JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 30 No. 1, pp. 149-161, January 2025
https://doi.org/10.9708/jksci.2025.30.01.149

A Study on Method of Monitoring the Status of Interfacing Equipment in Submarine Combat System based on Virtualization

Ye-Jun Jang*

*Engineer, Naval R&D Center, Hanwha System, Gumi, Korea

[Abstract]

Recent advancement in hardware performance, reductions in the system weight of naval vessels, and improvements in operability and security have led to increasing demands on combat systems. As a result, research on the application of virtualization to submarine combat systems, exploring both methods and feasibility, has been actively conducted. This paper proposes and validates a method for monitoring the status of interfacing equipment by applying the observer pattern to a virtualized submarine combat system. Existing combat systems operate applications on fixed nodes and use static ID to check the availability status of corresponding equipment. However, in the case of virtualized combat systems, dynamic execution of interface applications on non-fixed virtual machines is allowed to ensure operational efficiency. By verifying the functionality of a status monitoring method in an environment where the execution location of interface applications is dynamically managed, the feasibility of applying this new interfacing equipment status monitoring method to virtualized submarine combat systems is confirmed.

▸ **Key words**: Virtualization, Submarine Combat Management System, System Management, Design Pattern, Observer Pattern

[요 약]

최근 하드웨어의 성능 발달과 함정의 체계중량 감소, 운용성 및 보안성 향상 등 전투체계 요구사항 증가로 인해 잠수함 전투체계에 가상화를 적용하는 방안과 가능성 연구가 활발히 이루어지고 있다. 이에 본 논문에서는 가상화 기반 잠수함 전투체계에 옵저버 패턴(Observer Pattern)을 적용한 연동 장비의 상태감시 방법에 대해 제안하고 이를 검증하였다. 기존의 전투체계는 고정된 노드에서 각 연동단 응용이 실행되며, 그 노드의 정적 ID를 통해 대응되는 장비의 가용 상태를 확인하였다. 그러나 가상화 기반 전투체계에서는 함정 운용 효율성을 확보하기 위해 고정되지 않은 가상머신에 동적으로 연동단 응용을 실행할 수 있게 되었다. 이처럼 연동단 응용의 실행 위치가 동적으로 관리되는 환경에서 연동 장비의 상태를 감시하는 방법의 동작성을 검증함으로써, 가상화 기반 잠수함 전투체계에서 새로운 연동 장비 상태감시 방법의 적용 가능성을 확인하였다.

▸ **주제어**: 가상화, 잠수함 전투체계, 체계상태감시, 디자인 패턴, 옵저버 패턴

• First Author: Ye-Jun Jang, Corresponding Author: Ye-Jun Jang
*Ye-Jun Jang (yejunjang@hanwha.com), Naval R&D Center, Hanwha System
• Received: 2024. 11. 28, Revised: 2024. 12. 19, Accepted: 2024. 12. 27.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

해상이라는 특수한 환경에서 작전을 수행하는 함정을 운용하기 위해서는 함정에 탑재된 탐지 장비, 무장, 항해 지원 장비 등의 다양한 체계 정보의 통합이 필요하다. 모든 장비로부터 취득한 정보의 취합과 자동화를 통해 함정을 운용하는데 핵심 역할을 하는 것을 함정 전투체계라고 한다[1]. 함정 전투체계는 간단한 기능과 성능, 안정성, 생존성 등의 단순 요구사항 위주의 초기 전투체계에서부터 지속적인 요구사항의 증가로 전투체계 효용률, 체계의 중량 감소, 승조원의 생활 공간 확보, 운용성 및 보안성 향상 등의 요구사항을 만족하기 위해 꾸준히 발전해 왔다. 특히, 하드웨어의 비약적인 성능 발달로 제한된 자원의 효율적인 운용을 가능하게 하는 클라우딩 컴퓨팅 기술이 가능해지면서 이를 이용한 가상화 기술 적용을 위한 함정 전투체계 연구가 활발히 이루어지고 있다[2].

가상화가 적용되지 않은 기존의 잠수함 전투체계에서는 SBC(Single Board Computer) 단위의 고정적인 노드에 정해진 연동단 응용이 실행되고, 해당 응용이 장비와의 연동을 통해 필요 데이터를 취득한다. 이러한 방식은 특정 노드에 장애가 발생할 경우, 해당 노드와 연동되는 장비는 SBC를 교체하지 않으면 운용이 불가하다. 이는 함정이 정상적인 기능을 유지하는 확률인 가용도를 낮추고 고장 발생 시 체계를 정비하여 성능을 원상복구하는 시간인 정비도를 높이는 요인으로 작용한다[3]. 실제 함정 운용에 사용되는 SBC의 평균 고장 시간인 MTBF(Mean Time Between Failure)는 215,217시간으로 비교적 많은 시간이나 100개 이상의 SBC로 구성된 함정 전투체계의 운용환경을 고려하면 고장으로 인한 SBC의 교체는 필수적으로 발생한다. SBC에 고장이 발생하면 정상적인 SBC로 대체하기 위해서 OS 설치, IP 설정 등 전체적인 상황을 반영하였을 때 약 8시간이 소요된다. 또한, 가용한 SBC가 없다면 수리 및 신규 구매 비용에 많은 비용과 시간이 추가로 투입되므로 하드웨어 변경에 의한 유지보수 비용이 증가한다는 문제가 있다.

전투체계의 가상화 적용과 함께 AMSM(Application Management, System Monitoring for Naval CMS) 기반의 분산 체계관리 적용을 통해 연동 응용을 고정된 위치의 노드에 실행하지 않고 유후한 가상머신에 동적으로 실행함으로써 해당 문제를 해결할 수 있다[4-5].

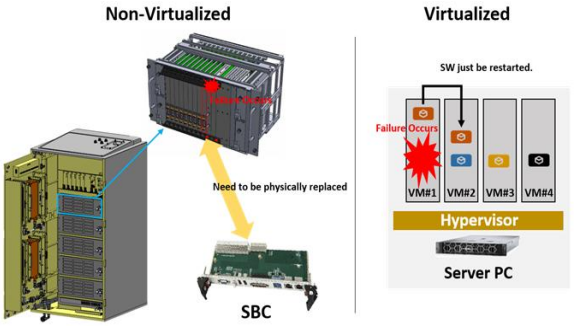

Fig. 1. Comparison of failure handling in Non-Virtualized System vs. Virtualized Systems

함정의 안정적인 임무 수행을 위해 체계의 상황을 분석하는 체계상태감시 방법은 함정 전투체계와 함께 꾸준히 발전해왔다. 단순히 장비의 LED 색상을 확인하는 방법에서부터 센서, 무장 체계의 연동 상태, 하드웨어 장비 등의 함정을 운용하는데 필요한 장비들의 상태를 통합된 전시화면을 통해 확인하는 형태로 발전하였다. 이를 통해 고장 발생의 위치, 유형을 한눈에 실시간으로 확인할 수 있게 되었으며 고장에 대한 빠른 조치가 가능하게 되었다. 이처럼 체계상태감시는 함정의 운용에 필요한 유지보수의 효율성을 크게 향상시켰다.

따라서 본 논문에서는 유후한 가상머신에 동적으로 연동단 응용이 실행되는 가상화 전투체계 환경에서의 체계상태감시 방법에 대해 제시하고 해당 기능의 동작성을 검증하였다. 또한, 신규 기능이 추가될 경우 객체의 독립적인 수정이 가능한 GoF(Gaong of Four) 패턴 중 옵저버 패턴(Observer Pattern)을 적용하여 패턴 적용의 이점에 대해 확인하였다.

본 논문은 다음과 같이 구성하였다. 2장에서 차세대 함정 전투체계에 대해 확인하고, 가상화 기반의 전투체계 연구 동향에 대해 알아보았다. 3절에서는 새롭게 제안하는 연동장비의 상태감시 방법에 대해 블록도 및 순서도를 통해 설명하였다. 4절에서는 로그 분석 을 통한 동작성 검증과 신규 연동 장비가 추가될 경우의 개발 소요를 확인하여 본 제안의 이점을 확인하였다. 마지막 5절에서 결론과 함께 앞으로의 방향에 대해 서술하며 논문을 마무리하였다.

II. Preliminaries

1. Next Generation Naval Combat System

아래 Fig. 2.는 서버 기반 가상화 방법인 VDI(Virtual Desktop Infrastructure) 기술 적용과 가상화 기반의 체계 통합을 통한 차세대 함정전투체계의 도식화이다.

