IMT-2000 시스템에서 확장된 Warm-standby sharing을 이용한 프로세서 이중화

이 종 찬[†] · 강 권 일[††] · 이 경 준[†††]

요 약

IMT-2000에서 RNC의 MCP는 호 처리를 담당하는 부분으로, 신뢰도와 실시간성이 요구된다. MCP는 높은 견고성을 갖도록 구현되지만 다소간의 오류 율(Fault late)은 존재할 수밖에 없으므로 프로세서를 이중화하여 활성화된 프로세서가 장애를 일으키더라도 대기중인 프로세서가 연속적인 서비스를 제공할 수 있어야 한다. Warm standby sharing에 비하여 Hot standby sharing은 데이터 손실이 없고 오류 데이터가 확산되지 않는 등의 다수의 장점을 갖지만 동기화 문제로 인하여 이를 시스템에 실제로 구현하는 것은 어렵다. 따라서 본 연구에서는 Hot standby sharing에 비하여 기존의 Warm standby sharing이 갖는 동기화의 장점에 데이터 손실 및 거짓 데이터의 확산 문제를 개선 함으로서, 실제 구현의 용이성 및 성능 향상이라는 결과를 얻으려 하였다.

The Processor Duplication using Extended Warm-Standby Sharing in IMT-2000 System

Jongchan Lee[†] · Kwonil Kang[††] · Kyung-Jun Lee[†††]

ABSTRACT

High reliability and real-time response are required in the Main Control Processor of Radio Network Controller for IMT-2000. In spite of its robustness some fault rates is inevitable, so the processors are duplicated for non-interrupted service and services are switched to the standby processor in case of faults. The hot-standby sharing scheme has advantages of no data loss and non-proliferation of error data in comparison with the warm-standby sharing scheme but it has difficulties with implementation due to synchronization problem. This paper proposes an expanded warm-standby sharing scheme based on a novel massage processing and error detection mechanism for improving performances.

키워드 : 이중화 프로세서, MFSR, Worm-standby, TCB, 메시지 처리(Message processing)

1. 서 론

산업 및 사회 전반에 무선 및 유선 통신의 의존도가 높아 감에 따라 복잡하고 다양한 형태의 고성능 통신 시스템이 개발되고 있고, 한 부분으로서 통신 시스템의 안정성에 대한 관심이 고조되고 있다. 특히 개발 중에 있는 IMT-2000시스템에서 RNC(Radio network Controller)는 무선 자원을 제어하는 시스템으로 RNC를 구성하는 MCP (Main Control Processor)의 결함 포용 능력은 전체 시스템의 안정성을 좌우하는 중요한 변수가 된다. 시스템의 안정성 확보 및 서비스 지속성 같은 고신뢰성을 달성하려면, MCP를 이중화 혹은 다중화 함으로써 활성화된 프로세서가 장애를 일으키더라도 대기중인 프로세서가 연속적인 서비스를 제공할 수 있도록 해야 한다. 이는 교환시스템으로의 호 처리 요청이 끊임없이 이루어지기 때문에 시스템의 일시적인 동작 중지는 사용자들에게 극도의 혼란을 야기시킬 수 있기 때문이다. 그러므로 기존의 교환시스템에서는 결함 감내 구조로서 Standby sharing 기법을 사용하고 있다.

Standby sharing 기법은 크게 세가지로 구분되는 데 Cold standby sharing, Warm standby sharing, Hot standby sharing이다[1-3]. Cold standby sharing 기법은 Standby side가 결함 발생으로 인해 Active 상태로 전환되기 전까지 전원 공급이 중단되어 있으므로 Active 기능을 수행하기까지에는 다소의 시간이 걸린다. 그러므로 고가용성을 요구하는 교환기의 결함 감내 시스템에는 이 기법이 적합하지 않다. Warm standby sharing 기법은 Active side가 시스템 정상 동작 시에 Concurrent write 방식을 사용하여 그 자신의 메모리 내용과 Standby side의 메모리 내용이

† 정 회 원 : 전자통신연구원 무선트래픽접속연구팀
†† 준 회 원 : 현대전자 통신시스템 연구 13팀
††† 정 회 원 : 삼성전자 IMT-2000 시스템 연구팀
논문접수 : 2001년 5월 4일, 심사완료 : 2001년 7월 5일

동일하도록 시스템을 동작시키기 때문에, 결함 발생 시에 Standby side가 Active 상태로 전환되고 본래의 정상 기능을 수행하는데 걸리는 시간은 Cold standby sharing 기법보다 매우 짧다. 그러나 결함의 종류와 정도에 따라서 다소의 데이터 손실(Data loss)이 발생하며 Active side와 Standby side간에 주기적인 결함 점검이 요구된다. Hot standby sharing 기법에서는 두 side가 active 하게 동작하므로 정상 동작 시 모든 시스템 모듈의 상태와 내용을 시스템 동기화를 통해서 동일하게 유지해야 한다. 이 기법에서는 외부 시스템과의 데이터 교환 시 한 모듈만이 Active로서 동작하게 된다. 임의의 한 모듈에서 결함이 발생하면, 결함 모듈을 시스템으로부터 제거하고 수행 중인 일을 계속 진행시킬 수 있으므로 결함 감지로부터 시스템 정상기능 재가동까지 걸리는 시간이 극히 짧다. 또한 단일 시스템 결함으로부터의 데이터 손실이 발생하지 않는 결함 감내 구조의 설계가 가능하다. 반면에 Warm Standby에 비하여 정상 동작 중에 결함 감내 시스템 모듈간의 동기화 유지가 어렵고 결함에서 복구된 시스템 모듈의 재정상 가동 등의 구현의 어려움이 해결해야 할 큰 문제점으로 지적되고 있다[4-6].

