태스크 기반 이중화 방안

정회원 이 종 찬*, 박 상 준**, 강 권 일***

Paper Duplication Method Supported by Task

Jong-chan Lee*, Sang-joon Park**, Kwon-il Kang*** *Regular Members*

요 약

IMT-2000에서 RNC의 Main Control Processor는 호 처리를 담당하는 부분으로, 고신뢰도와 실시간성이 요구되므로 결함 허용 시스템의 연구가 중요하다. 이를 위하여 본 연구에서는 태스크 기반 이중화 방안을 제안한다. 이 방안은 Active side의 태스크들이 메시지 단위로 동작하고, 동작 후 변경된 메모리 영역의 데이터를 Standby side에 전달하는 방식을 기본으로 하며, 절체 시 recovery를 위해 메시지를 logging하는 방식이다. 제안한 방식은 dual down 및 동기화 과정의 복잡성을 제거 할 뿐만 아니라, 태스크가 동기를 제어하므로 좀 더 정확한 동기화가 가능하다. 또한 효과적으로 태스크 기반 이중화를 수행하기 위한 결함 탐지 및 처리 방안을 제시한다. 이 방안은 결함 탐지 확률을 높이고 결함에 의하여 발생한 오류 데이터가 Standby side로 전송되는 것을 원천적으로 차단하는 것에 중점을 둔다.

ABSTRACT

In RNC of IMT-2000, main control processors such as ASP, ACP and OMP are responsible for call control function, and the high reliability and real-time property should be provided for it. So, the study of real-time fault-tolerant for it is needed. In this paper, we proposes an Task based duplication method, in which Tasks in active side operate on message unit and send the updated data to standby side after operation, log in the message to standby side for recovery during take-over. This scheme decreases the dual down and the complexity of synchronization procedure, and performs the synchronization more exactly because Tasks control the synchronization of system.

This paper also proposes the fault detection and the fault handling method for effective implementation of Task based duplication. This scheme focus on increasing the fault detection rate and intercepting originally that fault data is send to standby side.

Ⅰ. 서 론

복잡하고 다양한 형태의 고성능 컴퓨터 시스템이 개발되고, 산업 및 사회 전반에 컴퓨터의 의존도가 높아 감에 따라 컴퓨터 안정성에 대한 관심이 고조되고 있다. 특히 MT-2000시스템의 RNC(Radio Network Controller)[1]는 통신에 필요한 무선 자원을 제어하는 시스템으로, RNC의 MCP(Main Control Processor)는 다른 여타의 장치에 비하여 보다 높은 신뢰성이 보장되어야 하며, MCP의 결함은 해당 프로세서가 관리하는 장치들을 불능 상태로 만들 수 있으므로, MCP의 결함 포용 능력은 RNC의 안정성은 좌우하는 중요한 변수가 된다. MCP는 그 자체로서 높은 견고성을 가지도록 구현하지만, 다소간의 오류 율(fault rate)은 존재할 수 밖에 없으므로 MCP 자체에서 확보할 수 있는 범위 이상의 안정성을 확보하고 서비스 지속성을 달성하려면 MCP를 이중화(duplication), 혹은 다중화

* 한국전자 통신연구원 ** 숭실대학교 컴퓨터학과 *** 현대시스콤

논문번호 : 010240-0905, 접수일자 : 2001년 9월 5일

(Multiplication) 함으로서 활성화(Active)된 프로세서가 장애를 일으키더라고 대기(Standby)중인 프로세서가 연속적인 서비스를 제공해야 한다.

현재 프로세서 이중화를 위하여 활성-대기(Active-Standby), 주-보조(Master-Assistant) 그리고 부하분산(Load-Sharing) 등의 프로세서 이중화 방안이 연구되고 있지만, 안정성 확보를 최우선으로 하는 교환시스템과 같은 장치는 프로세서의 신뢰성을 높이기 위하여 Active-Standby 프로세서 이중화를 일반적으로 사용한다. 이 구조는 두 프로세서에 동일한 기억장소를 부여하고 절체에 대비하는 준비(Ready) 단계와 실제 활성-절체가 발생하는 절체(Take-over) 단계, 절체 후 다시 준비 단계로 들어가기 위하여 대기 프로세서를 복구하는 재준비(Re-ready) 단계를 거친다. 일반적으로 활성-대기-절체를 준비하는 준비 단계 또는 재준비 단계의 수행과정 및 방법에 따라 동시 동작(Concurrent Processing), 동시 쓰기(Concurrent Write) 그리고 상태 체크 포인팅(State Check-pointing)과 같은 기술을 적용하여 시스템을 구현하고 있다. 그러나 이런 기존의 방식들은 동기화 문제, dual down 그리고 이중화 전담 AP(Application Program)의 overhead 문제 등이 시스템을 구현에 많은 영향을 미치고 있다[2-3].

