論文 96-21-8-13

Warm Standby 고장감내 구조를 지원하는 교환 제어 시스템에서의 가동률 분석

正會員 송 광 석*, 여 환 근*, 한 창 호**, 문 태 수**,
이 광 배**, 김 현 욱**, 윤 충 화***

An Availability Analysis of Switching Control System with Warm Standby Fault Tolerant Architecture

Kwangsuk-Song*, Hwangeun-Yeo*, Changho-Han**, Taesoo-Moon**,
Kwangbae-Lee**, Hyenug-Kim**, Chunghwa-Yoon*** *Regular Members*

요 약

본 논문에서는 높은 가용성을 필요로 하는 교환 제어시스템의 이중화 모듈 구조에 적용 가능한 여러 가지 warm standby 고장감내 모델 및 그 동작 방법들을 설명하고, 각 모델의 4가지 구현 방식에 대한 시스템 불가동률을 마르코프 상태도를 이용하여 구한 후 그 성능을 비교 평가하였다. 본 연구 결과, 데이터 손실 면에서는 삼중 프로세서로 구성된 warm standby 모델이 가장 우수하며, 시스템 가동률 면에서는 대체적으로 이중 프로세서로 구성된 warm standby 모델이 standby 검사 방식을 사용하였을 때 가장 높은 가동률을 제공했다. 정기적인 절체는 시스템 불가동률을 증가시켰으나 standby 모듈에서의 정기적인 고장 검사는 단일 프로세서 warm standby 모델과 이중프로세서 warm standby 모델의 불가동률을 감소시켰다. 한편 데이터 복구 시간 및 운영 요원에 의한 복구율 변화는 시스템 불가동률에 거의 영향을 미치지 않았다.

ABSTRACT

In this paper, we describe several warm standby fault-tolerant models and their operation methods applicable to telephone switching control systems which have dual module structure and need high availability. Unavailabilities of the system implemented by four different methods for each model are computed by using the Markov state

* 한국전자통신연구소
Electronics and Telecommunications Research Institute
** 명지대학교 전자공학과
Dept. of Electronic Engineering, Myong-Ji University
*** 명지대학교 컴퓨터공학과
Dept. of Computer Science, Myong-Ji University
論文番號:95341-1004
接受日字:1995年 10月 4日

model, and then are compared for system performance evaluation. As the results of simulations, the warm standby model with triple processors is best in the aspect of data loss, while in most cases the warm standby model with double processors based on no standby check method provides the highest system availability. Periodic changeover increases the system unavailability, but the periodic standby check on standby module decreases the system unavailability of warm standby model with a single processor and with double processors. On the other hand, the variations of data recovery time and personnel recovery rate have little effects on the system unavailability.

I. 서 론

단순한 전화 교환 기능만을 갖고 있던 교환기가 다양한 서비스 기능과 통신 매체의 핵심적인 기능을 포함하도록 성장함에 따라 교환기내에서 처리되어야 할 데이터 양도 대폭 늘어나고, 그 처리방법도 복잡하게 되었다. 그 결과 오늘날 교환기 시스템내에서의 고장 발생은 경제 및 관련 산업 분야에 엄청난 피해를 주는 경우도 발생하고 있다. 앞으로 초고속 정보통신망의 구축은 이러한 피해의 규모 및 횟수가 기하급수적으로 증가할 것이 예상되므로 고신뢰도를 갖는 고장 감내 교환기 시스템의 개발이 필수적이라고 하겠다.

고장 감내 시스템은 시스템 하드웨어나 소프트웨어에 고장이 발생하더라도 계속해서 주어진 작업을 올바로 수행하도록 설계된 시스템으로, 고장 감내 기법은 크게 하드웨어, 소프트웨어 그리고 정보 기법으로 나눌 수 있다.[1, 2, 3, 4, 5, 6] 하드웨어 고장 감내 기법에는 패시브(passive), 액티브(active) 그리고 하이브리드(hybrid) 기법이 있다. 패시브 기법은 고장의 발생을 외부에 숨기고 고장으로 인한 에러 발생을 막는다. 이 기법은 고장의 발생을 감추기 위해서 보우팅(voting) 메카니즘에 의존하며, TMR(Triple Modular Redundancy) 기법이 가장 많이 쓰이고 있다.[1] 액티브 기법은 고장 발생을 감지했을 때 그 고장을 시스템으로 부터 제거하기 위한 조치를 위한다. 이 경우 주어진 시간내에 시스템이 정상 상태로 복귀한다면 일시적인 에러는 허용된다. 액티브 기법에는 이중 비교(duplication with comparison), 스탠드바이 스패어링(standby sparing), 페어 앤드 스페어(pair-and a spare) 기법 등이 있다.[1] 하이브리드 기법은 패시브 기법과 액티브 기법의 장점을 혼합하여 적용한 것으로써 스페어를 갖는 NMR(N-modular redundancy with spares), 셀프 퍼징 리던던시(self-purging redundancy) 그리고 시프트 아웃 모듈 리던던시(sift-out modular redundancy) 등이 그 기법에 속한다.[1]

근래에 와서 소프트웨어 고장 감내 기법에 대한 중요성이 부각되어 오늘날 이 분야에 대한 연구도 활발히 이루어지고 있다. 대표적으로 체크 포인팅(check pointing), 복구 블럭(recovery block),[2, 4] N 셀프 체킹(N-self checking),[3] N 버전 프로그래밍(N-version programming) 기법[1] 등이 사용되고 있다.

정보 고장 감내 기법은 대부분의 고장 감내 시스템에서 하드웨어 및 소프트웨어 기법의 보완책으로 널리 사용되고 있다. 이 기법은 실제 데이터에 여분의 정보를 추가함으로써 고장 발생 감지 및 일부 수정을 가능하게 한다. 패리티 코드(parity code), 체크섬(checksum), Hamming ECC(Hamming error-correcting code), 순환 코드(cyclic code), m-of-n code 등이 이 기법에서 사용되고 있다.

최근의 대부분의 고장 감내 시스템들은 하드웨어, 소프트웨어 및 정보 고장 감내 기법들을 모두 혼합하여 사용하고 있다. 이 경우에, 하드웨어 기법은 하드웨어 고장을 신속히 감지하는 데 사용되며 소프트웨어 기법은 고장으로 부터의 복구를 위해서 사용된다. 또한 정보 기법은 데이터 정보의 에러를 검출 및 수정하는 데 사용되고 있다. 현재 모든 교환기 시스템에서는 하드웨어 액티브 기법 중 하나인 스탠드바이 스패어링 기법을 기본 고장 감내 기법으로 채택하고 있는 데 이는 시스템 가동률이 교환기 시스템에서 가장 중요한 요소이기 때문이다.[6]

한편 외국의 일부 교환기 공급 회사에서는 교환기 전용 고장 감내 프로세서를 개발하여 사용해 왔으며 가장 대표적으로 AT&T사의 No. 1 ESS~No. 5 ESS에 사용된 프로세서 및 3B20D 프로세서 등이 이에 속한다. 반면에 국내에서는 상용 범용 프로세서를 이용하여 이중화 모듈 구조를 갖는 고장 감내 기능을 구현하고 있다.[5~11]

본 논문에서는 국내에서 개발한 교환기에서 채택하고 있는 제어계 이중화 방식인 warm standby 고장감내 기법을 근간으로하여 향후 제어시스템에서 적용 가능성이 있는 고장 감내 시스템 모델의 구조와 고장 감내 구현 방법 및 동작을 제시하고, 각 이중화 모델에 대한 불가동률을 마르코프 상태도를 이용하여 계산한 후 상호 비교 분석하였다.

본 논문의 구성은 서론에 이어 2장에서 기존 교환기 시스템에서 사용되고 있는 고장감내 기법에 관해서 간단히 살펴보고, 3장에서는 각 고장감내 이중화 모델의 구조 및 동작 방식을 설명한다. 4장에서는 각 모델 구조의 구현 방법에 따른 시스템 불가동률을 계산하기 위해 마르코프 상태도 모델을 제시하고 그에 대한 성능 평가를 한다. 끝으로 5장에서는 본 연구를 통해 얻은 결론과 향후의 연구 방향에 대해서 서술한다.

