# 전투체계 시스템을 위한 실시간 환경에서의 비동기 이중화 기법 연구

## A Study on Real Time Asynchronous Data Duplication Method for the Combat System

이 재 승* 류 존 하*
Lee, Jae-Sung Ryu, Jon-Ha

ABSTRACT

In a naval combat system, the information processing node is a key functional equipment and performs major combat management functions including control sensor and weapon systems. Therefore, a failure of one of the node causes fatal impacts on overall combat system capability. There were many methodologies to enhance system availability by reducing the impact of system failure like a fault tolerant method.

This paper proposes a fault tolerant mechanism for information processing node using a replication algorithm with hardware duplication. The mechanism is designed as a generic algorithm and does not require any special hardware. Therefore all applications in combat system can use this functionality. The asynchronous characteristic of this mechanism provides the capability to adapt this algorithm to the module which has low performance hardware.

주요기술용어(주제어) : Fault Tolerant System(내고장 시스템), Asynchronous Data Redundant(비동기 데이터 이중화), Real Time(실시간), Middleware(미들웨어)

## 1. 머리말

오늘날 해상 전장 환경은 다양하게 변화하고 있으며 위협 세력은 나날이 증가하고 있다. 이러한 변화는 함정 전투체계가 함정 탑재 센서와 무장에 대한 통합 능력을 가져야 하며, 최적화된 제어를 통해 정보를 융합하고 위협을 분석하고 이에 대응하기 위한 무장 할당 능력을 가질 것을 요구한다. 또한 함정 전투체계는 작전 운용 개념의 진화에 따른 함정 고유의 임무와 역할 변화에 대해 적응하는 능력을 필요로 한다. 이와 함께 다양한 획득 자료에 대한 효과적 처리를 위해 고도의 실시간 정보 처리 능력을 가져야 한다[1].

함정 전투체계는 다양한 상황에서 함정의 임무를 수행하기 위해 각종 체계를 이중화하여 사용하고 있다. 이러한 이중화에 대한 노력은 함정 전투체계의 생존성과 신뢰성으로 직결된다. 함정의 생존성은 함정의 전투력에 연관되어 있으며 함정의 전투력은 생

† 2006년 9월 26일 접수~2007년 3월 5일 게재승인
* 국방과학연구소(ADD)
주저자 이메일 : berise@add.re.kr

존성의 여부에 따라 큰 영향을 받는다.

함정 전투체계에서 정보처리장치는 다양한 임무를 처리한다. 이로 인해 정보처리장치의 고장은 함정 능력에 치명적인 손실을 가져다 줄 수 있다. 이에 따라 이러한 손실을 최소화하기 위해서 정보처리장치를 이중화하여 고장이나 외부로부터의 피해를 극복할 수 있는 방법이 요구된다.

기 개발된 이중화 방안은 장비 성능의 한계와 이를 극복하기 위해 전용 하드웨어 및 소프트웨어를 이용하여 이중화를 구현하였다[2~5]. 이러한 방법은 환경에 따라 최적의 결과를 얻을 수 있는 장점이 있지만 이를 개발하기 위한 비용 및 향후 유지 보수비용이 증가되는 단점이 있다. 배치 이후 수 십 년을 사용하는 체계에서는 유지 보수로 인한 비용 부담이 상대적으로 커지게 된다.

체계 개발이 전용 하드웨어에서 표준 제품을 기반으로 하는 개방형 구조(Open Architecture)를 참조 모델로 하여 개발하는 방식으로 점차 전환되고 있으며 이에 따라 많은 상용 제품들이 표준을 준수하는 방식으로 개발되고 있다. 체계 선진국에서는 이러한 개방형 구조를 도입하여 체계 개발 및 유지 보수비용을 줄이는 노력을 하고 있다.

