Journal of the KIMST, Vol. 18, No. 6, pp. 762-770, 2015
ISSN 1598-9127
DOI http://dx.doi.org/10.9766/KIMST.2015.18.6.762

Research Paper 정보 · 통신 부문

임무지향 컴퓨터를 위한 메시지패싱 고장감내 기법

김태현*,1) · 배정일1) · 신진범1) · 조길석1)

1) 국방과학연구소 제1기술연구본부

A Fault-Tolerant Scheme Based on Message Passing for Mission-Critical Computers

Taehyon Kim*,1) · Jungil Bae1) · Jinbeom Shin1) · Kilseok Cho1)

1) *The 1st Research and Development Institute, Agency for Defense Development, Korea*

(Received 25 June 2015 / Revised 6 October 2015 / Accepted 6 November 2015)

ABSTRACT

Fault tolerance is a crucial design for a mission-critical computer such as engagement control computer that has to maintain its operation for long mission time. In recent years, software fault-tolerant design is becoming important in terms of cost-effectiveness and high-efficiency. In this paper, we propose MPCMCC which is a model-based software component to implement fault tolerance in mission-critical computers. MPCMCC is a fault tolerance design that synchronizes shared data between two computers by using the one-way message-passing scheme which is easy to use and more stable than the shared memory scheme. In addition, MPCMCC can be easily reused for future work by employing the model based development methodology. We verified the functions of the software component and analyzed its performance in the simulation environment by using two mission-critical computers. The results show that MPCMCC is a suitable software component for fault tolerance in mission-critical computers.

Key Words : Engagement Control System(교전통제시스템), Fault-Tolerant Computer(고장감내 컴퓨터), Message Passing Synchronization(메시지패싱 동기화), Model-Based Development (모델기반 개발)

1. 서 론

방공용 교전통제시스템은 아군을 공격하는 적군 항공기나 유도탄과 같은 공중 위협에 대해 가용 유도탄을 할당하여 사격을 명령하고 통제하는 교전임무를 수행하는 시스템으로 빠른 응답성과 높은 신뢰도를 요구하는 경성 실시간 시스템이다. 교전임무를 수행하는 임무지향 컴퓨터에 고장이 발생하여 방공 임무가 중단되면 아군에 심각한 피해를 가져올 수 있으므로 안정적이고 신뢰성 높은 교전통제시스템을 위해서는 임무지향 컴퓨터의 고장감내 설계가 필수적이다.

* Corresponding author, E-mail: taehyonkim@add.re.kr

임무지향 컴퓨터의 고장감내 설계를 위해서는 고장이 발생할 수 있는 부분을 하드웨어적으로 중복하여 설계하는 것이 필수적이다. 그러나 하드웨어를 중복하여 적용할수록 시스템의 복잡도가 커져 개발 및 생산 비용이 크게 증가하므로 임무특성과 운용환경에 적합한 설계가 필요하다. 일반적으로 방공용 교전통제시스템은 전투기 보다 길고 우주항공 장비보다는 짧은 수일 정도의 임무지속 시간을 가진다. 시스템에 고장이 발생할 경우 자동으로 빠르게 복구 할 수 있는 고장감내 기능이 요구되지만 지상 혹은 함상에서 운용되므로 복구가 어려운 고장이 발생한 경우 임무를 중단하고 평균수리시간(MTTR)내에 정비수행을 통해 복구하는 것이 가능한 특성을 가진다[1]. 본 논문에서는 이러한 요구특성에 적합한 고장감내 기법을 제안한다. 제안하는 구조는 크게 이중화된 네트워크, 하드웨어적으로 동일한 두 대의 컴퓨터, 그리고 이에 내장된 고장감내 소프트웨어로 이루어진다.

이중화된 네트워크는 연동되는 외부장비와 컴퓨터간의 통신을 이중화하여 통신채널의 고장을 대비하며 이를 제어하는 내장형 통신 소프트웨어는 통신에 오류가 발생했을 경우 이를 복구하거나 재전송을 요청하여 정상적인 데이터를 수신할 수 있다. 임무지향 컴퓨터는 연동되는 외부장비와 데이터 입출력을 처리하고 교전계획을 생성하는 주 컴퓨터와 주 컴퓨터가 생성한 교전계획을 수신하여 동기화하는 부 컴퓨터로 나누어진다. 내장된 고장감내 소프트웨어는 모델기반으로 작성된 공통적인 메시지패싱 컴포넌트에 기반하며 주 컴퓨터에서 생성된 교전 정보를 부 컴퓨터로 동기화하여 하드웨어 고장을 대비한다. 주 컴퓨터에 고장이 발생할 경우 부 컴퓨터는 주 컴퓨터로부터 고장발생 메시지를 수신하거나 일정시간 이내에 장비 동작 신호인 Heartbeat를 수신하지 못하므로 부 컴퓨터는 동기화된 교전계획을 기반으로 교전계획을 복구하고 외부장비와 입출력 처리를 시작하여 교전중단 없이 고장감내를 수행하는 것이 가능하다.

