Journal of the KIMST, Vol. 22, No. 3, pp. 382-391, 2019
ISSN 1598-9127(print), 2636-0640(online)
DOI http://dx.doi.org/10.9766/KIMST.2019.22.3.382

Research Paper 정보 · 통신 부문

함정 전투체계에서 통합 통제 시스템의 고장 감내를 지원하기 위한 리더 선정 알고리즘 개발

서용진[1] · 조준영[1] · 김현수[*,1] · 고영근[2] · 김점수[2]

[1] 충남대학교 컴퓨터공학과
[2] 국방과학연구소 함정전투체계단

Development of Leader Selection Algorithm to Support Fault Tolerance of Integrated Management Systems in the Naval Combat System

Yongjin Seo[1] · Jun Young Jo[1] · Hyeon Soo Kim[*,1] · Youngkeun Go[2] · Chum-Soo Kim[2]

[1] *Department of Computer Science & Engineering, Chungnam National University, Korea*
[2] *The Naval Combat Systems PEO, Agency for Defense Development, Korea*

(Received 19 February 2019 / Revised 16 April 2019 / Accepted 17 May 2019)

ABSTRACT

The naval combat system is a distributed system in which various subsystems are integrated and operated together. The integrated management system(IMS) is a software system for systematically and consistently managing the application software which control and operate various devices in such a combat system. Since the malfunction or failure of such an IMS can disable the entire combat system, the IMS is more important than other application software of the combat system. In this paper, we propose a method to guarantee the stable and correct operation of the combat system. To this end, we propose a redundancy scheme composed of one leader and several followers so as to tolerate the failure situation of the IMS. We also propose a leader selection algorithm to select a new leader when the leader fails and can no longer perform its role. To verify the validity of the study, we verify the fault tolerance behavior of the system and the accuracy of the leader selection algorithm.

Key Words : Naval Combat System(함정 전투 시스템), Integrated Management System(통합 통제 시스템), Fault Tolerance(고장 감내), Leader Selection(리더 선정), Information Model(정보 모델)

1. 서 론

수상/수중함 전투체계는 적의 위협을 조기에 탐지하고 추적하는 센서체계, 함정의 여러 센서로부터 수집된 정보를 바탕으로 대응할 장비를 운용하기 위한 명령을 내리거나 지휘관의 의사결정을 지원하기 위한

* Corresponding author, E-mail: hskim401@cnu.ac.kr

통제체계, 방어를 위한 단계별 무기체계, 전술 통신 및 항해 장비 등과 같이 다양한 부체계들이 하나로 연동되어 통합 운용되는 분산 시스템으로[1] 다양한 제작사에서 제공되는 이질적 특성을 지니는 다수의 장비들과 이들을 제어하고 운영하기 위한 소프트웨어가 결합된 복합 시스템(System of Systems)이다[2]. 다양한 하부 체계들로 구성되는 통합 전투체계 시스템을 원활하게 관리하기 위해서는 각 하부 체계들의 다양성을 추상화하는 기술과 이들을 제어하고 운영하는 애플리케이션을 일관성 있게 관리하는 기술이 필요하다. 통합 통제 시스템은 바로 이러한 함정 전투체계에서 제어/운영 애플리케이션을 체계적이고 일관성 있게 관리하기 위한 소프트웨어 시스템이다.

통합 통제 시스템의 동작을 추상적으로 보자면, 각종 센서나 장비를 구동시키고 제어하는 역할을 수행하는 응용 소프트웨어를 관리 대상으로 삼아 응용 소프트웨어를 구동시키기 위한 명령을 내리거나 응용 소프트웨어의 실행 상황에 대한 모니터링 정보를 수집하는 역할을 수행한다. 이러한 통합 통제 시스템은 함정 전투체계의 다른 응용 소프트웨어에 비해 그 중요도가 더욱 크다고 할 수 있다. 시스템의 오작동이나 실패는 전체 함정 전투체계를 무력화 시킬 수 있기 때문이다.

그러나 지금까지 수행된 연구들은 함정 전투체계의 구성[1], 개방형 구조를 기반으로 한 아키텍처 설계[2], 복잡한 함정 전투체계를 지원하기 위한 통신 메커니즘[3], 함정 전투체계에서 처리되는 과다한 정보를 사람이 쉽게 다룰 수 있도록 지원하는 자동화 방안[4] 등에 관한 내용에 초점이 맞춰져 진행되었다.

본 논문에서는 함정 전투체계를 실제 운영하는 관점에서 고려해야 할 사항 중 하나인 함정 전투체계의 안정적이고 올바른 동작을 보장하기 위한 방안을 다루고자 한다. 이를 위해 먼저, 통합 통제 시스템의 다중화 방안을 제시한다. 통합 통제 시스템을 하나의 리더(leader)와 여러 개의 팔로워(follower)의 형태로 다중화하여 구성함으로써 통제 시스템의 실패로 인한 고장 상황을 감내할 수 있도록 통합 통제 시스템을 설계한다. 또한 통합 통제 시스템을 다중화 함으로 인해 새롭게 등장하는 문제인 리더가 실패하여 더 이상 제 역할을 수행할 수 없을 때 리더를 대체할 팔로워 중에서 새로운 리더를 선택하기 위한 방안으로 리더 선정 알고리즘을 제시한다. 또한, 연구의 타당성을 확인하기 위해 통합 통제 시스템의 프로토타입을 설계하고 구현한 내용을 기술하고 구현된 통합 통제 시스템에서 리더 동작에 관한 다양한 시나리오를 실행해 봄으로써 통합 통제 시스템의 고장 감내 동작 및 이를 위한 리더 선정 알고리즘의 정확성을 검증한다.