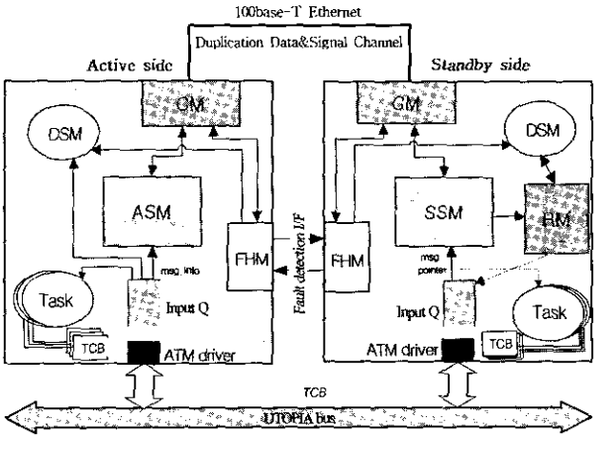
오늘날 고가용성을 지원하는 시스템들은 Warm 및 Hot standby 결함 감내 구조를 기본으로 채택하고 있다. Warm standby 구조는 시스템 구현이 Hot standby 구조에 비해 훨씬 용이하나 결함의 종류와 정도에 따라서 다소의 데이터 손실이 발생하며 현재 동작중인 모듈상에 감지되지 못한 오류 데이터(Fault data)가 궁극적으로 전체 시스템으로 확산되는 문제점을 지니고 있다. 그러나 현재 개발되어 사용되고 있는 AT&T사의 ESS-5 교환기, ETRI의 TDX-10 등의 대부분의 교환기들은 결함 감내 구조로서 Warm standby sharing 기법을 적용하고 있다. 그 이유는 결함 감내 구조로서 Hot standby Sharing 기법을 사용하는 교환 시스템은 데이터 무손실 등 많은 장점을 가지고 있으나, 동기화의 복잡성으로 인하여 실시간 교환 시스템으로의 구현 시에 이론적인 성능을 얻는 것이 어렵기 때문이다[7-10].

본 논문에서는 기존의 Warm standby 구조가 갖는 데이터 손실 및 오류 데이터의 확산 문제를 해결 함으로서 구현의 용이성과 구현 비용 그리고 성능의 향상이라는 결과를 얻고자 하였다. 실제로 시스템에 결함 포용을 실현함에 있어서, Hot standby가 갖는 이론상의 장점에서 벗어나 시스템으로의 구현이라는 문제에 직면하여 안전도와 신뢰도라는 측면 그리고 구현의 용이성이라는 측면을 비교하지 않을 수 없었다. 본 연구에서는 이중화 구조를 갖는 RNC에서 확장된 Warm standby sharing 결함 감내 구조를 구현함으로써 고성능 및 고가용성을 유지하는 시스템 개발에 기여하는 것을 목표로 하고 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문과 관련된 이중화 시스템 구조에 대해 간략히 살펴보고, 3장에서는 본 연구에서 제안한 확장된 Warm-standby sharing을 제시한다. 우선 하드웨어에 기반한 제어 구조를 제시하고 소프트웨어에 기반한 처리 구조를 제안한다. 4장에서는 마코프 모델을 이용하여 제안한 시스템의 성능을 평가한다. 마지막으로 5장에서는 결론 및 앞으로의 과제를 기술한다.

2. 시스템 구조

이중화 시스템의 프로세서는 RNC의 핵심 Controller로 System의 안정화를 위하여 이중화 되어야 한다. 이중화를 위한 Active side와 Standby side간에 데이터 및 제어 신호를 신뢰성 있게 전송하기 위하여, 이중화 path는 100 base-T Ethernet으로 구축하며 통신 모듈에서 관리한다. 동기화를 위하여, 소프트웨어 간의 통신으로 D/B의 내용을 일치 시킨다. 통신 방식으로는 직렬 인터페이스(Serial Interface)로 구현하고, 100Mbps Fast Ethernet 상으로 전송한다. (그림 1)에 본 연구에서 제시한 이중화 시스템 구조를 보인다.