2. 관련연구

고장감내 기법은 크게 하드웨어적인 기법과 소프트웨어적인 기법으로 구분된다. 하드웨어적인 기법은 고장이 발생할 경우 빠르게 고장을 탐지할 수 있는 장점이 있는 반면 부가적인 하드웨어로 인해 추가 비용이 발생하는 단점이 있다. 소프트웨어적인 기법은 하드웨어 기법보다 빠른 응답은 어렵지만 별도의 하드웨어를 요구하지 않고 구현된 모듈을 재사용할 수 있는 장점을 가진다. 잘 알려진 하드웨어 기반 고장감내 기법은 Triple Modular Redundancy[2] 및 Pair-and-Spare[3]가 있으며 소프트웨어 기반 고장감내 기법은 N-Version Programming[4], Recovery Block[5], 그리고 Process-Pair[6]가 있다.

Process-Pair는 고장에 대비하여 프로세스를 쌍으로 중복시키는 기법으로 프로세서 자원의 중복과 프로세스의 컨텍스트 보전 측면에서 Hot-Spare와 Hot-Standby로 나눌 수 있다[7]. Hot-Spare는 주 프로세스와 부 프로세스가 동시에 동일한 작업을 처리하도록 하여 이중화된 두 개의 프로세스 중 하나에 이상이 발생해도 다른 프로세스에 의해서 그 작업이 지속적으로 수행될 수 있도록 하는 기법이다. 두 프로세스가 동시에 동일한 작업을 수행하므로 고장이 발생하더라도 다른 프로세스를 통해 바로 복구가 이루어지므로 빠른 복구가 가능하나 다양한 외부장비와 서로 연동하고 주/부 프로세스가 서로 다른 컴퓨터에서 동작하는 복잡한 시스템에서는 동작 타이밍으로 인해 발생하는 프로세스간 입출력 동기화 문제를 처리하는 것이 까다로운 문제이다. Hot-Standby는 프로그램 수행을 주 프로세스에서만 수행하며 고장에 대비하여 자신의 상태를 주기적으로 부 프로세스로 전달하여 고장을 복구할 수 있는 정보인 Checkpoint를 생성한다. 주 프로세스에 고장이 발생하면 부 프로세스는 가장 최근의 Checkpoint를 이용하여 상태를 복구하고 그 시점부터 작업을 재개한다. 이 방식은 고장이 발생했을 경우 추가적인 복구절차를 수행해야 하므로 복구시간이 상대적으로 크나 평상시 동기화 절차가 상대적으로 간단하며 자원소모가 적은 장점이 있다. 본 논문에서 제안하는 메시지패싱 고장감내 기법은 Hot-Standby에 기반한다.

3. 메시지패싱 고장감내 기법

3.1 기본개념 및 동기화 컴포넌트

임무지향 컴퓨터에 내장되는 소프트웨어는 복잡도가 높고 다양한 기능이 요구되므로 단순한 기능을 가지는 여러 개의 CSC(Computer Software Component)로 나누어져 동작한다. 각 CSC는 일반적으로 운영체계 상에서 태스크나 스레드 등으로 구분되며 CSC간에 데이

터 동기화를 통해 전체 임무를 수행한다. 이러한 동기화 기법으로는 크게 공유메모리 기법과 메시지패싱 기법[9,10]이 있다. 공유메모리 기법은 동기화를 수행하는 CSC의 내용이 간단하고 개수가 적을수록 효율적인 방법이나 시스템이 커지면 동기화를 위한 오버헤드가 커지고 데드락의 문제가 존재한다. 메시지패싱 기법은 통신상의 메시지 전달을 통해 동기화를 수행하므로 간단한 시스템의 경우에도 기본적인 오버헤드를 가지는 단점이 있지만 시스템 확장에 따른 오버헤드가 작고 안전성이 높은 장점이 있다. 특히 CSC가 서로 다른 하드웨어에 있어도 투명하게 동기화를 수행할 수 있으므로 소프트웨어적으로 매우 직관적인 장점이 있다.

본 논문은 임무지향 컴퓨터의 고장감내를 위한 공통 소프트웨어 컴포넌트로 MPCMCC(Message Passing Component for Mission-Critical Computer)를 제안한다. 제안하는 소프트웨어 컴포넌트는 하드웨어적으로 분리된 임무지향 컴퓨터에 존재하는 공유데이터를 메시지패싱에 기반하여 단방향으로 동기화한다. 제안하는 소프트웨어 컴포넌트는 다음과 같은 조건에서 동작한다[1,7].

1) 동기화를 수행하는 CSC는 서로 다른 하드웨어에 존재하며 하드웨어는 동시에 고장 나지 않는다. 또한 고장복구 동안에 다른 고장이 발생하지 않는다.
2) 컴퓨터간 통신채널에서 오류나 메시지 오염이 발생하더라도 데이터 복구나 재전송을 통한 정상적인 데이터만을 수신한다.
3) 컴퓨터간 데이터 교환은 메시지로만 이루어진다. 컴퓨터간 공유되는 메모리를 통한 직접적인 데이터공유는 허용되지 않는다.
4) 내장되는 소프트웨어의 코드는 오류가 없다. 탐지되는 고장은 하드웨어의 고장이다.

