation is given.

1.2.1 Introduction

The next-generation air traffic control system for the United States is currently being built under contract to the U.S. government by the IBM Federal Systems Company (recently acquired by Loral Corp.). Advanced Automation System (AAS) [1] is a large distributed system that must function correctly, even if hardware components fail.

Design errors in AAS software are avoided and eliminated by a host of methods. This paper discusses one of them—the formal derivation of a protocol from its specification—and how it was applied in the AAS protocol for transferring authority to control a flight from one air-traffic controller to another. The flight hand-off protocol we describe is the one actually used in the production AAS system (although the protocol there is programmed in Ada). And, the derivation we give is a description of how the protocol actually was first obtained.

The formal methods we use are not particularly esoteric nor sophisticated. The specification of the problem is simple, as is the characterization of hardware failures that it must tolerate. Because the hand-off protocol is short, computer-aided support was not necessary for the derivation. Deriving more complex protocols would certainly benefit from access to a theorem prover.

\(^1\)Department of Computer Science, University of California San Diego, La Jolla, CA 92093. This author is supported in part by the Defense Advanced Research Projects Agency under NASA Ames grant number NAG 2-593, Contract N00140-87-C-8904 and by AFOSR grant number F496209310242. The views, opinions, and findings contained in this report are those of the author and should not be construed as an official Department of Defense position, policy, or decision.

\(^2\)Department of Computer Science, Cornell University, Ithaca, NY 14853. This author is supported in part by the Office of Naval Research under contract N00014-91-J-1219, AFOSR under proposal 93NM312, the National Science Foundation under Grant CCR-8701103, and DARPA/NSF Grant CCR-9014363. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not reflect the views of these agencies.

\(^3\)Loral Federal Systems, 9231 Corporate Blvd., Rockville, MD 20850.
We proceed as follows. The next section gives a specification of the problem and the assumptions being made about the system. Section 1.2.3 describes the formal method we used. Finally, Section 1.2.4 contains our derivation of the hand-off protocol.

1.2.2 Specification and System Model

The air-traffic controller in charge of a flight at any time is determined by the location of the flight at that time. However, the hand-off of the flight from one controller to another is not automatic: some controller must issue a command requesting that the ownership of a flight be transferred from its current owner to a new controller. This message is sent to a process that is executing on behalf of the new controller. It is this process that starts the execution of the hand-off itself.

The hand-off protocol has the following requirements:

\[ P1: \] No two controllers own the same flight at the same time.

\[ P2: \] The interval during which no controller owns a flight is brief (approximately one second).

\[ P3: \] A controller that does not own a flight knows which controller does own that flight.

The hand-off protocol is implemented on top of AAS system software that implements several strong properties about message delivery and execution time [1]. For our purposes, we simplify the system model somewhat and mention only those properties needed by our hand-off protocol.

The system is structured as a set of processes running on a collection of processors interconnected with redundant networks. The services provided by AAS system software include a point-to-point FIFO interprocess communication facility and a name service that allows for location-independent interprocess communication. AAS also supports the notion of a resilient process \( s \) comprising a primary process \( s.p \) and a backup process \( s.b \). The primary sends messages to the backup so that the backup’s state stays consistent with the primary. This allows the backup to take over if the primary fails.

A resilient process is used to implement the services needed by an air-traffic controller, including screen management, display of radar information, and processing of flight information. We denote the primary process for a controller \( C \) as \( C.p \) and its backup process as \( C.b \). If \( C \) is the owner of a flight \( f \), then \( C.p \) can execute commands and send messages that affect the status of flight \( f \); \( C.b \), like all backup processes in AAS, only receives and records information from \( C.p \) in order to take over if \( C.p \) fails.

AAS implements a simple failure model for processes [3]:

\[ S1: \] Processes can fail by crashing. A crashed process simply stops executing without otherwise taking any erroneous action.