2. 관련 연구

이 논문에서 제안하는 내용은 기본적으로 고장 감내(fault tolerance) 기법의 일종이다. 고장 감내 기법은 능동 다중화(active redundancy), 수동 다중화(passive redundancy), 예비(spare) 기법과 같이 세 가지 유형의 기법이 사용되어 왔다[5]. 이런 방법 모두에서 고려해야 할 것은 주 서버가 실패하였을 경우 다른 멤버 중에서 주 서버의 역할을 대신할 요소를 선정하는 것이다. 특히, 본 연구에서처럼 서버 그룹을 구성하는 멤버가 다수일 경우에는 수동 서버 중에서 주 서버의 역할을 수행할 멤버를 선택하기 위한 효율적인 알고리즘이 필요하다. 이와 같이 여러 개의 요소로 구성된 서버 그룹에서 주 서버가 실패할 경우 새로운 주 서버를 선정하는 일을 우리는 리더 선정(leader selection) 문제라 칭한다. 리더 선정과 관련하여 몇몇 연구들이 진행되었다.

Zhao[6]의 연구에서는 분산 애플리케이션에서 빠른 응답을 보장하기 위해 실패가 발생하였을 때 그것을 빨리 대처하기 위하여 백업 컴포넌트 중에서 주 컴포넌트를 선택하기 위해 리더-팔로워 복제 전략을 제시하고 있다. 여기서는 멤버가 그룹에 참여하는 순서에 의해 그룹의 멤버들에게 우선순위(precedence)를 부여한다. 현재 리더가 실패하면 팔로워의 우선순위에 따라 새로운 리더가 될 승계 순서가 결정된다. 이 연구에서는 우선순위 개념 외에도 복제본들에게 등급(rank)을 부여하고 있다. 부여된 등급의 순서에 따라 실패 감지 시간이 배정되며 이는 등급의 앞 순서 복제본이 뒤 순서 복제본보다 먼저 리더의 실패를 감지할 수 있도록 활용된다. 이를 통해 리더 경쟁에서 발생할 수 있는 경합 조건(race condition)을 줄일 수 있다. 그렇지만 우선순위뿐만 아니라 등급을 운영하므로 리더 선정에 대한 오버헤드가 높다는 문제점이 있다. 특히 리더가 교체되면 멤버의 등급을 재산정하는 과정이 동반되어야 하므로 리더 선정을 위한 예비 과정의 복잡도와 부담이 증가한다.

Lu[7]의 연구에서는 데드락(deadlock) 검출을 위한 방법에서 리더와 팔로워를 선택하는 방법을 제시하고

있다. 데드락 검출에 관여하는 제어 프로세스에게 리더, 팔로워, 후보자 세 가지의 역할을 부여한다. 처음에는 모든 프로세스가 팔로워로서의 역할을 수행하다가 리더 선정 투표에 참여하거나 리더 추천 메시지를 전송하고 나면 후보자로 역할이 바뀐다. 이 상황에서 리더 투표에서 이긴 프로세스는 리더가 되고 그렇지 못한 프로세스는 다시 팔로워가 된다. 그러다 네트워크의 단절로 인해 리더가 제 역할을 수행할 수 없게 되었을 때 팔로워들은 다시 리더 선정 투표에 참여하거나 리더 추천 메시지를 전송하면서 후보자로 역할이 바뀌게 되고 앞서 설명한 대로 리더 선정 투표에서 이기게 되면 리더로서의 역할을 수행하게 된다. 우선권(priority)을 기반으로 동작하며 높은 우선권을 갖는 프로세스는 낮은 우선권의 프로세스에게 부정의 투표 메시지를 전송하고, 낮은 우선권의 프로세스는 높은 우선권의 프로세스에게 긍정의 투표 메시지를 전송한다. 자신을 제외한 모든 프로세스로부터 긍정의 메시지를 받은 프로세스가 리더가 되며, 리더가 되자마자 모든 다른 프로세스들에게 승리 메시지를 전송하여 자신이 리더가 되었음을 알린다. 이 연구에서는 리더 선정을 위하여 여러 종류의 메시지(리더 선출 요청 메시지, 리더 찬성 메시지, 리더 반대 메시지, 승리 메시지, 리더 추천 메시지)를 운영하며, 각각의 메시지들이 상황에 따라 전송되어야 하므로 메시지 운영의 복잡도가 증가하고 아울러 전송되는 메시지 양도 증가한다. 연구의 목적이 데드락 검출이므로 필연적으로 리더는 다른 모든 팔로워와 연결되어 있어야 하므로 어느 하나의 팔로워와도 네트워크 단절이 발생하면 비록 리더가 존재하더라도 리더 선정이 다시 이루어져야 한다. 따라서 리더 선정 작업이 빈번하게 발생 할 수 있으므로 문제점이 될 수 있다.

Paxos[8] 알고리즘은 비동기 분산 시스템을 위한 리더 선정 알고리즘으로, 대다수 구성원이 리더에 투표하는 2 단계 커밋 전략을 사용하며 합의(consensus)를 기반으로 한 리더 선정 방식이다. 이 연구는 리더 선정을 위한 오래된 방법으로 종다수 기반의 다중 라운드 합의 알고리즘을 사용한다.