3.2 메시지패싱 동기화 방식

메시지패싱 동기화 방식은 Table 1과 같이 동기화 속도와 데이터 일관성에 따라 Full-Sync, Semi-Sync, Async의 세 가지 방식으로 나눌 수 있다. Fig. 1은 메시지패싱 동기화 방식에 따른 주/부 컴퓨터간 동기화 시퀀스를 나타낸 것이다. 주 컴퓨터는 동기화를 위한 정보인 교전계획을 부 컴퓨터에 전달하여 동기화 수행을 시작한다. Async 방식은 주 컴퓨터가 교전계획을 부 컴퓨터에 전달하면 바로 동기화가 된 것으로 판단한다. 동기화 속도는 매우 빠르지만 부 컴퓨터가 교전계획을 전달받기 전에 통신고장이 발생하는 경우 교전계획을 잃어버려 동기화에 실패할 가능성이 존재하므로 데이터 일치성보다 동기화 속도에 민감한 시스템에 사용되어야 한다. Semi-Sync 방식은 부 컴퓨터가 교전계획을 수신했다는 Ack메시지를 수신하면 동기화가 된 것으로 판단한다. 일반적으로 교전통제시스템의 주/부 컴퓨터간 통신 지연시간은 수 ms 이내로 매우 작으며 부 컴퓨터가 수신된 교전계획을 동기화하는데 실패할 가능성은 매우 낮기 때문에 교전통제 시스템을 위한 임무지향 컴퓨터에 적합한 방식이다. Sync방식은 부 컴퓨터가 수신된 교전계획을 모두 동기화 하고 응답 메시지를 전달하면 동기화가 된 것으로 판단하는 방법으로서 데이터 일치성이 가장 좋으나 동기화 속도는 Semi-Sync방식보다 크게 낮아질 수 있으므로 데이터 일치성이 무엇보다 중요한 환경에서 사용되어야 한다.

Table 1. A comparison of message-passing synchronization modes

동기화 방식	동기화 속도	데이터 일관성
Full-Sync	낮음	매우 높음
Semi-Sync	높음	높음
Async	매우 높음	낮음

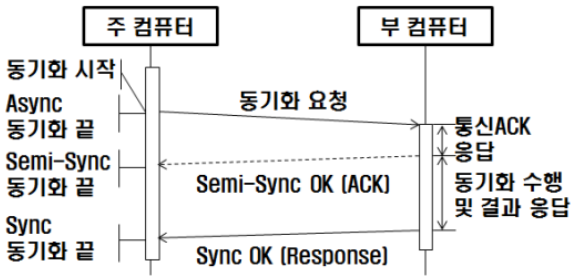

Fig. 1. Message-passing sequences between two computers according to synchronization modes

3.3 동기화 정보 및 고장감내 수행

메시지패싱 고장감내를 위해서는 고장이 발생했을 경우 정상상태로 복구할 수 있는 Checkpoint가 필요하며 방공용 교전통제 시스템의 경우 이 정보는 교전계획이 된다. 교전계획은 연동된 외부장비로부터 수신된

입력정보 및 내부 교전알고리즘을 사용하여 생성된 데이터셋이며 자신의 현재 위치 및 속도, 표적의 위치 및 속도, 교전상태 정보 등이 포함된다. 소프트웨어의 효율적인 동작 및 심플한 구조를 유지하기 위하여 데이터 교환을 관리하는 CSC가 MPCMCC를 통하여 다른 모든 CSC의 교전계획 복구를 위임받아 처리 할 수 있도록 하였으며 동일한 교전계획일 경우 동일한 복구 결과를 보장하여 교전중단 없이 고장감내가 가능하도록 하였다.

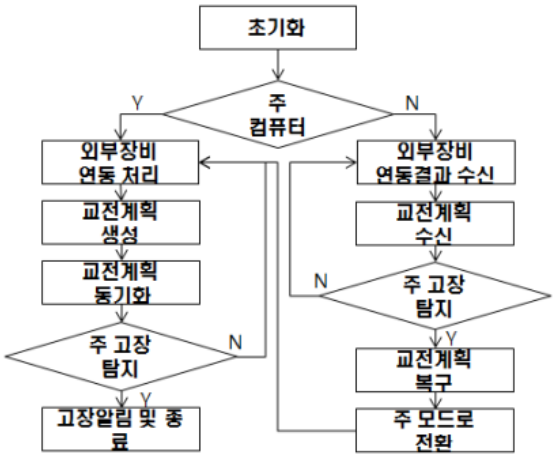

Fig. 2. A flowchart of important steps in the fault-tolerant algorithm

Fig. 2는 주/부 컴퓨터간 고장감내를 수행하는 방법을 나타낸 순서도이다. 먼저 컴퓨터의 초기화가 수행되면 설정을 확인하여 주 컴퓨터인 경우 주 모드로 동작하여 외부장비와 연동을 수행하며 교전계획을 생성하고 이를 부 컴퓨터로 동기화한다. 주기적으로 자신의 하드웨어 상태를 탐지하여 고장이 발생하지 않았으면 정상적인 동작을 수행하며 고장이 발생했으면 부 컴퓨터로 고장이 발생함을 알리고 동작을 종료한다. 부 컴퓨터인 경우 부 모드로 동작하며 외부장비와 직접 연동을 수행하지 않고 주 컴퓨터에서 연동한 결과만을 수신한다. 동기화를 위해 주 컴퓨터에서 생성된 교전계획을 수신하여 최신 교전계획을 유지하다가 주 컴퓨터의 고장을 탐지하면 최신 교전계획으로 내부 CSC를 고장복구하고 주 모드로 전환하여 동작한다.