Ⅱ. 기존의 교환기에서 사용된 고장 감내 기법

대표적인 교환기 공급회사인 AT&T사와 Ericsson사에서 개발된 교환 제어 시스템에서 사용된 고장감내 기법을 간단히 살펴본다. AT&T사에서는 ESS 교환기 시스템을 발표하였고, Ericsson사에서는 AXE 교환기 시스템을 개발하였다. 이들 교환기 제어시스템들은 기본적인 고장감내 기법으로서 스탠드바이 스패어링 기법을 사용하고 있다.[6]

2.1 AT&T ESS 계열 교환기

AT&T에서 개발된 교환기 시스템들은 ESS 시리즈로서 No. 1 프로세서부터 시작하여, No. 1A 프로세서, No. 2 프로세서, No. 3A 프로세서, 3B20D 프로세서, 5 ESS 프로세서, DMERT, attached 프로세서 등을 사용하였다. 프로세서는 자체점검 하드웨어를 이용하여 시스템의 운용중에 고장을 검색하며, central-control 부분과 main store 부분 등이 이중화되어있다. 에러 검출후 순차적인 처리와 관련된 부분이 마이크로프로그램화 되어있으며, 마이크로프로그램 저장부에서는 복수 인접 비트의 고장에 대한 검사를 효율적으로 수행하기 위하여 m-out-of-2m 방법과 패리티 방법을 혼용하여 사용하였다.

3B20D 프로세서는 프로세서의 소프트웨어와 하드웨어의 기본적인 구조가 이중화의 형태를 취하며, 동시 액세스 기능과 자체점검 기능을 가지고 있다. 전체적으로 중앙 제어부와 메모리, 입출력 보조기억장치들은 이중화의 형태를 취하고 있으며, 모두 스위칭이 가능한 요소들로서 그룹화되어있다.

2.2 Ericsson AXE 10

AXE 10 교환기 시스템은 동기화에 의해 이중화된 중앙제어 프로세서와 보다 간단한 구조를 갖는 주변제어 프로세서들로 구성되어 있으며, central 프로세서로서 APZ 210, APZ 211, 그리고 APZ 212 등이 존재한다. 이들 프로세서들은 각각 사용되는 용량 및 비용의 문제와 결부되어 DS(Data Store), PS(Program Store), RS(Reference Store) 등의 역할을 하나의 유니트로 통합시키거나, 그들을 다루는 유니트를 통합시키는 등의 방법을 취하고 있다. 전체적인 시스템의 구조는 이중화의 방법을 취하고 있으며, 중간에 MAU(Maintenance Unit)가 있어서, 어느 쪽 모듈에서 고장이 발생하였는지를 감지하는 기능을 가지고 있다.

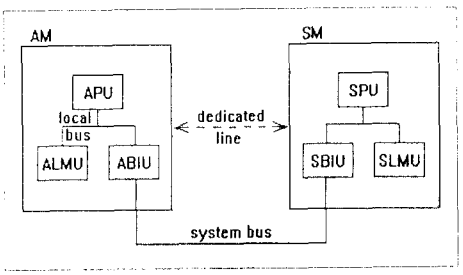
Ⅲ. warm standby 이중화 고장 감내 시스템 모델

기존의 단일 프로세서 warm standby 모델을 구현시키기 위해서 AT&T 사에서는 고장 감내 기능이 내장된 프로세서를 자체 개발하여 사용하고 있다. 그러나 고장감내 고성능 프로세서의 개발은 시간과 비용이 지수함수적으로 증가하는 추세이므로 상용 교환기로서 가격 경쟁을 갖추기 위해서는 고장 감내 기능이 내장되어 있지 않은 상용 프로세서를 사용하여야 한다. 그러므로 본 논문에서는 고장 감내 기능이 내장되지 않은 상용 프로세서를 사용하여 warm standby 모델을 구현시키는 방법들을 고려하고자 한다. 먼저 기존에 사용되고 있는 단일 프로세서 warm standby 모델과 이중 프로세서 warm standby 모델을 살펴보고, 이 모델의 문제점을 보완하기 위한 삼중 프로세서 warm standby 모델을 제시한 후 그에 대한 동작을 서술한다. 이 모델에 대해 메모리와 버스 인터페이스는 에러 점검 기능을 내장하고 있고 버스 라인 상에 에러는 발생하지 않는다고 가정하였다.

3.1 단일 프로세서로 구성된 warm standby 모델

단일 프로세서 warm standby 모델 구조는 그림 1에서와 같이 active와 standby 모듈 각각에 프로세서, 메모리, 버스 인터페이스 등이 하나씩 밖에 존재하지 않는다. 즉, 각 모듈내의 하드웨어 부품은 단일 유니트로 구성된다. 또한 active 모듈과 standby 모듈 사이에 존재하는 전용선은 주기적인 정상 상태 확인을 위해 사용될 뿐만 아니라 각 모듈간의 제어 신호 전송용 및 시스템 버스 사용 불가능시 일부 메시지 전송 대치용으로서 이용된다.

이 시스템 모델에서 데이터 읽기 동작시 APU는 ALMU로부터 데이터를 읽어오며, 데이터 쓰기 동작시 APU는 ALMU와 ABIU에 기록할 데이터를 전송하고, 이와 동시에 ABIU는 시스템 버스를 통하여 standby 모듈의 SLMU로 그 데이터를 전송한다. 즉 active 모듈에서의 한 번 메모리 쓰기동작으로 standby 내의 메모리에 동시쓰기가 이루어 진다.



AM : Active Module
APU : Active Processor Unit
ALMU : Active Local Memory Unit
ABIU : Active Bus Interface Unit
SM : Standby Module
SPU : Standby Processor Unit
SLMU : Standby Local Memory Unit
SBIU : Standby Bus Interface Unit

그림 1. 단일 프로세서 warm standby 구조
Fig. 1. single processor warm standby architecture

일부 교환 시스템에서는 active 모듈과 standby 모듈 간에 역할을 주기적으로 절체시켜 standby 모듈의 상태를 정기적으로 확인하는 방법도 적용되고 있다. 이 경우 두 모듈간의 정기적인 절체 동작은 다음과 같이 수행된다. 절체될 시점에서 우선 active 모듈은 정기 절체 동기 신호를 standby 모듈에 보낸다. 그리고 standby 모듈이 준비완료 상태임을 확인한 후 active 모듈은 모든 필요한 정보를 standby 모듈로 전송한다. 그후 standby 모듈은 그 정보를 이용하여 active 모듈로서 동작하기 위해 필요한 값들을 세트시키고, 준비가 완료되면 그 사실을 상대편 모듈에게 알린 후 자신은 active 모듈로서 동작하게 된다. 이때 상대편 모듈은 standby 상태로 세트된다.

하드웨어 고장은 전원부의 고장과 부품의 고장으로 크게 나뉘어 질 수 있으며, 반도체 기술의 발달로 대부분의 시스템에서 전원부 고장 발생률이 부품의 고장 발생률보다 훨씬 높게 나타난다. active 모듈에서의 전원부 고장발생시 standby 모듈이 상대측 전원부의 고장상태를 감지하고 active 상태로의 절체준비 작업이 완료될 때까지 active 모듈은 불가동 상태로 된다. 이때 처리중이던 프로세스의 데이터 손실은 불가피하다. APU의 고장 발생시 APU 자체는 이를 감지할 수 있는 기능을 가지고 있지 않으므로, 전용선을 통하여 주기적으로 정상 상태를 확인함으로써 그 고장을 감지할 수 있다. 이때 정상 상태를 확인하는 신호 주기사이에 처리된 데이터는 고장으로 인해 유실될 수 있다. ALMU 고장 발생시 ALMU는 이를 감지할 수 있는 기능을 내장하고 있으나 고장 발생시 수행되어져야 할 에러 처리 루틴들이 ALMU내에 있으므로 이 루틴들의 정상적인 처리가 불가능해 진다. 그러므로 고장 발생 감지 즉시 ALMU는 전용선을 통하여 그 사실을 standby 모듈에 통보한다. 이때에도 처리 중이던 프로세스의 데이터는 유실될 수 있다. ABIU 고장 발생시 ABIU는 이를 감지하여 APU에게 알리고, APU는 ALMU 내의 에러처리 루틴을 이용하여 그 사실을 standby 모듈에게 알리고 절체에 필요한 정보를 standby 모듈에 전송함으로써 standby 모듈이 active로서 동작할 수 있도록 한다. 이때에는 데이터 손실이 발생하지 않는다.