본 연구에서는 상위의 문제들을 근본적으로 해결하기 위하여, 태스크(Task) 기반 이중화 구조를 동기화 측면과 오류 탐지 및 처리 기능 측면에서 제시한다. 이는 태스크가 메시지 단위로 동작을 하면서 메모리의 변경 사항을 Standby side에 전달하는 방식을 기본으로 하며, 절체 시 recovery를 위해 메시지를 logging하는 방식이다. 이 방식은 레지스터 및 프로세서의 상태 정보를 필요로 하지 않고, 메시지 단위로 프로세서간의 동기화를 처리 때문에, 동기화 기능이 간략하고 구현이 용이하다. 또한 오 동작한 내용을 Standby side에 전달하지 않기 때문에 dual down문제도 해결할 수 있다는 장점을 갖는다. 그리고 각 B/D에서 순간적으로 발생하는 오류를 감지하기 위하여 하드웨어적으로 오류를 탐지할 뿐만 아니라 소프트웨어적으로도 오류를 탐지하는 이중 오류 탐지 방안을 제시한다. 하드웨어 탐지 구조는 Back Plane을 통해서 하드웨어적으로 상대 보드(Board)의 오류를 제어 함으로서 오류 발생 탐지 확률을 획기적으로 높였다. 소프트웨어 탐지 구조는 주기적으로 오류 여부를 조사 함으로서 하드웨어 탐지 기능에 오류가 발생했을 경우 등의 부가 기능을 수행한다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문과 관련된 이중화 관련 기술에 대해 간략히 살펴보고, 3장에서는 본 연구에서 제시한 태스크 기반 이중화 방안에 대하여 기술한다. 우선 태스크 기반 이중화 시스템 구조를 제시하고 이를 기반으로 동기화 구조와 장애 탐지 및 처리 구조를 제안한다. 4장에서는 마르코프 모델을 이용하여 제안한 시스템의 성능을 평가한다. 마지막으로 5장에서는 결론 및 앞으로의 과제를 기술한다.

Ⅱ. 관련 연구

프로세서 이중화는 프로세서의 신뢰성을 높이기 위한 가장 강력한 방법으로, 요구되는 안정도에 따라 Active-Standby 프로세서 이중화, Master-Assistant 프로세서 이중화, Load-Sharing 프로세서 이중화로 구분된다[4-5]. Active-Standby 프로세서 이중화 구조는 기능 수행에 필요한 하나의 프로세서 장치, 그리고 이와 동일한 중복(Redundancy)을 설치하고 둘 간의 제어 정보를 송수신하기 위한 제어 채널과 기억장소의 자료 및 Context 정보의 송수신을 위한 데이터 채널 등의 연결 장치를 부가하고 Active 프로세서에서 결함 발생 시 Standby 프로세서가 작업을 이어받아 연속적으로 서비스를 중단없이 수행하는 구조로서 추가 비용이 발생하지만 가장 안전하고 신뢰성 있는 구조이다.

Master-Assistant프로세서 이중화 방식은 Active-Standby프로세서와 같은 대칭(Balanced) 구조가 아니라 성능이나 안정성 면에서 주 프로세스보다 다소 떨어지는 프로세서를 중복시키는 비대칭 구조로서, 보조 프로세서는 주 프로세서에서 결함이 발생했을 때만 활성화되었다가, 주 프로세서가 정상동작을 시작하면 제어를 주프로세서로 넘기고 대기 상태로 되돌아가는 기능을 수행한다. 경제적인 면에서 Active-Standby 프로세서 이중화 구조보다 우수하지만 이중화 구조로서의 신뢰성은 떨어진다.

Load-Sharing프로세서 이중화 구조는 두개의 프로세서가 활성 상태를 유지하면서, 두 프로세서의 상태나 결함 여부에 따라 업무를 양분하여 수행하고 한 프로세서에서 결함이 감지되면 정상 동작중인 프로세서로 부하를 이동시킨다. 이런 구조는 시스템을 관리하거나 능동적인 작업을 수행하는 경우는 적용이 불가능하며 단순작업을 반복적으로 수행하는 장치에 적용 가능한 구조이다.

본 연구에서는 안정성이 가장 우수한 Active-Standby 프로세서 이중화 구조를 사용하며 이 구조는 동시 동작(Concurrent Processing), 동시 쓰기(Concurrent Write) 그리고 상태 체크 포인팅(State Checkpointing), 자동 체크 포인팅(Automatic Checkpointing)과 같은 기술들을 사용하여 구현하고 있다[6-8].

