분산 멀티미디어 환경에서 결함 허용 에이전트의 설계 및 구현

고 응 남† · 황 대 준††

요 약

본 논문에서는 분산 멀티미디어 환경에서 실행되는 FDRA(Fault Detection Recovery based on Agent)의 설계와 구현을 기술한다. 두레는 강의 하는 동안에 학생들과 교사 사이에 분산 멀티미디어 및 멀티미디어 원격 교육 시스템을 위한 좋은 예이다. 두레는 기본적인 서비스 에이전트들을 가지고 있다. 서비스 기능들은 객체 지향의 개념으로 구현되어 있다. FDRA는 멀티 에이전트 시스템이다. 그것은 효율적인 두레 환경의 구성과 실험을 위하여 설계되고 구축되었다. 멀티 에이전트 환경에서, 지능형 에이전트는 그들의 목적을 이루기 위하여 협조적 또는 비협조적으로 상호 교환한다. 중요한 생각은 폴링 방법을 사용하여 오류를 감지한다는 것이다. 이 방법은 세션과 직접 연관된 프로세서만을 주기적으로 폴링하여 오류 감지를 수행한다. 그리고, 학습 방법을 이용하여 자동적으로 오류 유형을 분류한다. 이 방법의 장점은 시스템을 복구하는 방법에 있어서 세션을 생성하는 것과 같은 방법으로 사용한다는 것이다. FDRA는 분산 멀티미디어 환경에서 소프트웨어 오류를 감지하고, 오류의 유형을 분류하여 자동적으로 복구하는 시스템이다.

A Design and Implementation of Fault Tolerance Agent on Distributed Multimedia Environment

Eung-Nam Ko† · Dae-Joon Hwang††

ABSTRACT

In this paper, we describe the design and implementation of the FDRA(Fault Detection Recovery based on Agent) running on distributed multimedia environment. DOORAE is a good example for distributed multimedia and multimedia distance education system among students and teachers during lecture. It has primitive service agents. Service functions are implemented with objected oriented concept. FDRA is a multi-agent system. It has been designed and implemented for construction and experiment of effective DOORAE environment. In multi-agent environment, intelligent agents interact with each other, either collaboratively or non-collaboratively, to achieve their goals. The main idea is to detect an error by using polling method. This system detects an error by polling periodically the process with relation to session. And, it is to classify the type of errors automatically by using learning rules. The merit of this system is to use the same method to recovery it as it creates a session. FDRA is a system that is able to detect an error, to classify an error type, and to recover automatically a software error based on distributed multimedia environment.

1. 서 론

멀티미디어는 교육, 원격 진료 등 다양하고 광범위한 분야에서 응용되고 있다. 멀티미디어 이용 분야 중에서 교육 분야에서도 많은 관심을 가지고 있다[1, 2, 3, 4, 5, 6]. 사회가 복잡해지고 컴퓨터 네트워크가 발달함에 따라 다양한 종류의 상호 참여가 요구되어지고 있다. 이에 따라 상호 참여를 제공하기 위한 화상 회의 시스템의 개발이 활발해지고 있다[7]. 화상 회의 시스

※ 본 논문은 한국과학재단 연구비지원(과제번호 : K-1-98046)에 의한 결과임
† 정 회 원 : 신성대학 컴퓨터계열 교수
†† 정 회 원 : 성균관대학교 정보공학과 교수
논문접수 : 1999년 6월 14일, 심사완료 : 1999년 9월 6일

템과 공동작업(CSCW : Computer Supported Cooperative Work) 환경은 원거리 전문가로부터 여러 장소에서 서로의 모습을 보며 음성과 연필로 서로의 의견을 교환할 수 있게 하였다[8]. 분산 멀티미디어 정보 시스템에 대한 요구는 매우 빠르게 상업, 생산, 교육, CAD(Computer Aided Design)/ CAE(Computer Aided Engineering), 의학, 기상 등 많은 분야에서 필요로 하고 있다. 멀티미디어 데이터는 그림, 동영상, 음성, 음향 및 문자 등의 데이터를 포함한다. 멀티미디어 시스템은 기존의 데이터 프로세스 보다 많은 기능과 능력을 요구한다. 이런 필요성은 VLSI, 광케이블, 네트워크 기술의 발달과 함께 분산 멀티미디어 시스템 구축을 가능하게 했다[6, 8]. 최근 들어 이러한 멀티미디어 교육 시스템의 공동 작업 환경이 증가하고 있는데 반하여 이러한 시스템에서의 전체적인 망 관리, 특히 응용 소프트웨어에 대한 결함을 발견 및 복구하는 연구는 미흡한 실정이다. 분산 시스템은 하나의 노드 또는 자원에 결함이 발생해도 전체 시스템에는 큰 손실을 입히지 않는 특성을 가지기 때문에 결함 허용 시스템의 설계를 위한 좋은 조건을 가진다[9]. 시스템에서 결함을 허용하는 정도에 따라 결함 허용 시스템은 소프트웨어 기법, 하드웨어 기법, 혼합 기법 등으로 분류할 수 있다[9]. 따라서, 본 연구에서는 분산 환경에서 응용 소프트웨어의 결함을 미리 감지하여 알려주고 복구할 수 있는 에이전트 시스템을 설계하여 제안한다.

이러한 문제를 해결하기 위하여 결함 허용 시스템에 멀티에이전트(Multi-Agent)의 개념을 도입 적용시킨다. 멀티 에이전트 시스템이란 작은 규모의 문제를 해결할 수 있는 에이전트들 간에 필요한 정보 교환을 통해 상호 협력을 하여 문제를 해결하는 시스템이다[10, 11, 12, 13]. 현재 항공 교통 제어, 분산 모니터링 등의 스케쥴링 및 계획 분야(planning)와 전자 메일 관리(예 : 미국 백악관), 인터넷 상의 정보 분류(예 : 전자 뉴스 팩토링), 전자 서비스 분야에서 멀티 에이전트 환경을 이용하는데 응용되고 있다[10]. 따라서 본 논문에서는 멀티 에이전트의 특성인 각 에이전트의 자치권을 보장하면서 서로간의 협력을 통해서 전체적인 망 관리, 특히 응용 소프트웨어에 대한 결함을 발견 및 복구하는 에이전트인 FDRA(Fault Detection Recovery based on Agent)를 제안한다. 이 에이전트를 통해서 전체적인 시스템의 성능을 향상시킬 수 있다.