본 논문에서는 표준 상용 제품을 이용하여 소프트웨어적인 방식을 이용한 이중화 방안을 제안한다. 하드웨어 플랫폼, 운영체제 및 통신 미들웨어 등의 제품을 표준 상용 제품으로 사용하며 이 상위에 전투체계를 위한 기반 라이브러리를 개발하고 이중화 기법을 구현한다. 이중화 기법은 별도의 하드웨어에 의존하지 않고 소프트웨어만을 이용하여 구현하였다. 본 논문에서 제안하는 데이터 보존 방안은 이중화 데이터를 관리하는 방법과 응용 프로그램이 직접 이중화 데이터를 제어할 수 있도록 하는 인터페이스의 설계로 구성된다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 이중화 방법에 대해 기술하고, 3장, 4장, 5장에서는 본 논문에서 제안한 이중화 시스템의 구조, 이중화 데이터의 동기화, 오류 감지에 대해 기술한다. 6장에서는 절체 방법 및 알고리즘 분석에 대해 기술하며 7장에서는 제안한 이중화 시스템을 시험하기 위한 환경과 시험 결과를 제시하고 8장에서 결론 및 향후 연구 과제를 제시한다.

## 2. 관련 연구

이중화 기법은 특정 시스템이 고장으로 인해 정지되더라도 다른 시스템이 작업을 이어받아 계속적으로 작업이 가능하도록 하는 것이 주된 목적이다. 이러한 기법은 고가용성을 필요로 하는 시스템에 많이 적용되고 있다. 이중화 기법에는 하드웨어와 소프트웨어를 이용한 다양한 방법이 존재하며 본 논문에서는 소프트웨어적인 방법을 사용한 이중화 기법만을 다룬다.

데이터를 이중화하기 위한 기법을 데이터 동기화 방식에 따라 나누면 동기식 이중화 방식과 비동기식 이중화 방식으로 나눌 수 있다. 동기식 이중화 방식은 액티브와 스텐바이가 데이터를 동일한 순서로 처리하여 그 결과를 비교하는 방식이며 비동기식 이중화 방식은 액티브에서 데이터를 처리하고 스텐바이는 액티브의 처리 결과 혹은 상태를 받아서 액티브와 일치시키는 방식을 말한다[6].

동기식 이중화 방식에는 주/부(Primary/Shadow) 처리 방식, 액티브 복제(Active Replication), 처리 후 투표(N-Version/Voting) 방식이 있다. 주/부 처리방식은 모든 노드들이 데이터를 동시에 처리하고 그 결과를 동기화시킨 후 주 노드(Primary Node)만 데이터를 전송하는 방식이다. 액티브 복제 방식은 모든 노드들이 데이터를 동시에 처리하고 그 결과를 모두 전송하는 방식이다. 이 결과를 필요에 따라 필터링하여 사용한다. 처리 후 투표 방식은 데이터를 처리하고 그 결과 전송 시 노드 간 투표를 통해 전달하는 방식이다. 위와 같은 동기식 이중화 방식은 액티브가 고장이 났을 때 빠르게 복구할 수 있는 장점이 있으나, 노드 및 네트워크에 부하가 많이 걸리는 단점이 있다. 따라서 체계 여유 가용도가 필수적인 경우에는 사용하기 어렵다.

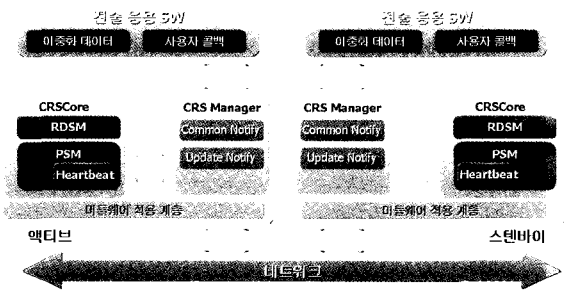
비동기식 이중화 방식에는 부하 분배(Load Sharing), 체크 포인트(Check Point), 수동적 복제(Passive Replication)의 방법이 있다. 부하 분배 방식은 처리할 데이터를 나누어 각각의 노드에서 처리하고 그 결

과를 종합하여 전달하는 방식이며 체크 포인트 방식은 엑티브 노드에서 주된 일을 처리하면서 스텐바이 노드에게 처리 점검 지점을 송신한다. 스텐바이는 이것을 수신하여 복구 지점을 정기적으로 수립한다. 고장이 발생한 경우 점검 지점을 기준으로 데이터를 복구한다. 수동적 복제 방식은 한 노드에서 데이터를 처리하고 나머지 노드들은 이 데이터를 동기화하는 방식이다. 위와 같은 방식은 부하가 작고 부가적인 처리 부하가 적은 대신에 복구시간이 오래 걸리는 단점이 있다.