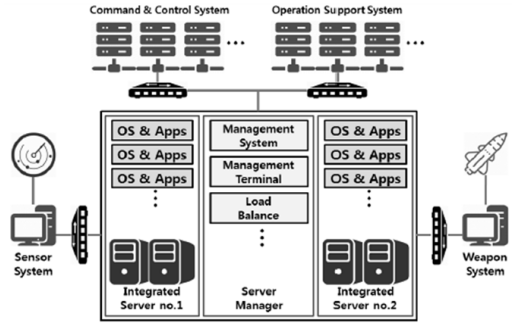

Fig. 2. Structure of applied virtualization for naval CMS

해당 모델은 크게 통합서버와 서버 매니저로 구성된다. 통합서버는 기존 체계별로 별도 구축된 서버 및 데스크톱의 컴퓨팅 자원을 대처하기 위한 하드웨어 풀이며 체계별로 운용되던 서버 및 저장장치를 통합하여 서버 가상화를 통해 구현하고 운용한다. 서버 매니저는 가상화 환경에서 통합서버 관리 기능을 수행하며 다양한 체계에 대한 가상머신을 위한 운영체제, 응용 관리, 서버 구동 및 부하 관리 및 가용 컴퓨팅 자원 관리를 통해 통합서버를 관리한다.

기존의 센서 및 무장은 통합서버의 가상머신과 연동하고 체계 내의 데이터 교환은 해당 가상머신 간에 이루어진다. 또, 지휘통제 및 작전지원 체계의 데스크톱 단말기는 규모의 최소화를 통해 통합서버에서 제공하는 가상 데스크톱을 통하여 기존의 기능을 수행한다. 그리고 적의 공격 등으로 인한 서버의 물리적 손상에도 기능 유지를 위해 통합서버는 함정 내부 분리 설치와 우선순위를 고려한 가상머신의 신속한 운용을 통해 운용 고가용성을 유지하도록 한다[6].

2. The research trends on Virtualization for Naval Combat System

국내 함정전투체계 분야는 가상화 기술 적용에 따른 기술적 충격을 완화하기 위해 서버 가상화, 데스크톱 가상화, 클라우드 형태로의 단계적으로 적용 중이다. 현재 서버 가상화에서 데스크톱 가상화로 가상화 기술 적용단계가 진화하면서 이를 위한 가상화 솔루션의 전투체계 적용성 및 신뢰성 확보를 위한 검증이 활발히 진행되고 있다[7].

2.1 Virtualization for Submarine Combat System

아래 Fig. 3.과 Fig. 4.는 OOO급 잠수함 전투체계에 HPC(High Performance Computing) 기반의 가상화 적용 시 컴퓨터 노드 수의 차이를 나타낸다. 총 13개의 컴퓨터 노드 수량이 5개로 감소함으로써 하드웨어 점유 공간 및 체계 중량 감소 가능성을 확인하였다[8].

기존 OOO급 잠수함의 전투체계는 하나의 SBC 자원(CPU/메모리)을 10% 미만으로 사용하여 100개 이상의 SBC로 구성되었다[9]. HPC 기반의 가상화 전투체계 적용은 자원의 활용성을 높여 실질적인 하드웨어 공간을 줄여 체계 중량 및 공간을 확보함으로써 더 나은 잠수함 운용 환경을 마련할 수 있다.

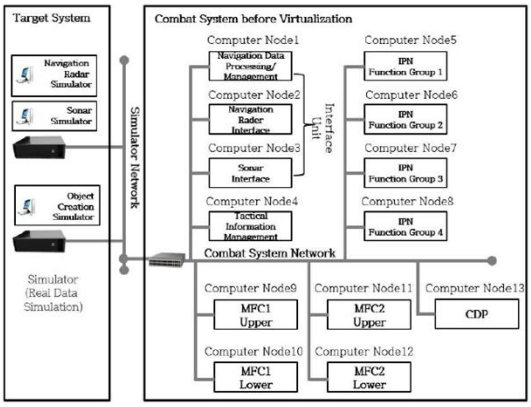

Fig. 3. Combat system configuration and number of computer node before virtualization

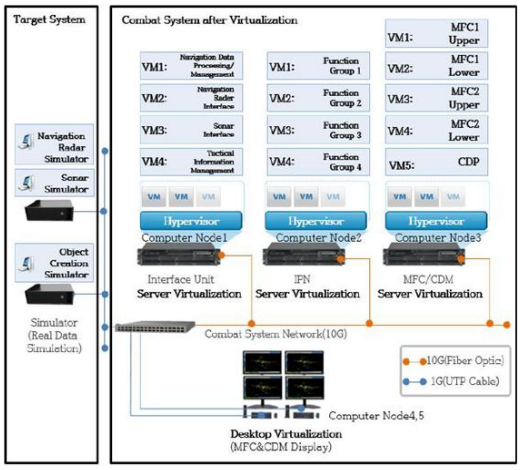

Fig. 4. Combat system configuration and number of computer node after virtualization

2.2 Various virtualization tools based on Hypervisor

하이퍼바이저는 가상화의 핵심인 소프트웨어로써 물리적인 하드웨어에 설치되어 가상머신과 하드웨어를 관리하는 역할을 한다. 하이퍼바이저 기반의 아키텍처는 Bare-metal과 Hosted로 분류된다. 두 타입의 차이는 호스트 서버의 존재 유무이다. Bare-metal 구조는 호스트 운영체제 없이 하이퍼바이저가 직접 하드웨어를 관리한다. 이러한 Bare-metal 구조의 특징은 호스트 운영체제에서 하이퍼바이저가 실행되어 추가적인 오버헤드가 필요한 Hosted 구조보다 실시간으로 많은 데이터가 운용되는 분산체계 기반의 전투체계에 더 적합한 것으로 판단된다[2].

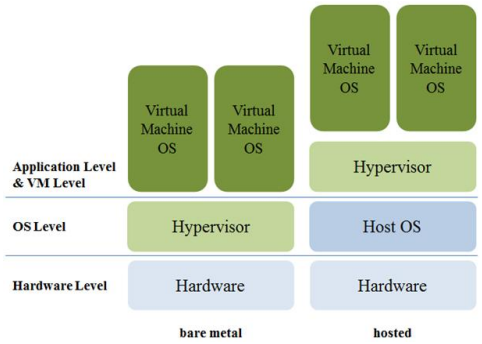

Fig. 5. Architecture of Hypervisor

하이퍼바이저(Hypervisor) 기반 중 Bare-metal 구조의 가상화 도구는 VMware VSphere/ESXi, Microsoft Hyper-V, Citrix Xen, RedHat KVM이 대표적이다[10]. 이 중 VMware가 제공하는 기능은 하이퍼바이저 기반의 가상화 적용으로 발생할 수 있는 여러 문제점을 해결해 준다. VMware는 빈번한 I/O 교환으로 발생하는 CPU 오버헤드를 줄이기 위한 VM DirectPath I/O 기능을 제공한다. 또한, 호스트와 게스트인 다수의 가상머신 간의 문제 발생 시 신속하게 복구하는 무중단서비스 기능은 여러 가상화 도구 중에 VMware만이 제공함으로써 전장이라는 특수한 환경에서 운용되는 함정 전투체계의 무중단성을 보장할 수 있는 가상화 도구로써 유일하다. DDS 기반의 전투체계 환경에서 메시지 응답 속도 측정에서도 VMWare가 가장 빠른 것으로 확인되었다[6]. 이에 본 논문에서도 VMWare의 ESXi를 이용한 가상화 환경에서 연구를 진행하였다. 이를 통해 가상화 적용으로 인해 발생할 수 있는 기술적 문제에 대한 고민을 배제하고 새롭게 제안하는 연동 상태감시 방법의 기능 검증에 초점을 맞춰 연구를 진행하였다.

2.3 Virtualization applied to Naval Combat System

함정 전투체계에 가상화 기술은 OO급 수상함 전투체계에 최초 적용되었다. 수상함 전투체계 일부 기능인 전시처리 장치에 데스크탑 가상화를 적용하여 사용자가 함정을 운용하기 위한 데스크탑 구성 및 설정을 사용자화할 수 있는 기능이 적용되었다[8-9]. 또한, 앞선 2절의 내용인 함정 전투체계의 가상화 연구 동향을 통해 전투체계에 가상화 기술을 적용하기 위한 가상화 솔루션의 성능 분석, 가상화 적용 가능성, 차세대 전투체계 컨셉 연구를 확인하였다. 이를 통해 함정 전투체계에 가상화 기술의 적용의 적합성을 명확히 확인할 수 있었으나 가상화 기술의 적용에 따라 변화된 환경에 대한 실질적인 응용 기능에 대한 연구가 필요하다고 판단하였다. 따라서 본 논문에서는 유동적인 가상머신에 연동단 응용이 실행되는 환경에서의 새로운 연동상태 감시 기능을 제안하고 확인함으로써 전투체계 가상화 적용에 따른 실질적인 응용 단위의 기능에 대해 검증하고자 한다.

3. Data Distribution Service(DDS)

DDS는 OMG(Object Management Group) 기구에서 제정한 미들웨어로 국방, 항공우주산업, 교통, 로봇 등 실시간성, 고신뢰성이 요구되는 분야에 적용된다. 분산 환경에서 네트워크 데이터 도메인에 자유로운 참여와 탈퇴가 가능하게 하여 노드 구성의 유연성을 극대화하고 발간(Publish)/구독(Subscribe) 기술을 기반으로 하여 실시간 데이터를 공유 및 배포하는 기능을 제공한다[11]. 이러한 기능은 응용 프로그램의 위치와 내부 구현과는 무관하게 상호 간 데이터 교환이 가능하므로 분산 환경에서의 응용 프로그램의 설계를 단순화할 수 있다[12]. 또한, 데이터 전송의 신뢰성, 실시간성 등 다양한 특성들을 보장하기 위해 22개의 QoS(Quality of Service) 정책이 표준화되어 있으며, 해당 QoS는 DDS에서의 데이터 단위인 토픽(Topic)의 크기와 주기 등에 따라 QoS 정책이 적용된다[13-14].