본 논문의 구성은 2에서 관련 연구를 기술하고, 3에서는 제안하는 오류 감지 및 복구 에이전트인 FDRA에 대해서 기술하고, 4에서는 시스템 평가, 5에서는 결론을 기술한다.

2. 관련 연구

결함 허용 시스템이란 하드웨어 오동작, 소프트웨어 오류 또는 정보 오염이 일어날지라도 주어진 임무를 올바르게 수행할 수 있는 시스템을 말한다[9]. 본 절에서는 기존의 결함 허용 기법, 기존 결함 허용 기법의 한계점 및 멀티 에이전트에 대해서 기술한다.

2.1 기존 결함 허용 기법

결함 허용성을 부여하는 방법에 따라 3가지로 나눌 수 있다. 첫째, 소프트웨어 기법은 운영체제에 의해 이루어지는 기법으로 소프트웨어에 의한 오버헤드로 시스템 성능 하락에 대한 희생이 따른다. 둘째, 하드웨어 기법은 하드웨어 다중화를 통해 결함 탐지 및 복구가 수행되는 기법이다. 셋째, 혼합 기법은 하드웨어로 결함을 탐지하고 소프트웨어로 결함 복구를 하게함으로써 소프트웨어 오버헤드와 하드웨어 비용을 줄일 수 있는 장점이 있다[14]. 소프트웨어 결함허용 기법은 소프트웨어 모듈의 중복이나 재수행(rollback and retry), 또는 이 두가지 방식의 혼용에 기초를 두고 있다. 검사점(Check pointing)은 소프트웨어 실행 중에 검사시점을 설정하여 오류가 발생했는지를 검사하여 이상이 없으면 계속 수행하고 이상이 감지되면 그 이전의 검사시점으로 되돌아가 재수행 하는 방식이다. 복구 블록(Recovery block)은 재수행 (rollback and retry)에 근거한 기법으로 검사시점에서 오류가 감지되면 지정된 이전 시점으로 되돌아가 같은 기능을 가진 다른 소프트웨어 모듈을 실행하는 방식으로 단일 프로세스내에 적용될 수 있다. Conversation은 복구 블록의 확장형으로 단일 프로세스가 아닌 서로 정보를 상호 교환하는 프로세스 들간에 적용 가능한 기법으로 복구 블록과 마찬가지로 재수행(rollback and retry)에 기초를 두고 있다.

분산 복구 블록(Distributed recovery block)은 복구 블록을 분산 환경으로 확장 적용하여 하드웨어 결함과 소프트웨어 결함을 동일한 방법으로 극복할 수 있도록 한 기법이다. N 자가 검사 프로그래밍(Self-checking programming)은 두 개 이상의 자가진단(self-checking)

부품이 실행되면서 그 중의 하나(수행부품)가 주어진 기능을 수행하고 나머지(대기 부품)는 여분으로 대기 상태에 있다가 자가 진단에 의해 수행 부품에 결함이 발견되면 대기 부품 중의 하나가 새로운 수행 부품이 되어 주어진 기능을 수행하는 기법이다. N-버전 프로그래밍(version programming)은 하드웨어 결함허용 기법 중 TMR(Triple Modular Redundancy)과 유사한 기법으로 N개의 독립적인 소프트웨어 모듈이 수행한 결과를 비교하여 다수의 동일한 결과를 채택하는 기법이다. TMR은 세 개의 동일한 모듈에서 실행 결과를 받아 이 들의 값을 서로 비교하여 세 개 중에 두 개 이상의 결과가 같으면 이를 출력으로 발생시키는 보팅(voting) 작동에 기초한다[20]. 이외에도 목적한 응용 분야에 적합하도록 설계된 여러 가지 형태의 소프트웨어 결함허용 기법이 있다[14, 15, 16].

2.2 기존 결함 허용 기법의 한계점 및 멀티 에이전트

기존 결함 허용 기법은 사용자가 직접 망 관리의 일부를 담당하기 때문에, 부당한 사용자의 요구로 인하여 전체 시스템에 문제를 일으킬 수 있는 여지가 발생한다. 또한 기존 결함 허용 기법을 그대로 적용하여 분산 시스템 상에서의 응용 소프트웨어의 결함을 감지하고 복구할 수 있는 시스템에 대한 연구는 미흡한 실정이다. 복잡하고 많이 연결되어 있는 분산 망에서 사용자가 결함이 발생한 응용 소프트웨어를 수동적으로 찾기란 쉽지 않다. 그러므로 멀티 에이전트의 특성인 각 에이전트의 자치권을 보장하면서 서로간의 협력을 통해서 전체적인 망 관리, 특히 응용 소프트웨어에 대한 결함을 발견 및 복구하는 에이전트가 필요하다.

에이전트란 사용자가 요구하는 어떠한 작업을 사용자를 대신해서 자율적, 자동적으로 수행하는 소프트웨어 모듈을 말한다. 일반적으로 에이전트를 표현하는 대표적인 성질은 다음과 같다[18, 19]. 자율성(Autonomy)은 인간/다른 에이전트의 직접적인 개입 없이 행동한다. 사회성(Social ability)은 다른 에이전트와 협조/교섭을 통해 상호 작용한다.

반응성(Reactionity)은 환경을 지각하고 어떤 변화에 적시에 반응한다. 자발성(Pro-activeness)은 스스로 주도권을 갖고 목표 지향적 행동을 취한다.