(그림 1) 이중화 시스템의 구조

이중화 지원 모듈(DSM; Duplication Support Module)은 이중화 블럭의 전체적인 상태(state)를 가지고 있으며, 이를 이용하여 다른 module을 control하고, 상위 프로세서와 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 구성되며 Active와 Standby에서 서로 다르게 동작한다. 동작 mode는 control module에서 지정하는 "system state"를 참조한다. ASM에서는 응용 프로그램(AP; Application Program)으로부터 Backup data와 Dump data를 받고, IPC로부터는 Message information을

받아 이들을 직렬화(serialize)하여, 통신 모듈(COM; Communication Module)로 전송한다. SSM은 Active side에서 전송된 Backup data/Dump data 및 Message info를 COM으로부터 받아 처리하고, IPC로부터 Standby side로 입력된 외부 Input message를 logging 및 이에 대한 Garbage collection을 수행한다. Input message information는 각 IPC (Inter-process Communication)를 통해 받은 메시지의 순서를 결정하는데 사용하며, Backup data는 응용 프로그램들의 Memory update 및 Message garbage collection에 사용한다. Data dump는 Standby가 실장된 초기에 Active와 Standby의 상태를 일치시키기 위한 것으로, Dump data를 이용하여 응용 프로그램의 메모리를 갱신한다.

장애 처리 모듈(FHM; Fault Handling Module)의 역할은 오류를 감지하여 적절한 조치를 취하는 것으로 이때 감지해야 할 오류는 상대 B'd(Board)의 치명적인 오류와 자기 B'd에서의 자원의 상태 변화이다. 상대 B'd의 오류 발생을 탐지하기 위한 모듈은 ISR(Interrupt Servive routine)로 구현한다. FD 인터럽트가 발생하면, 이에 해당하는 ISR이 FD register를 읽고, 메시지에 상대 B'd의 오류 내용을 담아 이중화 지원 모듈로 전송하여, 적절한 동작을 요구한다.

현재 구축중인 이중화 방식은 Active side과 Standby side에 적재되는 각 AP는 초기화 시에 이중화 모듈로 자신의 기억장소 영역을 등록한다. 이 과정은 Active side와 Standby side가 동일한 절차로 수행한다. 이후에 Active side 및 Standby side는 동시에 Call message를 받는다. Active side에 있는 IPC단은 이 메시지를 해당 AP로 전송하여 처리되도록 하고 동시에 Standby side에 메시지에 대한 정보를 전송한다. 그리고 메시지 처리 결과로 인해 기억장소의 내용이 변경되었다면 Backup message(변경된 내역 및 그 메시지에 대한 정보)를 이중화 모듈로 전송한다. 이중화 모듈은 이를 AP별로 관리하다가 그 AP가 checkpoint를 호출하면 이를 Standby side의 이중화 모듈로 전송한다. Checkpoint를 두는 이유는 AP가 Backup message를 관리할 수 있도록 융통성을 부여하는 것이다.

Standby side에서는 IPC단에서 외부 Tx를 막고, 외부로부터 입력된 메시지에 대해서도 해당 AP로 전송하지 않고 큐 형태로 저장한다. 그리고, 효율적인 메시지 관리 차원에서 각 메시지들에 대해서 AP ID를 키로 하는 해쉬 테이블을 작성하여 관리한다. Active side에서 메시지에 대한 정보를 보내오면 이를 AP별로 관리하는 해쉬 테이블에 삽입하고, Data backup message를 받으면 자신의 기억장소에 이를 반영한다. 그리고, Backup message와 함께 전송되어 온 메시지 정보를 이용해서 해쉬 테이블에서 일치하는 해당 AP를 조사해서 일치하는 메시지를 찾아 그 메시지까지 이전의 모든 메시지를 삭제한다.

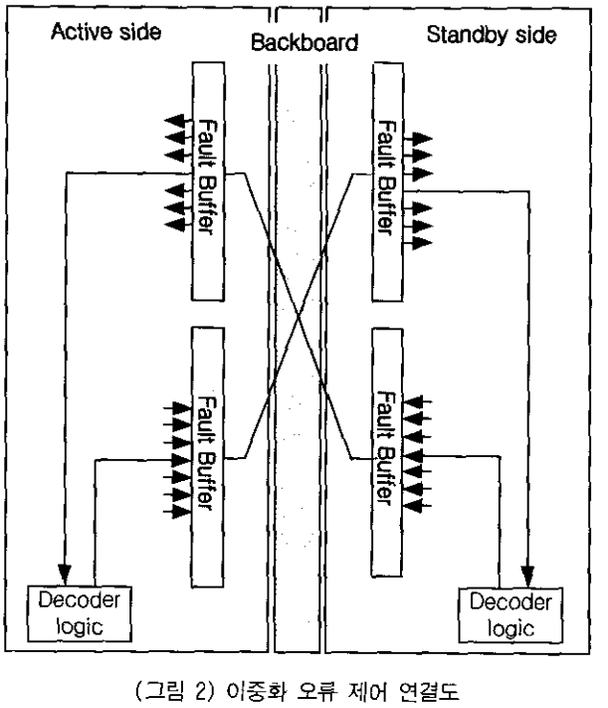
복구 모듈(RM; Recovery Module)은 Active side의 오류를 감지한 후, Standby side였던 B'd에서 수행되는 모듈로, 이전 Active side가 수행되었던 곳까지를 follow up한다. 이전의 Active side에서 Data backup이 완전이 수행되지 않은 경우, 새로이 Active가 된 side는 현재 백업되어 있는 데이터를 바탕으로, 절체된 순간 이전까지 Input queue에 저장된 Input message를 처리하여 절체되기 전의 Active가 수행했던 상태를 복구하며, IPC message의 유실을 방지한다.