3.4 모델기반 설계 및 구현

MPCMCC는 교전통제 소프트웨어 구현의 편리성과 메시지패싱 고장감내 기법의 성능개량 및 확장성에 유리하도록 UML2.0기반의 모델기반으로 설계 및 구현되었다. UML2.0 모델 기반 설계방법은 분석 및 설계 도메인에 적합한 모델링이 가능하며 내장형 소프트웨어에 적합하게 모델링이 가능하다[8].

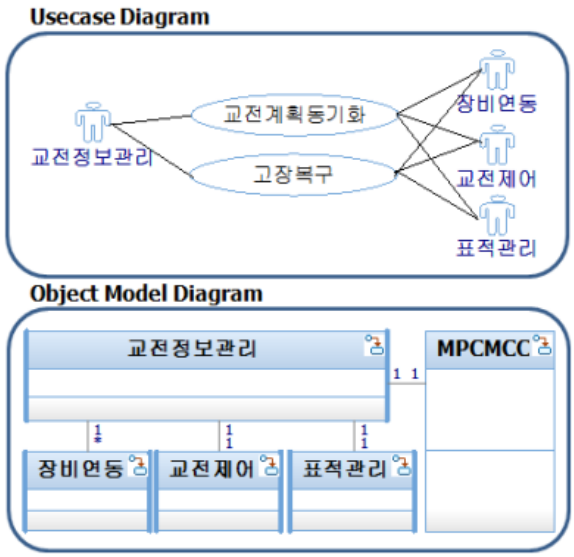

Fig. 3. Model based design and implementation-use case diagram and object model diagram of MPCMCC

Fig. 3은 MPCMCC의 요구조건 분석을 위한 유즈케이스 다이어그램과 구현을 위한 오브젝트 모델 다이어그램의 일부를 나타낸 것이다. 유즈케이스 다이어그램은 사용자 입장에서 바라본 시스템 동작에 대한 일종의 시나리오이다. 사용자는 MPCMCC가 제공해야 할 서비스, 즉 유즈케이스가 교전계획 동기화와 고장복구이고 이를 사용하는 액터들은 교전정보관리, 장비연동, 교전제어, 그리고 표적관리 CSC임을 직관적으로 확인할 수 있다. 오브젝트 모델 다이어그램은 소프트웨어의 클래스 구조 및 정적관계를 나타내주는 다이어그램이다. 사용자는 교전정보관리 인스턴스에 MPCMCC 인스턴스가 1:1로 연관되어 있는 것을 확인할 수 있다. 또한 교전정보관리 인스턴스와 장비연동, 교전제어, 그리고 표적관리 인스턴스가 1:n 혹은 1:1로 연관되어 있는 것을 확인할 수 있다. 이러한 모델기반 구현은 하나의 모델을 개발함으로서 하드웨어와 운영체제 및 사용언어의 변경에 관계없이 유연하게 여러 개

의 실행 가능한 코드로 자동 생성할 수 있으므로 설계된 모델을 여러 이기종 환경에서 쉽게 적용하는 것이 가능하다.

4. 임무지향 컴퓨터를 위한 고장감내 설계

4.1 하드웨어 및 소프트웨어 구조

Fig. 4는 임무지향 컴퓨터의 기본 하드웨어 구조를 나타낸다. 고장감내를 위하여 동일한 하드웨어로 구성된 두 개의 단일보드컴퓨터(이하 SBC : Single Board Computer)와 이더넷스위치를 포함하며 각 장치는 버스기판과 연결되어 전원을 공급받고 데이터버스로 연결된다. SBC는 전원인가 점검 및 사용자 요구점검 기능을 제공하고 이더넷스위치는 여기에 주기점검 기능을 추가적으로 지원하므로 원하는 시점에 고장을 탐지하는 것이 가능하다.

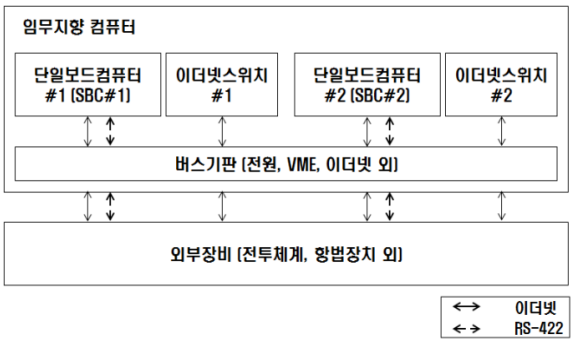

Fig. 4. The hardware architecture for fault tolerance in mission-critical computers

Fig. 5는 고장감내를 위한 임무지향 컴퓨터의 소프트웨어 구조를 나타낸다. 두 개의 SBC는 동일한 소프트웨어를 탑재하며 SBC의 장착위치 혹은 내부 설정에 따라 주 임무와 부 임무로 나뉘어 동작한다. 내부에 탑재되는 CSC는 크게 나누어 CSC간 데이터의 흐름을 관리하는 교전정보관리(이하 EIE : Engagement Information Exchange), 교전알고리즘에 기반하여 교전정보를 생성하는 교전제어(이하 EC : Engagement Control) 및 교전중인 표적정보를 관리하는 표적관리(이하 TM : Track Management), 그리고 외부장비와의 연동을 담당하는 장비연동(이하 Interface) 및 그 외의 CSC로 나눌 수 있다. EIE는 각 CSC의 내부 데이터 교환 및 MPCMCC의 인스턴스를 포함하여 교전계획의 동기화와 고장복구를 수행하는 매니저 기능을 수행한다.