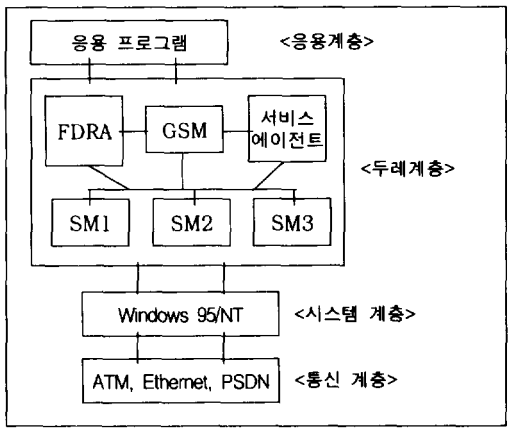
일상 생활의 복잡하고 정교한 작업 영역에서 전문적인 시스템이 인간을 대신해오고 있으며, 그 적용 영역을 넓혀가고 있다. 그러나 제어 영역이 다양화되고, 제어 기술이 점점 더 복잡해짐에 따라 전문가 시스템이 점점 더 커지고 중앙집중적인 하나의 시스템으로는 효율성에 한계가 있어 분산된 자치 시스템들의 협력으로 문제를 해결하려는 움직임이 생겼다[17, 18, 19].

3. FDRA

분산 멀티미디어 환경의 좋은 예는 성균관 대학교에서 1996년부터 개발 운영해온 두레(DOORAE)이다. 두레는 분산 멀티미디어 환경에서 상호 참여할 수 있는 응용 개발 환경이다. 따라서 본 논문에서는 분산 멀티미디어 환경에서 멀티 에이전트의 특성을 갖고 있으면서 서로간의 협력을 통해서 응용 소프트웨어에 대한 결함을 발견 및 복구할 수 있는 에이전트인 FDRA(Fault Detection Recovery based on Agent)를 제안한다.

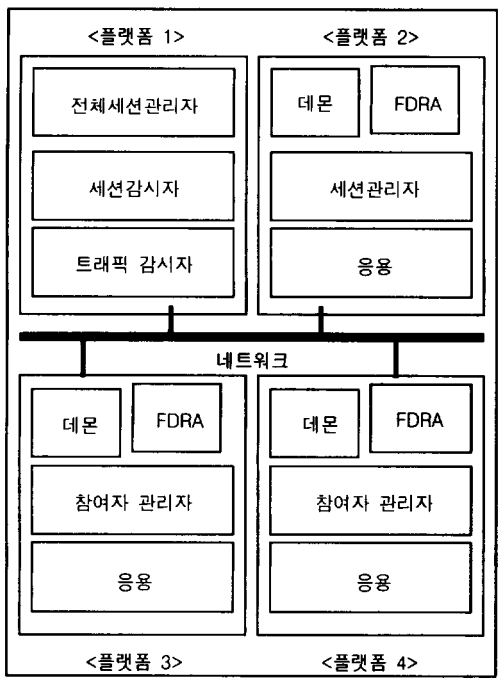
3.1 분산 멀티미디어 환경

두레는 상호 참여형 멀티미디어 응용 개발 환경으로 (그림 1)과 같이 4계층으로 구성되어 있다. 기존의 단일 멀티미디어 응용 개발에 발생하는 미디어 제어와 세션 관리에 대한 개발 과정의 비용을 줄임으로 상호 참여형 멀티미디어 응용의 개발을 가능하게 지원하는 미들웨어이다. 통신 계층은 분산 처리 환경의 메시지 전송을 담당하는 계층으로 MS-WINDOWS 95의 소켓 시스템(socket system)을 활용한다. 즉, 소켓 함수의 패밀리(family)로는 AF_INET를, 유형은 비연결형 또는 연결형을 사용한다. 시스템 계층은 기본적으로 기존의 운영체제(예 : Windows 95/NT) 기능을 사용한다.



(그림 1) DOORAE의 구조

두레 계층은 상호 참여형 멀티미디어 일반적인 응용을 개발하기 위해서 설계된 프레임워크이다. 두레 계층은 FDRA, 전체 세션관리자(GSM), 세션관리자(SM), 서비스 에이전트들로 구성된다. 두레에서 제공되는 서비스 에이전트에는 여러 개의 기능들을 가진다. 이 에이전트들은 상호 협력작업을 지원하기위한 것으로서 접근/동시성 제어 에이젼트, 오디오 혹은 미디어 자원의 공유를 가능하게 하는 미디어 제어 에이젼트, 공동작업 시 공동작업 공간(화이트보드 등)에서의 동일한 화면을 보게 하여 동시작업을 가능하게 하는 커플링 에이전트, 전자우편 혹은 인터넷 등 외부 네트워크와 접속을 담당하는 메일링 에이전트, 전체 세션에서 발생하는 세션의 종류, 이름, 참여자 명단, 통신의 양을 관리 하는 세션 감시 에이전트, 상용의 프리젠테이션 도구나 저작도구 등으로 개발된 소프트웨어를 공유하여 사용할 수 있게 해 주는 응용공유 에이전트 등이 있다. 또 이들의 외곽에는 통신 에이전트가 있어 여러 가지 통신 프로토콜을 지원 한다. (그림 2)는 근거리통신망(LAN)에서의 활성화된 하나의 세션을 나타낸다. 다중 세션이 활성화될 수도 있다.



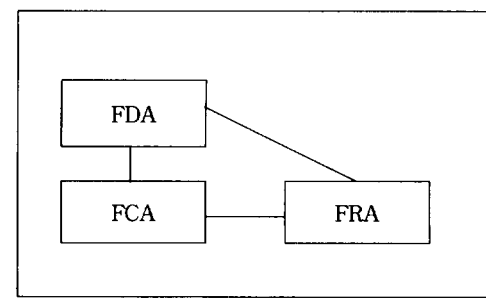
(그림 2) 근거리통신망에서의 단일 세션

플랫폼 1은 전체세션관리자(Global Session Manager)가 활성화되어 있으며 그 안에는 세션 감시자(Session Monitor)와 트래픽 감시자(Traffic Monitor)가 활성화되어 실행되고 있는 상태이다. 플랫폼 2는 세션의 초기 생성자인 세션관리자가 위치하는 곳이다. 플랫폼 2, 3, 4에는 데몬(Daemon)이 각각 하나씩 있는데 이 것은 응용 사용자에게 세션관리 서비스를 제공하기 위한 가장 기본적인 세션관리 서비스 객체이다. 이 데몬 생성시 동시에 오류를 감지하여 자동으로 복구시켜 주는 FDRA도 생성된다. 하나의 세션이 생성되면 이 세션에는 하나의 전체 세션 관리자, 하나의 세션 및 트래픽 감시자, 하나의 세션 감시자, 다수의 참여자 관리자가 생성된다.