본 연구의 리더 선정 방식도 기본적으로 우선순위를 기반으로 동작한다. 리더를 선정하기 위해 각각의 집행부들이 서로 메시지를 주고받는데 그 때 우선순위를 기반으로 메시지를 수신할지 여부를 판단한다. 그렇지만 본 연구에서는 와치독 타이머를 사용하여 와치독 타이머가 타임아웃 되었을 경우에만 메시지를 전송하도록 함으로써 과다한 통신 메시지의 교환을 줄여 통신 채널의 효율성을 높이도록 설계하였다. 또한 리더와 팔로워 관계가 주 컴포넌트와 백업 컴포넌트의 관계이므로 백업 컴포넌트 중 일부가 네트워크 단절이 된다 하더라도 리더 선정 작업을 다시 수행할 필요가 없다. 뿐만 아니라 종다수 기반의 다중 라운드 합의 알고리즘을 사용하지 않는다. 본 연구의 가장 큰 특징 중 하나는 리더 선정 과정에서 자신보다 높은 우선순위를 갖는 집행부의 존재를 인지하면 곧 바로 팔로워 상태로 돌아가서 리더 경쟁에 참여하지 않는다. 이렇게 함으로써 리더 선정 과정에서 경쟁 관계에 있는 집행부들 중 상대적으로 낮은 우선순위의 집행부들이 모두 팔로워 상태가 되고 나면, 가장 높은 우선순위를 갖는 나머지 집행부가 리더가 되므로 반드시 하나의 리더만 선정됨을 보장할 수 있다.

3. 통합 통제 시스템의 구조 및 리더 선정 알고리즘

3.1 통합 통제 시스템의 구조

통합 통제 시스템은 크게 두 종류의 요소로 구성되며, 수행하는 역할에 따라 집행부(executive)와 에이전트(agent)로 나눠질 수 있다. 여기서 집행부는 에이전트를 통해 관리 대상인 응용 소프트웨어(application software)를 제어하고 그들의 상태를 관리하는 역할을 수행하며, 에이전트는 집행부로부터 명령을 전달 받아서 응용 소프트웨어를 구동시키거나 응용 소프트웨어의 실행 상황에 대한 모니터링 정보를 집행부로 전달하는 역할을 수행한다. 집행부는 집행부 리더와 집행부 팔로워로 분류되는데, 리더는 다중화 형태로 구성된 집행부 중에서 주 모듈(primary)로 동작하며, 하나의 모듈만이 리더로 선정된다. 팔로워는 다중화 형태로 구성된 집행부 중에서 수동 모듈(passive)로 동작하며, 리더를 제외한 나머지가 모두 팔로워로 동작한다.

통합 통제 시스템은 Fig. 1과 같이 세 가지 형태로 구성되고 배치된다. 첫 번째 형태는 호스트 노드에 집행부 리더와 에이전트가 배치되는 형태이며, 두 번째는 호스트 노드에 집행부 팔로워와 에이전트가 배치되는 형태이다. 마지막으로 호스트 노드에 에이전트만 배치되는 형태도 존재한다. 에이전트는 응용 소프트웨어를 직접 실행시키는 모듈이기 때문에 소프트웨어에 대한 원활한 통제를 위해서는 모든 호스트 노드에 설치되어야 한다. 집행부는 통합 관리 기능을 수행하기

위해서 높은 수준의 가용성을 제공하여야 하며, 이를 위해 다중화 형태로 배치되어야 한다. 주 모듈에 해당하는 요소는 하나만 존재하여야 하므로 첫 번째 형태와 같은 배치 형태를 갖게 된다.

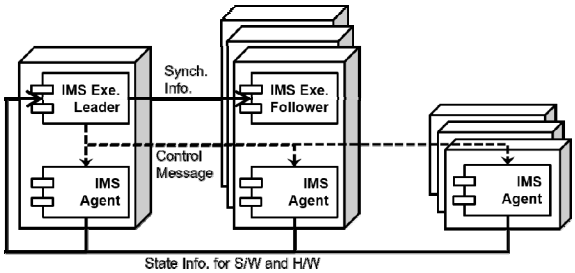


Fig. 1. IMS configuration and deployment

3.1.1 집행부 리더의 기능 및 상호작용에 대한 요구사항

집행부 리더는 다중화 형태로 구성된 집행부 중에서 주 모듈의 역할을 수행한다. 집행부 리더에게 주어진 역할은 (1) 관리 대상 응용 소프트웨어에 대한 상태 정보 수집을 통한 상태 감시, (2) 관리 대상 응용 소프트웨어에 대한 제어 기능, (3) 집행부 팔로워와의 동기화 기능, (4) 기능을 수행하는 과정에 발생된 로그 관리 기능 등이 있다.

3.1.2 집행부 팔로워의 기능 및 상호작용에 대한 요구사항

집행부 팔로워는 다중화 형태로 구성된 집행부 중에서 수동 모듈의 역할을 수행하는 모듈이다. 집행부 팔로워에게 주어진 역할은 (1) 집행부 리더와의 동기화 기능, (2) 집행부 리더의 상태 감시 등이 있다.

3.2 통합 통제 시스템에서의 리더 선정 알고리즘

집행부 리더 선정 알고리즘은 '집행부 리더는 다중화 형태로 구성된 집행부 중에서 주 모듈로 동작하며, 하나의 모듈만이 집행부 리더로 선정된다.'와 '집행부 팔로워는 다중화 형태로 구성된 집행부 중에서 수동 모듈로 동작하며, 집행부 리더를 제외한 나머지 집행부는 모두 집행부 팔로워로 동작한다.'라는 요구사항을 만족하기 위한 알고리즘이다. 집행부는 다수의 호스트에 분산 배치되기 때문에, 주 모듈로 동작하는 리더를 선정하는 과정은 매우 중요하다. 이 때, 리더 선정 알고리즘은 단순히 집행부 리더를 선정하는데 그쳐서는 안 된다. 리더의 이상 동작이 감지되면 팔로워 중 하나는 리더를 대체할 수 있어야 한다. 따라서 리더 선정 알고리즘은 다음을 만족하여야 한다.