3. 확장된 Warm-standby sharing

본 연구에서는 하드웨어적 측면과 소프트웨어적 측면에서 데이터 손실과 오류 데이터의 확산 문제를 해결하려 하였다. 하드웨어적 측면에서는 새로운 오류 탐지 구조 제시하고 소프트웨어적 측면에서는 새로운 메시지 처리 구조를 제시한다.

3.1 하드웨어에 기반한 제어 구조

(그림 2)는 데이터 손실 및 오류 데이터의 전체 시스템으로의 확산 문제를 해결하기 위하여 본 연구에서 제안한 이중화 결함 제어 연결도이다. 본 구조에서는 두 B'd가 자신의 상태뿐만 아니라 상대 B'd의 상태를 조사 함으로서 오류 발생을 탐지하지 못할 가능성을 획기적으로 줄였다. 즉 자기 B'd의 상태를 주기적으로 조사할 뿐만 아니라 Back Plane을 통해서 이중화 관련 Fault signals을 TTL로 Tx/Rx함으로서 하드웨어적으로 상대 B'd 오류를 Monitoring한다. 이를 통하여 자기 B'd 또는 상대 B'd가 Active

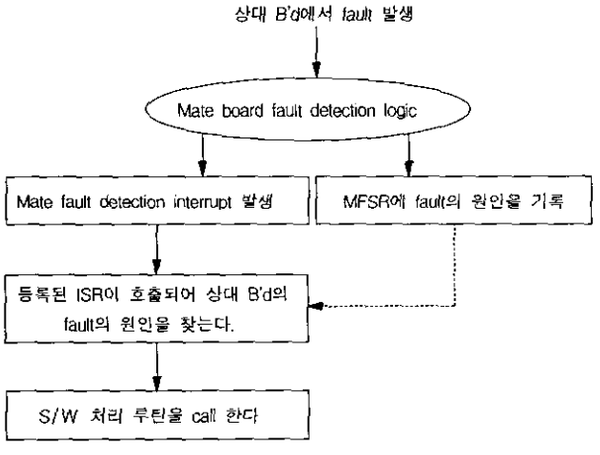


(그림 2) 이중화 오류 제어 연결도

또는 Standby 인지를 확인하고 소프트웨어에 의해 절체하게 된다.

이중화 절체는 B'd 자체적으로 검출되는 내부 상태 정보와 이중화로 구성된 상대 B'd로부터 보고되는 상태정보를 종합하여 활성 상태를 결정한다. 이를 위하여 Memory mapped register로서, 상대 B'd의 상태가 하드웨어적으로 setting되는 MFSR(Mate Fault State Register)과 자기 B'd의 상태가 하드웨어적으로 설정되는 MSSR(My Fault Status Register)을 둔다.

(그림 3)은 상대 B'd의 오류를 탐지하고 처리하는 과정을 상세화 한 것이다. Active side로 수행중인 상대 B'd에서 오류가 발생하면 상대 B'd 오류 탐지 logic은 MFDI(Mate fault detection interrupt)를 발생시키고, MFSR에 오류의 원인을 기록한다. MFDI가 발생하면, 등록되어 있는 ISR (Interrupt service routine)이 호출되어 MFSR을 읽어 상대 B'd의 fault의 원인을 찾고, 이를 소프트웨어 처리과정으로 넘긴다. 인터럽트가 disable되어 있는 system의 초기화 과정 등에서는 상대 B'd에 오류가 있어도 인터럽트가 발생하지 않는다. 이러한 경우, 제어 모듈에서는 인터럽트로부터의 호출없이도 MFSR을 access해서 상대 B'd의 오류를 검사한다.



(그림 3) 상대 B'd 오류 처리 과정

오류 데이터의 확산 문제를 해결하기 위하여 각 B'd는 Active indicator인 ACTIVE register를 두고, B'd에서 오류가 발생하면 하드웨어적으로 즉시 setting 한다. 즉 오류가 발생하면 하드웨어적으로 Active/Standby 상태 비트를 즉시 setting하고 해당 처리 루틴을 수행하여 Active side의 상태를 sleep 상태로 만들어 오류 데이터가 전송되는 것을 원천적으로 차단한다. 관련 동작은 다음과 같다.

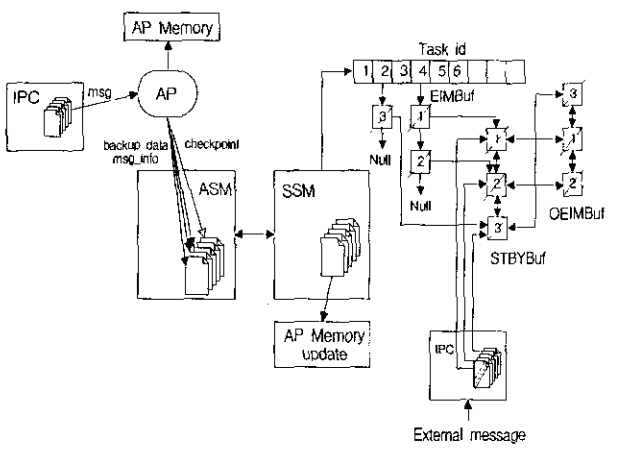
1. Active side와 Standby side에는 MFSR과 MSSR이 각각 존재한다.
2. 상대 B'd의 오류가 감지되면 그 상태를 MFSR에 저장하고 인터럽트를 발생시킨다.
3. 자기 B'd에서 오류가 감지되면 MSSR에 저장하고 상대 B'd로 상태를 알린다. 그리고 인터럽트를 발생시킨다.
4. Active/Standby 상태 비트를 setting한다.
5. setting된 정보를 바탕으로 바로 Active 상태는 sleep로 들어가고 Standby는 Active대기 상태로 전환되어 데이터의 교환을 차단한다.