3.2 FDRA의 구조

본 논문에서 제안하는 FDRA는 여러 기능의 에이전트가 존재하며 원활한 오류 감지 및 복구 기법을 수행하는 멀티 에이전트 시스템이다. 이들 에이전트 간의 상호 협조를 통하여 비정상적으로 종료되어 비 활성화되어 있는 응용 소프트웨어를 발견하고 이들 에이전트 간의 상호 협조를 통하여 자동적으로 활성화시켜주는 지능형 에이전트이다.

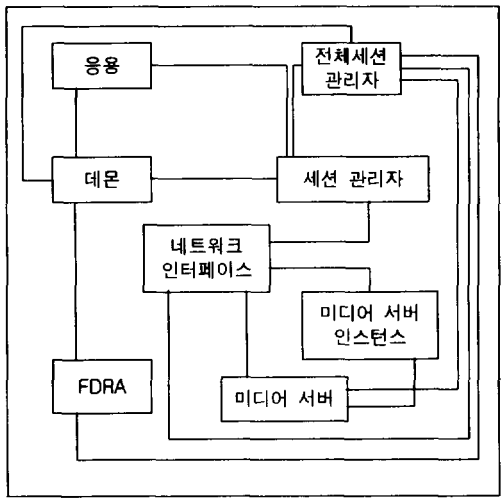
FDRA를 구성하는 구성 모듈로는 FDA(Fault Detection Agent), FCA(Fault Classification Agent), FRA(Fault Recovery Agent)이다. FDRA의 전체적인 구조는 (그림 3)과 같다.



(그림 3) FDRA의 구조

FDRA, 데몬, 세션 관리자, 전체 세션 관리자 및 응용과의 관계는 (그림 4)와 같다. 세션 관리는 다양한 서비스 객체들의 상호 작용에 의해서 지원된다. 최초 응용이 실행되면 응용은 데몬 객체의 존재를 찾아 응용으로써의 등록을 요구한다. 데몬은 해당 플랫폼에서

사용 가능한 응용의 고유 번호를 할당해 주게 된다. 데몬은 이 때 전체 세션 관리자에게 응용 등록함과 동시에 세션 생성에 필요한 미디어 자원과 발언권 방식 등 필요한 사항을 데몬에게 알려주고 세션의 생성을 요구한다. 전체 세션 관리자는 요구 받은 네트워크 자원을 데몬에게 할당한다. 데몬은 할당 받은 네트워크 자원을 가지고 세션관리자를 생성하게 되고 이때 생성된 세션 관리자는 요구 받은 미디어 자원에 대한 생성 요구를 각 미디어 서버에게 지정된 네트워크 자원을 통해 생성하도록 요구한다. 미디어 서버는 세션관리자로부터 부여 받은 네트워크 자원을 이용하여 이 후 응용으로부터 요구 받을 미디어 자원에 대한 서비스를 수행할 미디어 서버 인스턴스를 생성하게 된다. 미디어 서버 인스턴스가 생성되면 서버 인스턴스는 세션관리자에게 성공적으로 생성되었음을 알리게 되는데 이 때 자신을 접근할 수 있는 핸들을 전달한다. 세션 생성을 위한 모든 필요한 준비가 끝나게 되면 세션관리자는 응용 인터페이스를 통해 응용에게 세션이 생성되었음을 알려준다. 이후 응용은 세션에 대한 모든 서비스를 세션관리자를 통해 요구할 수 있게 된다.



(그림 4) FDRA, 데몬, 세션관리자 및 전체세션관리자와의 관계

3.3 FDRA의 알고리즘

FDRA에 대한 설명과 분석을 위해서 필요한 정의 및 표기는 다음과 같다.

(정의 1)

먼저 정수와 자연수의 집합을 정의한다.

Z : 정수의 집합

$Z = \{ \ldots, -2, -1, 0, 1, 2, \ldots \}$

N : 자연수의 집합

$N = \{ x \mid x \in Z, x > 0 \}$

(정의 2)

본 논문에 관계되는 에이전트들의 집합은 다음과 같다. 모든 에이전트들의 집합인 FDRA를 공집합이 아닌 임의의 집합이라고 하면, 집합 FDRA의 분할(partitions) πFDRA는 다음과 같은 집합이다.

$\pi FDRA = \{FDRA_1, FDRA_2, \ldots FDRA_i \ldots FDRA_k\}$ (i, k ∈ N)

여기서 πFDRA는 다음을 만족한다.

(1) i = 1, ..., k에 대하여 $FDRA_i$ 는 공집합이 아닌 집합 FDRA의 부분 집합이다.

(2) $FDRA = FDRA_1 \cup FDRA_2 \cup \ldots \cup FDRA_k (k \in N)$

(3) $FDRA_i$ 들 사이에서는 서로소이다. 즉, $i \neq k$ 이면 $FDRA_i \cap FDRA_k = \emptyset$ 이다.

같은 방법으로 FDA를 오류 감지 에이전트 들의 집합이라고 정의하면 집합 FDA와 분할 πFDA는 다음과 같다. 즉, $\pi FDA = \{FDA_1, FDA_2, \ldots FDA_i \ldots FDA_k\}$ (i, k ∈ N) 이고

$FDA = FDA_1 \cup FDA_2 \cup \ldots \cup FDA_k$ (k ∈ N) 이다.

같은 방법으로 FCA를 오류 분류 에이전트 들의 집합이라고 정의하면 집합 FCA와 분할 πFCA는 다음과 같다. 즉, $\pi FCA = \{FCA_1, FCA_2, \ldots FCA_i \ldots FCA_k\}$ (i, k ∈ N) 이고

$FCA = FCA_1 \cup FCA_2 \cup \ldots \cup FCA_k$ (k ∈ N) 이다.

