ICIUS-2010-0129

Experimental Analysis of Primary-Shadow Replication Scheme for Fault-Tolerant Operational Flight Program of Small UAV

Junyeong Kim†, Nodir Kodirov†, Doo-Hyun Kim‡, Chun-Hyon Chang*, Changjoo Kim , Yonghyun Kim

†School of Computer Science and Engineering
Konkuk University
Seoul, Republic of Korea
e-mail: {plaa, knodir}@konkuk.ac.kr

‡School of Internet Multimedia Engineering
Konkuk University
Seoul, Republic of Korea
e-mail: doohyun@konkuk.ac.kr (corresponding author)

*School of Computer Science and Engineering
Konkuk University
Seoul, Republic of Korea
e-mail: chchang@konkuk.ac.kr

School of Aerospace Engineering
Konkuk University
Seoul, Republic of Korea
e-mail: cjkim@konkuk.ac.kr

Department of Sensor Networks
Agency for Defense Development
Seoul, Republic of Korea
e-mail: yonghyun@add.re.kr

Abstract

The paper treats the Primary-Shadow TMO [1] Replication (PSTR) [2][3][4][5] scheme to develop a fault-tolerance Operational Flight Program (OFP) for the Unmanned Aerial Vehicle (UAV). The recent increase in UAV applications to various autonomous missions demands a highly reliable and extremely safe OFP to cope with unexpected system faults. This paper proposes the application of Primary-Shadow TMO Replication (PSTR) [2][3][4][5] mechanism for quick detection and rectification of system failures with minimum intervention of human pilots. For this purpose the PSTR method is integrated into the Hardware-In-the-Loop Simulation (HILS) environment with a UAV model. Various failure modes including receive UAV's sensor data, send calculated data to UAV's actuator and deadline violation (operational deadline miss problems and various faults such as sensor data interference or lost) are simulated and tested to show the enhanced fault-tolerance nature in the OFP. The test results show that 96% of our injected faults were successfully detected and recovered, and shadow OFP activation with a given deadline time success rate was 94%.

1 Introduction

There is an increasing demand for small scale UAVs, such as helicopter and airplane in the both civil and military domains. Unlike other mainstream embedded applications running on top of the embedded operation system, UAV OFP requires high reliability to confront timely deadline misses and various faults. Current industrial domain has two solutions overcome these issues; they are Boeing and Airbus fault tolerance mechanisms [7][9]. Airbus uses a kind of Replication Check [6] Mechanism (Fig. 1) to provide fault tolerance and Boeing's solution is a kind of Triple Modular Redundancy (TMR) [6], as it is illustrated in the Fig. 2.

The fault tolerance mechanism used by Airbus [7] checks the failure by threshold comparison of calculated data. The results are compared in the two systems, which are control and monitoring systems. The difference between their

ISBN 978-979-16955-1-0

values is compared against a given threshold. A failure is detected if the difference between the systems is above an allowable threshold. Similarly, fault tolerance mechanism used by Boeing [9], the computer system is triplicated (Fig. 2). Each computer operates in two roles: command role or monitor role. Only one computer is allowed to be in command role. The other computers are allowed to be in monitoring role. Like Airbus fault tolerance mechanism, Boeing is also monitoring the command system's result by checking them.