## 3. 시스템의 구조

상기 기술한 이중화 방식은 각각 방식이 장단점을 가지고 있지만, 고유의 기법을 그대로 전투체계에 적용하는 것은 어렵다. 이것은 전투체계 요구사항이 복합적인 기술의 결합을 요구하기 때문이며, 이를 위하여 본 논문에서는 데이터의 동시 수신을 위한 멀티캐스트(Multicast), 상태 체크 포인트(State Checkpoint), 이중화 상태 제어를 위한 콜백(Callback), 이중화 데이터 버퍼링 등을 사용한다. 또한, 전투체계 요구사항을 만족하기 위해서는 다음과 같은 요구 사항이 충족되어야 한다. 첫째, 엑티브 노드와 스텐바이 노드로 시스템을 구성하여 중요한 정보를 이중화할 수 있는 이중화 시스템을 구성해야 한다. 둘째, 정보를 처리하는 노드가 고장 났을 때, 이 시점까지의 이중화 데이터를 동기화 할 수 있어야 한다. 셋째, 동기화를 하지 못한 영역의 데이터를 복구할 수 있어야 한다.

본 논문에서 제안하는 이중화 시스템(CRS : Common Resilience System)의 구조는 그림 1과 같다.




[그림 1] 이중화 시스템의 구조

액티브와 스텐바이 간의 데이터버스는 이더넷을 이용하며 노드 간의 데이터 전송을 위해서는 상용 통신 미들웨어를 이용한다. 상용 통신 미들웨어는 QoS (Quality of Service)를 제어할 수 있는 기능으로 전투체계에서 요구하는 실시간 요구사항을 만족할 수 있는 기능을 구현한 제품이며 Publish/Subscribe 방식의 통신 지원으로 다자간 통신을 위한 다양한 API (Application Programming Interface), 자동화 매크로 및 다양한 편의 함수를 제공한다.

CRSManager는 상위 응용 프로그램에게 이중화 기능 인터페이스를 제공하는 계층이다. 이 계층을 이용하여 상위 응용 프로그램은 이중화 모듈을 제어할 수 있다. CRSManager는 이중화 데이터의 동기화 및 수신을 위한 각종 통지 장치(Notify) 기능을 제공한다.

CRSCore는 이중화 동기 관리자(RDSM : Resilience Data Synchronization Manager)과 상태 관리자(PSM : Process Status Manager) 및 Heartbeat로 구성된다. RDSM은 상위 응용 프로그램에서 설정한 이중화 데이터를 관리하며 외부로부터 들어오는 데이터를 입력 동기화하고 CRSManager의 통지 기능을 이용하여 액티브와 스텐바이 간에 데이터 동기화 시점을 관리한다. RDSM은 내부적으로 임시 버퍼를 사용하여 엑티브에서 이중화 데이터를 처리하는 동안 데이터를 저장하는 공간으로 사용한다. 이 임시버퍼는 스텐바이에서만 사용하며 스텐바이에서 데이터를 입력받은 후 엑티브에서 처리한 결과를 동기화하기 전까지 사용하는 큐(queue) 형태의 버퍼이다.

PSM은 노드의 ID(Identification), 노드 상태(Node Status), Heartbeat의 빈도 및 수신 상태를 이용하여 노드의 상태를 관리한다. PSM의 주 역할은 Heartbeat의 송수신을 통해 실시간 시스템의 오류를 감지하는 것이다. 시스템이 요구하는 데이터 복구 시간에 따라 Heartbeat을 점차 세밀하게 제어해야 할 필요가 있다. 하지만 Heartbeat의 전송 주기는 이중화 시스템의 성능에 영향을 줄 수 있으므로, Heartbeat의 교환 주기는 시스템의 성능 및 전술 응용 SW의 요구 사항에 따라 적절하게 설정해야 한다.

전술 응용 SW(Software)에서는 이중화 데이터와

사용자 콜백을 CRSManager의 API를 이용하여 이중화 시스템에 등록한다. 이중화 데이터는 응용 프로그램에서 필요로 하는 데이터를 이중화하는 공간이며 사용자 콜백은 이중화 시스템에서 전술 응용 프로그램으로 이벤트 및 데이터를 전달하기 위한 통로 역할을 담당한다.