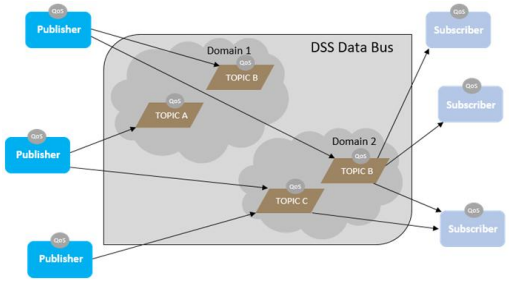

Fig. 6. DDS Overview

4. Application Management and System Monitoring for CMS Systems(AMSM)

AMSM은 DMTF(Distributed Management Task Force)에서 제정한 CIM(Common Information Model)을 확장한 통합 관리 표준이다. DDS를 제정한 OMG 기구에서 제정하였으며 CIM의 공통 모델을 기반으로 해군 함정 전투 관리 시스템에 맞게 확장하여 모델을 정의 한 것이다[4]. 본 논문의 잠수함 전투체계에는 XML(eXtensible Markup Language), DDS 모델로 AMSM가 구성되어있다. XML에 명시된 응용의 제어 및 감시 정책을 습득하고 이를 바탕으로 DDS를 통해 소프트웨어의 제어와 관리를 수행한다[5].

III. The Proposed Scheme

1. Overview of Existing System Status Monitoring Function Block

아래의 Fig. 7.은 기존의 체계상태감시의 전반적인 흐름을 블록도로 나타낸 것이다. 1.전투체계에 구성된 SBC 노드에는 고정적으로 실행되는 연동단 응용(Interface Control SW), 상태 보고 응용(Status Report SW)이 각 노드별로 탑재된 스크립트 파일에 의해 실행된다.

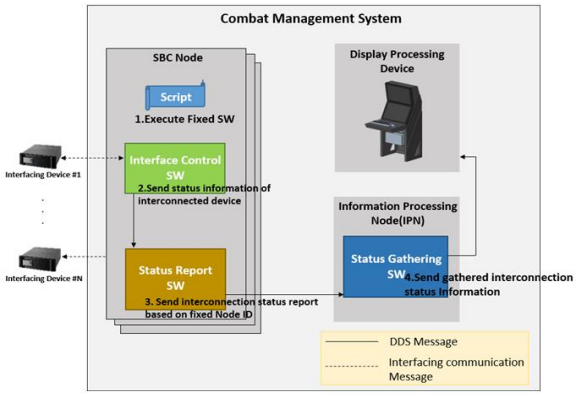

Fig. 7. Existing Block Diagram of existing status monitoring

2.연동단 응용은 장비의 연동상태를 상태 보고 응용으로 송신하고 3.해당 정보는 실행된 고정 노드 ID를 기준으로 정보처리장치(IPN)에 실행된 상태 취합 응용(Status Gathering SW)로 보고된다. 4.상태 취합 응용은 각 노드로부터 수신되는 정적 ID 기반의 상태 보고를 취합하고 전시 처리 장치로 송신함으로써 운용자가 체계상태를 감시할 수 있게 된다.

2. Proposal for new System Status Monitoring method

아래의 Table 1.은 체계상태감시를 위한 응용을 CSU(Computer Software Unit) 단위의 이름으로 설명한 것이다. 앞으로 기술할 내용의 이해를 돕기 위해 Table 1.을 통해 정의된 CSU 이름으로 새로운 제안을 설명하도록 한다. ICUSW는 연동단 응용을 포괄적으로 표현한 것이며 각 장비와 연동하는 실제 연동단 응용의 이름은 각자 달라진다.

Table 1. Description of CSU

No.	CSU Name	Description
1	NodeStatMgt	Status Report SW
2	ICUSW	Interface Control SW
3	SysStatMgt	Status Gathering SW
4	AMSServer	AMSM Server SW
5	AMSClient	AMSM Client SW

Table 1.의 AMSServer와 AMSClient는 가상화 환경에서 응용을 동적으로 실행하기 위한 CSU이다. 노드에 탑재된 스크립트에 의해 고정적인 응용이 실행되는 기존 방식과는 다르게, AMSServer와 AMSClient에 의해 동적인 응용 실행이 이루어진다. 해당 응용의 상세 동작은 본 논문에서 다루는 실질적인 내용이 아니므로 새로운 제안 방식 설명에서 간단히 다루도록 한다.

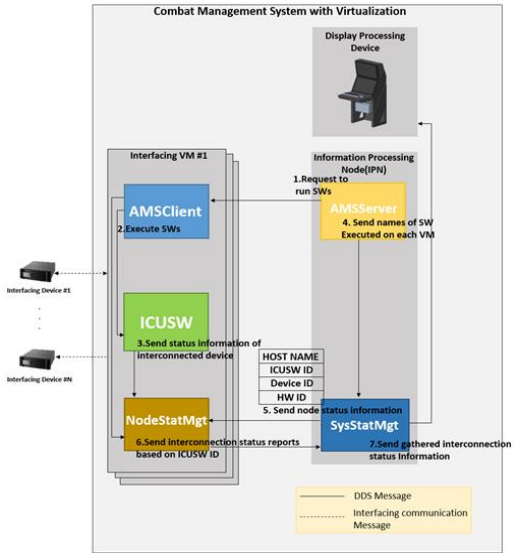

Fig. 8. Proposed Block Diagram of new status monitoring

위 Fig. 8.은 본 논문에서 제시하고자 하는 체계상태감시 방법의 블록도이다. 1.IPN의 AMSServer의 요청을 받아 각 가상머신에 실행된 2.AMSClient는 ICUSW와 NodeStatMgt를 실행한다. 3.실행된 ICUSW는 연동 장비와의 연동을 개시하고 연동 상태를 NodeStatMgt로 전송한다. 4.SysStatMgt는 AMSServer로부터 가상머신에 실행된 응용의 정보를 취득하여 5.가상머신의 이름, 해당 가상머신에 실행된 ICUSW의 ID, 연동 장비의 ID, 그리고 노드에 탑재되는 하드웨어 구성품 ID 정보를 NodeStatMgt로 송신한다. 6.해당 정보를 통해 NodeStatMgt는 상태 정보를 생성하여 SysStatMgt로 보고한다. 7.SysStatMgt는 취합된 상태보고 정보를 DDS 메시지로 송신한다.

해당 구현에 대한 상세 로직은 다음 장에서 클래스 다이어그램과 순서도를 통해 자세히 설명하였다.

3. Implementation & Design Pattern

아래 Fig. 9.와 Table 2.는 상태 보고 모듈인 NodeStatMgt의 클래스 구조이다.

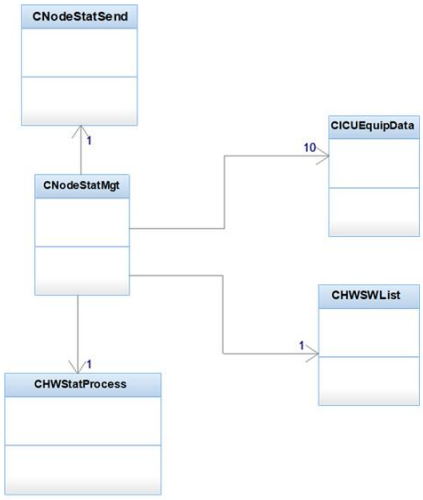

Fig. 9. Class Diagram of existing NodeStatMgt

Table 2. class description

No.	Class Name	Description
1	CNodeStatMgt	Object main class
2	CHWSWList	Get information for equipment items that require status report
3	CHWStatProcess	Collecting and managing the status of hardware
4	CICUEquipData	Manage the status of interfacing device.
5	CNodeStatSend	Send device status to SysStatMgt

본 논문에서 다루고자 하는 연동 장비의 상태를 관리하는 실질적인 클래스는 CICUEquipData 클래스이며 ICUSW로부터 연동 장비의 상태를 수신받아 연동 장비의 상태를 관리하는 역할을 한다.

아래의 Fig. 10.은 해당 기능을 유동적인 노드에 연동단 응용이 실행되는 가상화 전투체계 환경에서 CICUEquipData 클래스의 기능을 구현한 클래스 다이어그램이다. 해당 기능은 GoF의 디자인 패턴 중 행동 패턴에 해당하는 옵저버 패턴(Observer Pattern)을 적용하여 구현하였다. 옵저버 패턴은 어떤 주체(Subject)의 상태의 변화에 대해 주체 객체에 대해 의존성을 가진 옵저버(Observer)가 그 변화에 대해 통지받고 그에 따른 행동을 수행할 수 있도록 하는 특징을 가진 디자인 패턴이다[15].