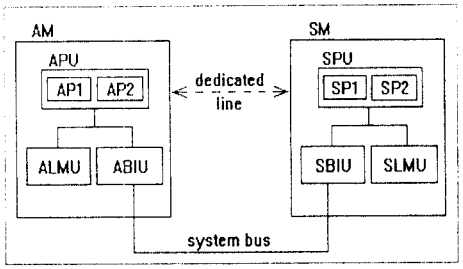
standby 모듈은 active 모듈의 고장 발생을 확인한 경우 필요에 따라 데이터 복구 동작을 수행한 후 active 모듈로서 동작하게 된다.

한편 고장난 active 모듈은 고장 발생 신호 및 필요한 관련 정보를 standby 모듈로 전송한 후 그 자신은 고장복구 동작에 들어간다. 그러한 고장 복구 동작은

다음의 순서로 수행한다. 먼저 고장난 모듈에 대한 진단시험 및 재시동을 통하여 자체 복구를 시도하고 불가능한 경우에는 운용요원에게 수리를 요청한다. 수리완료된 모듈은 시스템에 재구성되어 초기화를 수행한 후 현재 active로 동작중인 모듈에게 standby 상태로 진입하기 위한 요구신호를 보내고, 동기를 맞추기 위하여 메모리 내용을 변경시킬 준비를 갖춘다. active 모듈은 변경된 메모리의 내용을 standby 모듈과 동일하게 유지하기 위하여 정상 동작과 병행하여 수리된 모듈의 메모리 내용을 변경시킨다. 이러한 일련의 데이터 복구동작이 완료되면 수리된 모듈은 standby 모듈로 세트되고 전체 시스템은 완전히 정상 상태에서 동작하게 된다.

3.2 이중 프로세서 warm standby 모델

단일 프로세서 warm standby 모델은 프로세서 고장 발생 즉시 감지할 능력이 없으므로 오동작으로 인해 시스템 전체가 거짓 데이터에 오염될 수 있다. 그러므로 이를 보완하기 위해서 이중 프로세서 warm standby 모델을 고려하였다. 그림 2에서 알 수 있듯이 단일 프로세서 warm standby 모델 구조에서는 각각의 프로세서 유니트에 하나의 프로세서만이 존재하였으나, 이 구조에서는 각각의 프로세서 유니트에 두개씩의 프로세서들이 존재한다.



AP1 : Active Processor 1

AP2 : Active Processor 2

SP1 : Standby Processor 1

SP2 : Standby Processor 2

그림 2. 이중 프로세서 warm standby 구조
Fig. 2. Double processor warm standby architecture

이중 프로세서 warm standby 모델 구조에서 AP1은 정상 프로세서로서 동작을 하게 되며 AP2는 AP1에서의 고장 발생을 검출하는데 사용된다. 본 논문에서 제시한 APU에 대한 상세한 내부 구조는 그림 3에 나타내었다. 이러한 구성은 앞의 단일 프로세서 warm standby 모델 구조에서의 취약했던 프로세서 고장 발생에 대한 감지 능력이 보강되어 프로세서에서의 고장이 정상 상태 확인 신호 구동시에 감지되는 대신에 고장발생 시점에서 감지되므로 처리중인 데이터의 손실이 감소될 뿐만아니라 거짓 데이터 오염으로부터 시스템을 보호할 수 있다.

두 모듈간에 절체 동작과 고장 발생시 복구동작은 단일 프로세서 warm standby 모델과 동일하다.

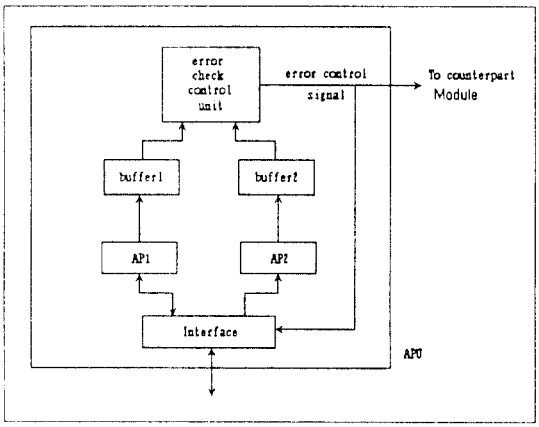


그림 3. APU의 내부구조
Fig. 3. internal structure of APU

3.3 삼중 프로세서 warm standby 모델

고장 발생으로 인한 데이터 손실을 최소화시키는 warm standby 모델로서 그림 4와 같은 구조를 갖는 삼중 프로세서 warm standby 모델을 제시하였다. 이 모델에서는 APU 및 SPU는 각각 세개의 프로세서로 구성되고, ALMU와 SLMU는 각각 두개의 메모리 유니트로 구성된다. 또한 ABIU와 SBIU는 각각 두개의 버스 인터페이스로 구성됨으로써 이전까지는 BIU의 고장시 그 기능을 전용선이 대신하였지만, 이 구조에서는 하나의 버스 인터페이스 고장시 다른 버스 인터페이스를 이용하여 고장난 active 모듈이 필요한 정보

및 데이터를 시스템 버스를 통하여 standby 모듈에 전송할 수 있다. 이 구조에서도 전원부의 고장을 감지하기 위하여 전용선을 통한 정상 상태 확인 동작이 주기적으로 수행된다. APU에 대한 상세한 내부 구조는 그림 5에 나타내었으며 AP와 에러검사 유니트사이에 버퍼를 두어 고장검출 기능으로 인한 시스템의 성능 저하가 최소화되게 구성하였다.

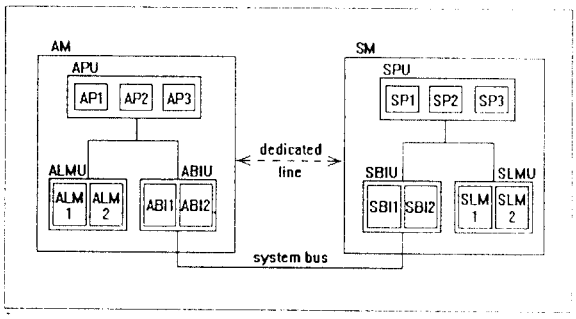


그림 4. 삼중 프로세서 warm standby 구조
Fig. 4. Triple processor warm standby architecture

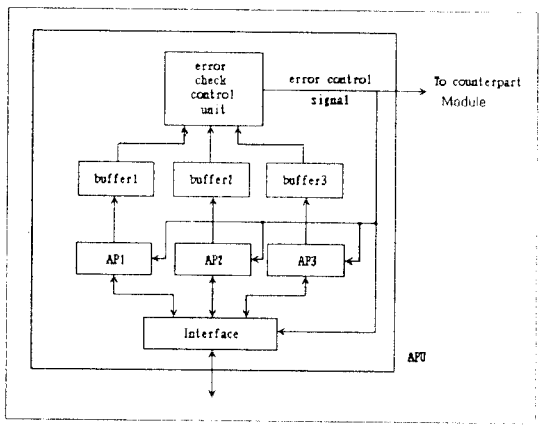


그림 5. APU의 내부 구조
Fig. 5. internal structure of APU

정상 동작중에 데이터 읽기 동작시 ALMU의 ALM1은 APU로 데이터를 보내며, 이때 APU내의 삼중화된 프로세서(AP1, AP2, AP3) 모두 이 데이터를 받아서 동시에 일을 계속 수행한다. 데이터 쓰기 동작시 APU내의 AP1만이 ALMU와 ABIU로 변경될 데이터를 보내며, 이때 ALMU내의 ALM1과 ALM2 그리고 ABIU내의 ABI1과 ABI2 모두가 이 데이터를 받는다. 그 후 ABIU의 ABI1만이 그 데이터를 시스템 버스를 통해 SBIU의 SBI1과 SBI2에 전송하고, SBI1은 그 데이터를 SLMU의 SLM1과 SLM2로 전송한다. 두 모듈의 절체 동작은 앞의 단일 프로세서 warm standby 모델과 동일하다.