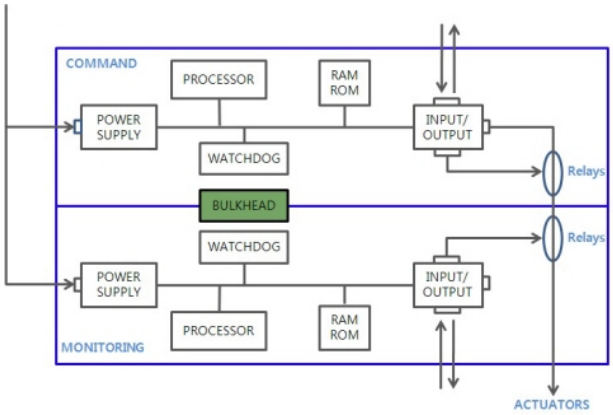

Figure 1: Airbus basic computer global architecture

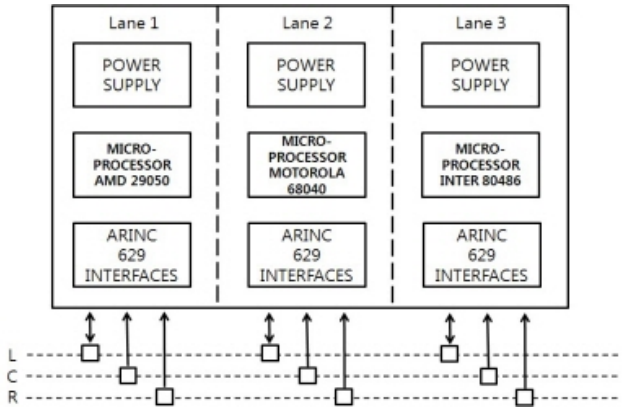

Figure 2: Boeing basic computer global architecture

However, both solutions have the same problem at detecting fault. Their fault tolerance mechanism have high possibility to miss a critical point for accurately detect the fault in the time of occurrence. The reason of this weakness is at their fault detection mechanism, which relies on comparing calculated data in each system. We provide a solution to quickly detect a fault and recover, via PSTR [2][3][4][5] motivated mechanism.

And we define the likelihood of failure (fault) and analyze the failure (fault) possibility places (failure mode) on Operational Flight Program using Failure Mode and Effects Analysis (FMEA) [10][11][12]. We assume the likelihood of failure in input, calculation and output process of OFP. Based on these we define following failure modes. They are sensor data interference, failed to receive the sensor data, failed to send the control data and failed to pass the acceptance test. In this paper, we discuss our fault-tolerance mechanism about our assumed failure (fault) mode.

2 Operational Flight Program (OFP) Mechanism and Periodic thread Assignment

Our OFP is a piece of software written in C, used as a flight control program of the small scale UAV. The OFP's architecture illustrated in the Fig. 3 and below we will give description to its characteristics.

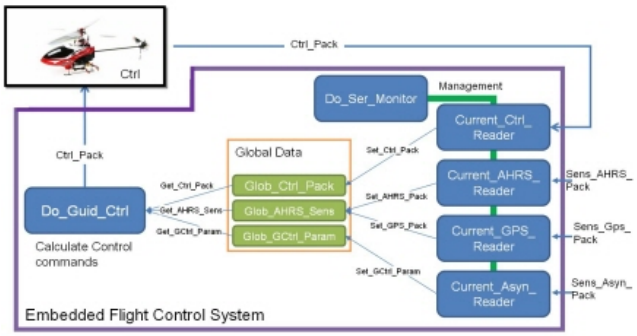

Figure 3: OFP thread interaction

OFP consists of the six threads for reading and writing data and control logic. The first which interacts with Helicopter is Control logic thread, marked as the Do_Guid_Cntrl (Guidance and Control thread) in the Figure 3. It is OFP's core thread running on 50Hz frequency. Its task is to calculate control signal based on input sensor data and control algorithm, and send them to the UAV's Ctrl (Control) module. The second one is Do_Ser_Monitor (Serial monitor thread) for polling asynchronous I/O ports. The other four threads are to read sensor data and store it into the global storage. These four reader threads manage Control Signal, Navigation, GPS, and Command message respectively. The Current_Ctrl_Reader (Control reader thread) thread receives control signal which is fed back from OFP. The Current_AHRS_Reader (Attitude Heading Reference System reader) thread's job is to receive AHRS sensor data sent on 100Hz frequency. Received data includes current angle (roll, pitch, yaw), three-axis acceleration and angular velocity. Reading data from Sens_AHRS_Pack, Current_AHRS_Reader analyzes it and stores in Global data (Glob_AHRS_Sens). The Current_ GPS_Reader (GPS reader) thread's job is to receive GPS sensor data sent on 10Hz frequency and save it into Global data (Glob_AHRS_Sens). Current_GPS_Reader received data includes location values, and velocity of three-axis (longitudinal, latitudinal and altitudinal). Finally, Current_ Asyn_Reader (Asynchronous Data Reader thread) receives command data including waypoint, operational mode and UAV information, which are sent from Ground Control System (GCS). All received data by Current_Asyn_Reader are saved into the Global storage. Content of the global data storage will be used to calculate control signal for proper navigation and stability of the UAV.