이와 유사한 패턴으로는 이벤트 버스 패턴(Event Bus Pattern)과 발행-구독 패턴(Publish-Subscribe Pattern)이 있다. 세 개 패턴은 모두 이벤트 기반의 정보 갱신을 통해 데이터를 처리한다는 유사성을 보인다. 그러나 옵저버 패턴은 주체와 옵저버가 직접적으로 데이터를 주고 받지만 이벤트 버스 패턴과 발행-구독 패턴은 메시지 브로커를 통해 발행자와 구독자의 데이터 전달이 이루어진다는 차이점이 있다. 메시지 브로커의 존재는 불필요한 시스템의 복잡도를 증가시키므로 본 논문의 주제에 맞지 않다고 판단하여 옵저버 패턴을 채택하여 적용하였다.

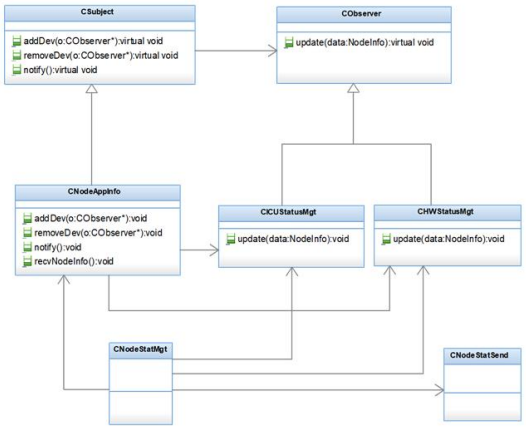

Fig. 10. Class Diagram of new NodeStatMgt

노드에 탑재된 하드웨어 구성품의 상태를 확인하기 위한 CHWStatProcess 클래스에 디자인 패턴을 추가 적용함으로써 옵저버 패턴의 객체 간 느슨한 결합도를 확인하였다. 하드웨어 구성품의 상태는 구성품의 타입에 따라 다양한 방법으로 수집된다. CPU, 메모리, 그래픽카드와 같이 IPMI(Interlligent Platform Management Interface)를 사용하거나 특정 카드는 진단 라이브러리를 사용하는 등 타입에 따라 달라지며 CHWStatProcess는 노드에 탑재된 하드웨어 구성품의 ID를 통해 하드웨어 타입을 구분하고 그에 맞는 상태 취득 함수를 호출해준다.

Table 3. class description

No.	Class Name	Description
1	CSubject	Defining an interface to manage observers
2	CNodeAppInfo	Manage the information of nede received from SysStatMgt and notify each observer in case of a change in the state of the node
3	CObserver	Definition of the interface required to update changes in the state of the subject
4	CICUStatusMgt	Maintain the state regarding the updates of interfacing application information executed at the node and report the status of the related interfacing device
5	CHWStatusMgt	Maintain the status regarding the updates of the hardware devices configured at the node and report the status of the respective hardware devices

가상머신에 동적으로 실행되는 ICUSW 및 하드웨어 구성품의 상태를 각 옵저버에게 알리는 정보를 추상화한 CNodeAppInfo를 ConcreteSubject로 설계하고 연동단 장비의 상태 관리 클래스와 하드웨어 구성품의 상태 관리 클래스인 CICUStatusMgt, CHWStatusMgt를 ConcreteObserver로 구현하여 주체와 옵저버 간에는 추상적인 결합만이 존재하도록 하였다. 이를 통해 주체 및 옵저버에 수정이 필요할 경우 독립적인 코드 수정이 가능하다. 또한, 각각의 객체는 독립적인 재사용이 가능하며 추후 새로운 형태의 상태 보고를 위한 새로운 옵저버 객체를 독립적으로 추가 가능한 이점이 있다[16].

아래의 Fig. 11.은 ConcreteSubject인 CNodeAppInfo의 동작 순서도이다. 최초 상태 보고 모듈이 실행되면 SysStatMgt로부터 해당 노드에 실행된 ICUSW와 탑재된 하드웨어 구성품에 대한 정보를 수집한다. 해당 정보는 ICUSW와 하드웨어 장치로 구분된다. 수신된 ICUSW와 하드웨어 구성품이 아닌 경우의 옵저버가 구독된 상태라면 해당 옵저버를 구독 해제함으로써 상태보고를 더 이상 수행하지 않도록 한다. 수신된 ICUSW와 하드웨어 구성품에 대한 옵저버가 구독중이지 않다면 해당 항목에 대한 신규 구독자를 추가한다. 그 후 구독중인 모든 옵저버에게 갱신된 정보를 전달하여 갱신된 노드의 정보를 통해 각 옵저버가 상태보고를 수행할 수 있도록 한다.

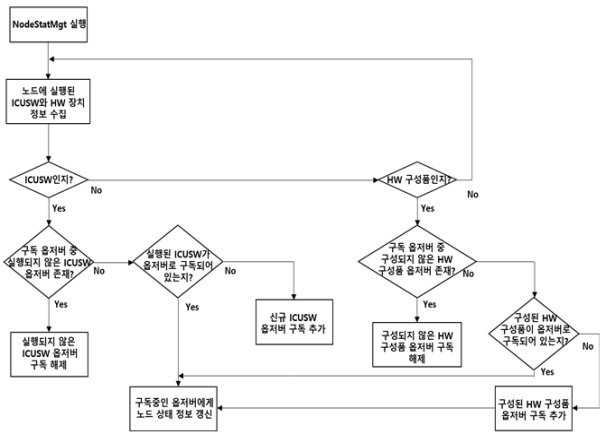

Fig. 11. Flow Chart of CNodeAppInfo

위 기능의 CNodeAppInfo는 ConCreteObserver인 CICUStatusMgt, CHWStatusMgt의 기능이 어떻게 동작하는지 알 필요가 없다. 이러한 추상적 결합을 통해 특정 클래스의 내부 동작이 변경되더라도 타 클래스에게 영향을 미치지 않는다. 또한, 각 ConcreteObserver의 동작은 서로 독립적이다. 아래의 Fig. 12.는 CICUStatusMgt, CHWStatusMgt 간의 독립적인 동작에 대한 순서도를 나타내었다. CICUStatusMgt 클래스는 주체에서 정보의 갱신이 이루어지면 갱신된 노드에 실행된 연동단 응용과 현재 콜백으로 수신되는 연동 장비의 ID가 매칭될 때만 상태보고 메시지를 송신하여 연동 장비의 상태보고 안정성을 높였다. CHWStatusMgt 클래스는 타입에 따른 상태 수집 함수를 호출하여 하드웨어 구성품에 대한 상태를 수집한 후 상태보고 메시지를 송신하게 된다.

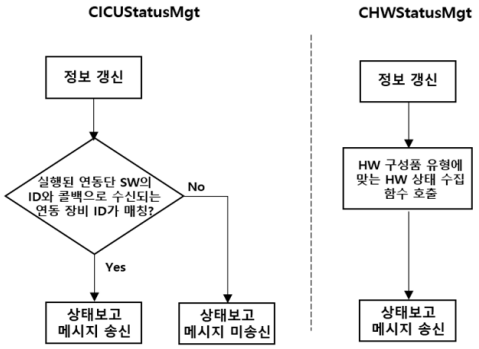

Fig. 12. Flow Chart of CICUStatusMgt and CHWStatusMgt

마지막으로 IPN에 실행되는 SysStatMgt가 각 노드의 NodeStatMgt의 상태 보고 데이터를 수집하고 처리하는 방법을 설명한다. SysStatMgt는 AMSServer로부터 각 가상머신에 실행된 ICUSW의 실행 응용명을 수집한다. 그리고 정보처리장치(IPN)에 탑재된 설정 파일로부터 ICUSW 실행 파일명에 매칭되는 SW_ID, 연동장비의 DEV_ID를 취득한다. 그리고 다기능 콘솔처럼 소프트키판넬(Soft Key Panel)과 같이 실제 하드웨어 구성품이 탑재되는 장치의 이름과 그 장치의 하드웨어 구성품 정보를 취득한다. 이를 통해 아래 Table. 4.의 구조체의 메시지를 NodeStatMgt로 송신한다.

Table 4. Node information struct send to each node by SysStatMgt

SM_NODE_STATUS

struct SM_NODE_STATUS {
SMSG_HEADER stMessageHeader;
char hostname[20];
unsigned short SW_ID[10];
unsigned short DEV_ID[10];
unsigned short HW_ID[10];
}

hostname 필드에 가상머신의 이름이나 다기능콘솔과 같은 장비의 이름으로 초기화된다. 그리고 가상머신이라면 SW_ID 필드에 실행된 ICUSW의 ID, 다기능콘솔과 같은 장비라면 HW_ID 필드에 해당 장비에 탑재된 하드웨어 구성품의 ID로 초기화 되게 된다. 해당 구조체는 hostname

을 key 값으로 가지는 테이블 타입의 DB 형태로 DDS 메시지로 송신되며 각 노드에 실행된 NodeStatMgt가 해당 정보를 참조하여 연동 장비와 하드웨어 구성품의 상태보고를 수행한다. SysStatMgt는 NodeStatMgt의 상태보고 메시지를 통해 취합된 상태보고 결과를 송신한다.