3.2 소프트웨어에 기반한 제어 구조

Data backup 과 Recovery 시 데이터 손실 및 오류 데이터의 확산을 막기 위한 방안으로, 메시지의 삭제 및 재 구성을 위한 메시지 처리 구조를 제안한다. 이 구조는 EIMBuf (External Input Message Buffer), OEIMBuf(Ordered External Input Message Buffer) 그리고 STBYBuf(Stnadby Buffer)로 구성된다. Active side와 동일하게 전달된 IPC message는 SSM의 EIMBuf에 연결하고, OEIMBuf를 구성한다. IPC에서 전달된 메시지를 저장하고 있는 buffer인 EIMBuf 를 두고 AP id를 key로 갖는 hash table로 관리한다. 각 EIMBuf는 AP id당 하나씩 할당되며, 해당 AP id를 목적지(destination)로 갖는 IPC message를 STBYbuf를 통해 저장한다. 이때, IPC message는 IPC로부터 할당 받은 buffer인 IPCBuf를 그대로 사용하며, EIMBuf는 IPC message의 포인터만을 관리한다. OEIMBuf는 AP로의 Input injection order를 저장하기 위해 사용되는 buffer로서 AP의 우선순위에 의하여 바뀌는 메시지의 처리 순서를 일치시키기 위하여 STBYbuf를 Active side의 처리 순으로 pointing 함으로서 전체 메시지를 관리한다. STBYBuf는 Standby side로 전송되었지만 수행되지 않은 IPC message에 대해 순서적으로 포인터를 저장한다.

TCB(Task Control Block)가 오류 확산의 완충작용으로 확산 문제를 해결한다. 메시지 처리 결과로 생성된 Backup data의 저장을 Active side로부터 최종 데이터가 전송될 때까지 지연한 후, Standby side에 최종 데이터가 도착하면 해당 AP 메모리에 데이터를 저장한다. 마지막 데이터가 수신될 때까지 TCB에서 이 Backup data를 관리하고, 마지막 데이터가 수신되기 전에 Active side에 오류가 발생한다면 해당 데이터를 TCB에서 삭제한다. 이를 위하여, Backup data의 관리는 AP별로 수행되며, TCB를 구축해 백업될 영역에 대한 Start address와 size를 관리하며, Standby side의 Garbage collection을 위해 최근의 External input message에 대한 정보를 저장한다. AP의 Backup message 와 Checkpoint에 의하여 TCB에 등록되어 있는 일련의 Backup data를 이중화 path로 전달하기 위해 ASM은 일단 패킷(packet)의 크기(1024-byte)에 맞게 데이터를 분리하고, 이들이 재조합 가능하도록 Sequence number를 붙인다. 그리고 COM으로 전달한다. 이중화 path를 통하여

Backup data를 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 Backup data를 Start address와 동일한 메모리 공간에 저장 함으로서 동기를 유지한다. 만약, 마지막 패킷을 전달 받기 전에 절체가 된다면, 이 불완전한 데이터를 삭제 함으로서 오류 데이터의 확산을 방지한다. 이를 위하여 Data backup 과 Recovery 시 Data loss 및 Fault data의 확산을 막기 위한 방안으로, 메시지의 삭제 및 재구성을 위하여 제안된 메시지 처리 구조가 (그림 4)에 보인다.



(그림 4) SSM의 메시지 처리 동작

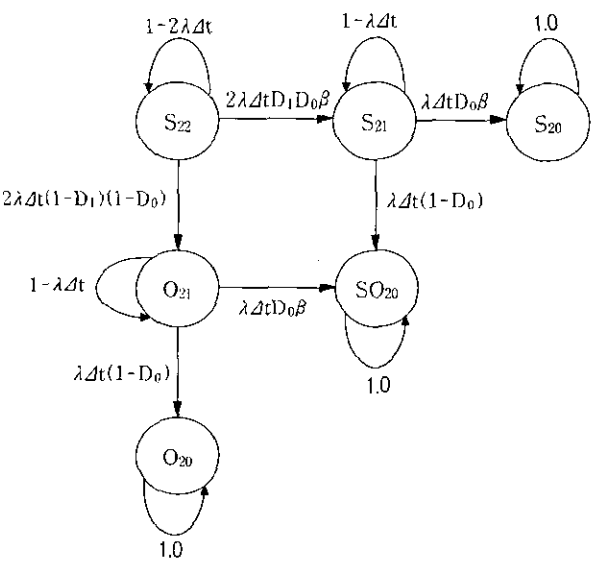
Data backup 시에는, Standby side가 Active side로부터 마지막 패킷의 헤더 정보에 포함된 Backup message를 받으면, Data backup에 관여했던 모든 메시지들을 EIMbuf로부터 삭제해야 한다. 이를 위하여 백업된 데이터와 연관된 AP id를 찾고, 이 AP id로부터 해당 EIMbuf를 찾는다. 그리고, 이 EIMbuf에 연결된 메시지 중에서 Data backup에 관여했던 메시지를 찾고, 이 메시지 이하를 모두 free 시킨다. 만약, Backup message를 받지 못하고 절체되어 Active side가 된다면 해당 메시지를 재수행하여 데이터를 회복한다.