- 집행부 중에서 리더로 동작할 모듈을 선정할 수 있어야 한다.
- 집행부 리더의 이상 유무를 파악할 수 있어야 한다.
- 집행부 리더의 이상을 감지한 뒤, 새로운 리더가 선정될 수 있어야 한다.

본 연구에서는 위와 같은 문제를 해결하기 위해서 생명신호(heartbeat) 기반의 리더 선정 알고리즘을 개발하였다. 생명신호는 특정 대상의 동작 여부를 감시하기 위한 기법으로, 감시가 필요한 대상이 주기적으로 생명신호 메시지를 전송함으로써 자신이 동작 중임을 알린다. 감시자는 생명신호 메시지가 전송되지 않으면 감시 대상이 동작하지 않는다고 판단한다. 집행부는 운용 기간 동안 리더와 팔로워 역할을 모두 수행할 수 있으므로, 감시자와 감시 대상의 역할을 모두 수행하여야 한다. 즉, 모든 집행부들은 생명신호 메시지를 송수신할 수 있어야 한다.

하지만 생명신호 메시지는 리더의 이상 유무를 파악하기 위한 것이므로, 특별히 데이터를 담아 전송하진 않는다. 본 연구에서는 생명신호 메시지의 데이터 구조를 Table 1과 같이 정의하고 우선순위를 부여한다. 생명신호 메시지는 전송자의 식별자와 전송자의 동작 상태로 구성된다. 여기서 전송자의 동작 상태는 초기화 상태, 팔로워 상태, 리더 상태로 나뉜다.

Table 1. Data structure of a heartbeat message

이름	자료형	설명
Id	String	집행부가 동작하는 하드웨어 노드의 식별자
State	IMS_ImsState	집행부의 상태 • IMS_INITIAL • IMS_LEAD • IMS_FOLLOW

[정의] 생명신호 메시지의 우선순위

생명신호 메시지의 우선순위는 전송자의 동작 상태가 리더 상태일 경우 가장 높으며, 동작 상태로 구분

할 수 없을 때는 더 작은 식별자를 갖는 경우가 높다.

이렇게 정의된 우선순위는 리더를 선정할 때 사용된다. 모든 집행부들은 자신보다 높거나 같은 우선순위를 갖는 생명신호 메시지만을 생명신호로 받아들인다. 자신보다 높은 우선순위를 갖는 생명신호 메시지를 전송하는 집행부가 존재하는 경우에는 감시자 역할을 수행하는 집행부 팔로워로 동작한다.

다만, 팔로워로 동작한다는 것이 더 이상 생명신호 메시지를 보내지 않는다는 것은 아니다. 리더로 선정된 집행부가 더 이상 동작하지 않을 때를 대비하여 팔로워도 계속 생명신호 메시지를 전송하여야 한다. 하지만 모든 집행부들이 계속 생명신호 메시지를 전송하는 것이 효율적이지는 않다. 따라서 집행부의 동작에 크게 영향을 미치지 않으면서도 효율성을 높이기 위해 집행부가 와치독(Watchdog) 형태로 동작하도록 하였다. 와치독 형태로 동작하며 주기적으로 생명신호 메시지를 전송하는 집행부 리더 선정 알고리즘은 다음과 같다.

[리더 선정 알고리즘]

① 실행과 동시에 집행부의 와치독 타이머가 동작한다.
② 와치독 타이머의 타임아웃이 발생되면 집행부는 식별자 및 동작 상태 정보가 담긴 생명신호 메시지를 전송한다(broadcast).
③ 생명신호 메시지를 전송한 이후에는 다시 와치독 타이머를 동작시킨다.
④ ①~③ 과정 중에 다른 집행부로부터 전달된 생명신호 메시지를 수신하면 자신의 생명신호 메시지와의 우선순위를 비교한다.
⑤ 만일 다른 집행부의 생명신호 메시지의 우선순위가 자신의 생명신호 메시지보다 낮은 경우에는 이를 무시하고 ①~③의 과정을 반복한다.
⑥ 만일 다른 집행부의 생명신호 메시지의 우선순위가 자신의 생명신호 메시지보다 높은 경우에는 집행부 팔로워로 전향한 뒤 ①~③의 과정을 반복한다. 단, 와치독 타이머의 타임아웃 시간을 두 배로 설정한다.
⑦ 생명신호 메시지를 세 번 전송하는 동안 자신보다 높은 우선순위를 갖는 집행부를 발견하지 못한 경우에는 집행부 리더로 전향하고 ①~③의 과정을 반복한다.

리더 선정 알고리즘은 크게 생명신호 메시지 전송과 생명신호 메시지 수신 과정을 기반으로 동작하는데 이를 그림으로 표현하면 Fig. 2와 같다. Fig. 2의 왼쪽 부분은 각각의 집행부에서 와치독 타이머가 타임아웃이 되었을 때 생명신호 메시지를 전송하는 과정을 UML (Unified Modeling Language)의 액티비티 다이어그램(activity diagram)으로 표현한 것이다. Fig. 2의 오른쪽 부분은 각각의 집행부가 생명신호 메시지를 수신하는 과정을 액티비티 다이어그램으로 표현한 것이다. Fig. 3은 Fig. 2의 ToFOLLOW와 ToLEAD 상태에서의 세부 활동을 액티비티 다이어그램으로 표현한 것이다.