전원부의 고장은 앞의 경우와 마찬가지로 정상상태 확인 신호 구동시 감지되며, 고장 발생시 처리중이던 데이터의 손실은 불가피하다. 부품 유니트의 고장시 APU에서 고장이 발생한 경우에는 먼저 시스템의 현재 동작을 일시 중지시키고 고장나지 않은 프로세서 중에 하나를 선택하여 그 프로세서의 주관하에 필요한 정보를 standby 모듈에 전송한다. APU 고장으로 인한 데이터 손실이 발생하지 않도록 하기 위해서 여분의 참조 버퍼를 APU에 포함시킬 수 있다.

ALMU에서 고장이 발생한 경우에는 현재의 시스템 동작을 일시 중지시킨 후 고장나지 않은 메모리로부터 APU가 에러 처리 루틴을 수행하고, 필요한 정보를 standby 모듈로 전송함으로 이때 데이터 손실은 발생하지 않는다. ABIU에서 고장이 발생한 경우, APU는 현재의 시스템 동작을 일시 중지시킨 후 ABIU 내부의 전송되지 않은 데이터를 고장나지 않은 버스 인터페이스를 이용해서 standby 모듈에 전송한 후 필요한 정보를 standby 모듈에 전송한다. 이 경우에 고장으로 인한 데이터 손실은 발생하지 않는다. 한편 고장 발생시 standby 모듈의 동작과 고장난 모듈의 고장 복구 동작은 단일 프로세서 warm standby 모델의 경우와 동일하나, 단지 고장복구 동작중 현재 active로 동작하는 모듈에 고장이 발생한 경우에는 앞의 두 모델의 경우와 같이 시스템을 완전히 정지시키는 것이 아니라 여분의 부품을 사용하여 고장난 모듈이 완전히 복구될 때까지 시스템을 구동시킨다. 이때, 고장난 모듈이 완전히 복구되면 현재 동작중인 고장난 active 모듈은 고장 복구된 모듈에 필요한 정보를 전송한 후 고장복구 동작에 들어간다.

Ⅳ. 성능 평가

4.1 마르코프 상태도를 이용한 불가동률의 계산

본 연구에서는 앞절에서 설명한 각 모델에 대해 4

가지 이중화 동작 방식을 대상으로 각각에 대한 불가동률을 마르코프 상태도를 이용하여 구하였다. 이들 4가지 동작 방식은 다음과 같다.:정기 절체, 비정기 절체, standby 검사, 혼합 방식.

정기 절체방식은 정기적으로 active 모듈과 standby 모듈간에 역할을 바꾸어 구동시키는 방식이며, 비정기 절체방식에서는 active 모듈의 고장 발생시에만 standby 모듈이 active로서 동작하게 된다. standby 검사 방식은 비정기 절체 방식을 취하면서 standby 모듈이 자체 고장 검사를 수행하는 방식이며, 혼합 방식은 정기절체와 standby 검사 방식을 혼용한 방식이다.

본 논문에서는 마르코프 상태도를 이용한 각 모델별 불가동률을 계산할 때 고장 감지 기능을 갖는 하드웨어는 고장 발생 즉시 이를 감지할 능력이 있다고 가정하였다. 한편 standby 검사 방식과 혼합 방식을 사용하는 경우, 정기적인 standby 모듈의 에러발생 검사는 대부분 고장 발생 즉시 이루어지지 않으므로 고장 발생시 고장을 감지할 수 있는 확률(coverage)을 고려하였다.

그림 6과 그림 7은 단일 프로세서및 이중 프로세서로 구성된 warm standby 모델에 대한 상태도를 나타내었다. 그림 6은 정기절체와 비정기절체 방식에 대한 상태도를 보인 것으로, 이 두 방식에서 마르코프 상태도의 형태는 동일하며 단지 천이선(line) 상의 일부 변수와 값들이 서로 다른 차이점이 있다. 그림 6에서 P0는 정상 상태, P1은 고장이 발생한 것을 감지하지 못한 상태, P2는 고장 발생을 감지한 상태, P3는 standby 모듈이 active 상태로 천이되고 고장모듈은 하드웨어 복구를 시도하는 상태, P4는 고장 모듈의 하드웨어가 완전히 수리되고 데이터 복구를 수행하는 상태이며 P5는 시스템 전체가 완전히 다운(down)된 상태를 의미한다. 그림 7은 standby 검사와 혼합 방식에 대한 마르코프 상태도를 보인 것으로, 그림 6의 상태와 비교하여 P6 상태가 추가되었다. P6은 standby 모듈에서의 고장발생을 감지하지 못한 상태를 의미하며, standby 모듈에서의 고장이 감지되었을 경우에는 P3 상태로 천이한다.

그림 6과 그림 7의 천이선 상의 파라미터는 표 1에 나타내었다.

표 1에서 λ_0는 시스템에서의 고장률로서 λ_1과 λ_2의 합을 의미하고, CS는 standby 모듈에서의 고장발생

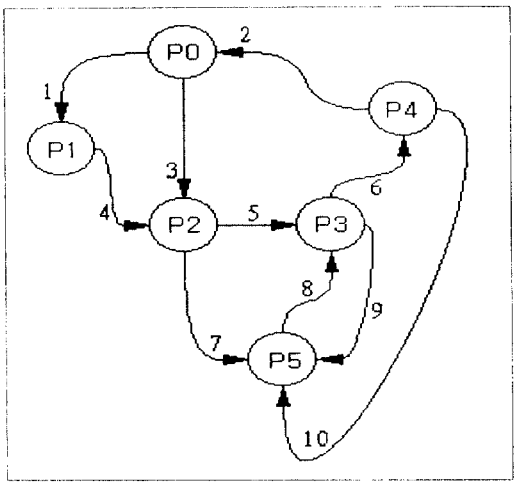


그림 6. 단일 프로세서 warm standby 및 이중 프로세서로 구성된 warm standby 모델에 대한 상태도 1 (정기절체, 비정기절체)

Fig. 6. state diagram 1 for the warm standby model with a single processor and double processors (periodic changeover, no periodic changeover)

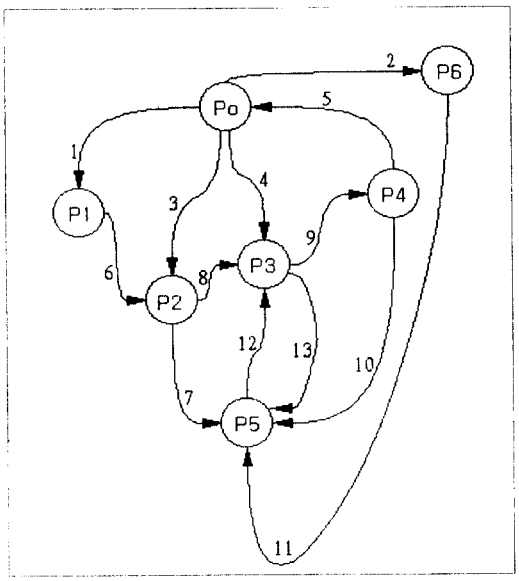


그림 7. 단일 프로세서 및 이중 프로세서로 구성된 warm standby 모델에 대한 상태도 2 (standby 검사, 혼합 방식)

Fig. 7. state diagram 2 for the warm standby model with a single processor and double processor(standby check, mixed method)

표 1. 그림 6과 그림 7의 천이 변수
Table 1. Transition parameters of Fig. 6 and 7

이중화방식 / 천이번호	정기 절체 비 정기 절체	혼합 방식
1	λ_2	λ_2
2	μ_4	$\lambda_0(1-C_s)$
3	λ_1	λ_1
4	μ_1	$\lambda_0 C_s$
5	μ_2	μ_4
6	μ_3	μ_1
7	λ_0	λ_0
8	μ_5	μ_2
9	λ_0	μ_3
10	λ_0	λ_0
11		λ_0
12		μ_5
13		λ_0

시 고장을 감지할 수 있는 확률이며, μ는 회복률을 의미한다.