Data backup 동안에 발생하는 데이터 손실 문제를 해결하기 위하여, 백업의 완료 시점을, 메시지 처리 결과로 생성된 데이터들이 Standby side의 해당 AP의 메모리에 모두 저장되는 시점으로 보고, 저장 완료 후에만 Standby side의 동일 메시지를 삭제 가능하게 한다. 백업하지 못하고 절체 한다면 이 메시지를 재수행함으로서 손실된 데이터를 복구한다.

절체시, OEIMbuf를 참조하여 남아있는 메시지를 STBY buf에서 찾고, STBYbuf와 연결된 EIMbuf에서 해당 AP id를 찾은 후, 실제 IPCBuf에 있는 메시지들을 해당 AP로 전송하고 STBYbuf와 OEIMBuf 의 해당 메시지를 IPC Buf에서 삭제한다. 이 작업은 OEIMbuf에 메시지가 없을 때까지 수행한 후 Active side로서 동작한다. 이런 일련의 과정을 통하여, Active side에서는 처리가 됐으나 백업 받지 못함으로서 발생하는 관련 데이터의 손실을 방지한다.

4. 이중화 프로세서에 대한 마코프(Markov) 모델

B'd에서 오류가 발생할 확률 λ는 지수분포에 의한 hazard 함수 $Z(t)=\lambda$이다. 상대 보드 결함 탐지 확률(Mate Fault Detection Probability) D_1이며, 자기 보드 상태 탐지 확률(My Fault Detection Probability)은 D_0, 시간 증가 인자는 Δt 그리고 결함 회복 확률은 β, 여기서 결함 회복 확률은 오류를 탐지한 후에 손실 없이 시스템을 정상 상태로 회복시킬 수 있는 확률을 말한다. 제안된 시스템의 마코프 모델은 (그림 5)와 같다.



(그림 5) 확장된 Warm-standby의 Markov 모델

본 연구에서 제시한 시스템은 Markov 모델을 적용할 경우 6개의 시스템 상태를 나타낸다. 시간 $t+\Delta t$에서 시스템이 임의의 상태 S일 확률은 시스템이 임의의 상태에서 상태 S로 전이할 수 있는 확률과 자신의 상태를 계속 유지할 확률로 정의할 수 있다. 이 경우, 제안된 모델을 수식화하면 다음과 같이 정의할 수 있다.

상태 $P_{S_{22}}(t+\Delta t)$는 두 B'd가 하드웨어적으로 완전한 상태로서 호 처리 및 백업이 완전하게 수행중인 상태를 나타낸다. 이는 Markov 모델에 의하여 다음과 같이 나타낼 수 있다.

$$P_{S_{22}}(t+\Delta t)=(1-2\lambda\Delta t)P_{S_{22}}(t) \quad (1)$$

상태 $P_{S_{21}}(t+\Delta t)$는 Active side에서 오류가 발생하고 Standby side가 성공적으로 Active로 실행된 경우, Standby side에 오류가 발생했지만 오류가 검출된 상태를 나타낸다.

$$P_{S_{21}}(t+\Delta t)=2\lambda\Delta t D_1 D_0 \beta P_{S_{22}}(t)+(1-\lambda\Delta t)P_{S_{21}}(t) \quad (2)$$

상태 $P_{S_{20}}(t+\Delta t)$는 Active side와 Standby side에서 오류가 발생하고 오류가 검출된 상태를 나타낸다.

$$P_{S_{20}}(t+\Delta t)=\lambda\Delta t D_0 \beta P_{S_{21}}(t)+P_{S_{20}}(t) \quad (3)$$

상태 $P_{O_{21}}(t+\Delta t)$는 Active side 또는 Standby side에서 오류가 발생했지만 검출하지 못한 상태를 나타낸다.

$$P_{O_{21}}(t+\Delta t)=2\lambda\Delta t(1-D_1)(1-D_0)P_{S_{22}}(t)+(1-\lambda\Delta t)P_{O_{21}}(t) \quad (4)$$

상태 $P_{O_{20}}(t+\Delta t)$는 Active side와 Standby side에서 오류가 발생했지만 검출하지 못한 상태, 그리고 Active side에서 오류가 발생했지만 Standby side의 사용이 실패한 상태를 나타낸다.

$$P_{O_{20}}(t+\Delta t)=\lambda\Delta t(1-D_0)P_{O_{21}}(t)+P_{O_{20}}(t) \quad (5)$$

상태 $P_{SO_{20}}(t+\Delta t)$는 Active side의 오류 검출 실패, Standby side의 오류 검출 실패, Active side에서 오류가 발생했지만 Standby side의 사용이 실패(Standby side의 오류 검출 실패)한 상태를 나타낸다.