같은 방법으로 FRA를 오류 복구 에이전트 들의 집합이라고 정의하면 집합 FRA와 분할 πFRA는 다음과 같다. 즉, $\pi FRA = \{FRA_1, FRA_2, \ldots FRA_i \ldots FRA_k\}$ (i, k ∈ N) 이고

$FRA = FRA_1 \cup FRA_2 \cup \ldots \cup FRA_k$ (k ∈ N) 이다.

다중 세션에서 i번째 활성화되어 있는 세션이 있을 때 이때 실행하는 에이전트들을 $FDRA_i$ 라고 정의한다. 정의된 오류 감지, 오류 분류 및 오류 복구에이전트들 ($FDRA_i$, FDA_i, FCA_i, FRA_i) 사이의 관계는 다음과 같다. 분할 $\pi FDRA_i = \{FDA_i, FCA_i, FRA_i\}$이고 $FDRA_i = FDA_i \cup FCA_i \cup FRA_i$ (i ∈ N)이다.

(정의 3)

다중 세션에서 i번째 활성화되어 있는 세션이 있을 때 이때 실행하는 프로세서를 P_i 라고 정의하기로 한다. 즉, 오류 감지 및 복구 대상이 되는 미디어, 미디어 인스턴스 및 응응 프로그램들의 집합은 다음과 같다.

VD : 비디오 서버 들의 집합,

VDI : 비디오 서버 인스턴스 들의 집합,

AD : 오디오 서버 들의 집합,

ADI : 오디오서버 인스턴스 들의 집합,

WB : 화이트보드 들의 집합,

AP : 응용 공유 들의 집합,

A : 응용 프로그램 들의 집합이라고 정의 한다.

모든 프로세서들의 집합인 P_i 를 공집합이 아닌 임의의 집합이라고 하면, 집합 P_i 의 분할(partitions) πP_i 는 다음과 같은 집합이다.

πP_i = {VD, VDI, AD, ADI, WB, AP, A} 이고

P_i = VD ∪ VDI ∪ AD ∪ ADI ∪ WB ∪ AP ∪ A 이다.

$VD = \{p_i \mid p_i \in P_i , i \in N\}$

$VDI = \{p_k \mid p_k \in P_i , k \in N\}$

$AD = \{p_m \mid p_m \in P_i , m \in N\}$

$ADI = \{p_n \mid p_n \in P_i , n \in N\}$

$WB = \{p_w \mid p_w \in P_i , w \in N\}$

$AP = \{p_x \mid p_x \in P_i , x \in N\}$

$A = \{p_z \mid p_z \in P_i , z \in N\}$ 이고

$VD \cap VDI \cap AD \cap ADI \cap WB \cap AP \cap A = \varnothing$ 이다.

(정의 4)

E_i는 에이전트들 $FDRA_i$ 와 프로세서 p_i가 실행하고 있을 때 발견되는 오류(error)들의 집합으로 정의한다.

즉, $E_i = \{e_i \mid i \in N\}$이다.

F_i는 에이전트들 $FDRA_i$ 와 프로세서 p_i가 실행하고 있을 때 오류의 원인이 되는 결함(fault) 들의 집합으로 정의한다. 즉, $F_i = \{f_i \mid i \in N\}$ 이다.

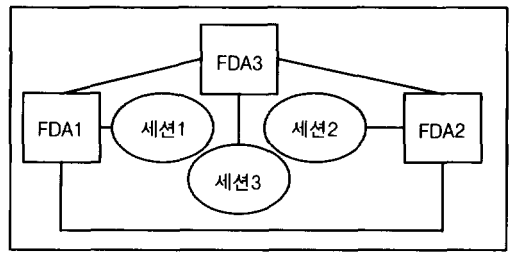
R_i는 에이전트들 $FDRA_i$ 와 프로세서 p_i가 실행하고 있을 때 발견되는 오류(error)를 복구하는 모듈들의 집합으로 정의한다. 즉, $R_i = \{r_i \mid i \in N\}$ 이다.

C_i는 에이전트들 $FDRA_i$ 와 프로세서 p_i가 실행하고 있을 때 발생하는 오류의 정보를 갖고 있는 프로세서 데이터베이스(PDB)에서의 리턴 코드 값을 집합으로 정의한다. 즉, $C_i = \{c_i \mid i \in N\}$이다.

K_i는 에이전트들 $FDRA_i$ 와 프로세서 p_i가 실행하고 있을 때 발생하는 오류의 유형에 대한 정보를 갖고 있는 지식베이스(KB)에서의 리턴 코드 값을 집합으로 정의한다. 즉, $K_i = \{k_i \mid i \in N\}$이다.

FDRA와 데몬과의 사이에 다음과 같은 알고리즘이 실행된다. 데몬이 실행된다. 데몬은 공유 메모리를 생성한다. 데몬은 공유 메모리에 정보를 삽입한다. 즉, 데몬 핸들(handle) 및 각 실행 파일 이름 등이다. FDRA를 실행한다. FDRA는 데몬에게 생성과 핸들을 알린다. 데몬은 세션 생성시와 종료 시에 세션 정보를 알린다. FDRA는 서버 비정상 종료 시에 종료된 사실을 알린다. 데몬이 서버 재실행 후에 세션을 형성한다. FDRA는 데몬이 비정상 종료 시 공유 메모리에 세션 형성을 위한 정보를 실은 후에 데몬 실행 후 실행시킨다. 데몬은 데몬이 재 실행 된 것과 핸들을 알린다. 데몬은 데몬이나 서버의 정상적 종료 시 메시지로 알린다.

세션이 활성화 되어 있는 동안에 FDRA에 있는 에이전트인 FDA, FCA 및 FRA 들이 순서대로 실행된다. FDA와 세션과의 관계는 (그림 5)와 같다.