## 4. 이중화 데이터 동기화

데이터 동기화는 엑티브와 스텐바이 간에 시스템 복구를 위한 유지하는 데이터를 일치시키는 작업으로 엑티브의 고장이 있을 경우 스텐바이가 엑티브의 작업을 이어받아 수행하기 위해 필요한 작업이다.

엑티브와 스텐바이 간에 동기화를 수행하기 위해서 공통의 메모리 주소 공간이 필요하다. 이 주소 공간은 응용 프로그램에서 이중화에 필요한 메모리 주소 공간을 제공한다. 할당된 메모리 주소 공간은 CRS에게 맵핑되고 초기화 작업을 수행한다. 따라서 전술 응용 프로그램은 자신의 프로그램 컨텍스트에 맞는 주소 공간을 할당하고 관리할 수 있다.

초기화 작업이 종료된 후 엑티브는 외부 입력 데이터를 처리하고 그 결과를 스텐바이에게 전달한다. 스텐바이는 이 데이터를 동기화한다. 이때 엑티브 시스템의 부하로 인해 이중화 데이터의 동기화가 영향을 받아서는 안 된다. 이를 방지하기 위해 엑티브-스텐바이 간에 이중화 작업은 별도의 쓰레드를 이용하여 처리되며 우선순위를 높게 설정한다.

이중화 데이터 동기화를 동기화하기 위한 절차는 크게 초기 데이터 동기화와 중간 데이터 동기화 두 부분으로 나눌 수 있다. 초기 데이터 동기화는 최초 시스템을 운용할 때 엑티브와 스텐바이 간의 데이터 전체를 동기화 하는 작업이다. 이 과정은 이중화 데이터 공간의 크기에 따라 동기화 시간이 달라진다. 중간 데이터 동기화는 최초 동기화 시점 이 후에 전술 응용 프로그램에서 필요에 따라 변경된 부분을 갱신하는 절차이다. 전술 응용 프로그램은 콜백을 통하여 필요시 마다 수행한다.

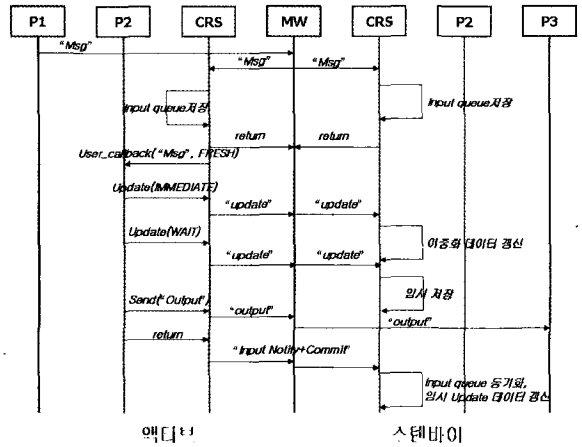
중간 데이터 동기화의 경우는 주로 스텐바이의 임시 버퍼를 사용하여 동기화한다. 스텐바이는 이중화 데이터의 갱신을 위해 임시 버퍼를 이용하여 엑티브에서 이중화 데이터를 처리하는 동안 입력되는 데이터를 저장한다. 엑티브에서 데이터를 처리하는 동안 스텐바이는 대기하게 되며 엑티브로부터 데이터 처리 종료 메시지를 전달받기 전까지의 입력 데이터들은 모두 이 임시 버퍼로 저장된다. 임시 버퍼의 크기는 고장이 일어난 후 고장 감지 시간이 길어질수록 더 큰 버퍼의 크기가 필요하다. 이 버퍼의 크기 Q는 데이터 처리 요청 메시지의 발생 빈도(F)와 고장 전환 시간(T)의 곱으로 정의되며, 임시 버퍼의 크기는 이중화 시스템의 성능에 영향을 주는 요소이기 때문에 이중화 시스템의 Heartbeat의 주기와 데이터의 크기나 발생 빈도 등의 특성에 맞게 설정되어야 한다.

엑티브의 고장이 감지될 경우 스텐바이는 프로세스 상태 관리자로부터 ONLINE(HOT_STANDBY)에서 ONLINE(OPERATIONAL)로의 상태 변경 통지를 받게 되며, 엑티브에서 처리한 마지막 메시지 이후부터 메시지를 처리한다.