IV. Test & Validation

1. Test Environment Configuration

본 논문에서 제시한 유동적인 노드에 연동단 응용이 동적으로 실행되는 가상화 기반 잠수함 전투체계에서 상태 감시 방법의 동작성 검증을 수행하였다.

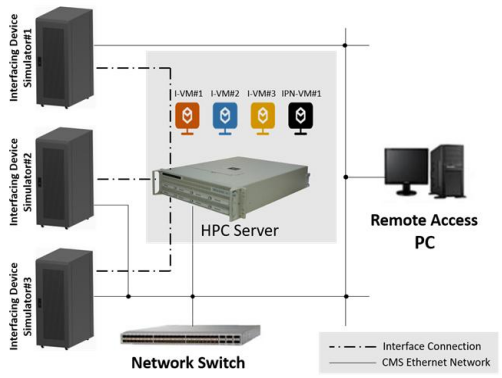

Fig. 13. Configuration of Test Bed

연동 장비 시뮬레이터 3대, HPC Server 1대, 네트워크 스위치 1개, 원격 접속 PC 1대를 이용하여 시험 환경을 구성하였다. 연동 장비 시뮬레이터는 실제 연동 장비의 데이터를 모의한다. 각 연동 장비의 인터페이스에 의해 시뮬레이터는 HPC Server와 연결된다. 연동단 응용이 실행되는 가상머신 3개, IPN 역할의 가상머신 1개로 각 노드를 구성하였다. 네트워크 스위치를 이용하여 테스트 환경을 하나의 네트워크로 구성하였고 원격 접속 PC를 이용한 SSH(Secure Shell) 원격 접속을 통해 각 가상머신의 시스템 로그를 모니터링하여 동작성을 확인하였다.

아래의 Table 5.는 시험 환경의 사양이다.

Table 5. Specification of Test Environment

No.	Category		OS	Specification
1	HW	HPC Server	VMWare ESXi 7.0.3	CPU:36 Core, Mem 128GB
2		Interface Device Simulator #1~3	Windows 10	CPU:4 Core, Mem:32GB
3		Remote Access PC	Windows 10	CPU:4 Core, Mem:32GB
4	VM	I-VM#1~3	RTST Linux	CPU:8 Core, Mem:16GB
5		IPN-VM	RTST Linux	CPU:8 Core, Mem:16GB

가상머신의 CPU 코어 수와 메모리는 기존 OOO 잠수함의 SBC 사양과 동일하게 설정하였다.

2. Verification of the functionality of the proposed method

아래의 Table 6.과 같이 4개의 테스트 케이스를 구성하였다. 테스트 케이스는 전투체계 운용환경에서 빈번히 일어나는 고장 형태로 구성하였으며 각 테스트 케이스를 통해 정확한 상태보고가 이루어지는지 확인하였다. 해당 테스트 케이스 마다 10회 반복 시험을 통해 상태정보 갱신의 정상 동작을 확인하였다.

Table 6. Specification of Test Case

No.	Test Case	Description
1-1	Failure of Interfacing Device	The status of the Interfacing device changes from normal to fault.
1-2	Normalization of the Interfacing Device	The status of the Interfacing device changes from fault to normal.
2	Abnomal termination of ICUSW	ICUSW terminated abnormally instead of shutting down normally through a proper command.
3	Shutdown of virtual machine node	Shutdown of the virtual machine node running ICUSW.

테스트 케이스의 수행 환경은 아래의 Fig. 14.와 같다. I-VM#1(100), I-VM#2(110), I-VM#3(120)에 ICUSW-A(200), ICUSW-B(210), ICUSW-C(220)가 각각 실행된 상태, 각 I-VM에는 상태보고를 위한 NodeStatMgt, IPN-VM에는 상태 취합을 위한 SysStatMgt가 실행된다.

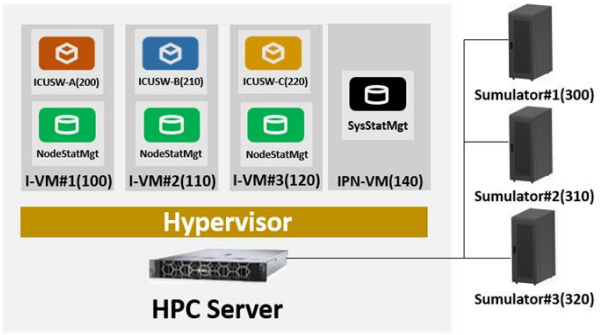

Fig. 14. Block Diagram of Test

각 ICUSW-A,B,C가 실행되는 I-VM은 달라질수 있다. 해당 환경에서 모든 상태보고가 정상적으로 들어오는 조건을 기본 조건(Default Condition, DC)으로 설정하고 각 테스트를 수행하였다.

아래의 Table 7.은 테스트 케이스 분석에 필요한 시스템 로그의 문구에 대한 설명이다.

Table 7. Description of System Log

No.	Log words	Description
1	SubSystemID	ID of the Status Reporting Subject(SW_ID, DEV_ID)
2	EQUIP-Subsystem ID	DEV_ID
3	OperStatus	Operating Status
4	TechStatus	Physical connection status with the Interfacing Device
5	FaultInfo	Fault Status Information
6	COMM_SUBSYSTEM_STATUS_INFO	Interconnection status information message of the interfacing device transmitted by ICUSW
7	SM_ALL_SYS_STATUS_INFO	Status information meessage transmitted by SysStatMgt
8	SM_NODE_INFO	Node status information transmitted by SysStatMgt

2.1 Test Case 1-1

해당 테스트 케이스는 연동 장비가 비가용 상태가 됐을 경우 그에 대한 상태가 정확히 보고되는지 확인한다. 장비의 전원이 인가되지 않거나 장비의 장애로 연동이 끊어졌을 때의 상황에서 장비의 연동상태가 비가용으로 전환되는지 확인한다. 아래의 Fig. 15.는 테스트 케이스 1-1의 I-VM#1(100)에 실행된 NodeStatMgt 로그에 대한 캡처 화면이다. 앞서 정의한 DC 상태에서 I-VM#1에 실행된 ICUSW-A(200)와 연동하는 시뮬레이터#1(300)의 모의를 종료하였다.

```
234  Oct 30 17:51:28 I-VM-100 _MGT[3031]: Observer[200]-Start ReportStatus()----------
235  Oct 30 17:51:28 I-VM-100 _MGT[3031]: Checking... Observer - SW_ID : 200 / Received DEV_ID : 300
236  Oct 30 17:51:28 I-VM-100 _MGT[3031]: DEV_ID : 300, E_TechStat : 1, E_OperStat : 1, FaultInfo : 0
237  Oct 30 17:51:28 I-VM-100 _MGT[3031]: Observer[200]-End ReportStatus()----------
238  Oct 30 17:51:28 I-VM-100 _MGT[3031]: MGT-Start Process_COMM_SUBSYSTEM_STATUS_INFO()
239  Oct 30 17:51:28 I-VM-100 _MGT[3031]: i_pData->DevId : 300,  i_pData->enTechStatus : 0, i_Data->enOperStatus : 0
240  Oct 30 17:51:28 I-VM-100 _MGT[3031]: i_pData->ulFaultInfo :2,  i_pData->enINFHistoryInfo : 0
241  Oct 30 17:51:28 I-VM-100 _MGT[3031]: MGT-End Process_COMM_SUBSYSTEM_STATUS_INFO() : DEV_ID:300...
242  Oct 30 17:51:29 I-VM-100 _MGT[3031]: Observer[200]-Start ReportStatus()----------
243  Oct 30 17:51:29 I-VM-100 _MGT[3031]: Checking... Observer - SW_ID : 200 / Received DEV_ID : 300
244  Oct 30 17:51:29 I-VM-100 _MGT[3031]: DEV_ID : 300, E_TechStat : 0, E_OperStat : 0, FaultInfo : 2
245  Oct 30 17:51:29 I-VM-100 _MGT[3031]: Observer[200]-End ReportStatus()----------
```

Fig. 15. LOG of the NodeStatMgt on I-VM#1

Line 238~240 : NodeStatMgt가 ICUSW-A로부터 장비와의 미연동 상태를 수신

Line 243 : 옵저버(200)가 SW_ID(200)와 DEV_ID(300)의 ID 매칭 검사 수행

Line 244 : 변경된 연동 상태정보를 DDS 메시지로 송신

```
1174  Oct 30 17:51:29 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:200, EQUIP-SubsystemID:300
1175  Oct 30 17:51:29 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 200, OperStatus : 1
1176  Oct 30 17:51:29 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 300, TechStatus : 0
1177  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetOperStatusNotAvailable() - SubsystemID : 300, OperStatus : 0
1178  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 300, OperStatus : 0
1179  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:210, EQUIP-SubsystemID:310
1180  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 210, OperStatus : 1
1181  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 310, OperStatus : 1
1182  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:220, EQUIP-SubsystemID:320
1183  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 220, OperStatus : 1
1184  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 320, OperStatus : 1
1185  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:200, EQUIP-SubsystemID:300
1186  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 200, OperStatus : 1
1187  Oct 30 17:51:30 T-VM-140 SYSTEMMNG[2951]: SetTechStatus() - SubsystemID : 300, TechStatus : 0
1188  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(210) - Not changed
1189  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(310) - Not changed
1190  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(220) - Not changed
1191  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(320) - Not changed
1192  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(200) - Not changed
1193  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(300) - changed
1194  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:210, EQUIP-SubsystemID:310
1195  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 210, OperStatus : 1
1196  Oct 30 17:51:31 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 310, OperStatus : 1
```

Fig. 16. LOG of the SysStatMgt on IPN-VM

위 Fig. 16.은 IPN-VM(140)에 실행된 SysStatMgt의 로그 캡처 화면이다. 각 라인의 설명은 다음과 같다.