그림 8과 그림 9는 삼중 프로세서로 구성된 warm standby 모델에 대한 상태도를 나타내었다. 그림 8은 앞의 그림 6과 대응하며, 그림 9는 앞의 그림 7과 대응해서 고려될 수 있다. P6, P7, P8은 이전 고장 발생으로 인한 고장복구 동작중에 현재 active 모듈에 고장이 발생한 경우를 나타내는 상태들이다. 이 경우에는 앞에서 설명한 바와 같이 시스템을 완전히 다운시키는 것이 아니라 현재 고장복구중인 모듈의 수리가 완료될 때까지 시스템이 치명적이지 않는 한 고장난 상태로 계속 동작한다.

표2는 삼중 프로세서로 구성된 warm standby 모델에서의 상태도에 대한 천이 변수를 나타내었으며, λ_{30}은 시스템에서의 고장률로서 λ_{31}과 λ_{32}의 합을 의미하고, CS는 standby 모듈에서의 고장발생시 고장을 감지할 수 있는 확률이며, μ는 회복률을 의미한다.

그림 6의 상태천이도로 부터 시스템이 각 상태로 존재할 확률을 구할 수 있으며 그에 대한 식은 다음과 같다.

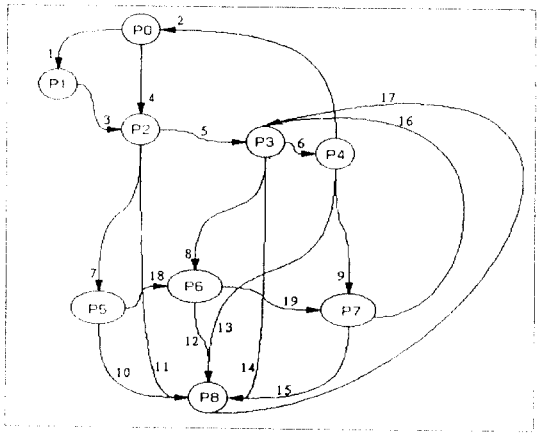


그림 8. 삼중 프로세서로 구성된 warm standby 모델에 대한 상태도 1 (정기절체, 비정기절체)
Fig. 8. state diagram 1 for the warm standby model with triple processors(periodic changeover, no periodic changeover)

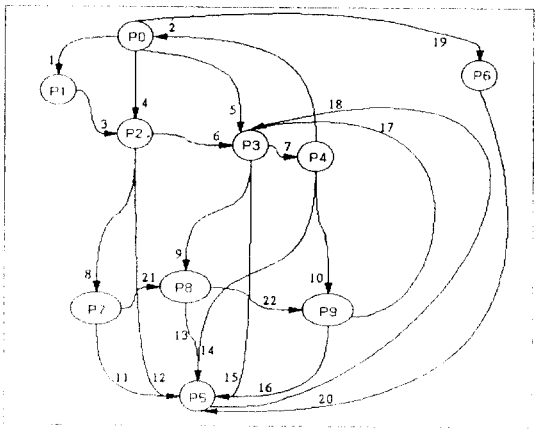


그림 9. 삼중 프로세서로 구성된 warm standby 모델에 대한 상태도 2 (standby 검사, 혼합방식)
Fig. 9. State diagram 2 for the warm standby model with triple processors(standby check, mixed method)

표 2. 그림 8과 그림 9의 천이 변수
Table 2. Transition parameter of Fig. 8 and 9

이중화방식 / 선번호	정기 절체 비 정기 절체	혼합 방식
1	λ_{32}	λ_{32}
2	μ_{34}	μ_{34}
3	μ_{31}	μ_{31}
4	λ_{31}	λ_{31}
5	μ_{32}	$\lambda_{30}Cs$
6	μ_{33}	μ_{32}
7	λ_{33}	μ_{33}
8	λ_{33}	λ_{33}
9	λ_{33}	λ_{33}
10	λ_{34}	λ_{33}
11	λ_{35}	λ_{34}
12	λ_{34}	λ_{35}
13	λ_{35}	λ_{34}
14	λ_{35}	λ_{35}
15	λ_{34}	λ_{35}
16	μ_{36}	λ_{34}
17	μ_{35}	μ_{36}
18	μ_{32}	μ_{35}
19	μ_{33}	$\lambda_{30}(1-Cs)$
20		λ_{30}
21		μ_{32}
22		μ_{33}

$$P0(\lambda_1+\lambda_2)-P4\mu_4=0 \quad (1)$$
$$P1\mu_1-P0\lambda_2=0 \quad (2)$$
$$P2(\lambda_0+\mu_2)-P0\lambda_1-P1\mu_1=0 \quad (3)$$
$$P3(\lambda_0+\mu_3)-P2\mu_2-P5\mu_5=0 \quad (4)$$
$$P4(\lambda_0+\mu_4)-P3\mu_3=0 \quad (5)$$
$$P5\mu_5-\lambda_0(P2+P3+P4)=0 \quad (6)$$

위의 식들은 서로 의존적(dependent)이므로 한 식을 선택하여 P0 + P1 + P2 + P3 + P4 + P5 = 1 식으로 대체한 후 계산하면 그 시스템 불가동률이 다음과 같다.

$$Su_1 = P2 + P4 + P5$$

그림 7의 상태천이도로 부터 시스템이 각 상태로 존재할 확률을 구하는 식은 다음과 같다.

$$P0\{\lambda_1+\lambda_2+\lambda_0Cs+\lambda_0(1-Cs)\}-P4\mu_4=0 \quad (7)$$
$$P1\mu_1-P0\lambda_2=0 \quad (8)$$
$$P2(\lambda_0+\mu_2)-P0\lambda_1-P1\mu_1=0 \quad (9)$$
$$P3(\lambda_0+\mu_3)-P0\lambda_0Cs-P2\mu_2-P5\mu_5=0 \quad (10)$$
$$P4(\lambda_0+\mu_4)-P3\mu_3=0 \quad (11)$$
$$P5\mu_4-\lambda_0(P2+P3+P4+P6)=0 \quad (12)$$
$$P6\lambda_0-P0\lambda_0(1-Cs)=0 \quad (13)$$

위의 식 중 한 식을 선택하여 P0 + P1 + P2 + P3 + P4 + P5 + P6 = 1 식으로 대체한 후 그 시스템 불가동률을 계산하면 다음과 같다.

$$Su_2 = P2 + P4 + P5$$

그림 8의 상태천이도로 부터 시스템이 각 상태로 존재할 확률을 구하는 식은 다음과 같다.