$$P_{SO_{20}}(t+\Delta t)=\lambda\Delta t(1-D_0)P_{S_{21}}(t)+\lambda\Delta t D_0 \beta P_{O_{21}}(t)+P_{SO_{20}}(t) \quad (6)$$

시스템의 신뢰도(Reliability Degree)는 시스템이 안전하게 동작 가능한 상태로서 식 (7)과 같은 확률로 나타낼 수 있으며, 안전도(Safety Degree)는 B'd에서 오류가 발생하여 기능을 중단하는 경우에도, 효과적인 방법으로 오류를 취급하여, B'd가 안정 상태를 유지하는 것을 말한다. 이는 식 (8)과 같다.

$$R_M(t)=P_{S_{22}}(t)+P_{S_{21}}(t)+P_{O_{21}}(t) \quad (7)$$

$$S_M(t)=P_{S_{22}}(t)+P_{S_{21}}(t)+P_{S_{20}}(t)+P_{O_{21}}(t)+P_{O_{20}}(t) \quad (8)$$

5. 성능 평가

4절에서 제안한 모델에 대하여, 하드웨어적인 결함을 방지하는 능력이 시스템의 신뢰도와 안전도에 미치는 영향을 분석한다. 제안된 모델을 평가하기 위하여 두 개의 Coverage Factor가 고려된다.

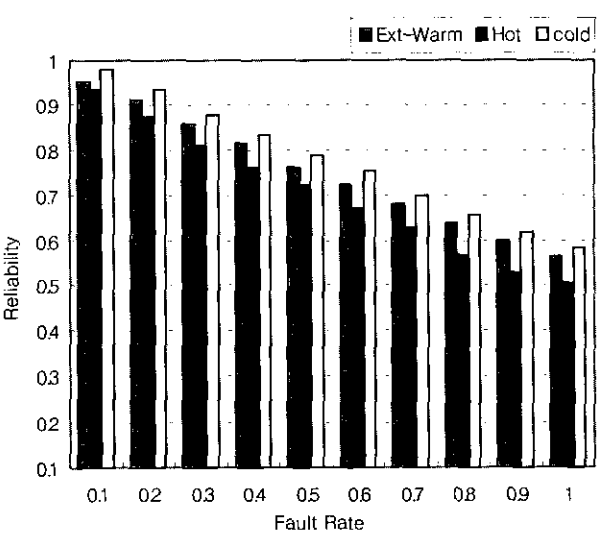
1) 상대 B'd의 오류를 탐지할 수 있는 확률
2) 자기 B'd의 오류를 탐지할 수 있는 확률

본 시스템은 결함 검출과 recovery 절차에 있어서 상대 B'd 결함 검출 확률과 자기 B'd 결함 검출 확률의 합에 의존한다. 성능 평가를 간략화 하기 위하여 다음의 가정을 고려한다.

① 소프트웨어 오류는 고려하지 않는다.
② 시스템 내에 감지되지 않는 어떠한 에러도 존재하지 않는다.
③ 시스템이 두 side의 결함에 의하여 강등되었을 때, 시스템은 두 side를 동시에 회복하고 그 때 정상 상태로 복귀된다.
④ 전자적인 오류율은 Gate Level에서 입증된 것으로 고려하지 않는다.

(그림 5)의 Markov로 모델화한 시스템은 오류 발생률 λ, 상대 B'd 결함 탐지 확률 D_1과, 자기 B'd 상태 탐지확률 D_0를 갖는 하드웨어 모듈로 구성되어 있으므로 신뢰도와 안전도에 대한 모듈식은 식 (7)과 식 (8)에 대한 결과이다.

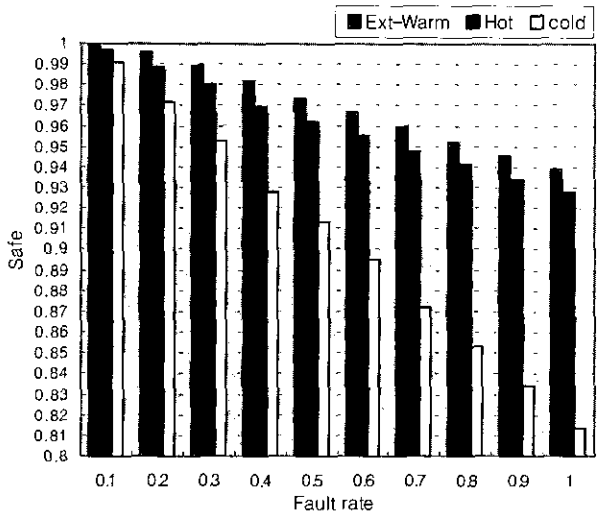
(그림 6)은 기존 방안과 확장된 Warm-standby의 신뢰도를 비교한 것이다. 각 방안의 운영 특성상, 주 모듈이 작업을 수행하고 있을 때, 대기하거나 진단 등의 예비 작업을 수행하는 Cold-Standby의 시스템 신뢰도가 가장 우수함을 알 수 있다. 그리고 확장된 Warm-standby는 오류로부터 시스템이 완전하게 동작 가능한 확률이 Hot-Standby에 비하여 우수하므로 신뢰도가 더 높음을 알 수 있다.