Line 1174 : NodeStatMgt로부터 수신한 장비 상태 DDS 메시지의 Callback 함수를 호출

Line 1175~1178 : Callback 함수를 수행하여 변화된 상태 정보를 갱신

Line 1193 : 갱신된 상태정보를 DDS 메시지로 전송

2.2 Test Case 1-2

테스트 케이스 1-1 종료 시점에 이어 테스트 케이스 1-2 수행을 위해 연동 장비 시뮬레이터#1의 모의를 재시작하였다. 장비의 전원이 재인가되거나 연동 장비의 장애 상태가 해결된 상황을 모의하여 장비의 연동상태가 비가용에서 가용으로 전환되는 것을 확인하였다.

```
343  Oct 30 17:52:17 I-VM-100 _MGT[3031]: Observer[200]-Start ReportStatus()----------
344  Oct 30 17:52:17 I-VM-100 _MGT[3031]: Checking... Observer - SW_ID : 200 / Received DEV_ID : 300
345  Oct 30 17:52:17 I-VM-100 _MGT[3031]: DEV_ID : 300, E_TechStat : 0, E_OperStat : 0, FaultInfo : 2
346  Oct 30 17:52:17 I-VM-100 _MGT[3031]: Observer[200]-End ReportStatus()----------
347  Oct 30 17:52:18 I-VM-100 _MGT[3031]: MGT-Start Process_COMM_SUBSYSTEM_STATUS_INFO()
348  Oct 30 17:52:18 I-VM-100 _MGT[3031]: i_pData->DevId : 300,  i_pData->enTechStatus : 1, i_Data->enOperStatus : 1
349  Oct 30 17:52:18 I-VM-100 _MGT[3031]: i_pData->ulFaultInfo :0,  i_pData->enINFHistoryInfo : 0
350  Oct 30 17:52:18 I-VM-100 _MGT[3031]: MGT-End Process_COMM_SUBSYSTEM_STATUS_INFO() : DEV_ID:300...
351  Oct 30 17:52:18 I-VM-100 _MGT[3031]: Observer[200]-Start ReportStatus()----------
352  Oct 30 17:52:18 I-VM-100 _MGT[3031]: Checking... Observer - SW_ID : 200 / Received DEV_ID : 300
353  Oct 30 17:52:18 I-VM-100 _MGT[3031]: DEV_ID : 300, E_TechStat : 1, E_OperStat : 1, FaultInfo : 0
354  Oct 30 17:52:18 I-VM-100 _MGT[3031]: Observer[200]-End ReportStatus()----------
```

Fig. 17. LOG of the NodeStatMgt on I-VM#1

Line 347~349 : NodeStatMgt가 ICUSW-A(200)로부터 장비와의 연동 상태를 수신

Line 353 : 변경된 연동 상태정보를 DDS 메시지로 송신

```
1497  Oct 30 17:52:18 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:200, EQUIP-SubsystemID:300
1498  Oct 30 17:52:18 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 200, OperStatus : 1
1499  Oct 30 17:52:18 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 300, OperStatus : 1
1500  Oct 30 17:52:18 T-VM-140 SYSTEMMNG[2951]: SetTechStatus() - SubsystemID : 300, TechStatus : 1
1501  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:210, EQUIP-SubsystemID:310
1502  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 210, OperStatus : 1
1503  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 310, OperStatus : 1
1504  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:220, EQUIP-SubsystemID:320
1505  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 220, OperStatus : 1
1506  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 320, OperStatus : 1
1507  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(210) - Not changed
1508  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(310) - Not changed
1509  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(220) - Not changed
1510  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(320) - Not changed
1511  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(200) - Not changed
1512  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(300) - changed
1513  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:200, EQUIP-SubsystemID:300
1514  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 200, OperStatus : 1
1515  Oct 30 17:52:19 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 300, OperStatus : 1
1516  Oct 30 17:52:20 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:210, EQUIP-SubsystemID:310
1517  Oct 30 17:52:20 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 210, OperStatus : 1
1518  Oct 30 17:52:20 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 310, OperStatus : 1
```

Fig. 18. LOG of the SysStatMgt on IPN-VM

Line 1497 : NodeStatMgt로부터 수신한 장비 상태 DDS 메시지의 Callback 함수를 호출

Line 1499~1500 : Callback 함수를 수행하여 변화된 상태 정보를 갱신

Line 1512 : 갱신된 상태정보를 DDS 메시지로 전송

2.3 Test Case 2

해당 테스트 케이스는 연동 장비와 연동하는 ICUSW가 비정상 종료되는 경우이다. ICUSW-B의 비정상 종료가 발생했을 때 추가적인 작업 없이 연동상태 보고가 정상적으로 이루어지는 것을 확인한다. 기존 전투체계에서는 비정상 종료된 ICUSW-B가 실행된 SBC를 재부팅하여 연동 및 연동상태 보고를 복구하였으나 새로운 방법에서는 유후한 가상머신에 비정상 종료된 ICUSW-B가 재실행되어 연동 및 연동상태 보고가 정상적으로 복구됨을 확인하기 위한 시험이다.

DC 상태에서 I-VM#2(110)에 실행된 시뮬레이터 #2(310)와 연동하는 ICUSW-B(210)를 강제 종료하였다. 종료된 ICUSW-B(210)은 I-VM#1(100)에 재실행 되었으며 아래 내용으로 해당 결과를 설명한다.

```
504  Oct 30 18:07:22 I-VM-110 _MGT[2697]: MGT- Start Process_SM_NODE_INFO()
505  Oct 30 18:07:22 I-VM-110 _MGT[2697]: rcv - SM_NODE_INFO()  [hostname = I-VM-110]
506  Oct 30 18:07:22 I-VM-110 _MGT[2697]: There is no running Software!!
507  Oct 30 18:07:22 I-VM-110 _MGT[2697]: Unsubscribe Observer[210] - DEV_ID : 310
508  Oct 30 18:07:22 I-VM-110 _MGT[2697]: MGT- End Process_SM_NODE_INFO()
```

Fig. 19. LOG of NodeStatMgt LOG on I-VM#2

Line 504 : SysStatMgt로부터 I-VM#2(110)의 SM_NODE_INFO 수신

Line 507 : ICUSW-B 옵저버(210) 구독 해제

```
712  Oct 30 18:07:22 I-VM-100 _MGT[3031]: MGT- Start Process_SM_NODE_INFO()
713  Oct 30 18:07:22 I-VM-100 _MGT[3031]: rcv - SM_NODE_INFO()  [hostname = I-VM-100]
714  Oct 30 18:07:22 I-VM-100 _MGT[3031]: SW_ID[0] : 200, DEV_ID[0] : 300
715  Oct 30 18:07:22 I-VM-100 _MGT[3031]: SW_ID[1] : 210, DEV_ID[1] : 310
716  Oct 30 18:07:22 I-VM-100 _MGT[3031]: Subscribe new Observer[210] - DEV_ID : 310
717  Oct 30 18:07:22 I-VM-100 _MGT[3031]: MGT- End Process_SM_NODE_INFO()
718  Oct 30 18:07:23 I-VM-100 _MGT[3031]: MGT-Start Process_COMM_SUBSYSTEM_STATUS_INFO()
719  Oct 30 18:07:23 I-VM-100 _MGT[3031]: i_pData->DevId : 300,  i_pData->enTechStatus : 1, i_Data->enOperStatus : 1
720  Oct 30 18:07:23 I-VM-100 _MGT[3031]: i_pData->ulFaultInfo :0,  i_pData->enINFHistoryInfo : 0
721  Oct 30 18:07:23 I-VM-100 _MGT[3031]: MGT-End Process_COMM_SUBSYSTEM_STATUS_INFO() : DEV_ID:300...
722  Oct 30 18:07:23 I-VM-100 _MGT[3031]: Observer[200]-Start ReportStatus()---------------------------------
723  Oct 30 18:07:23 I-VM-100 _MGT[3031]: Checking... Observer - SW_ID : 200 / Received DEV_ID : 300
724  Oct 30 18:07:23 I-VM-100 _MGT[3031]: DEV_ID : 300, E_TechStat : 1, E_OperStat : 1, FaultInfo : 0
725  Oct 30 18:07:23 I-VM-100 _MGT[3031]: Observer[200]-End ReportStatus()-----------------------------------
726  Oct 30 18:07:23 I-VM-100 _MGT[3031]: MGT-Start Process_COMM_SUBSYSTEM_STATUS_INFO()
727  Oct 30 18:07:23 I-VM-100 _MGT[3031]: i_pData->DevId : 310,  i_pData->enTechStatus : 1, i_Data->enOperStatus : 1
728  Oct 30 18:07:23 I-VM-100 _MGT[3031]: i_pData->ulFaultInfo :0,  i_pData->enINFHistoryInfo : 0
729  Oct 30 18:07:23 I-VM-100 _MGT[3031]: MGT-End Process_COMM_SUBSYSTEM_STATUS_INFO() : DEV_ID:310...
730  Oct 30 18:07:23 I-VM-100 _MGT[3031]: Observer[210]-Start ReportStatus()---------------------------------
731  Oct 30 18:07:23 I-VM-100 _MGT[3031]: Checking... Observer - SW_ID : 210 / Received DEV_ID : 310
732  Oct 30 18:07:23 I-VM-100 _MGT[3031]: DEV_ID : 310, E_TechStat : 1, E_OperStat : 1, FaultInfo : 0
733  Oct 30 18:07:23 I-VM-100 _MGT[3031]: Observer[210]-End ReportStatus()-----------------------------------
```