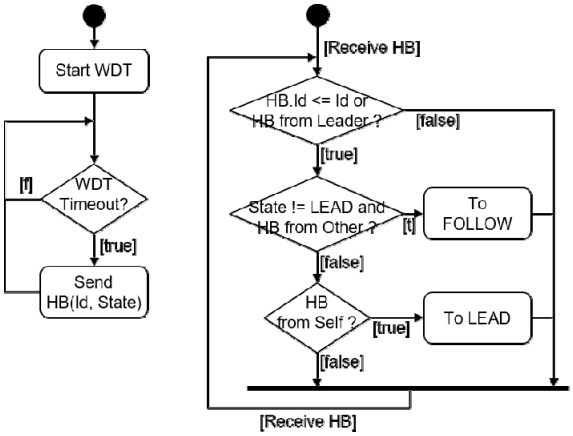


Fig. 2. Activity diagram for leader selection

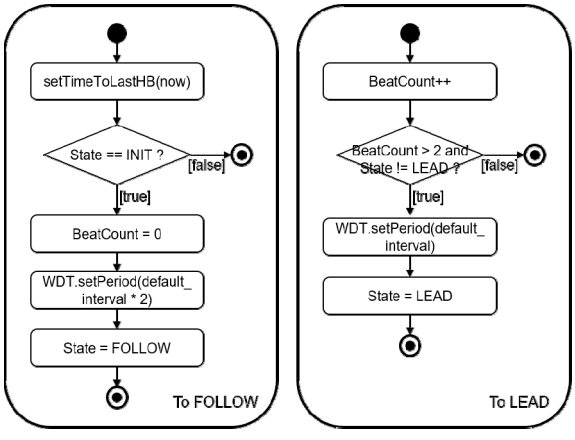


Fig. 3. Detailed activities in ToFOLLOW & ToLEAD states in Fig. 2

위의 알고리즘에서는 집행부 리더가 선정된 직후부터 집행부 팔로워는 와치독 타이머의 타임아웃이 발

생되지 않기 때문에 특정 시점부터 무의미하게 전송되는 생명신호 메시지가 줄어든다. 집행부 리더가 동작하지 않는 경우에만 팔로워의 와치독 타이머에서 타임아웃이 발생하여 새로운 집행부 리더를 선정하는 과정이 수행된다. 따라서 본 연구의 리더 선정 알고리즘은 통신 채널의 효율성을 높이면서도 집행부 리더를 선정할 수 있다.

4. 통합 통제 시스템 설계 및 구현

4.1 집행부 설계 및 구현

본 연구에서 제안하는 집행부는 Fig. 4와 같은 구조 및 동작 체계를 가지며, 다섯 가지의 주요 요소로 구성된다.

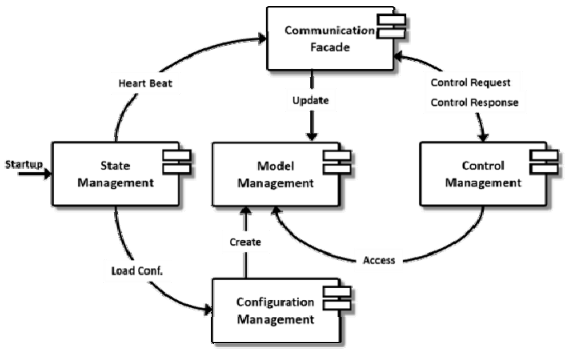


Fig. 4. Components of an executive and its operation mechanism

- 정보 모델 관리자(Model management) : 관리 대상 시스템의 소프트웨어 및 하드웨어에 대한 상태 및 실행 정보를 관리한다.
- 집행부 상태 관리자(State management) : 집행부의 동작 상태를 관리하며 리더 선정을 위한 생명신호 메시지를 생성한다.
- 설정 관리자(Configuration management) : 통합 통제를 수행하기 위해 필요한 시스템 구성 정보를 XML 파일로부터 읽어 오는 역할을 수행한다.
- 통신 파사드(Communication Facade) : 에이전트 등과 같은 외부 요소와의 통신을 담당한다.
- 제어 관리자(Control management) : 외부 요소의 요청과 관리 대상의 이상 감지 등을 통해 관리 대상 소프트웨어의 상태를 제어하는 역할을 담당한다.

본 논문의 통합 통제 시스템은 정보 모델을 기반으로 제어와 모니터링을 수행한다. 정보 모델은 하드웨어의 플랫폼이나 응용 소프트웨어의 종류에 관계없이 필요한 정보를 추출하여 추상적으로 관리한다. 정보 모델 관리자에서 관리하는 정보 모델의 종류는 Table 2와 같다.

Table 2. Managed information model

정보 모델	설명
Software System	• 소프트웨어로 구성된 시스템을 표현하기 위한 요소 • 하위 요소로 Software System, Application, ESE를 가짐
Application	• 특정 기능을 수행하는 다수의 소프트웨어 그룹을 표현하기 위한 요소 • 하위 요소로 Application, Redundancy Group, ESE를 가짐
Redundancy Group	• 다중화 요소를 표현하기 위한 요소 • 하위 요소로 ESE를 가짐
ESE (Executable Software Element)	• 단위 기능을 수행하는 소프트웨어를 표현하기 위한 요소 • 소프트웨어가 탑재되는 하드웨어 노드의 정보를 포함함
Host	• 하드웨어 시스템을 구성하는 단위 하드웨어 노드를 표현하기 위한 요소 • 탑재되어 있는 소프트웨어의 목록 정보를 포함함

집행부 상태 관리자는 (1) 집행부 초기화 수행, (2) 집행부 상태 관리, (3) 집행부 리더 선정을 위한 생명신호 메시지 송신 등의 기능을 수행한다. 집행부 초기화 작업은 설정 파일을 바탕으로 관리 대상의 정보 모델을 생성하는 작업과 외부 요소와의 통신을 위한 통신 초기화 작업으로 나뉜다. 각각은 설정 관리자와 통신 파사드를 통해 수행된다. 집행부는 Table 3과 같이 세 가지 상태를 갖는다.