데이터 동기화는 비동기 이중화 시스템에서 가장 많은 시스템 자원을 소모한다. 이로 인해 비동기 이중화 시스템의 최대 이중화 데이터 처리 용량은 시스템의 자원에 의존적이다. 따라서 최종 시스템의 이중화 데이터 용량은 시스템의 성능, 시스템의 평균 부하, 그리고 주변장치의 속도 등을 고려하여 결정해야 할 필요가 있다.

그림 2와 같이 전술 응용 SW인 P1에서 P2를 거쳐 P3으로 데이터를 전달하는 경우가 있다고 가정하자.

P2에서는 데이터에 대한 이중화 처리를 한다. P1에서 Msg를 전달(Publish)하면 미들웨어(Middleware, MW)에 의해 적절한 수신자에게 수신(Subscribe)한다. MW는 액티브의 P2와 스텐바이의 P2에게 데이터를 전달하게 되고 각각에서 이중화 시스템(CRS : Common Resilience Service)은 데이터를 수신하여 저장하고 통지메시지를 User_callback을 통해 P2에게 데이터의 수신을 알려준다. P2는 데이터를 가공한 후 필요에 따라 Update를 이용하여 스텐바이에게 이중화 데이터의 갱신을 수행할 수 있다. Update는 이중화 데이터를 즉시 갱신하는 IMMEDIATE, 나중에 갱신하는 WAIT를 인자로 사용할 수 있다.




[그림 2] 동기화 절차

이중화 처리가 끝난 데이터는 P3에게 전달되고 Commit을 이용하여 이중화 데이터의 처리가 끝났음을 스탠바이에게 통지한다. 이 통지를 통해 스탠바이는 Update(WAIT)를 수행한 데이터에 대한 동기화를 수행한다.

## 5. 오류 감지와 상태 관리

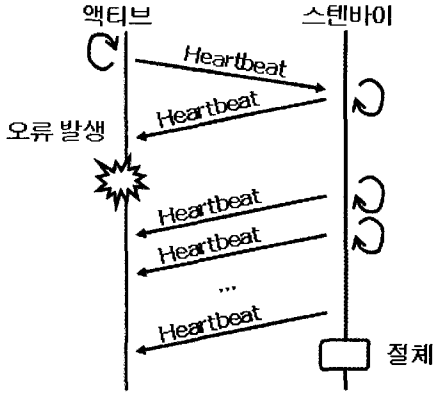
순간적인 노드의 오류를 감지할 수 있는 기능은 치명적인 시스템 문제를 피하기 위해서 필요하다.

Heartbeat은 상대 프로세스의 상태를 결정하거나 고장을 감지하거나 자신의 상태를 점검하기 위해서 사용된다. Heartbeat 교환은 시스템에 대한 지연 시간을 최소화하기 위해서 별도의 쓰레드를 생성하여 교환한다. 액티브와 스탠바이 간에 Heartbeat 교환은 그림 3과 같다.

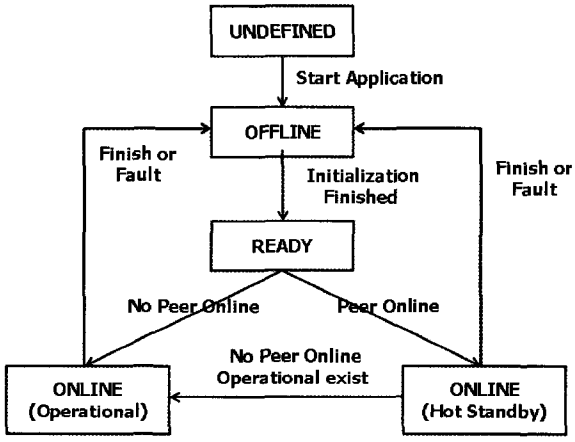
자신의 상태를 위한 Heartbeat와 상대 노드를 감시하기 위한 Heartbeat를 송신하고 있다. Heartbeat은 반드시 응답을 필요로 하며 응답이 없는 경우 오류로 간주하며 오류 감지를 위한 절차를 수행한다.

오류가 발생한 경우 이중화 시스템은 시스템의 설정 값에 따라 Heartbeat를 재송신하고, 응답이 없을 경우 절체를 수행하여 이중화 데이터를 복구하고 응용 프로그램의 동작이 끊어지지 않도록 한다.