Ⅲ. 태스크 연동 이중화 방법

1. 시스템 구조

이중화 시스템의 프로세서는 RNC의 핵심 제어기(Controller)로 시스템의 안정화를 위하여 이중화 되어야 한다. 이를 위하여 본 연구에서 제안한 이중화 시스템의 구조를 그림 1에 보인다. Active side와 Standby side간에 데이터 및 제어 신호를 신뢰성 있게 전송하기 위하여 이중화 경로(path)는 100base-T Ethernet으로 구축하며 통신 모듈에서 관리한다. 기존의 이중화 방식은 대개 Memory Copy 방식이므로 메모리 액세스 속도가 증가하고 캐쉬(Cache)가 Write-Through 방식이면 전체 성능을 저하시킨다. 본 구조에서는 소프트웨어간의 통신으로 D/B의 내용을 일치 시킨다. 통신 방식으로는 Serial I/F와 Dual Port RAM을 이용한 공유 메모리(Shared Memory) 방식을 사용할 수 있으나, 이는 속도 문제와 모듈 자체의 이중화 문제 때문에 구현하기 어렵다. 따라서 이중화는 직렬 I/F로 구현하고, 100Mbps Fast Ethernet 상으로 전송한다. 해당 속도를 적용하기 위해 PCI BUS에 연결하며 과도한 데이터 전송 시의 CPU 부담을 줄이기 위해 DMA를 이용한다.

이중화 지원 모듈(DSM; Duplication Support Module)은 이중화 모듈의 전체적인 상태 정보(State Information)를 관리하며, 이 정보를 이용하여 다른 모듈을 제어하고, 상위 프로세서와의 통신을 담당한다.

동기화 모듈(SM; Synchronization Module)은 Active side와 Standby side 사이의 상태 동기를 맞추기 위한 모듈로 Active 동기화 모듈(ASM; Active Synchronization Module)과 Standby 동기화 모듈(SSM; Standby Synchronization Module)로 구성된다. ASM은 태스크로부터 백업 데이터(Backup Data)와 덤프 데이터(Dump Data)를 받고, IPC(Inter-process Communication)로부터 메시지 정보(Message Information)를 받아 이들을 직렬화(serialize)하고, 통신 모듈(COM; Communication Module)로 전송한다. SSM은 Active side의 COM에서 전송된 백업 데이터/덤프 데이터/메시지 정보를 받아 처리하고, IPC로부터 Standby side로 입력된 외부 입력 메시지(Input message)의 logging 및 이에 대한 Garbage collection을 수행한다. 입력 메시지 정보는 각 IPC를 통해 받은 메시지의 순서를 결정하는데 사용하며, 백업 데이터는 응용 프로그램들의 메모리 갱신 및 메시지 garbage collection에 사용한다. 데이터 덤프는 Standby가 실장된 초기에 Active와 Standby의 상태를 일치시키기 위한 것으로, 덤프 데이터를 이용하여 응용 프로그램의 메모리를 갱신한다.

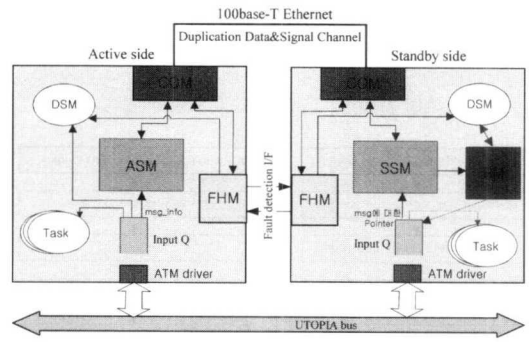


그림 1. 이중화 시스템의 구조

장애 처리 모듈(FHM; Fault Handling Module)은 두 프로세서의 오류를 감지하여 적절한 조치를 취한다. 이때 감지해야 할 오류는 상대 보드의 치명적인 오류와 자기 보드의 자원(Resource)의 상태 변화이다. 상대 보드의 오류 발생을 탐지하기 위한 모듈은 ISR(Interrupt Servive routine)로 구현한다. FD 인터럽트가 발생하면, 이에 해당하는 ISR이 오류 탐지 레지스터(Fault Detection Register)를 읽고, 메시지에 상대 보드의 오류 내용을 담아 이중화 지원 모듈로 전송하여, 적절한 동작을 요구한다.

복구 모듈(RM; Recovery Module)은 Active side로 전환된 보드에서 수행되는 모듈로, 이전 Active side가 수행되었던 곳까지를 follow up한다. 이전의 Active side에서 데이터 백업이 완전히 수행되지 않은 경우, 현재 백업되어 있는 데이터를 바탕으로, 절체된 순간 이전까지 입력 큐(Input Queue)에 저장된 입력 메시지를 처리하여 절체되기 전의 Active가 수행했던 상태를 복구하며, IPC 메시지의 유실을 방지한다.

2. 동기화 구조

태스크 기반 이중화 방안은 태스크를 기반으로 결함 허용 시스템을 구축하기 위해, 태스크를 이용하여 동기화를 수행한다. 제시한 방식은 동기화를 메시지 단위로 수행하므로, 기존 방식에 비하여 태스크의 Context 상태 정보 및 레지스터 값 등이 필요하지 않다. 또한 태스크가 직접 메시지를 처리한 후, 그 메시지에 관한 log 정보를 작성하고 이를 Standby side로 전송하여 동기화 함으로서, 더욱 정확하고 간단하게 동기화를 수행하며 프로그램의 구현도 용이하다. 또한, 오동작으로 인하여 치명적인 장애가 발생할지라도, 오류 정보를 Standby side로 전송하지 않으므로 dual down을 막을 수 있다. 그림 2에 태스크 기반 이중화 과정을 보인다.