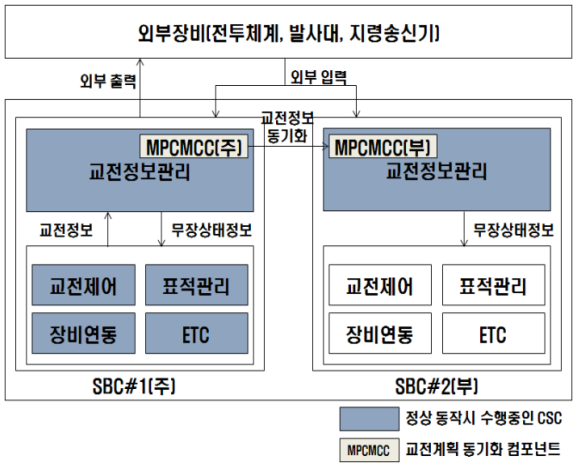

Fig. 5. The software architecture for fault tolerance in mission-critical computers

4.2 동기화 메시지의 신뢰성 판단

MPCMCC는 통신채널의 오류로 인한 메시지 오염이 발생하더라도 정상적인 메시지를 수신하는 것을 가정하였다. 실제 환경에서는 메시지의 내용이 변경되거나 사라질 가능성이 존재하므로 이를 해결하기 위하여 동기화 메시지를 MPCMCC로 전달하기 전 수신된 메시지가 신뢰할 수 있는지 여부를 판단하는 단계가 필요하다. 먼저 통신 채널에서 수신된 메시지 패킷에 오류가 발생하였는지를 판단하기 위해 패킷에 포함된 체크섬, 패킷타입, 시퀀스넘버, 프레임길이를 확인하여 모두 정상일 경우 해당 패킷에 오류가 없다고 판단한다. 다음으로 이중화된 패킷을 검사하여 패킷의 내용이 동일한 경우 해당 패킷은 신뢰할 수 있는 것으로 판단한다. 내용이 다를 경우 이중화 채널에 재전송을 요청하여 정상 패킷을 수신하도록 한다. 만약 하나의 채널에서만 패킷이 수신되었으면 이는 정상적인 패킷으로 간주하고 수신되지 않은 통신채널의 오류카운트를 증가시켜 각 통신채널의 동작 상태를 감시할 수 있도록 한다. 이러한 동작은 EIE에서 처리되며 외부장비와의 통신에도 공통적으로 적용된다.

4.3 정상상태 시 동작

주 컴퓨터는 외부장비의 연동과 내부 상태정보에 기

반하여 교전계획을 수립하고 그 결과를 MPCMCC를 통해 부 컴퓨터로 전달하여 최신 교전계획을 동기화한다. 주 임무로 동작하는 경우 모든 CSC가 정상 동작하지만 부 임무로 동작하는 경우에는 EIE외의 CSC는 동작을 대기하며 외부장비의 연동 결과만 처리하도록 하여 잘못된 정보를 생성하지 않도록 한다.

4.4 하드웨어 고장탐지

각 SBC는 CPU와 메모리 등의 하드웨어 컴포넌트가 기능적으로 잘 동작하는지 여부를 주기적으로 점검하여 임무 수행 중 하드웨어 고장이 발생했는지를 판별한다. 이러한 점검 정보는 CBIT(Continuous Built In Test)에 포함되어 주기적인 상호 교환을 통해 상대방의 고장을 탐지할 수 있다. 부 SBC가 지정된 시간 내에 CBIT나 Heartbeat정보를 수신하지 못하거나 수신된 정보에 이상이 있을 경우 부 SBC는 주 SBC가 고장이 발생했다고 판단하고 고장감내 절차를 수행한다. 주 SBC가 부 SBC의 Heartbeat 메시지를 수신하지 못하면 부 SBC에 고장이 났다고 판단하지만 추가적인 고장감내 절차는 수행하지 않는다. 네트워크 라인 단절 등으로 인한 네트워크 고장은 Heartbeat가 아닌 네트워크 이중화로 복구되며 이는 장비와의 연동을 담당하는 Interface CSC에서 처리를 담당한다.