이중화 시스템은 이중화 노드의 상태를 나타내기




[그림 3] Heartbeat 교환




[그림 4] PSM의 상태 천이도

위해 PSM을 사용한다. 자신이 속한 노드와 상대 노드의 상황에 따라 PSM 상태를 정의한다. PSM의 상태는 모두 5가지로 UNDEFINED, OFFLINE, READY, ONLINE(OPERATIONAL), ONLINE(HOT STANDBY)로 나눌 수 있다.

PSM는 최초 UNDEFINED 상태에서 시작하며, 이중화 응용 프로그램을 실행하면 OFFLINE으로 변경된다. 이 후 액티브와 스탠바이 간의 이중화 데이터 초기화 작업이 끝나면 READY상태가 된다. READY 상태에서는 상대 노드의 PSM의 존재 유무를 판단하고, 만약 상대 PSM가 존재하지 않으면 엑티브로 동작하는 ONLINE(OPERATIONAL) 상태로 전환하고 그렇지 않으면 스탠바이로 동작하는 ONLINE(HOT_STANDBY) 상태로 변경된다. ONLINE(HOT_

[표 1] 프로세스 상태 관리자의 상태

| 상 태 | 설 명 |
|---|---|
| UNDEFINED | 프로그램이 구동되지 않은 상태 |
| OFFLINE | 프로그램 구동 후 초기화 이전의 상태 |
| READY | 프로그램 내부 초기화를 완료하고 Peer 상태 확인을 위한 준비를 마친 상태 |
| ONLINE (OPERATIONAL) | Peer가 존재하지 않거나 Peer의 상태를 확인할 수 없는 경우 |
| ONLINE (HOTSTANDBY) | Peer(Online)가 존재하고 Standby가 동작하는 상태 |

STANDBY) 상태에서는 상대 PSM이 감지되지 않은 경우 ONLINE(OPERATIONAL) 상태로 전환된다. 시스템이 종료하거나 오류로 인해 종료될 경우에는 OFFLINE으로 상태가 변경된다. PSM의 상태 천이 그래프는 그림 4와 같으며 상태에 따른 동작은 표 1에 기술되어 있다.

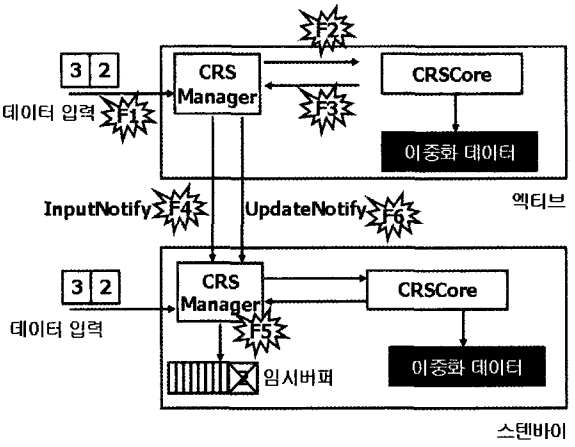
## 6. 절체 방법 및 알고리즘 분석

시스템에서 일어나는 오류는 PSM에서 감지하고 이중화 처리 컴포넌트에게 전달한다. 이와 함께, PSM은 이중화 시스템의 현재 상태 및 이중화 데이터의 처리 상황을 CRSManager에게 통보한다. CRSManager는 오류 상황에 따라 시스템의 복구 단계를 결정하고 시스템을 복구한다.

이중화 데이터는 액티브와 스텐바이에게 동시에 전달된다. 전달된 이중화 데이터는 CRSManager에 의해 InputNotify로 액티브와 스텐바이 간에 데이터 수신을 확인하게 된다. 이 때 스텐바이에서는 큐 형태의 임시 버퍼에 저장하게 된다. 이 후 액티브에서는 이중화 데이터를 CRSCore로 전달하여 이중화 처리를 한 후 필요에 따라 UpdateNotify를 이용하여 스텐바이의 이중화 데이터를 갱신한다. 스텐바이는 UpdateNotify 메시지를 통해 엑티브의 데이터 처리가 끝났음을 인지하고 이중화 데이터를 갱신하며 임시 버퍼에 저장된 이중화 데이터를 삭제한다.