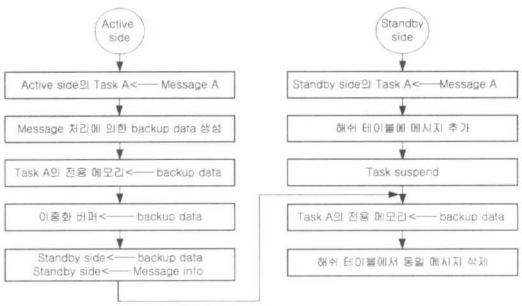


그림 2. 태스트 기반 이중화 방안

태스크 기반 이중화 방식은 Active side과 Standby side에 적재되는 각 태스크들이 초기화 될 때, 이중화 모듈로 자신의 기억장소 영역을 등록한다. 이 과정은 Active side와 Standby side가 동일한 절차로 수행한다. 이후에 Active side 및 Standby side는 동시에 호 메시지를 받는다. Active side에 있는 IPC단은 이 메시지를 해당 태스크로 전송하여 처리하고 동시에 Standby side에 메시지에 대한 정보를 전송한다. 메시지 단위로 시스템을 운영하므로 한 메시지의 처리가 끝나면, 태스크는 메시지 처리 결과로 인해 기억장소의 내용이 변경되었을 경우, 백업 메시지(시작 주소와 크기를 포함한 변경된 내역 및 그 메시지에 대한 정보)를 이중화 모듈로 전송한다. 이중화 모듈은 이를 태스크별로 관리하다가 해당 태스크가 Checkpoint를 호출하면 이를 Standby side의 이중화 모듈로 전송 함으로서 동기화를 수행한다. 태스크에게 Checkpoint를 사용하여 백업 여부를 관리 하게 함으로서 동기화의 융통성을 부여한다.

백업 데이터의 관리는 태스크별로 수행되며, TCB (Task Control Block)를 구축해 백업될 영역에 대한 시작 주소(Start Address)와 크기(Size)를 관리하며, Standby side의 Garbage collection을 위해 최근의 외부 입력 메시지(External input message)에 대한 정보를 저장한다. 태스크의 백업 메시지 와 Checkpoint에 의하여 TCB에 등록되어 있는 일련의 백업 데이터를 이중화 경로로 전달하기 위해 ASM은 일단 패킷(Packet)의 크기(1024-byte)에 맞게 데이터를 분리하고, 이들이 재조합 가능하도록 순서 번호(Sequence Number)를 붙인다. 그리고 COM으로 전달한다. 이중화 경로를 통하여 백업 데이터를 수신한 SSM은 데이터를 재구성하기 위해 Active side로부터 전달된 패킷을 ASM에서와 같이 이중화 모듈의 TCB를 이용하여 관리한다. 즉, SSM에서 데이터가 재구성되기 전까지 Active side로부터 전달된 패킷을 TCB에 매달아 놓는다. 그리고, 데이터의 마지막 패킷이 도착하게 되면, 해당 백업 데이터를 시작 주소와 동일한 메모리 공간에 저장 함으로서 동기를 유지한다. 만약, 마지막 패킷을 전달 받기 전에 절체 된다면, RM에서 이 불완전한 데이터를 삭제한다.

Standby side에서는 IPC단에서 외부 Tx를 막고, 입력된 메시지를 해당 태스크로 전송하지 않고 큐 형태로 저장한다. 그리고 효율적인 메시지 관리를 위하여, 태스크 ID를 키로 하는 해쉬 테이블을 작성하고 각 메시지들을 관리한다. 백업 메시지와 함께 전송된 메시지 정보를 이용하여 해쉬 테이블에서 일치하는 해당 태스크를 조사하고 일치하는 메시지를 찾아 그 메시지와 이전의 모든 메시지를 삭제한다.

3. 오류 탐지 및 처리 구조

각 B/D에서 순간적인 각 side의 오류를 감지할 수 있는 기능은 치명적인 시스템 문제를 피하기 위하여 필요하다. 결함 포용 구조가 이러한 요구에 부흥하기 위하여, 본 시스템에서는 하드웨어적 방법과 소프트웨어적 방법을 병행하여 사용한다.

자기 보드내의 오류 감지, 이중화된 상대 보드의 오류 유무 및 종류를 판별하고 이를 처리하기 위한 FHM의 오류 탐지 구조는 그림 3과 같다. 오류를 탐지하기 위하여 SMI(System Monitoring Interrupt)를 두고, SMI에 의하여 상대방의 오류가 감지되면 그 상태를 저장하고 인터럽트를 발생시키며 자기

보드에서 발생한 오류는 상대 보드로 상태를 알린다.