4.5 고장복구

주기적인 Heartbeat절차에 실패하여 주 SBC에 고장이 발생되었다고 판단되면 부 SBC는 일련의 고장복구 절차를 수행하고 즉시 주 임무로 전환되어 동작하게 된다. Fig. 6은 주 SBC 고장시 고장복구 과정을 나타내는 시퀀스 다이어그램으로 고장복구 수행 순서는 다음과 같다. 먼저 부 SBC의 EIE는 MPCMCC에서 수신된 최신의 교전계획을 분석하여 외부장비와의 연동을 담당하는 Interface CSC를 복구한다. 예를 들어 발사대 Interface는 각 탄별 장착상태 및 진행 중인 발사절차의 정확한 복구를 보장한다. 지령송신기 Interface는 현재 업링크 중인 교전을 확인하고 동일한 주기로 업링크를 수행할 수 있도록 업링크 주기 테이블을 복구한다. 외부장비 Interface CSC의 복구가 완료되면 EIE는 TM의 표적정보 데이터베이스와 EC의 교전정보 데이터베이스를 복구하며, 이 과정은 TM과 EC의 데이터베이스 구현에 따라 상이할 수 있다. 부 SBC의 모든 CSC가 복구절차를 완료하면 EIE는 주 임무 전환 이벤트를 각 CSC에 전달하여 주 임무로 전환하고 외부장비와 연동을 재개한다. 동기화된 교전계획에 기반한 고장복구 절차를 통해 부 SBC는 교전중단 없이 외부장비와 임무를 재개할 수 있다.

부 SBC에 고장이 발생하면 주 SBC는 Fig. 7과 같이 단순히 MPCMCC를 통한 교전계획 동기화를 중단하고 그 외의 동작은 모두 정상적으로 수행한다. 주 SBC는 부 SBC가 고장이 발생해도 정상동작에 영향을 받지 않으므로 추가적인 고장감내 절차가 필요하지 않다. 고장이 발생된 SBC는 고장을 확인하고 초기화하기 전에는 다시 정상상태로 전환되지 않는다.

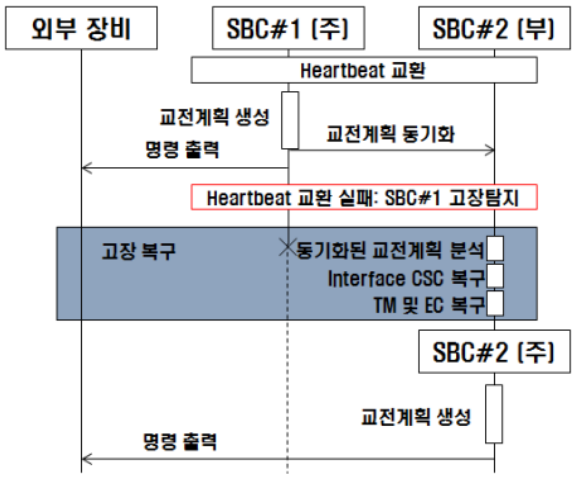

Fig. 6. Failover sequence diagram when the master computer goes down

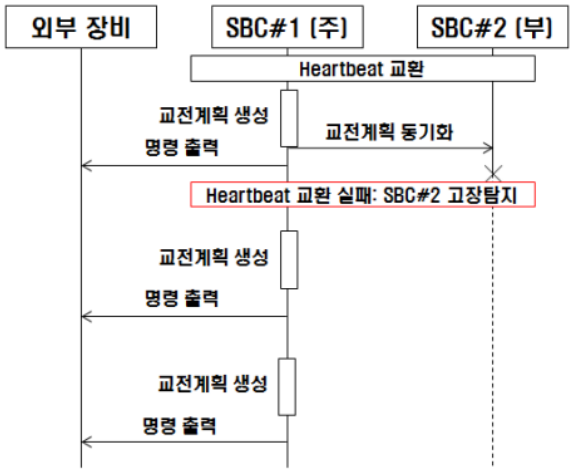

Fig. 7. Failover sequence diagram when the secondary computer goes down

5. 시험 및 성능분석

5.1 모의 시험환경 구성 및 설정

제안하는 메시지패싱 고장감내 기법의 성능을 분석하기 위하여 모의 시험환경을 구축하였다. 모의 시험환경은 크게 외부장비 모의기, 네트워크 스위치, 데이터 분석장치, 그리고 통제컴퓨터로 이루어진다. 통제컴퓨터는 단일보드컴퓨터용 회로카드조립체, 입출력용 회로카드조립체, 이더넷스위치용 회로카드조립체를 각각 2개씩 포함하였다. 하드웨어 구성품의 고장을 탐지하기 위해 수행하는 CBIT의 실행 주기는 5000 ms로 모든 회로카드조립체에 동일하게 설정하였으며 고장정보를 전송하기 위한 Heartbeat의 주기는 20 ms로 설정하였다. CBIT 결과에 오류가 발생하거나 Heartbeat 신호를 5회 연속 미수신하면 해당 하드웨어는 고장이 발생한 것으로 판단한다.

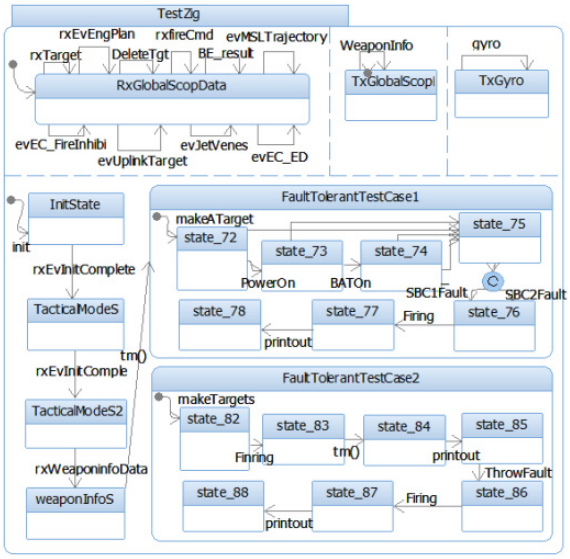

Fig. 8. Model based test bench for functional verification

5.2 고장감내 기능 검증

구현된 고장감내 소프트웨어의 단위기능을 검증하기 위한 시험치구를 모델기반으로 작성하였다. Fig. 8은 모델기반으로 작성된 시험치구이다. 시험치구를 사용하여 시험 시나리오에 따른 이벤트를 발생시켜 출력되는 이벤트와 소프트웨어 내부 상태머신의 상태천이 과정을 확인하여 단위기능을 검증하였다.