시스템에 고장이 일어날 수 있는 부분으로 구분하면 F1~F6까지 이중화 데이터의 갱신 시점에 따라 6단계로 나눌 수 있다. 고장 시점은 이중화 데이터의 갱신이 일어나기 전 단계인 C1과 이중화 데이터의 갱신이 일어난 후 단계인 C2로 나눌 수 있다. 이를 기준으로 이중화 시스템에서의 고장을 구분하면 다음과 같다.

F1은 데이터가 입력되는 단계에서 고장이 일어나는 시점으로 C1에 포함된다. F2와 F3은 이중화 시스템의 내부 처리 과정에서 일어나는 고장으로 C1에 포함된다. 마찬가지로 F4, F5의 경우 입력을 위한 동기화 메시지인 InputNotify와 스텐바이에서 임시 버퍼를 처리하는 과정으로 C1에 포함된다. F6은 스텐바이에 데이터 갱신 메시지인 UpdateNotify가 전달되었을 경우 C2가 되고 전달되지 않은 경우 C1의 경우가 된다. UpdateNotify가 정상적으로 스텐바이에게 전달된 경우에는 스텐바이의 CRSManager는 임시 버퍼의 항목에서 엑티브에 의해 처리된 데이터를 제거한다. 만약 UpdateNotify가 전달되지 않은 경우는 엑티브의 이중화 데이터가 동기화되지 못한 상황이 된다. 이 경우 스텐바이는 임시 버퍼에 저장된 순서대로 데이터를 복구하면 된다. 따라서 고장이 일어난 경우 스텐바이는 UpdateNotify의 수신 시점을 기준으로 데이터를 복구하면 된다.




[그림 5] 오류 감지와 시스템 복구 시점

이와 같은 이중화 데이터의 입력 및 처리 흐름은 그림 5와 같다.

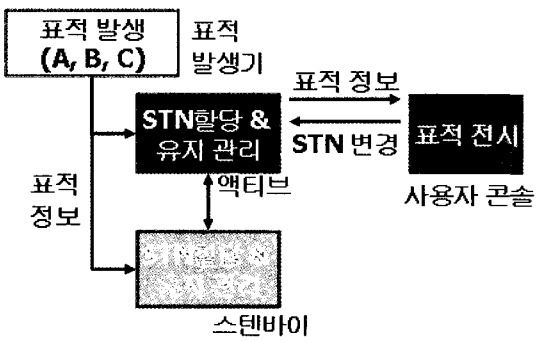
## 7. 실험 및 평가

본 논문에서 제안한 데이터 이중화 기능을 확인하기 위한 실험 환경은 표적 발생기, 정보처리장치, 사용자 콘솔로 구성된다. 실험 환경에 사용한 시스템으로는 인텔 CPU 2.0GHz 기반의 Windows 2000 시스템과 VxWorks MVME 5100 시스템을 사용하였다.

표적 발생기는 표적 정보로 사용할 A, B, C 형의 표적을 각각 생성하고 각각의 표적들을 임의의 속도와 방향으로 이동시킨다. 정보처리장치는 각 표적에 대해 STN(System Track Number)을 할당하고 사용자로부터 요청이 있을 경우 STN을 변경할 수 있어야 한다. 사용자 콘솔은 IPN(Information Processing Node)이 보내는 각 표적을 해당 속성에 따라 아이콘을 출력하고 표적의 세부 정보를 출력할 수 있어야 하며 STN을 변경 요청을 수행할 수 있어야 한다.

표적 발생기에 의해 발생된 표적 정보는 CRS가 운용 중인 정보처리장치에게 전달된다. 이 중 액티브 노드는 입력되는 데이터에 대해 이중화 처리를 수행하고 사용자 콘솔로 전달한다. 사용자 콘솔은 표적 정보를 화면에 출력하고 필요시 표적의 STN을 변경한다. 이와 같은 실험 환경은 그림 6과 같다.

CRS의 이중화 지원 능력의 평가는 크게 이중화 데이터 관리 능력과 이중화 처리 능력으로 나누어 평가하였다. 이중화 관리 능력은 이중화 노드의 상태 감시 능력, 전송 메시지 손실 여부, 이중화 데이터 동기화, 노드/프로세스 단위의 이중화 기능 지원 및 이중화 지원 능력을 평가하였다. 이중화 처리 능력은 이중화 데이터를 처리하는 능력, 프로세스 종료, 네트워크 단절 및 전원 차단에 대해 CRS의 동작 여부를 통해 평가하였다. 실험 결과 CRS는 표 2 이중화 데이터 관리 능력 항목을 모두 만족하였다.