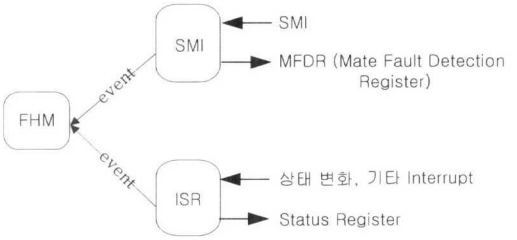


그림 3. 오류 탐지 구조

하드웨어적으로 두 보드는 자신의 상태뿐만 아니라 상대 보드의 상태도 조사함으로서 오류를 탐지하지 못할 가능성을 획기적으로 줄였다. 즉 자기 보드의 상태를 주기적으로 조사할 뿐만 아니라 Back Plane을 통해서 이중화 관련 오류 신호(Fault Signal)를 TTL로 Tx/Rx함으로서 하드웨어적으로 상대 보드의 오류를 감시한다. 이를 통하여 자기 보드 또는 상대 보드가 Active 또는 Standby인지를 확인하고 소프트웨어 절차에 의해 절체하게 된다. 또한 각 보드는 Active indicator인 ACTIVE 레지스터를 두고, 보드에서 오류가 발생하면 하드웨어적으로 즉시 설정한다. 즉 오류가 발생하면 하드웨어적으로 Active/Standby 상태 비트를 즉시 설정하고 해당 처리 루틴을 수행하여 Active side의 상태를 sleep 상태로 만들어 오류 데이터가 전송되는 것을 원천적으로 차단한다.

소프트웨어 탐지 모듈은 주기적으로 오류 발생 여부를 조사 함으로서, 하드웨어적인 오류 탐지가 불완전할 경우의 탐지 기능을 수행한다. 특히 인터럽트가 disable되어 있는 시스템의 초기화 과정 등에서는 상대 보드에 오류가 있어도 인터럽트가 발생하지 않는다. 이러한 경우, FHM에서는 인터럽트로부터의 신호 없이도 소프트웨어 오류 탐지를 이용하여 MSSR(My Fault State Register) 또는 MFSR(Mate Fault State Register)을 조사하여 보드의 오류를 검사한다.

그림 4에 장애 처리 흐름도를 보인다. Active side로 수행중인 보드에서 오류가 발생하면 자기 보드 오류 탐지 모듈 또는 Standby side의 상대 보드 오류 탐지 모듈은 FDI(Fault Detection Interrupt)를 발생시키고, MFSR또는 MSSR에 오류의 원인을 기록한다. FDI가 발생하면, 등록되어 있는 ISR을 호출하여 MFSR 또는 MSSR 을 읽어 보드의 오류의 원인을 찾고, 이를 소프트웨어 처리과정으로 넘긴다. 즉시 ACTIVE 레지스터를 조사하여, 절체했다면 Active 상태 여부에 따라 Recovery(Standby에서 active로 전환) 또는 reinitialize(Active에서 Standby로 전환)를 수행한다. 절체하지 않았다면 오류 여부를 조사하여 reinitialize한다.

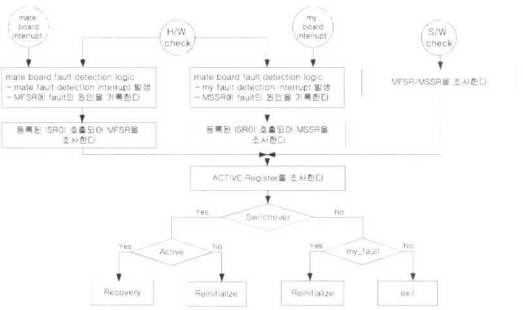


그림 4. 장애 처리 구조

결함 탐지과정을 거친 후, 메시지 처리에 근거한 Recovery를 위하여 Active side와 동일하게 입력되는 IPC 메시지는SSM의 MSGBuf(Message Buffer)에 연결한다. 각 MSGBuf는 태스크 id당 하나씩 할당된다. 이때, IPC 메시지는 IPC로부터 할당 받은 버퍼를 그대로 사용하며, MSGBuf는 IPC 메시지의 포인터만을 관리한다. 그림 5에 MSGBuf와 IPC 메시지와의 연관 관계를 나타낸다.

데이터 백업 시에는, Standby side가 Active side로부터 마지막 패킷의 헤더 정보에 포함된 백업 메시지를 받으면, 데이터의 백업에 관여했던 모든 메시지들을 MSGbuf로부터 삭제해야 한다. 이를 위하

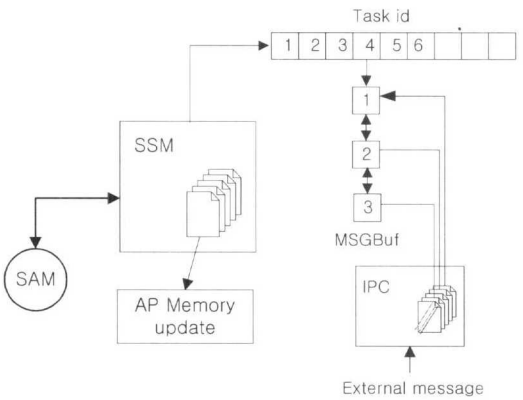


그림 5. 메시지 처리 구조

여 백업된 데이터와 연관된 태스크 id를 찾고, 이 태스크 id로부터 해당 MSGbuf를 찾는다. 그리고, 이 MSGbuf에 연결된 메시지 중에서 데이터의 백업에 관여했던 메시지를 찾고, 이 메시지 이하를 모두 삭제한다. 만약, 백업 메시지를 받지 못하고 절체되어 Active side가 된다면 해당 메시지를 재수행하여 데이터를 복구한다.