Fig. 20. LOG of NodeStatMgt on I-VM#1

Line 712~715 : SysStatMgt로부터 I-VM#2 노드에 실행된 소프트웨어 정보를 수신

Line 716 : ICUSW-B(210)을 신규 옵저버로 구독 추가

Line 726~729 : 새롭게 실행된 ICUSW-B로부터 연동 장비(310)의 상태를 수신

Line 731 : 옵저버(210)가 SW_ID(210)와 DEV_ID(310)의 ID 매칭 검사 수행

Line 732 : 변경된 연동 상태정보를 DDS 메시지로 송신

```
4259  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:210, EQUIP-SubsystemID:310
4260  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 210, OperStatus : 1
4261  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 310, OperStatus : 1
4262  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetTechStatus() - SubsystemID : 310, TechStatus : 1
4263  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:220, EQUIP-SubsystemID:320
4264  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 220, OperStatus : 1
4265  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 320, OperStatus : 1
4266  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:200, EQUIP-SubsystemID:300
4267  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 200, OperStatus : 1
4268  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 300, OperStatus : 1
4269  Oct 30 18:07:24 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(210) - changed
4270  Oct 30 18:07:24 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(310) - Not changed
4271  Oct 30 18:07:24 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(220) - Not changed
4272  Oct 30 18:07:24 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(320) - Not changed
4273  Oct 30 18:07:24 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(200) - Not changed
4274  Oct 30 18:07:24 T-VM-140 SYSTEMMNG[2951]: Send_SM_ALL_SYS_STATUS_INFO(300) - Not changed
4275  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: statusCallback Call!! - SubsystemID:210, EQUIP-SubsystemID:310
4276  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 210, OperStatus : 1
4277  Oct 30 18:07:23 T-VM-140 SYSTEMMNG[2951]: SetOperStatus() - SubsystemID : 310, OperStatus : 1
```

Fig. 21. LOG SysStatMgt of IPN-VM

Line 4259 : NodeStatMgt로부터 수신한 장비 상태 DDS 메시지의 Callback 함수를 호출

Line 4260~4262 : Callback 함수를 수행하여 변화된 상태 정보를 갱신

Line 4269 : 갱신된 상태정보를 DDS 메시지로 전송

2.4 Test Case 3

해당 테스트 케이스는 연동 장비와 실질적으로 연동된 노드의 장애가 발생했을 때를 재현하였다. 노드의 장애 발생으로 인해 연동 장비와의 정상적인 연동이 불가할 때의 동작성을 확인하는데 시험 목적이 있으며 정상 연동 및 상태보고가 정상적으로 복구 되는지 확인하였다.

앞선 테스트 케이스와 동일하게 DC 상태에서 가상머신의 비정상 종료를 재현하기 위해 I-VM#3(120)를 종료하였다. I-VM#3에 실행되어 있던 ICUSW-C(220)은 I-VM#2(110)에 재실행 되어 아래의 내용과 같이 정상 동작함을 확인하였다.

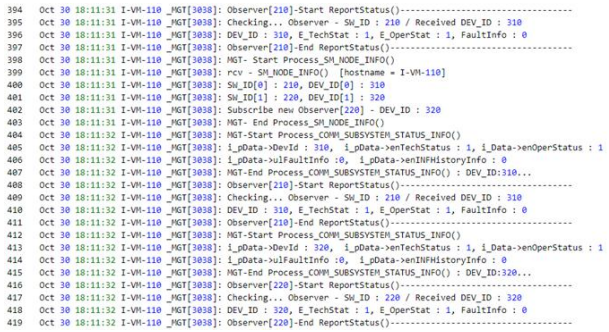

Fig. 22. LOG of NodeStatMgt on I-VM#2

Line 398~401 : SysStatMgt로부터 I-VM#2 노드에 실행된 소프트웨어 정보를 수신

Line 402 : ICUSW-C(220)을 신규 옵저버로 구독 추가

Line 412~415 : 새롭게 실행된 ICUSW-C로부터 연동 장비(320)의 상태를 수신

Line 417 : 옵저버(220)가 SW_ID(220)와 DEV_ID(320)의 ID 매칭 검사 수행

Line 418 : 변경된 연동 상태정보를 DDS 메시지로 송신

Fig. 23. LOG SysStatMgt of IPN-VM

Line 5280~5281 : I-VM#3 종료로 상태보고 메시지 타임아웃 발생

Line 5288~5291 : Callback 함수를 수행하여 변화된 상태 정보를 갱신

Line 5302~5303 : 갱신된 상태 정보를 DDS 메시지로 전송

2.3 Test Result Evaluation

Table 8. Test Results

No.	Test Result (Success/Attempt)	Success Rate
1-1	10/10	100%
1-2	10/10	100%
2	10/10	100%
3	10/10	100%

위 Table 8.은 테스트 케이스 전체의 시험 결과이다. 각 테스트 케이스마다 10번의 시험을 수행하였으며, 상황 발생으로부터 5초 이내에 변경된 상태가 반영되는지를 확인하였다. 장비 상태의 확인은 실시간성이 필수적인 요소가 아니므로 운용자가 해당 변화를 파악할 수 있는 5초 내의 상태 반영을 Pass/Fail의 기준으로 설정하였다. 전체 4개의 테스트 케이스를 10회 반복하였으며, 모든 테스트 케이스의 결과가 Pass로 도출되었다. 해당 결과는 새로운 연동 장비 상태감시 방법의 정상적인 동작을 검증한다.

3. Development Effort Analysis for Adding new Interfacing Device

다음은 잠수함 전투체계에 새로운 연동 장비가 추가되는 상황을 가정하고 신규 개발 소요 분석과 동작 검증을 수행하였다. 아래의 Fig. 24.는 SysStatMgt가 정보를 취득하는 IPN에 탑재된 설정 파일이다. XML로 작성되어 있으며 하드웨어 장비의 이름과 그 장비에 구성되는 하드웨어 구성품의 ID, ICUSW의 ID와 실행 응용명, 그리고 ICUSW와 연동하는 장비의 ID로 구성된다.

아래 Fig. 24.의 설정 파일은 SysStatMgt가 각 가상머신에 실행된 응용 정보를 구성하기 위한 최소한의 정보로 구성하였다. 이는 SysStatMgt가 해당 설정 파일을 통해 각 가상머신에 실행된 연동단 응용의 SW_ID와 DEV_ID 정보만을 취득하여 각 가상머신의 연동단 실행 정보를 구성하도록 한다. 따라서 추가되는 신규 장비의 개수는 장비 추가 기능과는 독립적이므로 동시에 다수의 장비가 추가되는 상황에 대응할 수 있다. 또한, 설정 파일은 패키지 형태의 일괄 관리를 통해 SysStatMgt가 실행되는 IPN-VM에 탑재되게 하였다. 해당 설정 파일은 IPN-VM에 실행된 SysStatMgt 응용만이 참조하도록 구성하여 설정 파일의 동시 접근에 대한 충돌상황을 배제할 수 있도록 하였다.

```
<?xml version="1.0" encoding="EUC-KR"?>
<ROOT>
  <HW>
    <MFC1_CONTAINER name="MFC">
      <item hwid="100"></item>
      <item hwid="101"></item>
      <item hwid="102"></item>
      <item hwid="103"></item>
    </MFC1_CONTAINER>
  </HW>
  <SW>
    <DEFAULT>
      <SWInfo name ="SW">
        <item swid ="200" appName="GNSS.out" devid="300"></item>
        <item swid ="210" appName="EMLog.out" devid="310"></item>
        <item swid ="220" appName="DSDS1.out" devid="320"></item>
        <item swid ="230" appName="DSDS2.out" devid="330"></item>
      </SWInfo>
    </DEFAULT>
  </SW>
</ROOT>
```