[그림 6] 실험 환경

이중화 데이터 처리 능력을 평가하기 위해 메시지 빈도를 초당 100개에서 700개까지 증가시키면서 실험하였다. 실험 결과, Windows 기반의 시스템의 경우 초당 500개까지는 메시지 손실이 없었으나 이 이상의 경우 메시지를 잃어버리는 경우가 발생하였다. VxWorks 시스템의 경우 초당 400개까지 정상적으로 처리하였으나, 이 이상의 메시지는 일부 손실하는 경우가 발생하였다. 이와 같은 결과는 플랫폼 간 서로 다른 하드웨어와 소프트웨어로 인한 처리 능력의 차이로 판단된다. 하드웨어 성능 차이 요소로는 중앙처리장치, 메모리, 하드디스크 등이 있으며 소프트웨어 요소로는 네트워크 드라이버, 네트워크 스택, 운영체제의 성능 차이로 인한 것으로 분석된다.

[표 2] 이중화 기능 평가 항목

| 평가 내용 | | 비고 |
|---|---|---|
| 이중화 데이터 관리 능력 | · Active/Standby 상태 감시 | |
| | · 프로세스 단위의 이중화 기능 지원 | |
| | · 노드 단위 이중화 기능 지원 | |
| | · Active, Standby 데이터 동기화 | |
| | · 전송 메시지 손실 및 과전송 발생 여부 | |
| | · 운용자 명령 이중화 지원 | |
| 이중화 데이터 처리 능력 | · 이중화 데이터 처리 | 메시지 빈도 100~700 (개/초) |
| | · 프로세스 종료 | |
| | · 네트워크 단절 | |
| | · 전원 차단 | |

## 8. 결 론

과거 개발된 이중화 방안은 장비 성능의 한계와 이를 극복하기 위해 전용 하드웨어 및 소프트웨어를 이용하여 이중화를 구현하였다. 이러한 방법은 환경에 따라 최적의 결과를 얻을 수 있는 장점이 있지만 이를 개발하기 위한 비용 및 향후 유지 보수비용이 증가되는 단점이 있다.

본 논문에서 제안하는 이중화 기법은 최근 증가하고 있는 체계개발 비용 문제를 해결하기 위해 개방형 구조를 도입하여 표준 상용 제품을 사용하였으며 전투체계의 생존성 유지, 전투력 확보 및 개발에 따른 확장성을 만족하기 위해 기 개발된 이중화 방안의 장단점을 분석하고 필요한 기법들을 사용하였다.

전술 응용 프로그램의 필요에 따라 이중화 데이터 공간을 할당하고 관리하는 방법과 이에 따른 데이터 동기화 방법과 고장을 감지했을 때의 절체 방법을 설명하였다.

향후 과제로는 보다 향상된 동기화 메모리의 관리 기법, 이중화 데이터 갱신 기법과 고장 발생 시 엑티브의 자동선출 방법 등이 연구되어야 한다.

## 참 고 문 헌

[1] D. C. Schmidt, "Applying Adaptive Real-Time Middleware to Address Grand Challenges of COTS-based Mission-Critical Real-Time Systems", 1st RT Mission-Critical Systems, Nov. 30, 1999.

[2] 문경록, 김재문, "프로세서 이중화를 통한 로켓 발사통제시스템 시퀀스 컨트롤러 구현", 대한전자공학회, Vol. 26-1, pp.2795~2798, 2003.

[3] 신진욱, 박동선, "핫 스탠바이 스페어링 기법을 이용한 고장 감내 시스템 설계", 한국통신학회, Vol. 29, pp.1113~1122, 2004.

[4] 김종호, 이제헌, "상용 운영체제 기반 이중화 시스템 설계", 한국통신학회, Vol. 25, pp.1104~1114, 2000.

[5] Barry W. Johnson, "Design and Analysis of Fault-Tolerant Digital Systems", Addison-Wesley, 1989.

[6] Naval Surface Warfare Center Dahlgren Division, "Open Architecture Computing Environment Design Guidance", Dahlgren, 2004.