Recovery할 때, MSGBuf에 있는 메시지들을 해당 태스크로 전송하고 메시지를 MSGBuf에서 삭제한다. 이 작업은 MSGbuf에 메시지가 없을 때까지 수행한 후 Active side로서 동작한다. 이런 일련의 과정을 통하여, Active side에서는 처리가 됐으나 백업하지 못하여 발생하는 관련 데이터의 손실을 방지한다.

Ⅳ. 성능 평가 및 결과

1. 성능 평가 파라미터

본 연구에서는 기존 방법들과의 성능평가를 위하여 시스템 불가동률 사용하며, 마르코프 상태도를 이용하여 구하였다. 각 모델에 사용되는 천이 변수 식 내의 변수 중 결함의 원인은 표 1과 같이 한정하여 정의하였다.

표 1. 오류의 원인과 파라미터.

파라미터(발생률)	결함 원인
F_b	Board 착/탈장
F_p	power fail
F_r	Reset(H/W, S/W)
F_w	Watch-dog Timer-Out
F_a	ATM link fail
F_m	Management take-over

그 외에 오류 제어 관련 파라미터는 표 2와 같이 정의한다.

2. 성능 모델

그림 6은 제시한 태스크 기반 이중화 모델의 상태 천이도로서 Active side의 결함 발생 시에만 Standby side가 Active로서 동작하는 비정기 절체 방식이다. 각 상태와 천이 파라미터에 대한 정의는 표 3과 표 4에 보인다.

표 2. 오류 제어 파라미터.

파라미터	기능
Pfd	전원 결함 발생 감지 확률
hwda	하드웨어 에러 감지 기능의 정확도
swct	정상 상태 점검 주기
fht	결함 감지 시 기본 에러 처리 시간
reco	Standby side의 데이터 복구 시간
pct	운영요원의 새 모듈 대체 시간
pcp	운영 요원의 결함 발생 모듈 대체 확률
hrdrt	하드웨어 복구 후, 기본데이터의 복구 시간
dct	하드웨어 복구 후, 액티브 모듈의 메모리 내용과 동일하게 데이터를 복구하는 시간
srt	결함 발생 시 하드웨어 자체 복구 시간

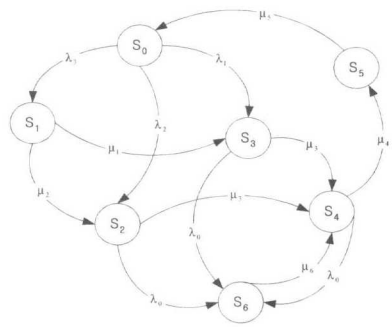


그림 6. 마르코프 상태 천이도

그림 6의 상태천이도로부터 시스템이 각 상태로 존재할 확률은 다음 식들로부터 구한다.

$S_0+S_1+S_2+S_3+S_4+S_5+S_6=1$
$S_0(\lambda_1+\lambda_2+\lambda_3)-S_5\,\mu_5=0$
$S_1(\mu_1+\mu_2)-S_0\,\lambda_3=0$
$S_2(\lambda_0+\mu_3)-S_0\,\lambda_2-S_1\,\mu_2=0$
$S_3(\lambda_0+\mu_3)-S_0\,\lambda_1-S_1\,\mu_1=0$
$S_4(\lambda_0+\mu_4)-S_2\,\mu_3-S_3\,\mu_3.S_3\,\mu_6=0$
$S_5\,\mu_5-S_4\,\mu_4=0$
$S_6\,\mu_6-S_2\,\lambda_0-S_3\,\lambda_0-S_4\,\lambda_0=0$

표 3. 상태 정의 파라미터.

파라미터	기능
S_0	정상 상태
S_1	한 보드의 결함이 탐지가 안된 상태
S_2	한 보드의 결함이 S/W적으로 탐지된 상태
S_3	한 보드의 결함이 H/W적으로 탐지된 상태
S_4	단일 보드로서 동작 준비가 된 상태
S_5	결함 발생 보드의 하드웨어가 복구된 상태
S_6	두 보드 결함 발생으로 시스템 정지 상태

표 4. 상태 천이 파라미터.