$$P0(\lambda_{31}+\lambda_{32})-P4\mu_{34}=0 \quad (14)$$
$$P1\mu_{31}-P0\lambda_{32}=0 \quad (15)$$
$$P2(\lambda_{33}+\lambda_{35}+\mu_{32})-P0\lambda_{31}-P1\mu_{31}=0 \quad (16)$$
$$P3(\lambda_{33}+\lambda_{35}+\mu_{33})-P2\mu_{32}-P8\mu_{36}-P5\mu_{35}=0 \quad (17)$$
$$P4(\lambda_{33}+\lambda_{35}+\mu_{34})-P3\mu_{33}=0 \quad (18)$$
$$P6(\lambda_{34}+\mu_{32})-P2\lambda_{33}=0 \quad (19)$$
$$P7(\lambda_{34}+\mu_{33})-P3\lambda_{33}-P6\mu_{32}=0 \quad (20)$$
$$P8(\lambda_{34}+\mu_{36})-P4\lambda_{33}-P7\mu_{33}=0 \quad (21)$$
$$P5\mu_{35}-\lambda_{35}(P2+P3+P4)-\lambda_{34}(P6+P7+P8)=0 \quad (22)$$

위의 식 중 한 식을 P0 + P1 + P2 + P3 + P4 + P5 + P6 + P7 + P8 = 1 식으로 대체한 후 그 시스템 불가동률을 계산하면 다음과 같다.

$$Su_4 = P2 + p4 + P5 + P7 + P9$$

그림 9의 상태천이도로 부터 시스템이 각 상태로 존

재할 확률을 구하는 식은 다음과 같다.

$$P0\{\lambda_{31}+\lambda_{32}+\lambda_{30}Cs+\lambda_{30}(1-Cs)\}-P4\mu_{34}=0 \quad (23)$$
$$P1\mu_{31}-P0\lambda_{32}=0 \quad (24)$$
$$P2(\lambda_{33}+\lambda_{35}+\mu_{32})-P0\lambda_{31}-P1\mu_{31}=0 \quad (25)$$
$$P3(\lambda_{33}+\lambda_{35}+\mu_{33})-P0\lambda_{30}Cs-P2\mu_{32}-P9\mu_{36}-P5\mu_{35}=0 \quad (26)$$
$$P4(\lambda_{33}+\lambda_{35}+\mu_{34})-P3\mu_{33}=0 \quad (27)$$
$$P7(\lambda_{34}+\mu_{32})-P2\mu_{33}=0 \quad (28)$$
$$P8(\lambda_{34}+\mu_{33})-P3\lambda_{33}-P7\mu_{32}=0 \quad (29)$$
$$P9(\lambda_{34}+\mu_{36})-P4\lambda_{33}-P8\mu_{33}=0 \quad (30)$$
$$P5\mu_{35}-\lambda_{35}(P2+P3+P4)-\lambda_{34}(P7+P8+P9)-P6\lambda_{30}=0 \quad (31)$$
$$P6\lambda_{30}-P0\lambda_{30}(1-Cs)=0 \quad (32)$$

위의 식 중 한 식을 선택하여 P0 + P1 + P2 + P3 + P4 + P5 + P6 + P7 + P8 + P9 = 1 식으로 대체한 후 그 시스템 불가동률을 계산하면 다음과 같다.

$$Su_4 = P2 + P4 + P5 + P7 + P9$$

4.2 분석 및 고찰

시스템 불가동률 계산에 사용되는 고장률 λ와 관련된 값들은 sun-2/50 워크스테이션에 대한 Lambda 출력 데이터[6]를 msec당 고장률로 환산하여 사용하였으며, 회복율 μ와 관련된 값들은 현재 국내 교환기에 적용되고 있는 값들에 근거를 두어 정하였다. 그림 6, 7, 8 및 9에 대응하는 천이 변수식은 다음과 같다.

(그림 6과 7) (단일 프로세서 warm standby 비정기 절체 및 standby 검사)

$\lambda_1 = \lambda_m + \lambda_b + \lambda_k + \lambda_e * Pd$
$\lambda_2 = \lambda_p + \lambda_e * (1 - Pd)$
$\lambda_0 = \lambda_1 + \lambda_2$
$1/\mu_1 = mct/2$
$1/\mu_2 = te + srt1$
$1/\mu_3 = pt * pr + st * (1 - pr)$
$1/\mu_4 = bdt + dt$
$1/\mu_5 = te + 1/3\{(1/\mu_3 + bdt) + (0.5 * 1/\mu_3 + bdt) + 0.5 * bdt\}$

(이중 프로세서 warm standby 비정기 절체 및 standby 검사)

$\lambda_1 = 2\lambda_p + \lambda_{pec2} + \lambda_m + \lambda_b + \lambda_k + \lambda_e * Pd$
$\lambda_2 = \lambda_e + (1 - Pd)$
$1/\mu_2 = te + srt2$

다른 천이 변수식은 단일 프로세서 warm standby 비정기 절체 경우와 동일

(단일 프로세서 warm standby 정기 절체 및 혼합 방식)

$\lambda_1 = (\lambda_m + \lambda_b + \lambda_k + \lambda_e * Pd) * 2$
$\lambda_2 = \{\lambda_p + \lambda_e * (1 - Pd)\} * 2$
$\lambda_0 = \lambda_1 + \lambda_2$

다른 천이 변수식은 단일 프로세서 warm standby 비정기 절체 경우와 동일

(이중 프로세서 warm standby 정기 절체 및 혼합 방식)

$\lambda_1 = (2\lambda_p + \lambda_{pec2} + \lambda_m + \lambda_b + \lambda_k + \lambda_e * Pd) * 2$
$\lambda_2 = \lambda_e * (1 - Pd) * 2$
$\lambda_0 = \lambda_1 + \lambda_2$

다른 천이 변수식은 이중 프로세서 warm standby 비정기 절체 경우와 동일

(그림 8과 9) (삼중 프로세서 warm standby 비정기 절체 및 standby 검사)

$\lambda_{31} + 3\lambda_p + \lambda_{pec3} + 2\lambda_m + 2\lambda_b + \lambda_k + \lambda_e * Pd$
$\lambda_{32} = \lambda_e * (1 - Pd)$
$\lambda_{30} = \lambda_{31} + \lambda_{32}$
$\lambda_{33} = (3\lambda_p + 2\lambda_m + 2\lambda_b)\lambda_{30}/\lambda_{30} + (\lambda_k + \lambda_e + \lambda_{pec3})$
$(3\lambda_p + 2\lambda_m + 2\lambda_b)$
$\lambda_{34} = \lambda_e + \lambda_k + \lambda_{pec3}$
$\lambda_{35} = (\lambda_e^2 + \lambda_k^2 + \lambda_{pec3}^2)/\lambda_{30}$
$1/\mu_{31} = mct/2$
$1/\mu_{32} = te + srt3$
$1/\mu_{33} = pt * pr + st * (1 - pr)$
$1/\mu_{34} = bdt + dt$
$1/\mu_{35} = te + 1/3\{(1/\mu_{33} + bdt) + (0.5 * 1/\mu_{33} + bdt) + 0.5 * bdt\}$
$1/\mu_{36} = 1/\mu_{32} + 1/\mu_{34}$

(삼중 프로세서 warm standby 정기 절체 및 혼합 방식)

$\lambda_{31} = (3\lambda_p + \lambda_{pec3} + 2\lambda_m + 2\lambda_b + \lambda_k + \lambda_e * Pd) * 2$
$\lambda_{32} = \lambda_e * (1 - Pd) * 2$
$\lambda_{30} = \lambda_{31} + \lambda_{32}$

다른 천이 변수식은 삼중 프로세서 warm standby 비정기 절체 경우와 동일

천이 변수식에 사용되는 변수의 의미는 다음과 같다.