ISBN 978-979-16955-1-0

3 Time-driven Fault-Tolerance Mechanism

In order to provide fault tolerance mechanism suitable with OFP characteristics, we designed fault tolerance mechanism being motivated from PSTR mechanism [2][3][4][5]. It is used in the periodical execution based OFP. The main concept of Time-driven Fault-Tolerance mechanism is illustrated in the Fig 4. Here, shadow system runs together with a primary system and analyzes its output during each execution period; thus monitoring primary OFP's behavior.

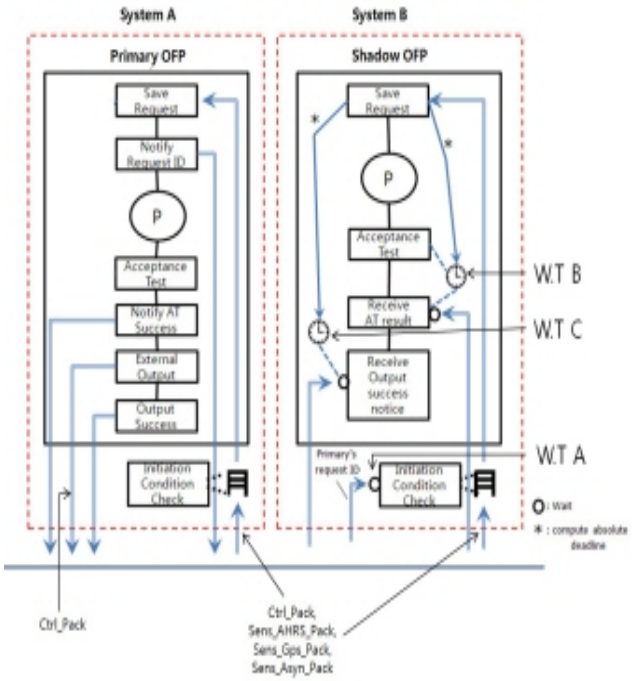

Figure 4: Time-driven Fault-Tolerance Mechanism Arcitecture

Below we explain more specific details of each step for our Fault-Tolerance mechanism, at the same time comparing descriptions at Fig. 3 and Fig. 4. In both figures input data is the same, indicated as a Ctrl_Pack, Sens_AHRS_Pack, Sens_Gps_Pack and Sens_Asyn_Pack respectively. "Save Request" state means to store an input data into the Global data storage. In the Primary OFP, Request ID means the unique ID of each input data. For example, Ctrl_Pack's request ID number could be 1, Sens_AHRS_Pack's 2, Sens_Gps_Pack's 3, and Sens_Asyn_Pack request ID number could have a value equal to 4. Once all data is saved to the Global data storage, notification message will be sent to the Shadow OFP during "Notify Request ID". Step "P" in the Fig. 4 is equal to the Do_Guid_Ctrl step in the Fig. 3, while in this step control signal of the OFP is calculated. After the step "P", "Acceptance Test (AT)" will be executed. It is a self-checking test to verify whether estimated fixed time-deadline of the "P" step has passed or not. Once we keep estimated deadline, we reach "Notify AT Success" step. In this step we will send the AT test success message to the Shadow OFP. The "External Output" step is to send the Ctrl_Pack to the Ctrl, which is a small scale helicopter. Lastly, "Output Success" is to send Output Success message to the Shadow OFP, informing about successful completion of all computation by the Primary OFP. In Shadow OFP, Wait-for-response Time (W.T) means, if there is no acknowledgement notification message received from Primary OFP, Shadow will wait only for W.T before replacing Primary. Below we will provide execution procedure of our technique.