파라미터	기능
λ_1	결함 발생 시 하드웨어적으로 감지될 확률
λ_2	결함 발생 시 소프트웨어적으로 감지될 확률
λ_3	결함 발생 시 에러가 즉시 감지되지 않을 확률
λ_0	두 모듈이 연속적으로 결함을 발생할 확률
μ_1	감지못한 결함을 하드웨어적으로 발견할 확률
μ_2	감지못한 결함을 소프트웨어적으로 발견할 확률
μ_3	한 모듈의 결함 발견 후, 나머지 모듈이 단일 모듈로서 동작할 확률
μ_4	결함 발생 모듈 하드웨어의 완전 수리 확률
μ_5	결함 발생 모듈의 정상 모듈과의 내용일치확률
μ_6	두 모듈의 결함에서 정상 상태로 복구될 확률

상위의 각 상태로 존재할 확률을 이용하여 시스템 불가동률 S_w을 다음과 같이 구할 수 있다.

$$S_w = S_2 + S_3 + S_6$$

천이 변수식은 다음과 같다.

$$\lambda_1 = F_b + F_r + F_w + F_a + F_m + F_p * Pfd$$

$$\lambda_2 = F_b + F_r + F_w + F_a + F_m + F_p * Pfd$$

$$\lambda_3 = F_b + F_r + F_w + F_a + F_m + F_p * (1-Pfd)$$

$$\lambda_0 = \lambda_1 + \lambda_2 + \lambda_3$$

$$1/\mu_1 = hwda/2$$

$$1/\mu_2 = swct/2$$

$$1/\mu_3 = fht + reco$$

$$1/\mu_4 = pct * pcp + srt * (1-pcp)$$

$$1/\mu_5 = dct$$

$$1/\mu_6 = fht + 1/\mu_4((1/\mu_4 + hrdrt) + (0.5 * 1/\mu_4 + hrdrt) + 0.5 * hrdrt)$$

3. 평가 결과

그림 7은 결함 지속 시간의 변화에 따른 불가동률을 나타내고 있다. CP-based는 결함지속시간이 증가할수록 각 태스크의 context가 불일치할 가능성이 증가하고 이에 따라 동기화의 복잡성이 증가하여 불가동률이 증가하게 된다. 또한 CW-based는 결함 지속 시간이 증가할수록, Active side의 오동작이 Standby side로 전이될 확률이 높아지고 이에 따라dual down이 발생활 확률이 높아지므로 불가동률이 증가하게 된다. 이에 반하여 Task-based는 자체적인 오류탐지 기능의 정확성, 그리고 메시지 단위의 동기화 등으로 인하여 우수한 성능을 보인다.

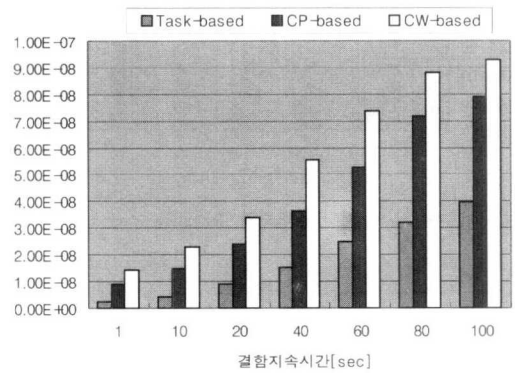


그림 7. 결함지속시간의 변화와 불가동률

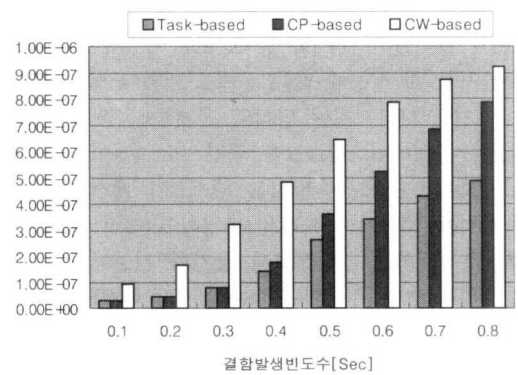


그림 8. 결함발생빈도수의 변화와 불가동률

그림 8은 결함 발생 빈도 수의 변화에 따른 불가동률을 나타내고 있다. Task-based는 메시지 logging 방식이므로 결함이 전이될 확률이 적고 하드웨어 또는 소프트웨어적으로 감지되지 않은 결함을 발견할 확률이 높다. 이에 따라 정상상태 복구확률이 증가 함으로서 불가동률이 두 기법에 비하여 향상됨을 알 수 있다.

그림 9는 데이터 복구 시간의 변화에 따른 불가동률을 나타내고 있다. CP-based는 동기화의 복잡성으로 인하여 데이터 복구 시간이 증가하지만 우수한 성능을 보인다. Task-based는 메시지를 logging하여 데이터를 복구하므로 데이터복구시간이 증가할수록 CP-based와 거의 동일한 성능을 유지함을 알 수 있다.