고장 시나리오는 크게 단일표적과 다중표적으로 나누어 작성하였다. 단일표적 시나리오는 유도탄 전원인가 전, 가역, 비가역, 유도탄 이탈과 같이 유도탄 발사절차 중 고장이 발생할 수 있는 모든 시점을 포함하였으며 고장유형을 주/부 SBC 고장으로 구분하여 결과를 확인하도록 하였다. 다중표적 시나리오는 연속적인 유도탄 발사절차를 진행하고 특정한 시점에 고장을 발생시켜 모든 표적 및 각 유도탄에 대해 정상적으로 고장감내 기능이 동작하여 교전이 유지되는지를 확인하도록 하였다. Table 2에 시험 시나리오와 이에 따른 고장감내 단위시험 결과를 나타내었다. 시험결과 모든 시나리오에서 고장감내 기능이 정상적으로 동작하여 교전이 유지됨을 확인하였다.

Table 2. Test scenarios and experimental results in the unit test

시험 시나리오			시험 결과
표적	고장 장비	고장모의 시점	
단일 표적	주 SBC	전원인가전	교전유지
		가역	
		비가역	발사중지 및 유도탄 재할당[1] 교전유지
		탄이탈	
	부 SBC	전원인가전	교전유지
		가역	
		비가역	
		탄이탈	
다중 표적 (단발 발사)	주 SBC	전원인가전, 가역, 비가역, 탄이탈 시점을 포함	각 유도탄에 대해 단일표적인 경우와 동일
다중 표적 (연발 발사)	주 SBC	전원인가전, 가역, 비가역, 탄이탈 시점을 포함	

1) 유도탄 안전처리에 따른 절차

5.3 고장감내 성능분석

임무 소프트웨어의 모든 기능을 통합하고 고장감내

통합시험을 수행하여 고장감내 소프트웨어의 실제 성능을 확인하였다. 고장감내 소프트웨어에서 생성되는 CBIT, Heartbeat, 그리고 동기화 이벤트는 모두 데이터 분석장치에 전달되어 로깅되며 CSV(Comma Seperated Value)파일 포맷으로 변환이 가능하다. 로깅된 메시지를 분석하여 구현된 고장감내 기법의 성능을 확인하였다. 데이터 분석장치 로깅시간의 해상도는 1 ms로 두 이벤트의 시간차이가 1 ms보다 크지 않으면 동일한 시간으로 기록된다.

Fig. 9는 교전계획 동기화에 소요되는 시간을 분석한 것이다. 교전데이터 동기화는 주 컴퓨터가 부 컴퓨터에게 동기화할 교전계획을 전달하고 부 컴퓨터로부터 통신 Ack를 받는데 소요되는 시간이다. 기록된 동기화 시간은 1 ms 이내로, 데이터 분석장치의 해상도를 고려하면 동기화 수행에 소요되는 시간은 최대 2 ms를 초과하지 않는 것을 확인할 수 있다.

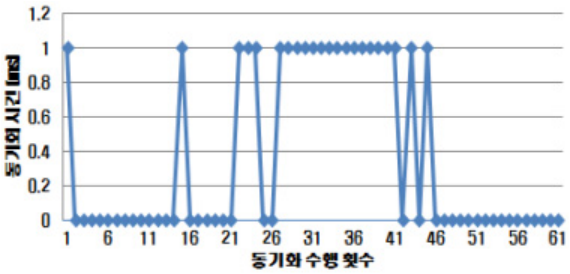

Fig. 9. Elapsed time for synchronization of engagement plan between two computers

Table 3은 고장감내 통합시험 중 고장모의 전후에 로깅된 이벤트를 분석한 것이다. CBIT이 SBC#1에서 정상적으로 수행된 것을 기준으로 4999 ms 후 CBIT이 다시 SBC#1에서 수행된 것을 확인할 수 있다. 5011 ms에 Heartbeat와 외부장비 상태요청을 SBC#1이 수행하였고 5018 ms에 SBC#1의 고장이 모의되었다. 5136 ms에 외부장비 상태요청을 SBC#2가 수행한 것을 확인하였는데 이는 SBC#2가 SBC#1으로부터 5011 ms 이후 Heartbeat를 연속 5회 미수신하여 5111 ms(5011 ms + 100 ms)에서 고장감내 소프트웨어가 동작하였고 고장감내 처리를 완료 한 후 SBC#2가 외부장비 상태요청을 수행한 것이다. 고장감내가 완료된 후 외부장비 상태요청을 곧바로 시작한 시간이 5136 ms이므로 25 ms (5136 ms − 5111 ms = 25 ms)이내에 고장복구가 완료되었다고 판단할 수 있다. 이러한 성능은 교전통제시스템을 위한 임무지향 컴퓨터의 고장감내 요구조건을 충분히 충족시키는 수준이다. 이후 SBC#2는 CBIT를 수행하며 SBC#1의 고장을 확인한 것을 알 수 있다.