As it can be seen from the Fig. 4, initially system A hosts primary OFP and system B shadow OFP. Both of them receive the same data from UAV sensor modules and store it into their buffer. Once input data arrives, both of the system will execute Initiation Condition Check (ICC) mechanism to check whether input data is available to be consumed or not. If ICC step is successfully passed, both of the systems store their data and Primary OFP sends Request ID notification to the shadow OFP. In the next step, both of Primary and Shadow OFP concurrently processes saved data independently, using the same Acceptance Test (AT) routine. Once Primary OFP successfully passes AT routine, it sends AT success message to the Shadow OFP. Lastly, Primary OFP sends its processed output to the UAV actuator modules and send Output Success notice to the Shadow OFP. Thus, confirming successful handle of the request by Primary OFP. Our fault tolerance mechanism's execution process is briefly illustrated in Fig. 5 and Fig. 6.

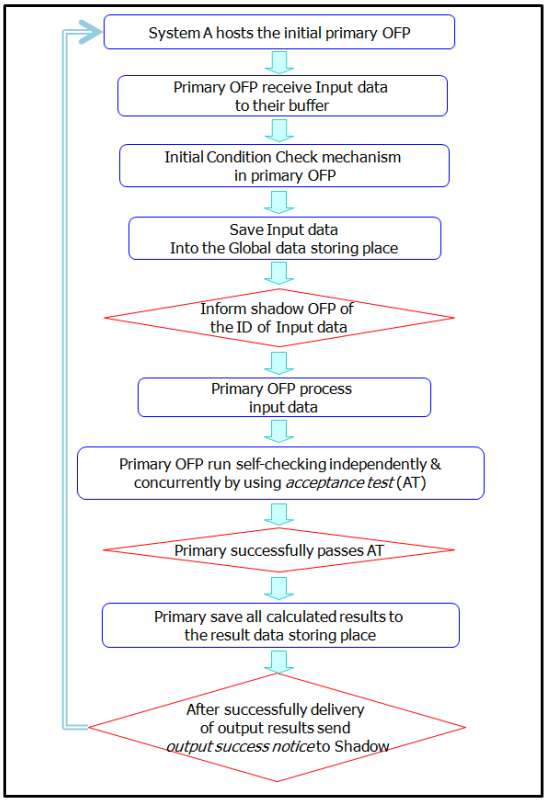

Figure 5: Primary OFP Execution Process Flow

ISBN 978-979-16955-1-0

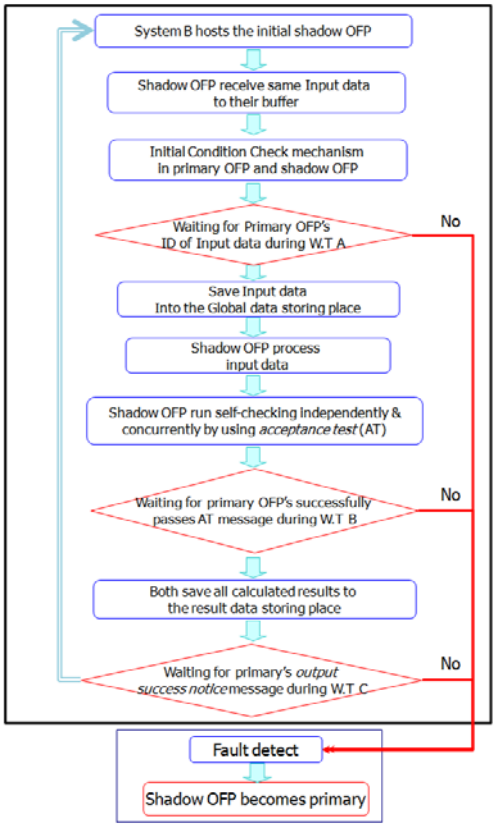

Figure 6: Shadow OFP Execution Process Flow

4 EXPERIMENTAL RESULTS AND ANALYSIS

We constructed HIL simulation test environment for reliability and accuracy measurement. In this environment we check system reliability and accuracy in the execution deadline time, where periodical execution based fault tolerance mechanism is utilized. Fig. 7 illustrates overall system view.