λ_p : 프로세서 고장률
λ_m : 메모리 고장률
λ_b : 버스 인터페이스 고장률
λ_k : clock 고장률
λ_e : 전원 고장률
Pd : 전원 고장 발생시 감지할 확률
mct : 정상 상태 점검 주기
te : 고장 감지시 기본 에러 처리 시간
srt1 : 단일 프로세서 warm standby에 대한 고장 감지 시 standby 모듈의 데이터 복구 시간
srt2 : 이중 프로세서 warm standby에 대한 고장 감지 시 standby 모듈의 데이터 복구 시간
srt3 : 삼중 프로세서 warm standby에 대한 고장 감지 시 standby 모듈로의 제어 정보 전송 시간
pt : 운영 요원이 고장 모듈을 새 모듈로 대처하는데 걸리는 시간
pr : 운영 요원이 고장 모듈을 대체할 확률
st : 고장 발생시 하드웨어를 복구한 후 기본 데이터를 복구시키는데 소요되는 시간
bdt : 고장 발생시 하드웨어를 복구한 후 active 모듈의 메모리 내용과 동일하게 데이터를 복구시키는데 소요되는 시간
λ_{pec2} : 이중 프로세서 warm standby에서의 프로세서 에러 점검 유닛트의 고장률
λ_{pec3} : 삼중 프로세서 warm standby에서의 프로세서 에러 점검 유닛트의 고장률

본 논문의 성능평가를 위하여 λ_p, λ_m, λ_b, λ_k 값은 sun-2/50 워크스테이션에 대해 수행된 데이터 결과[6]를 인용하였으며, Tandem 컴퓨터 시스템에서의 고장 감내 테스트 결과 전원 고장률이 전체 하드웨어 고장률의 60~70%를 점유한다는 보고서[6]를 근거로 하여 본 논문에서는 전원 고장률이 전체 하드웨어 고장률에 차지하는 비중을 60%로 가정하여 계산하였다.

전원 고장 발생시 감지할 확률은 0.1로 가정하였고, 정상 상태 점검 주기는 국내 교환기의 경우와 동일하게 30ms로 책정하였다. st는 ft-sparc[12]의 경우와 같이 100ms로 정하였다. 또한 te와 bdt는 각각 0.1 ms로 가정하였다. λ_{pec2}는 $\lambda_b/4$, λ_{pec3}는 $1.5/4 * \lambda_b$로 가정하여 계산하였다. srt2는 한 프로세서의 데이터를 복구하는 데 걸리는 시간을 의미하며 1ms로 가정하였고, srt3도 1ms 정도 소요된다고 가정하였다. 반면에 srt1은 고장 발생 시간과 고장 감지 시간간에 간격에 따라 변화하며, pr도 교환기 설치 위치에 따라 변화한다. dt는 데이터 복구 동작이 background 프로세서로 동작하므로 소요 시간이 가변이라 할 수 있다.

그림 10은 active 모듈에 고장이 발생했을 때 단일 프로세서로 구성된 warm standby 모듈의 데이터 복구시간의 변화에 대해 pr = 0.001, dt = 100000 ms, Cs = 0.9로 가정하여 각 모델의 불가동률을 계산한 것이다.

그림에서 보는 바와 같이 이중 프로세서와 삼중 프로세서로 구성된 warm standby 모델은 srt1 값에 변동이 없으므로 동일한 불가동률을 가지며, 단일 프로세서로 구성된 warm standby 모델만이 srt1이 증가함에 따라 거의 선형적으로 불가동률이 증가되는 현상을 볼 수 있다.

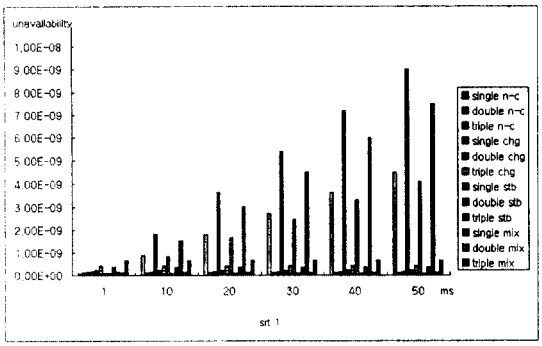


그림 10. srt1에 대한 불가동률
Fig. 10. Unavailability vs srt1

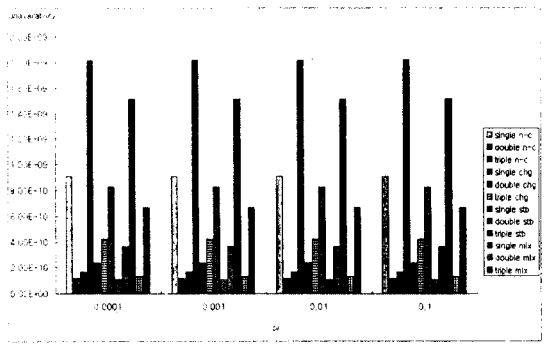


그림 11. 운용요원에 의한 복구율에 대한 불가동률
Fig. 11. Unavailability vs personnel recovery rate

그림 11은 active 모듈에서 고장이 발생했을 경우 직접 운용요원에 의해 수리되어 복구될 확률(pr)에 대해 srt1 = 10 ms, dt = 100000 ms, Cs = 0.9로 가정하여 각 모델의 불가동률을 구하였다.

그림에서 보는 바와 같이 전체적으로 각 모델에 대한 시스템 불가동률은 운용요원에 의한 복구율에 거의 영향을 받지 않음을 알 수 있다.

그림 12는 고장 모듈의 하드웨어 수리가 완료한 후 active로 동작중인 모듈의 메모리 내용과 고장 모듈의 메모리 내용을 상호 일치시키는 데 걸리는 데이터 복구 시간(dt)에 대해 srt1 = 10 ms, pr = 0.01, Cs = 0.9로 가정하여 각 모델의 시스템 불가동률을 보여주고 있다.

그림에서 보는 바와 같이 데이터 복구 시간 dt의 변화는 각 모델에 대한 시스템 불가동률에 거의 영향을 미치지 않음을 알 수 있다.

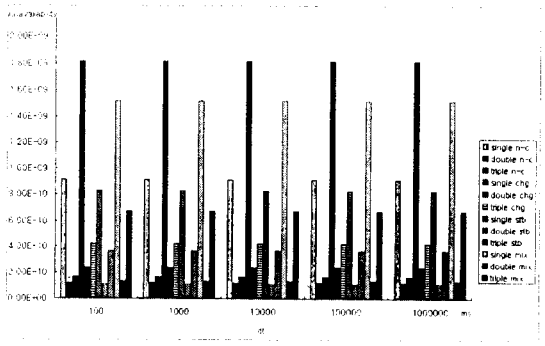


그림 12. 데이터 복구 시간에 대한 불가동률
Fig. 12. Unavailability vs data recovery time

그림 13은 standby 모듈에서의 고장 발생시 고장을 감지할 수 있는 확률(C_s)의 변화에 대해 srt1 = 10 ms, pr = 0.01, dt = 100000 ms로 가정하여 각 모델의 시스템 불가동률을 그래프로서 나타낸 것이다. 위의 그래프로부터 단일 프로세서로 구성된 warm standby 모델은 standby 모듈 고장 감지율(Cs)이 증가할 수록 오히려 시스템 불가동률이 증가하며, 이중 프로세서로 구성된 warm standby 모델도 Cs의 증가에 따라 시스템 불가동률이 약간 증가하였다. 반면에 삼중 프로세서로 구성된 warm standby 모델은 Cs의 증가와 더불어 시스템 불가동률이 감소하였다.

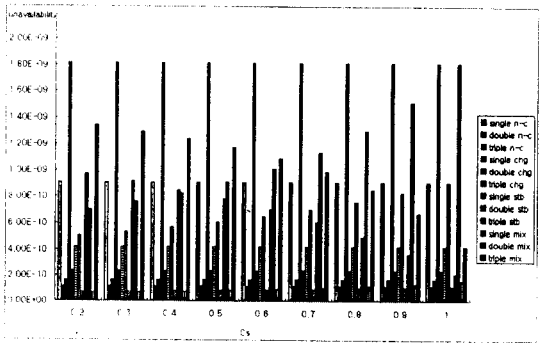


그림 13. C_s에 대한 불가동률
Fig. 13. Unavailability vs C_s

V. 결 론

본 논문에서는 교환기의 제어시스템에서 채택하고 있는 warm standby 이중화 고장감내 기법을 기본으로 세 가지 고장감내 제어시스템 모델을 고려하고 각 모델의 구조와 동작 방법을 설명하였으며, 성능 평가를 위해 각 모델별로 구현 가능한 4가지 이중화 방식--비정기절체, 정기절체, standby 검사 및 혼합 방식--에 대한 시스템 불가동률을 마르코프 상태도를 이용하여 구한 후 상호 비교 분석하였다.