(그림 5) FDA 및 세션과의 관계

FDA는 오류를 감지하는 핵심 에이전트로 고장 감지 정보 흐름은 패킷 정보를 보내어 그 응답 결과 상태를 분석하여 오류의 발생 여부를 분석한다. 세션의 복원을 진행하기 위해서는 먼저 오류 감지를 하기위한 방법이 필요하다. MS-Windows 95/NT의 시스템에서 오류 감지를 위한 방법은 실행된 프로세서의 상태를 보관하는 프로세서 데이터 베이스를 주기적으로 검사하는 방법이다. 그러나 이것은 두레를 이용한 세션의 상태와는 무관한 프로세서까지 데이터베이스를 검사해야 한다는 단점이 있다. 그래서 두레를 이용한 시스템에서는 데몬이나 세션 메니저가 생성한 프로세서에 대한 정보를 결함 감지기에게 통보하여 결함 허용 시스템에서 세션과 직접 연관된 프로세서만을 주기적으로 폴링(polling)하여 오류 감지를 수행한다. PDB(Process Database)의 구조 중 프로세서의 살아있는 상태를 알려면 종료 상태의 내용을 보고 알 수 있다. 예를 들면, 하나의 프로세스가 여전히 활동하고 있으면 STILL_

ACTIVE(핵사 코드값 : 103)이다. PDB의 이 값을 알려면 API 함수[21] 중 GetExitCodeProcess를 사용했을 때 리턴(return)되는 코드값으로 알 수 있다.

Set of Detection = {Set of error, Set of fault, Set of error detector}

여기에서 Set of error = {E, T, D}

- E : 발생하는 오류
- T : 시간 간격
- D : 발생하는 도메인 위치

Set of fault= {F, T, C}

- F : 발생하는 고장(fault)
- T : 시간 간격
- C : PDB에서 찾은 오류 코드(error code), 즉, GetExitCodeProcess 실행 후에 리턴 (return) 되는 코드 값

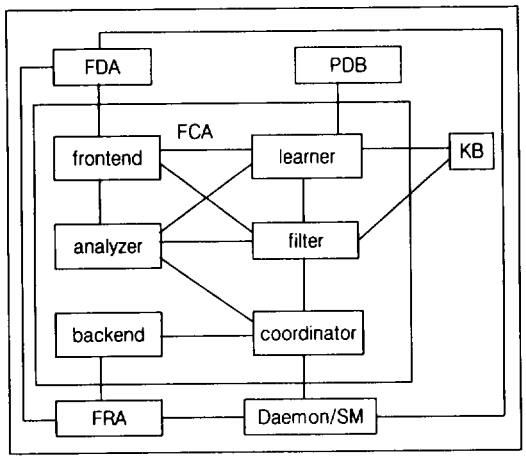
Set of error detector = {Addr_FDA, Func_FDA}

- Addr_FDA : FDA의 주소 정보
- Func_FDA : FDA의 기능(function)은 관계 R의 원소의 순서 쌍에서 모든 원소의 집합을 정의역 (domain)이라하고 Dom(R)로 표시하고, 또한 한 원소의 집합은 치역(range)이라 하고, Ran(R) 로 표시한다.

$Dom(R) = \{ (p_i, e_i) \mid (p_i, e_i) \in R \} \subseteq P_i \times E_i$

$Ran(R) = \{ c_i \mid c_i \in R \} \subseteq C_i$

FCA의 구조는 (그림 6)과 같다. FCA는 사용자와 결함 감지 및 복구를 위한 FDA 및 FRA와 같이 에이전트를 기반으로 한 응용 분야와의 상호 작용을 위한 인터페이스로서의 역할을 하는 에이전트이다.



(그림 6) FCA의 구조

FCA는 FDA에서 접수된 오류와 PDB를 이용하여 자동으로 분류할 수 있는 지적 대리인이다. 분류된 데이터를 가지고 지식베이스(Knowledge Base)를 통해 분석적인 정보를 생성하고 학습의 기능도 가진다. front-end는 오류 감지를 받아들이는 일과 그 오류 복구의 결과를 사용자에게 제공하는 기능을 가진다. backend는 FRA에게 정보를 주거나 받는 역할을 한다. coordinator는 세션관리자나 데몬을 통하여 다른 FDA와의 정보 교환을 한다. filter는 PDB(Process Data Base)에서 얻은 정보를 지식베이스(KB)를 사용하여 요구에 맞게 정리 목록화 하여 지식베이스에 저장하는 기능이다. analyzer는 FDA로부터 frontend를 통해 받은 오류 정보를 분류하는 역할을 한다. learner는 시스템 오류 유형의 이력(history)을 보관하며 이후의 오류와 비교하며 학습 자료로 사용한다. 이를 위하여 신경 망에서의 학습 알고리즘을 도입한다. 본 논문에서 제안하는 학습 알고리즘은 Q-learning이다. 학습 에이전트가 동작하는 초기에는 오류 진단 및 분류를 위한 지식이 없으므로 PDB(Process Data Base)를 통하여 장애 진단에 필요한 인지 정보를 받는다. 이 것을 기초로 지식베이스가 쌓이면 장애를 좀더 효과적으로 진단할 수 있다. 각 FCA는 오류가 발생한 유형과 이력을 다음과 같은 형태로 가지고 있다. FCA는 PDB에서 얻은 정보를 바탕으로 지식베이스에 오류의 이력을 보관하여 이후의 질의와 비교하며 학습자료로 사용한다.

Set of History = {C, Information of fault, Set of classifier}

여기에서

C : 오류코드,즉 PDB에서 찾은 내용으로서 지식베이스의 키워드로 사용된다.

Information of fault = {D, T, F, N, conf(C)}

- D : 고장이 발생되는 도메인의 위치
- T : 고장의 유형
- F : 고장이 발생하는 빈도수
- N : 고장난 노드의 수
- conf(C) : 이 정보의 키워드에 대한 PDB의 신뢰도

Set of classifier = {Addr_FCA, Func_FCA, conf(FCA)}

- Addr_FCA : 키워드와 관련된 오류 정보를 가진 FCA의 정보

- conf(FCA) : 그 FCA에 대한 신뢰도
- Func_FCA : FCA의 기능은 관계 R의 원소의 순서 쌍에서 모든 원소의 집합을 정의역(domain)이라하고 Dom(R)로 표시하고, 또한 한 원소의 집합은 치역(range)이라 하고, Ran(R)로 표시한다.