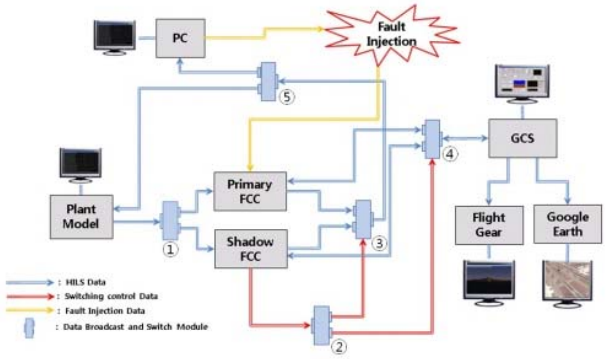

Figure 7: Fault Tolerance Experimental Environment

Our experimental environment is composed of combinations of several modules, such as Plant model, Primary Flight Control Computer (FCC), Shadow FCC, GCS, Flight Gear, Google Earth, Data Broadcast and Switch module based on RS-232 communication. Below we provide more specific details of our each module.

Plant model equipment generates a virtual sensor signal based on small scale helicopter's dynamic model, while receiving the real control signal from FCC module. Primary and Shadow FCC equipments are x86-based embedded board and they are running both Primary and Shadow OFP respectively. The Ground Control System (GCS), Flight Gear and Google Earth visualize UAV state, using UAV's current flight information. The PC's role is to calculate data logging and inject false fault into the Primary FCC. Finally, there are five data broadcast and switch equipments based on RS-232 communication (called as RS-232 equipment for the rest of the work). The first RS-232 equipment is broadcasting the virtual sensor signal generated by plant model to Primary and Shadow FCC. The second RS-232 equipment sends a control signal to the third and fourth RS-232 equipment, which will control the switching path of those equipments. The third and fourth RS-232 equipment relay the data to the fifth and GCS respectively and they will control switch path of the output. For example, if there is no fault, both of them will take the result of Primary FCC. On the other hand, in case of fault in the Primary FCC, the third and the fourth equipments will switch the path to consume the result provided by the Shadow FCC. Finally, the fifth RS-232 equipment is to broadcast the output data which is sent by the third equipment, to the PC and Plant model.

In our experiment we implemented Time-driven Fault-Tolerance mechanism in eCos (Embedded Configurable OS) [13] Real-Time Operating System (RTOS). We assumed failure mode in introduction. They are receiving the sensor data, calculating the control data and sending the control data of the OFP. To set the shadow OFP's waiting-for-response time at each failure mode, we measure each stage's execution Period and Worst Case Execution Time (WCET) in our experimental environment. In the Table. 1 we have an execution sequence illustrated for an each stage.

Table 1 : Experimental Measurement Period and WCET for each stage

Function	Periodic	Worst Case Execution Time
Receiving the sensor data	2 ~ 3 ticks	2 ticks
Calculating the control data	3 ticks	1 tick
Sending the control data	2 ~ 3 ticks	1 tick

Based on this analytical knowledge, we configured shadow OFP's wait-for-response time of A (W.T A) to be two clock ticks, wait-for-response time of B (W.T B) and C (W.T C) to be one clock tick (Fig. 4). During our experiment, 96% of our injected faults were successfully detected and recovered, and shadow OFP activation with a given

ISBN 978-979-16955-1-0

deadline time success rate was 94%. During our experiment 100 faults were injected to the system. Fig 8 and Fig 9 illustrates UAV altitude-change in the hovering mode when fault is enabled.