집행부의 상태는 Fig. 5와 같은 전이 관계를 갖는다. 집행부 상태 관리자는 세 가지 상태와 이들 간의 상태 전이를 통해 집행부의 상태를 관리하며 IMS_LEAD와 IMS_FOLLOW에 따라 집행부의 일부 기능을 활성화 혹은 비활성화 한다.

Table 3. States of an executive

상태	설명
IMS_INITIAL	집행부가 처음 실행될 때의 상태, 아직 리더 및 팔로워 선정이 진행되지 않은 상태
IMS_LEAD	집행부가 리더로 동작할 때의 상태
IMS_FOLLOW	집행부가 팔로워로 동작할 때의 상태

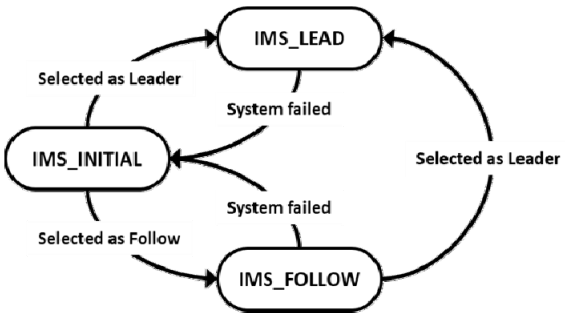


Fig. 5. State transition of an executive

4.2 통합 통제 시스템 검증

이번 절에서는 본 연구를 통해 개발된 통합 통제 시스템 프로토타입에 대한 검증 내용을 소개한다. 통합 통제 시스템은 다수의 하드웨어 노드 위에 분산 배치되어 동작하는 시스템이므로, 단일 컴퓨터에서 검증을 수행하기에 적합하지 않다. 그렇지만 본 연구에서는 도커(docker) 시스템을 이용하여 분산 환경을 모사하고 그 위에서 검증을 진행하였다. 검증 환경의 구성과 수행한 검증 시나리오는 세부 절에서 설명한다.

4.2.1 검증 환경

도커는 애플리케이션을 신속하게 구축/검증/배포할 수 있는 소프트웨어 플랫폼으로, 소프트웨어를 컨테이너(container)라는 표준화된 단위로 패키징하여 관리한다. 컨테이너를 구성할 때, 소프트웨어를 실행하기 위해 필요한 라이브러리, 시스템 도구, 코드 등을 포함한다. 이와 같은 패키징 정보는 도커파일(dockerfile)이라는 요소를 통해 명세할 수 있다.

통합 통제 시스템을 검증하기 위해서는 시스템이 다수의 하드웨어 노드 위에 분산 배치되어 상호작용하는 것을 확인하여야 한다. 도커파일은 하나의 노드에 해당되는 컨테이너만을 생성할 수 있기 때문에 본 연구의 검증에서 원하는 다수의 하드웨어 노드를 모사할 수 없다. 이런 상황을 극복하기 위해 도커 컴포즈(docker compose) 기법을 사용한다. 도커 컴포즈는 별도의 스크립트를 통해 다수의 컨테이너를 한꺼번에 실행시킨다.

Fig. 6은 여러 컨테이너를 탑재한 도커 환경을 보여준다. 그림에서 보는 바와 같이 컨테이너는 두 가지 형태인데 하나는 DDS 통신 모듈, IMS Executive 모듈, IMS Agent 모듈, 여러 응용 프로그램에 해당하는 ESE 모듈들로 구성되고, 다른 하나는 이전 구성에서 IMS Executive 모듈을 제외한 형태이다. 이러한 컨테이너의 구성은 Fig. 1의 통합 통제 프레임워크의 전체 구성에 대응된다고 볼 수 있다. 통합 통제 시스템의 전체 테스트 환경은 도커 컴포즈를 통해 이렇게 구현될 수 있다.

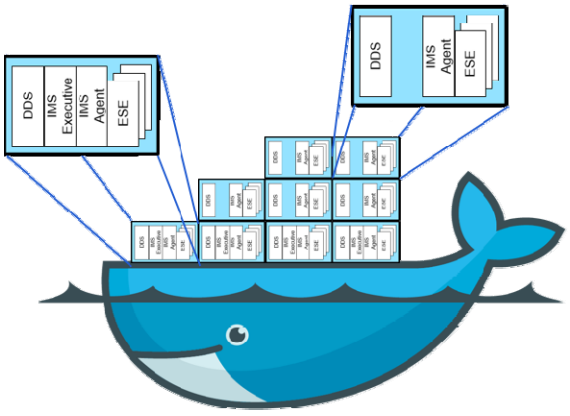


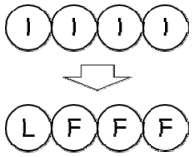
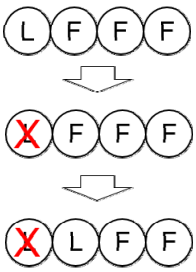
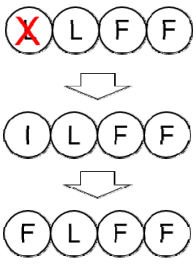
Fig. 6. Docker environment with multiple containers

4.2.2 검증 시나리오

본 연구에서는 도커 컴포즈를 이용하여 다수의 하드웨어 노드가 존재하는 시스템을 모사하여 검증을 진행하였다. 통합 통제 시스템의 기능을 검증하기 위해 각 요구사항 별로 세부 시나리오를 작성하여 검증을 수행하였는데 이 논문에서는 Table 4와 같이 리더 선정, 리더 전환, 재진입에 대한 시나리오를 다룬다.