$Dom(R) = \{ (c_i, k_i) \mid (c_i, k_i) \in R \} \subseteq C_i \times K_i$

$Ran(R) = \{ f_i \mid f_i \in R \} \subseteq F_i$

제안하는 시스템에서는 검색된 오류 정보에 대한 PDB의 피드백을 가지고 신뢰도를 나타낸다.

Conf(C)의 신뢰도는 다음과 같이 표현된다.

$Conf_n = Conf_n + \alpha * quality\_of\_feedback$

- α : PDB 피드백에 대한 Conf 민감도를 지정하기 위한 학습 비율이다.
- quality_of_feedback : 제공하는 오류 정보에 대한 사용자의 만족도

Conf(FCA)의 신뢰도는 다음과 같이 표현된다.

$Conf_n = Conf_n + \beta * quality\_of\_feedback$

- β : PDB 피드백에 대한 Conf 민감도를 지정하기 위한 학습 비율이다.
- quality_of_feedback : 제공하는 오류 정보에 대한 PDB의 만족도

본 논문에 제안하는 학습알고리즘은 강화 학습(Reinforcement Learning)[24]이고 그 중에서 Q-Learning이다. 이러한 메카니즘을 통하여 사용자 지향의 형태로 진화되어 간다. 그 내용은 다음과 같다.

$Q(s, a) = Q(s, a) + \alpha(r + \max_{a'}(s', a') - Q(s, a))$

(Q : Q value, r : reward, s : situation, a : action, s′: next situation of s, a′: next action of a)

FRA는 FCA로부터 오류 분류 정보를 바탕으로 오류를 복구하는 모듈이 실행된다. 고장 복구 정보흐름은 패킷 정보를 보내어 오류가 발생한 응용 소프트웨어를 정상적인 상태로 복구 시켜 준다. 주기적인 폴링에 대하여 감지 대상으로부터의 일정시간이나 일정 횟수의 응답이 없는 경우 결함 허용 시스템은 비정상적인 상황으로 간주하고 감지 대상을 생성한 프로세서에게 그 사실을 통보한다. 즉, 세션의 진행 과정 중 세션의 미디어 서비스 인스턴스가 비정상적으로 종료되는 경우가 있다. 이런 경우 세션의 진행을 중단할 수도 있지만 허용하는 한 미디어 서비스 인스턴스를 재 활성화시켜 사용자에 대한 보호를 한다.

오류가 발견된 후에 본 시스템에 나타나는 오류의 유형은 다음과 같다. 오류의 유형은 복구 방법에 따른 분류이다. 비정상적인 상황의 통보에 대하여 복원의 과정을 수행한다. 복원의 과정을 수행하기 전에 FCA에 의해서 유형별로 자동 분류된다. 오류의 유형에 따라 FRA가 복구 방법에 대한 개요(scheme)는 다음과 같다.

Set of Recovery = {C, Set of recovery module, Set of recovery agent }

여기에서

C : 오류코드, 즉 PDB에서 찾은 내용으로서 지식베이스의 키워드로 사용된다.

Set of recovery module = {T, R}

- T : 고장의 유형
- R : 고장시 실행 모듈

Set of recovery agent = {Addr_FRA,Func_FRA}

- Addr_FRA : FRA의 주소 정보
- Func_FRA : FRA의 기능은 관계 R의 원소의 순서 쌍에서 모든 원소의 집합을 정의역(domain)이라하고 Dom(R)로 표시하고, 또한 한 원소의 집합은 치역(range)이라하고, Ran(R)로 표시한다.

$Dom(R) = \{ (f_i, r_i) \mid (f_i, r_i) \in R \} \subseteq F_i \times R_i$

$Ran(R) = \{ p_i \mid p_i \in R \} \subseteq P_i$

결함의 유형에 따른 복구 방법에 대한 개요(scheme)는 다음과 같다. 복원이 가능한 경우와 복원이 불가능한 경우로 나누어 고려한다.

(경우 1) 복원할 수 없는 경우

복원할 수 없는 것은 오디오/비디오와 같은 하드웨어 자원에서 오류가 발생한 경우이다.

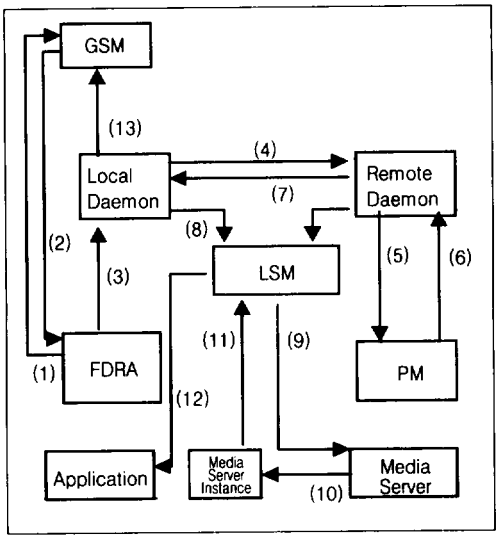
(경우 2) 복원이 가능한 경우

먼저 복원이 가능한 경우는 오디오/비디오 등 응용 프로그램과 같이 단순 재실행과 정보 재전송으로 복원할 수 있을 때이다.

단순 재 실행인 경우 데몬은 할당 받은 네트워크 자원을 가지고 세션관리자를 생성하게 된다. 이 때 생성된 세션관리자는 요구 받은 미디어 자원에 대한 생성을 요구한다. 미디어 서버는 미디어 서버 인스턴스를

생성한다.

세션이 진행 중에 발생하는 오류에 대하여 복구 를 위한 메시지 흐름도는 (그림 7)과 같다. 동시 다 세션을 잘 유지하기 위한 복구 알고리즘은 세션에 대한 정보를 잘 알고 있어야 한다. 이 때 도미노효과(Domino Effect)는 고려하지 않는다. 프로세서들 간의 정보 교환과 복구점이 조화되지 않으면 프로세스 사이에 계속적인 롤백 전달의 사태가 일어나는 것을 도미노 효과라 한다[22].