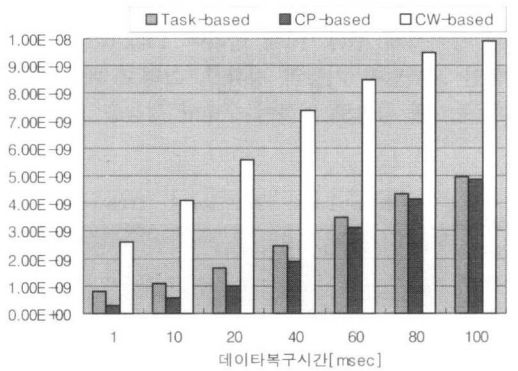


그림 9. 데이터복구시간의 변화와 불가동률

V. 결 론

호 처리 및 무선 자원 관리를 담당하는 RNC와 같은 중요한 장치는 프로세서의 신뢰성을 높이기 위하여 Active-Standby 프로세서 이중화를 일반적으로 사용한다. 그러나 이를 구현하기 위하여 적용되는 기존의 방식들은 동기화 문제, dual down 문제 등 실제 구현 시 성능에 많은 문제를 가지고 있었다.

본 연구에서는 기존 방식의 문제들을 근본적으로 해결하기 위하여, 태스크를 근간으로 한 이중화 구조를 제시하였다. 이 방식은 태스크가 메시지 단위의 동작에 의하여 메모리를 변경하면 즉시 Standby side에 전달하는 방식으로서 동기화 기능이 간략하고 구현이 용이하며 dual down문제도 해결할 수 있었다. 또한 이중 오류 탐지 방안을 제시하였다. Back Plane을 통해서 하드웨어적으로 상대 보드의 오류를 제어하고 하드웨어 탐지 기능에 오류가 발생했을 경우 등의 부가 기능을 소프트웨어 탐지 기능이 수행함으로서 오류 발생 탐지 확률을 획기적으로 높였다.

본 구조는 태스크간의 연동을 기반으로 결함 포용 시스템을 구축함으로, 태스크간의 긴밀한 인터페이스가 필수적이다. 따라서 태스크와 IPC간 인터페이스 그리고 태스크간의 인터페이스를 좀 더 효율적으로 구성하는 방안을 향후에 제시해야 할 것이다.

참 고 문 헌

[1] Kap-Dong Kim. "The NPA Fault-Tolerance Method of IMT-2000 BSC," The 4th CDMA International Conference & Exhibition, vol. 2, PP. 429-433, 1999

[2] S.J. Upadhyyaya and K.K. Saluja "An Experimental Study to Determine Task Size for Rollback Recovery Systems," IEEE Transactions on Computers, vol. 37, no. 7, PP. 872-877, 1988

[3] A.M. Tyrrell and G.F. Carpenter, "CSP Methods for Identifying Atomic Actions in the Design of Fault Tolerant Concurrent Systems," IEEE Trans.on SE,Vol.21,No.7,pp.59-68,Jul. 1995

[4] M. Russinovich, and Z. Segall, "Fault Tolerance for Off-The-Shelf Applications and Hardware," Proc.of FTCS-25,pp.67-71,Jun.1995

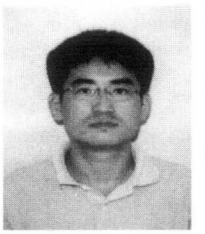
[5] 여환근외 4인, "고장 감내형 고성능 프로세서 시스템 구조에 관한 연구," 한국전자통신연구원 위탁보고서, 명지대학교, Dec. 1996.

[6] 양승민외 3인, "BSC용 제어 프로세서의 실시간 미들웨어에 관한 연구," 한국전자통신연구원 위탁보고서, 숭실대학교, Dec. 1999.

[7] 김상하외 4인, "Fault Tolerance(duplex) 방식 연구," 한국전자통신연구원 위탁보고서, 충남대학교, Dec. 1999.

[8] 박동선외 6인, "프로세서 시스템의 고가용성 및 고신뢰성 구현에 관한 연구," 한국전자통신연구원 위탁보고서, 전북대학교, Nov, 1999.

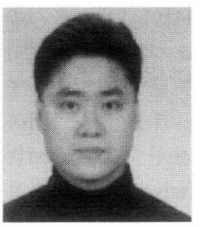
이 종 찬(Jong-chan Lee) 정회원



1994년 2월 : 군산대학교 컴퓨터과학과 졸업
1996년 8월 : 숭실대학교 전자계산학과 석사
2000년 8월 : 숭실대학교 전자계산학과 박사

2000년 10월~현재 : 한국전자 통신연구원 선임연구원
<주관심 분야> Duplex system, Mobile Tracking

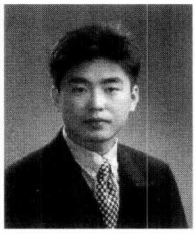
박 상 준(Sang-joon Park) 정회원



1996년 2월 : 동국대학교 전산학과 졸업
1998년 2월 : 숭실대학교 컴퓨터학과 석사
1998년 3월~현재 : 숭실대학교 컴퓨터학과 박사과정

<주관심 분야> 위치 관리, 이동 Ad Hoc 네트워크

강 권 일(Kwon-il Kang) 정회원



1997년 2월 : 대전대학교 컴퓨터공학과 졸업
1999년 2월 : 성균관대학교 전기전자 및 컴퓨터공학과 석사
1999년 3월 ~ 현재 : 현대시스콤 연구원

<주관심 분야> Duplex system, Neural Networks