Table 5는 Table 4의 리더 선정 검증 시나리오에 따라 시스템을 수행한 결과를 정리한 표이다. 표에는 집행부의 상태에 변화가 발생하였을 때 그것을 쉽게 알 수 있도록 표시가 되어 있다. Table 5의 수행 결과를 통해 초기 수행에는 집행부의 팔로워가 먼저 결정된 다음

Table 4. Verification scenarios

시나리오	설명	개념도
리더 선정	두 개 이상의 집행부가 실행될 경우, 하나는 LEAD 상태로 동작하고, 나머지는 FOLLOW 상태로 동작한다.	
리더 전환	집행부 리더가 비정상 종료될 때, FOLLOW 상태로 동작하던 집행부 중 하나는 LEAD 상태로 동작하고, 나머지는 FOLLOW 상태로 동작한다.	
재진입	리더 전환 후 비정상적으로 종료된 집행부가 실패 원인을 수정한 후 다시 실행을 시작하면 그 집행부는 FOLLOW 상태로 동작한다.	
I: Initial, L: Lead, F: Follow 상태를 나타내며, X 표시는 집행부가 비정상 종료된 상황을 의미함.		

리더가 결정된다는 것과 여러 집행부 중 오직 하나만 리더로 선정된다는 사실을 알 수 있다. 여기서 주기는 1초로 다소 길게 느껴질 수 있으나 그것은 조정 가능한 값이다. 그런데 주기 시간 설정은 설계 시점이 아니라 실제 시스템을 운영하는 시점에 정하는 것이 바람직하다. 이 시스템은 기본적으로 네트워크로 연결된 분산 환경에서 운영되므로 각 노드의 처리 속도, 네트워크의 속도 등이 복합적으로 고려되어야 한다. 만일 실제 환경을 고려하지 않고 그 값을 짧게 지정하면 네트워크에서의 단순 지연이 리더의 실패로 간주되어 리더 선정 작업이 진행되고 이로 인해 리더를 교체하려는 시도가 불필요하게 계속 발생할 수 있기 때문이다.

Table 6은 Table 4의 리더 전환 검증 시나리오의 수행 결과를 정리한 표이다. 표를 통해서 리더가 비정상적으로 종료되었을 때 팔로워 중에서 하나가 리더로 전환됨을 볼 수 있다. 주목할 사항은 팔로워 중에서 리더로 전환된 집행부만 Lead 상태로 상태가 전환되고 나머지는 여전히 팔로워로 있으므로 상태 변화가 발생하지 않아서 상태 변화 메시지를 출력하지 않는다는 것이다.

Table 7은 Table 4의 재진입 검증 시나리오의 수행 결과를 정리한 표이다. 비정상적으로 종료된 리더가 실패 원인을 해결한 후 다시 실행을 시작하면 Initial 상태에서 시작하지만 리더 전환 시나리오에 의해 선택된 리더가 이미 존재하므로 바로 팔로워로 전환된다. 다른 집행부들은 상태 변화가 발생하지 않으므로 상태 변화 메시지를 출력하지 않는다.

Table 5. Execution results for leader selection scenario

Log(Executive-1)	Log(Executive-2)	Log(Executive-3)	Log(Executive-4)
08:07:11| Start	08:07:11| Start	08:07:11| Start	08:07:11| Start
08:07:11| HB Gen Start	08:07:11| HB Gen Start	08:07:11| HB Gen Start	08:07:11| HB Gen Start
08:07:13| Send HB	08:07:12| Send HB	08:07:12| Send HB	08:07:12| Send HB
08:07:14| Send HB	08:07:13| Send HB	08:07:13| Listen HB	08:07:13| Listen HB
08:07:15| Send HB	08:07:14| Listen HB	08:07:13| Follow	08:07:13| Follow
08:07:15| Lead	08:07:14| Follow	08:07:14| Listen HB	08:07:14| Listen HB
08:07:16| Send HB	08:07:15| Listen HB	08:07:15| Listen HB	08:07:15| Listen HB
08:07:17| Send HB	08:07:16| Listen HB	08:07:16| Listen HB	08:07:16| Listen HB
이하 생략	이하 생략	이하 생략	이하 생략

Table 6. Execution results for leader switching scenario

Log(Executive-1)	Log(Executive-2)	Log(Executive-3)	Log(Executive-4)
이전 생략	이전 생략	이전 생략	이전 생략
08:51:50| Send HB	08:51:50| Listen HB	08:51:50| Listen HB	08:51:50| Listen HB
08:51:51| Send HB	08:51:50| Follow	08:51:50| Follow	08:51:50| Follow
08:51:51| Lead	08:51:51| Listen HB	08:51:51| Listen HB	08:51:51| Listen HB
08:51:52| Send HB	08:51:52| Listen HB	08:51:52| Listen HB	08:51:52| Listen HB
중간 생략	중간 생략	중간 생략	중간 생략
08:52:08| Send HB	08:52:08| Listen HB	08:52:08| Listen HB	08:52:08| Listen HB
08:52:09| Send HB	08:52:09| Listen HB	08:52:09| Listen HB	08:52:09| Listen HB
x	08:52:11| Send HB	08:52:11| Send HB	08:52:11| Send HB
x	08:52:12| Send HB	08:52:12| Listen HB	08:52:12| Listen HB
x	08:52:13| Send HB	08:52:13| Listen HB	08:52:13| Listen HB
x	08:52:13| Lead	08:52:14| Listen HB	08:52:14| Listen HB
x	08:52:14| Send HB	08:52:15| Listen HB	08:52:15| Listen HB
	이하 생략	이하 생략	이하 생략