Table 3. Event logs analysis in the system integration test

로깅 타임 (ms)	메시지	데이터	비고
0	CBIT	SBC1_BIT:1 SBC2_BIT:1	SBC#1수행
4999	CBIT	SBC1_BIT:1 SBC2_BIT:1	SBC#1수행
5011	Heartbeat	-	SBC#1수행
5016	외부장비 상태요청	-	SBC#1수행
5018	고장모의	-	SBC#1 고장모의
5136	외부장비 상태요청	-	SBC#2수행
5293	CBIT	SBC1_BIT:0 SBC2_BIT:1	SBC#2수행

6. 결 론

본 논문은 임무지향 컴퓨터를 위한 저비용 고효율의 고장감내를 위하여 메시지패싱 기반의 고장감내 기법을 제안하였다. 제안하는 소프트웨어 컴포넌트인 MPCMCC는 하드웨어적으로 분리된 컴퓨터간의 공유데이터인 교전계획을 메시지패싱을 통해 단방향으로 동기화하므로 공유메모리 기법에 비해 오버헤드가 적고 안정적이며 모델기반으로 개발되어 쉽게 재사용이 가능하다. 임무지향 컴퓨터는 고장감내를 위한 하드웨어 자원을 최소한도로 포함하며 하드웨어 고장을 탐지하고 MPCMCC에 기반하여 교전계획을 동기화하고 고장복구를 수행하도록 설계하였다. 고장감내 기능 검증을 위해 통제컴퓨터를 포함한 모의실험 환경에서 유도탄 발사절차의 모든 단계에서의 고장 모의 및 다중표적 환경에서의 고장을 모의하는 시험 시나리오를 수행하였다. 이를 통해 발생 이벤트들과 소프트웨어 내부 상태머신의 상태천이 과정을 확인하여 고장감내

기능이 정상적으로 동작함을 검증하였다. 또한 임무 소프트웨어의 모든 기능을 통합하고 고장감내 통합시험을 수행하여 제안한 기법의 성능을 분석하였으며 그 결과 임무지향 컴퓨터에 적합한 성능을 가지는 것을 확인하였다.

MPCMCC는 하드웨어 고장이 동시다발적으로 발생하지 않고 통신채널에 오류나 메시지 오염이 발생하더라도 데이터 복구나 재전송을 통해 정상적인 데이터를 수신하는 것을 가정하며 소프트웨어 오류가 없는 환경에서의 동작을 보장한다. 향후 연구과제로 이와 같은 복잡한 고장 상황에서도 고장감내 기능을 정상적으로 수행할 수 있도록 보완하는 것이 필요하다.

References

[1] J. Shin, S. Kim, “Reliability Analysis of The Mission-Critical Engagement Control Computer Using Active Sparing Redundancy,” The KIPS Transactions: Part A, Vol. 15, No. 6, pp. 309-316, 2008.

[2] R. E. Lyons, W. Vanderkulk, “The Use of Triple-Modular Redundancy to Improve Computer Reliability,” IBM Journal of Research and Development, Vol. 6, No. 2, pp. 200-209, 1962.

[3] J. Gray, D. P. Siewiorek, “High-Availability Computer Systems,” Computer, Vol. 24, No. 9, pp. 39-48, 1991.

[4] A. Avizienis, “The N-Version Approach to Fault-Tolerant Software,” Software Engineering, IEEE Transactions on, Vol. SE-11, No. 12, pp. 1491-1501, 1985.

[5] K. Shin, Y. Lee, “Evaluation of Error Recovery Blocks Used for Cooperating Processes,” Software Engineering, IEEE Transactions on, Vol. SE-10, No. 6, pp. 692-700, 1984.

[6] I. Lee, R. K. Iyer, “Software Dependability in the Tandem GUARDIAN System,” Software Engineering, IEEE Transactions on, Vol. 21, No. 5, pp. 455-467, 1995.

[7] D. Song, C. Lee, “An Implementation of Fault-Tolerant Message Passing Interface on Parallel Computers,” Journal of KIISE : Computing Practices and Letters, Vol. 6, No. 3, pp. 319-328, 2000.

[8] M. Yoo, et. al., “Development of the Engagement Control Software Architecture Based on UML 2.0 Model,” Journal of the Korea Institute of Military Science and Technology, Vol. 10, No. 4, pp. 20-29, 2007.

[9] B. Rajappa, Y. Motiwala, “Message Based Redundancy Approach using Totem Protocol for Telecom Applications and Protocol Stacks,” Communication Systems Software and Middleware, 2nd International Conference on, pp. 1-6, Jan. 2007.

[10] R. Batchu, et. al., “MPI/FT: A Model-Based Approach to Low-Overhead Fault Tolerant Message-Passing Middleware,” Cluster Computing, Vol. 7, No. 4, pp. 303-315, 2004.