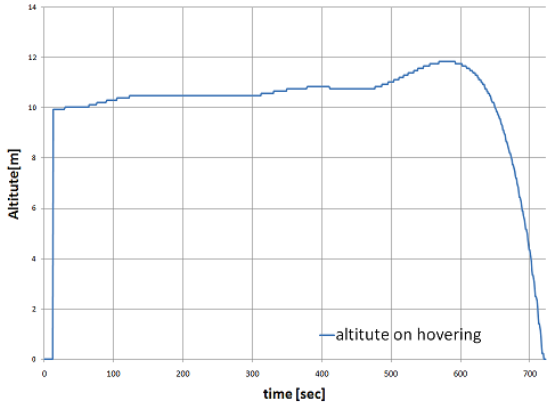

Figure 8: Altitude-change without Fault-Tolerance

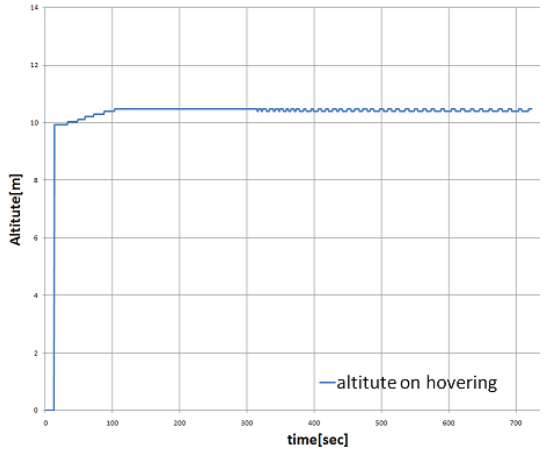

Figure 9: Altitude-change with Fault-Tolerance

5 Conclusion and Future Works

In this paper, we discussed about fault-tolerance issue in Unmanned Aerial Vehicle, and suggested Time-driven Fault-Tolerance mechanism motivated by TMO's PSTR scheme. Using the Time-driven Fault-Tolerance mechanism, we tested Operational Flight Program on the HILS environment. As a result, we affirm practical applicability of our fault-tolerance mechanism and it's suitability to improve reliability.

As future work we try to have more accurate value of WCET and make Time-driven Fault-Tolerance mechanism more reliable and scalable. We will keep this trend to make further enhancement in the Time-driven Fault-Tolerance and accompanying our research with practical results.

Acknowledgement

This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract. (UD060048AD)

References

[1] K. H. (Kane) Kim, Masaki Ishida, Juqiang Liu, *An Efficient Middleware Architecture Supporting Time-Triggered Message-Triggered Objects and an NT-based Implementation*, ISORC, Second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp.~54, 1999

[2] K. H Kim, C. Subbaraman, *An Integration of the Primary-Shadow TMO Replication Scheme with a Supervisor-Based Network Surveillance Scheme and Its Recovery Time Bound Analysis*, Proc. 17th IEEE Symposium on Reliable Distribute System, West Lafayette, Indiana, pp.~168, 1998

[3] K. Kim, C. Subbaraman, *A Modular Implementation Model of the Primary-Shadow TMO Replication Scheme and a Testing Approach using a Real-Time Environment Simulator*, ISSRE, The Ninth International Symposium on Software Reliability Engineering, pp.247, 1998

[4] K.H Kim, C. Subbaraman, *The PSTR/SNS Scheme for Real-Time Fault Tolerance via Active Object Replication and Network Surveillance*, IEEE Trans. On Knowledge and Data Engr., Vol. 12, No.2, pp.~145-159, 2000

[5] K. H. (Kane) Kim, Jeff J.Q. Liu, *Techniques for Implementing Support Middleware for the PSTR Scheme for Real-Time Object Replication*, ISORC, pp.~163-172, Seventh IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'04), 2004

[6] Pankaj Jalote, *Fault Tolerance in Distributed System*, Perntice-Hall Inc,A peason Education Company, 1998

[7] D. Briere, P. Traversem, *Airbus A320/A330/A340 electrical flight controls – a family of fault tolerant systems*, Proc. 23rd IEEE Int. Syymp. On Fault-Tolerant Computing (FTCS-23), Toulouse, France, pp.~ 293-307, 1993

[8] Pullum, Laura, Software Fault Tolerance Techniques and implementation, Artech House computing library, 2001

[9] Yeh Y.C, *Triple-Triple Redundant 777 Primary Flight Computers*, Proc. IEEE Aerospace Applications Conference, Aspen, CO, USA, 3-10 fevrier, pp.~293-307, 1996

[10] D.H Stamatis,*Failure Mode and Effect Analysis: FMEA from Theory to Execution*, ASQ Quality Press, 2003

[11] Raymond J. Mikulak, Robin McDermott, Michael Beauregard, *The Basics of FMEA*, Productivity Press, 1996

[12] J.D Andrews, T.R Moss, *Reliability and Risk Assessment*, Professional Engineering Publishing Limited, London and Bury St Edmund, UK, 2002

ISBN 978-979-16955-1-0

[13] eCos home page – http://ecos.sourceware.org

ISBN 978-979-16955-1-0