Table 7. Execution results for reintroduction scenario

Log(Executive-1)	Log(Executive-2)	Log(Executive-3)	Log(Executive-4)
이전 생략	이전 생략	이전 생략	이전 생략
08:52:09| Send HB	08:52:09| Listen HB	08:52:09| Listen HB	08:52:09| Listen HB
x	08:52:11| Send HB	08:52:11| Send HB	08:52:11| Send HB
x	08:52:12| Send HB	08:52:12| Listen HB	08:52:12| Listen HB
x	08:52:13| Send HB	08:52:13| Listen HB	08:52:13| Listen HB
x	08:52:13| Lead	08:52:14| Listen HB	08:52:14| Listen HB
x	08:52:14| Send HB	08:52:15| Listen HB	08:52:15| Listen HB
	중간 생략	중간 생략	중간 생략
08:57:21| Start	08:57:21| Send HB	08:57:21| Listen HB	08:57:21| Listen HB
08:57:21| HB Gen Start	08:57:22| Send HB	08:57:22| Listen HB	08:57:22| Listen HB
08:57:22| Send HB	08:57:23| Send HB	08:57:23| Listen HB	08:57:23| Listen HB
08:57:23| Listen HB	08:57:24| Send HB	08:57:24| Listen HB	08:57:24| Listen HB
08:57:23| Follow	08:57:25| Send HB	08:57:25| Listen HB	08:57:25| Listen HB
이하 생략	이하 생략	이하 생략	이하 생략

5. 결 론

다양한 이기종 시스템이 통합되어 운용되는 대규모 함정 전투체계에서 하부 시스템들을 효과적으로 관리하기 위해서는 표준화된 통합 관리 시스템이 필요하다. 이에 본 연구에서는 함정 전투체계를 위한 통합 통제 시스템의 프로토타입에 대해 설계하고 구현하였다. 이 시스템은 추상화된 정보 모델을 이용하기 때문에 이기종 환경에 영향 받지 않고 함정 전투체계의 모든 시스템들에 대한 통합 관리를 수행할 수 있다. 그런데 이런 통합 통제 시스템의 오작동이나 실패는 전체 함정 전투체계를 무력화 시킬 수 있기 때문에 통합 통제 시스템은 함정 전투체계의 다른 응용 소프트웨어에 비해 그 중요도가 더욱 높다 할 수 있다. 본 논문에서는 함정 전투체계를 실제 운영하는 관점에서 고려해야 할 사항 중 하나인 함정 전투체계의 안정적이고 올바른 동작을 보장하기 위한 방안을 제시하였다. 이를 위해, 통합 통제 시스템의 집행부를 하나의 리더와 여러 개의 팔로워로 구성되는 형태의 다중화 방안을 제시하였고, 통합 통제 시스템을 다중화 함으로 인해 새롭게 등장하는 문제인 리더 선정 알고리즘을 개발하였다. 리더 선정 알고리즘은 다중화 형태로 동작하는 집행부 중 통합 통제 기능을 수행할 주 모듈을 선정하는 작업을 수행한다. 또한, 통합 통제 시스템의 검증을 위해 분산 시스템을 모사한 테스트 환경을 구축하였으며, 여러 검증 시나리오를 통해 통합 통제 시스템의 고장 감내 동작 및 이를 위한 리더 선정 알고리즘의 정확성을 검증하였다. 향후에는 정형 검증 기법을 적용해서 리더 선정 알고리즘의 정확성에 대한 추가 검증을 수행할 예정이며, 충분한 검증 노력을 통해 발견된 이슈를 해결함으로써 통합 통제 시스템의 프로토타입에 대한 보완 작업을 수행할 예정이다.

후 기

본 연구는 국방과학연구소의 지원으로 수행되었습니다(계약번호 UD170004DD).

References

[1] S. J. Ko and D. H. Park, “An Examination on Overseas Technology Trend and Domestic Development Pattern of the Naval Combat Management System,” Journal of the Korea Association of Defense Industry Studies, Vol. 16, No. 2, pp. 237-258, 2009.

[2] B. K. Min, H. S. Kim, S. H. Kuk, C. S. Kim, and W. G. Han, “Development of Information Model based Integrated Management and Monitoring System for Naval Ship Combat System with Heterogeneous Distributed Environments,” Journal of the Korea Institute of Military Science and Technology, Vol. 15, No. 4, pp. 381-389, 2012.

[3] J. W. Lee, “Development of Message Define & Management System based on Distributed Processing Environment for Naval Combat Systems,” KIISE Tran. on Computing Practices, Vol. 23, No. 12, pp. 670-676, 2017.

[4] H. F. R. Arciszewski, T. E. de Greef, and J. H. van Delft, “Adaptive Automation in a Naval Combat Management System,” IEEE Tran. on System, Man, and Cybernetics, Vol. 39, No. 6, pp. 1188-1199, 2009.

[5] L. Bass, P. Clements, and R. Kazman, “Software Architecture in Practice,” 3rd Ed., Addison-Wesley, pp. 91-92, 2013.

[6] W. Zhao, P. M. Melliar-Smith, and L. E. Moser, “Low Latency Fault Tolerance System,” Electrical Engineering & Computer Science Faculty Publications, 264, pp. 1-26, 2012.

[7] W. Lu, Y. Yang, L. Wang, W. Xing, X. Che, and L. Chen, “A Fault Tolerant Election-based Deadlock Detection Algorithm in Distributed Systems,” Software Quality Journal, 26, pp. 991-1013, 2018.

[8] L. Lamport, “The Part-Time Parliament”, ACM Tran. on Computer Systems, Vol. 16, No. 2, pp. 133-169, 1998.