세가지 warm standby 모델에서 삼중 프로세서로 구성된 모델, 이중 프로세서로 구성된 모델 그리고 단일 프로세서로 구성된 모델 순으로 고장 발생시 데이터 손실은 증가하고, 사용되는 하드웨어 소자 수, 즉 비용은 반대로 감소하게 된다.

시스템 불가동률 성능 평가 결과, 이중 프로세서로 구성된 warm standby 모델이 단일 프로세서로 구성된 warm standby 모듈의 데이터 복구 시간(srt1)이 1ms 보다 더 큰 경우 가장 우수한 시스템 가동률을 제공하였으며, 정기적인 절체는 오히려 시스템 불가동률을 증가시키는 단점을 나타내었다. 반면에 standby 모듈에서의 정기적인 고장 검사 방법은 시스템 가동률면에서 단일 프로세서 및 이중 프로세서로 구성된 warm standby 모델에 대해 긍정적으로 작용하였으나, 삼중 프로세서로 구성된 warm standby 모델에 대해서 부정적으로 작용하였다. 또한 srt1의 증가는 단일 프로세서로 구성된 warm standby 모델의 불가동률을 증가시켰으며, 운영 요원에 의한 복구율(pr)의

변화는 각 모델에 대한 시스템 불가동률에 거의 영향을 미치지 않았다. 한편, standby 모듈 고장 감지율(Cs)의 증가는 단일 프로세서 모델과 이중 프로세서 모델의 불가동률을 증가시킨 반면 삼중 프로세서 모델의 불가동률을 감소시켰다.

본 연구 결과, 시스템 가격과 가동률 및 데이터 손실 등을 고려할 때 제안된 세 모델 중에서 이중 프로세서로 구성된 warm standby 모델이 향후 개발할 교환 제어시스템에 가장 적합한 모델로 생각되며 향후 연구되어야 할 과제로는 하드웨어 고장이외의 다른 고장을 내포한 경우에 대한 시스템 가동률을 구함으로써 최적인 교환 제어시스템 모델과 이중화 구현 방식을 찾는 데 있다.

참 고 문 헌

1. Johnson, B. W., Design and Analysis of Fault Tolerant Digital Systems, Addison Wesley, 1989.
2. Randell, B., "System Structure for Software fault tolerance," IEEE Trans. on Software Engr., June 1975, pp. 220~232.
3. Laprie, J-C, et al., "Definition and Analysis of Hardware and Software Fault-Tolerant Architectures," Computer, pp. 39~51, July 1990.
4. Shem-Tov Levi and Ashok K. Agrawala, Fault Tolerant System Design, McGraw-Hill, 1994.
5. W. N. Toy, "Fault-Tolerant Design of Local ESS Processors," Proceedings of the IEEE, pp. 1126~1145, Oct. 1978.
6. Daniel P. Siewoirek and Robert S. Swarz, Reliable Computer Systems, Digital Press, 1992.
7. R. M. Wolfe and N. A. Martellotto, "Telecommunications Data Base Application with the 3BTM20 Processor," ISS '84 Florence, 7-11 May 1984.
8. Frank, R. J., Siegel, El H. Jr. and Watters, R. J., "Attached Processor for 4ESSTM and 1A ESSTM Switches," ISS '84 Florence, 7-11 May 1984.
9. Jan-Erik Andersson, Per-Erik Gustafsson and Lennart Soderberg, "New Operation and Mainternace Functions in AXE10," Erricsson Rev. No. 3, pp. 104~111, 1986.
10. Thomas Edsbacker, "New Regional Processor for AXE10," Ericsson Rev. No. 3, pp. 112~118, 1986.
11. Ingmar Jonsson, "Control System for AXE10," Ericsson Rev. No. 4, pp. 146~155, 1984.
12. ft sparc Technical Description, IMP Ltd, 1995.

송 광 석(Kwangsuk-Song) 정회원

1979년 : 고려대학교 전자공학과 졸업(학사)
1981년 : 고려대학교 대학원 전자공학과(석사)
1992 : 고려대학교 대학원 전자공학과(박사)
1992년 : Georgia Institute of Technology 객원연구원
1982년~현재 : 한국전자통신연구소 제어시스템연구실장
※관심분야 : Fault Tolerant Control System, Computer Architecture, ATM Switching System

여 환 근(Hwangeun-Yeo) 정회원

1981년 : 경북대학교 전자공학과 졸업(학사)
1983년 : 경북대학교 대학원 전자공학과(석사)
1992년~현재 : 경북대학교 대학원 컴퓨터공학과 박사과정
1993년~1994년 : U. of Maryland SRI 객원 연구원
1983년~현재 : 한국전자통신연구소 제어시스템연구실 선임연구원
※관심분야 : Fault Tolerant Computing System, Distributed Computer Architecture, ATM Switching System

한 창 호(Changho-Han) 정회원

1994년 : 명지대학교 전자공학과 졸업(공학사)
1994년~1996년 : 명지대학교 전자공학과 졸업(공학석사)
1996년~현재 : 한국컴퓨터통신(주) 연구원
※관심분야 : ATM통신, 고장감내형 시스템 설계, 멀티미디어(영상 처리)

문 태 수(Taesoo-Moon) 정회원

1989년 : 명지대학교 전자공학과 졸업(공학사)
1994년 : 명지대학교 전자공학과 졸업(공학석사)
1994년~현재 : 명지대학교 전자공학과 박사 재학
1996년~현재 : 성남 기능대학 전기기술학과 교수
※관심분야 : 컴퓨터 시스템, 네트워크 구조 및 설계, 데이타 통신

이 광 배(Kwangbae-Lee) 정회원
1979년 : 고려대학교 전자공학과 졸업(공학사)
1979년~1981년 : 고려대학교 전자공학과 졸업(공학석사)
1981년~1982년 : 삼성반도체연구소
1982년~1983년 : 금성연구소
1984년~1986년 : Univ. of Southern Califonia, Computer Engineering 전공(공학석사)
1986년~1991년 : Arizona State Univ., Electronical Engineering전공(공학박사)
1992년~현재 : 명지대학교 전자공학과 부교수
※관심분야 : 멀티미디어(영상 및 음성 신호처리), 병렬처리 및 고속 컴퓨터(prolog 방식), Communication System(고장 감내형)

김 현 욱(Hyenug-Kim) 정회원
1978년 : 고려대학교 전자공학과 졸업(공학사)
1978년~1980년 : 고려대학교 전자공학과 졸업(공학석사)
1980년~1987년 : 고려대학교 전자공학과 졸업(공학박사)
1980년~1981년 : 동양공업전문대학 전자과 전임강사
1981년~1988년 : 명지대학교 전자공학과 전임강사, 부교수
1988년~1990년 : Arizona State Univ., Computer Science전공(Adijunct Faulty)
1990년~현재 : 명지대학교 전자공학과 교수
※관심분야 : 멀티미디어(영상 및 음성 신호처리), 병렬처리 및 고속 컴퓨터(prolog 방식), Communication System(고장 감내형)

윤 충 화(Chunghwa-Yoon) 정회원
1979년 : 서울대학교 수학과 졸업(학사)
1984년 : U. of Texas at Austin 전산학과 졸업(석사)
1989년 : 루이지에나 주립대학 전산학과 졸업(박사)
1990년~현재 : 명지대학교 컴퓨터공학과 조교수
※관심분야 : 신경회로망, 전문가 시스템, 성능평가