Fig. 24. Setting file on IPN-VM

시험 수행을 위해 신규 ICUSW-D(230)과 연동장비 시뮬레이터#4(330)를 추가하기 위해 Line 17을 추가하였다. I-VM#3(120)에 ICUSW-D를 실행하여 테스트 케이스를 재수행하였으며 정상 동작을 확인하였다.

```
Nov 04 10:11:32 I-VM-120 _MGT[2997]: MGT-Start Process_COMM_SUBSYSTEM_STATUS_INFO()
Nov 04 10:11:32 I-VM-120 _MGT[2997]: i_pData->DevId : 330,  i_pData->enTechStatus : 1, i_Data->enOperStatus : 1
Nov 04 10:11:32 I-VM-120 _MGT[2997]: i_pData->ulFaultInfo :0,  i_pData->enINFHistoryInfo : 0
Nov 04 10:11:32 I-VM-120 _MGT[2997]: MGT-End Process_COMM_SUBSYSTEM_STATUS_INFO() : DEV_ID:330...
Nov 04 10:11:32 I-VM-120 _MGT[2997]: Observer[230]-Start ReportStatus()---------------------------------
Nov 04 10:11:32 I-VM-120 _MGT[2997]: Checking... Observer - SW_ID : 230 / Received DEV_ID : 330
Nov 04 10:11:32 I-VM-120 _MGT[2997]: DEV_ID : 330, E_TechStat : 1, E_OperStat : 1, FaultInfo : 0
Nov 04 10:11:32 I-VM-120 _MGT[2997]: Observer[230]-End ReportStatus()-----------------------------------
```

Fig. 25. Sample LOG of NodeStatMgt on I-VM#2

본 논문에서 설계한 Fig. 10.의 NodeStatMgt의 클래스 수정 없이 단순 설정 파일에 내용 추가를 통한 정상 상태 보고 동작을 수행함으로써 신규 장비 추가에 대한 확장성을 확인하였다. 이를 통해 잠수함 전투체계에 신규 장비가 추가될 경우 소프트웨어의 수정 없이 단순 설정 파일의 변경만으로 장비를 추가함으로써 공수 절감 효과를 기대할 수 있다.

V. Conclusions

함정 전투체계에서 체계상태를 감시하는 기능은 함정을 운용하는데 필요한 전반적인 시스템의 상태를 파악하여 유지 보수하기 위한 필수적인 기능이다. 이러한 체계상태 감시를 수행하기 위한 방법은 전투체계에 가상화가 적용됨에 따라 변화하는 시스템 정책에 맞게 변화하여야 한다.

따라서 본 논문에서는 연동단 응용이 유동적인 위치에 실행되는 가상화 기반 잠수함 전투체계에서 연동장비의 상태 감시 방법을 제시하고 동작 검증을 통해 새로운 방법의 적용 가능성을 확인하였다. 또한, 디자인 패턴 중 옵저버 패턴(Observer Pattern)을 적용함으로써 옵저버 객체의 구독/해제를 통해 이루어지는 상태보고 동작과 느슨한 결합도를 적용한 설계로 각 클래스 간의 독립성을 확인하였다.

3개의 연동단 응용이 실행되는 Bare-Metal 구조의 하이퍼바이저 기반 가상화 환경에서의 테스트를 통해 동작을 검증하였다. 검증 적합성을 위해 기존 잠수함 전투체계에서 발생하는 고장 형태로 테스트 케이스를 구성하여 시험을 수행하였으며 시스템 로그 분석을 통해 정상동작을 확인하였다.

또, 상태보고가 필요한 장비가 신규로 추가됐을 때, 별도의 코드 수정 없이 설정 파일에 새로운 장비의 ID 정보만을 추가하여 정상적인 상태보고가 이루어지는 것을 확인하였다. 이는 잠수함 전투체계에 신규 장비가 추가될 경우 공수를 절감하는 효과로 이어질 수 있다.

본 논문에서 제안한 실행 위치가 유동적인 연동단 응용의 연동상태 보고 기능의 적용으로 가상화 기반의 잠수함 전투체계에서 효율적인 장비의 연동상태 감시를 통해 함정 운용과 유지보수에 있어 이점으로 작용할 것이다.

현재 함정 전투체계 가상화에 대한 사업과 연구들이 활발히 진행되고 있다. 본 논문의 연구를 시작으로 잠수함 전투체계에 가상화 적용에 따른 시스템의 변화에 대응하는 실질적인 응용 기능의 후속 연구가 이루어질 것이라고 예상한다. 또한, 전투체계의 AI 전력화를 목표로 하는국방부에 맞춰 GBR, SVR과 같은 머신러닝 모델의 전투체계 적용으로 연구를 확장하여 후속 연구를 진행해 나갈 예정이다.

REFERENCES

[1] J. G. Lee, “A Study on the Standard Architecture of Weapon Control Software on Naval Combat System,” Journal of the Korea Society of Computer and Information Vol. 26, No. 11, pp. 101-110, November 2021. DOI: 10.9708/jksci.2021.26.11.101

[2] S. M. Kwon, and Seung-Mo Jung, “Virtualization based high efficiency naval combat management system design and performance analysis,” Journal of The Korea Society of Computer and Information Vol. 23, No. 11, pp. 9-15, November 2018. DOI: 10.9708/jksci.2018.23.11.009

[3] R. H. Lee, “The design of a Portable Automatic Test Equipment for Operational Availability of Combat System,” Journal of the Korea Academia-Industrial cooperation Society, Vol. 21, No. 3, pp. 453-459, March 2020. DOI: 10.5762/KAIS.2020.21.3.453

[4] 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 Korea Institute of Military Science and Technology, Vol. 15, No. 4, pp.

381-389, August 2012.

[5] Y. G. Hong, "Applicability Study of Distributed System Management Based on AMSM for Naval Combat System," Paper of Korea Institute of Military Science and Technology, Autumn Annual Conference, November 2018.

[6] J. Y. Im, and D. S. Kim, "Performance Evaluation of Virtualization Solution for Next Generation Naval Combat Systems," Journal of The Institute of Electronics and Information Engineers, Vol. 56, No. 2, pp. 41-49, February 2019. DOI: 10.5573/ieie.2019.56.2.41

[7] K. S. Song, "A Study of Feasibility and Performance Analysis of VDI Based on GPU Acceleration for Naval Combat System," Journal of The Institute of Electronics and Information Engineers, Vol. 58, No. 11, November 2021. DOI: 10.5573/ieie.2021.58.11.86

[8] D. W. Lee, B. K. Bae, and K. S. Cho, "A feasibility study of virtualization for Submarine Combat System," Journal of The Korea Society of Computer and Information, Vo. 27, No. 9, pp. 121-129, September 2022. DOI: 10.9708/jksci.2022.27.09.121

[9] S. G. Son, "A study of submarine combat management system docker-based server virtualization design and performance analysis," Journal of The Korea Society of Computer and Information, Vol. 27, No 12, pp. 121-129, December 2022 DOI: 10.9708/jksci.2022.27.12.121

[10] T. H. Kim, H. J. Kim and J. C. No, "VDI Real-Time Monitoring System for KVM-Based Virtual Machine Resource Usage Analysis," Journal of The Institute of Electronics and Information Engineers, Vol. 52. No. 1, pp. 189-190, January 2015 DOI: 10.5573/ieie.2015.52.1.069

[11] Y. W. Jeong, "A Study on the Usages of DDS Middleware for Efficient Data Transmission and Reception," Journal of The Korea Society of Computer and Information, Vol. 23, No. 11, pp. 59-66, November 2018. DOI: 10.9708/jksci.2018.23.11.059

[12] J. Y. Im, "Performance Evaluation of Discovery and Message Transmission of DDS(Data Distribution Service) Security," Journal of the Korea Institute of Information and Communication Engineering, Vol. 25, No. 5, pp. 701~708, May, 2021. DOI: 10.6109/jkiice.2021.25.5.701

[13] Y. G. Hong, G. T. Choi, and S. G. Back, "Analysis of Secure DDS Discovery Time And Probability on Various-Sized Distributed Systems," Journal of the Korea Institute of Information and Communication Engineering, Vol. 27, No. 10, pp. 1270-1277, October 2023. DOI: 10.6109/jkiice.2023.27.10.1270

[14] DDS, *https://www.omg.org/omg-dds-portal/*

[15] J. S. Kim, "Derivation of Kotlin Flow API from Observer Pattern," Journal of Korean Institute of Information Scientist and Engineers, Vol. 50, No. 9, pp.821-826, September 2023. DOI: 10.5626/JOK.2023.50.9.821

[16] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, *"Design Patterns: Elements of Reusable Object-Oriented Software,"* PEARSON Education Korea, pp. 382-394, 2009

Authors

Ye-Jun Jang received the B.S. degrees in Information and Communication Engineering from Yeungnam University, Korea, in 2014. He is currently working in Hanwha Systems Co. from 2022.

He is interested in Naval Combat Management System, Design Pattern, Server Virtualization and so on.