(그림 6) 오류율에 따른 신뢰도 비교

(그림 7)은 안전도를 비교한 것이다. Cold-Standby는 고장 검출 및 시스템의 재구성을 자기 진단 계산에만 의존하므로 안전도가 매우 낮음을 알 수 있다. 또한 두 프로세서 사이의 출력 결과를 비교하여 오류 여부를 확인하고, 각 프로세서가 자기진단계산(self-diagnostic routine)을

수행하여 고장 프로세서를 찾는 Hot-Standby는 하나의 모듈이 고장이고 자기 진단에 의하여 검출되지 않는다면 시스템의 안전도를 감소시키는 결과를 낳는다. 반면에 제안된 방안은 실시간으로 자신 및 상대 B'd의 오류를 탐지하고 처리를 수행하므로 안전도가 가장 우수함을 알 수 있다.



(그림 7) 오류율에 따른 안전도 비교

6. 결 론

본 연구에서는, 각 프로세스가 메시지 단위로 동작하면서 갱신된 데이터를 Standby side에 전달하는 Task기반 이중화방식을 기본으로, 기존의 Warm standby 구조가 갖는 데이터 손실 및 거짓 데이터의 확산 문제를 해결 함으로서 성능의 향상을 도모하였다. Warm standby는 Hot standby에 비해 구현이 용이하나 결함 발생 시의 데이터 손실 문제 그리고 오류 데이터가 시스템으로 확산되는 문제점을 가지고 있었다. 반면에 Hot standby 구조는 데이터 무손실 등의 다수의 장점이 있지만 이중화 모듈간에 빈번한 동기로 인한 전체 시스템 성능 저하 문제로 인하여 구현 시 이론적인 성능을 얻기 어렵다.

본 연구에서는 Warm standby의 동기화의 단순함에 따른 구현의 용이성이라는 측면에 기존 Warm standby의 단점을 개선함으로서 효율적인 이중화 시스템을 구축하였다. H/W 적인 측면에서는, 각 B'd가 자신 및 상대 B'd의 상태를 실시간으로 조사 함으로서, 오류 탐지 확률을 높이고 오류가 발생하면 H/W적으로 상태 비트를 설정하여 데이터 전송을 원천적으로 차단하였다. S/W적인 측면에서는, TCB와 EIMBuf, OEIMBuf를 이용한 메시지 처리 구조를 제시하여 데이터 손실 및 오류 데이터의 확산 문제를 해결하였다. 이를 통하여 자기진단계산에만 의존하고 오류로부터 완전하게 동작할 확률이 낮은 Hot standby에 비하여 확장된 Warm standby의 신뢰도 및 안전도가 우수함을 알 수 있었다. 향후에는 H/W 탐지 기능에 오류가 발생했을 경우 등의 부가 기능을 위한 S/W 오류 탐지 기능을 제시해야 한다.

참 고 문 헌

[1] D. P. Siewiorek, "Architecture of fault - tolerant computers : an historical perspective," Proc. IEEE, Vol.79, No.12, pp.1710-1734, Dec. 1991.

[2] J. C, Laprie, et al., "Definition and Analysis of Hardware and Software Fault Tolerant Architectures," Computer, pp. 39-51, July. 1990.

[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] L. Young, "Software Fault Tolerance at the Operating System Level," Proc.of COMPSAC '95, pp.393, 1995.

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

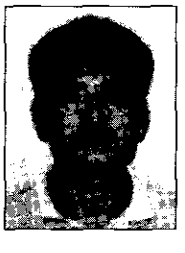
[6] B. J. Min, S. S. Shin and K. W. Rim, "Design and Analysis for a Multiprocessor System with Extended Fault Tolerance," IEEE FTDCS., pp.301-307, Aug. 1995.

[7] "부하분담 방식의 이중화 및 절체 방법과 이를 수행하기 위한 시스템", 현대전자 특허 96-42408, Dec. 1996.

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

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

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



이 종 찬

e-mail : chan2000@etri.re.kr

1994년 군산대학교 컴퓨터과학과(공학사)

1996년 숭실대학교 대학원 전자계산학과 (공학석사)

2000년 숭실대학교 대학원 컴퓨터과학과 (공학박사)

2000년~현재 전자통신연구원 무선브래픽접속연구팀 선임연구원

관심분야 : Duplex system, Mobile Tracking, Wireless Multimedia.

강 권 일

e-mail : kangki@nature.skku.ac.kr

1997년 대전대학교 컴퓨터공학과 졸업 (학사)

1999년 성균관대학교 전기전자 및 컴퓨터 공학과 졸업(석사)

1999년~현재 현대시스콤

관심분야 : 프로세서 이중화, Neural networks

이 경 준

e-mail : kyungjun@telecom.samsung.co.kr

1998년 한림대학교 컴퓨터공학과 졸업

2000년 한림대학교 컴퓨터공학과 대학원 졸업

2000년~현재 삼성전자 통신연구소 IMT-2000 시스템연구팀

관심분야 : IMT-2000 system, Fuzzy system