(그림 7) 복구를 위한 메시지 흐름

(1) FDRA는 전체세션관리자(GSM)에게 모든 진행 중인 세션에 대한 정보를 요청한다.

(2) 전체세션관리자(GSM)는 FDRA에게 세션에 대한 정보를 알려준다.

(3) FDRA는 그 위치에 있는 데몬(Daemon)에게 복구할 것을 알린다.

(4),(5) 세션 진행 중에 복구 요청을 받은 데몬 (Daemon)은 원격지의 데몬(Daemon)들에게 세션 진행중에 복구 요청이 있다는 것을 알린다.

(6) 원격지 데몬(Daemon)은 자신의 세션관리자 (SM)에게 세션 진행 중 에 복구 요청이 있다는 것을 알린다.

(7) 원격지 세션관리자(SM)는 자신의 데몬(Daemon)을 통해 오류가 발생한 곳의 데몬(Daemon)에게 응답을 보낸다.

(8-11) 원격지에 모두 알린 데몬(Daemon)은 자신의 참여자 관리자(PM : Participant Manager)를 생 성하고 이어서 세션에 필요한 미디어를 실행시킨다.

(12) 사용할 자원의 생성이 모두 끝나면 응용을 실행시킨다.

(13) 응용의 실행이 모두 끝나면 오류 복구 처리는 완료한다.

복구된 프로그램들은 세션의 정상적인 참여자가 된다.

4. 시스템 평가

제안된 시스템은 Visual C++로 설계 및 구축 가능하다. 오류 감지에서 제안된 방법의 나은 점을 DEVS 형식론을 이용하여 모델링 및 시뮬레이션을 통하여 비교한다. DEVS(Discrete Event System Specification)는 Bernard P. Zeigler에 의해 개발된 이산 사건 모델들의 계층 구조적 모듈화 방법을 제공해주는 형식론이다. 시스템을 작은 모듈들로 나누고 그것들로 전체 시스템을 계층적으로 구성해 나간다. 각 모듈들은 원자(atomic) 모델로 표현되며 그것들의 계층적 구성은 커플(coupled) 모델로 표현된다. DEVS 형식론에서 가장 기본이 되는 모델인 원자 모델은 다음과 같은 집합으로 표현된다[23].

$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$

X : 외부 입력 사건들의 집합

S : 상태 변수들의 집합

Y : 외부 출력 사건들의 집합

δ_{int} : 내부적 상태 변환 함수

δ_{ext} : 외부적 상태 변환 함수

λ : 출력 함수

t_a : 시간 진행 함수

원자모델을 결합하여 새로운 커플모델을 형성한다.

(기존 방식)

DEVS 형식론에서 기존 방식의 변수를 정의하면 다음과 같다. 원자 모델은 EF, RA1, UA1, ED1이고 이 원자 모델들을 결합하여 새로운 커플모델을 형성한다. 처음에 외부 입력 사건들 중에서 폴링 시간을 받아들인다. 그 값이 RA1과 UA1에서 각각 입력 값이 되어 실행 한 후에 그 결과 값이 ED1의 입력 값이 되어서 그 결과 값이 EF의 transducer를 통해서 관측될 수 있다.

모델(방법 1)	상태 변수	목 표
EF(genr)	Poll-int	Polling interval
RA1	App-cnt Ra-re-t-a	두레 관련 응용프로그램의 개수 반응시간 누적
UA1	App-cnt2 Ua-re-t-a	두레에 무관한 응용프로그램의 갯수 반응시간누적
ED1	Ra-re-t-a Ua-re-t-a Tat1-t-a	ra 반응시간 누적 ua 반응시간 누적 ra 반응시간 누적 + ua 반응시간 누적

(제안된 방식)

DEVS 형식론에서 제안된 방식의 변수를 정의하면 다음과 같다. 원자 모델은 EF, RA2, ED2이고 이 원자 모델들을 결합하여 새로운 커플모델을 형성한다. 처음에 외부 입력 사건들 중에서 폴링 시간을 받아들인다. 그 값이 RA2에서 입력 값이 되어 실행 한 후에 그 결과 값이 ED2의 입력 값이 되어서 그 결과 값이 EF의 transducer를 통해서 관측될 수 있다.

모델(방법 2)	상태 변수	목표
EF(genr)	Poll-int	Polling interval
RA2	App-cnt Ra-re-t-a	두레 관련 응용프로그램의 개수 반응시간 누적
ED2	Ra-re-t-a Sm-t-a Tat2-t-a	ra2 반응시간 누적 SM 등록된 정보를 주는 시간 누적 Ra 반응시간 누적 + SM 등록된 정보를 주는 시간

본 논문에서는 전체 응용에 대해서 폴링 방식을 사용하는 방식과 폴링을 사용하되 기존 시스템의 정보를 갖고 있는 세션 관리자를 이용하여 등록되어 있는 필요한 응용만 찾아서 폴링 시간을 줄이는 방식 2가지를 비교하여 효율성 검토를 한다. 시뮬레이션 모델을 통한 관측 목표와 관측 값 계산에 관련된 변수를 상태 변수로 가지는 모델이다.

대상 (object)	관측 지수		관측 모델
	기존 방식	제안된방식	
각 작업 (Each job)	ED1는 EF에 RA1, UA1 반응시간 총합을 준다(종료 시점에서).	RA2는 반응시간 누적과 SM 등록된 정보를 주는 시간 누적을 준다(종료 시점에서).	transducer

기존 방식과 제안된 방식의 비교는 다음과 같이 할 수 있다. 오류 감지 부분에서 기존 방법의 감지 시간(Tat1-t-a)은 다음과 같다.

$$Tat1\text{-}t\text{-}a = Poll\_int*(App\_cnt + App\_cnt2)$$

제안된 방법의 감지 시간(Tat2-t-a)은 다음과 같다.

$$Tat2\text{-}t\text{-}a = Poll\_int*(App\_cnt) + Sm\_t\_a$$

만일 프로세서 간의 메시지가 전달될 때 걸리는 시간을 t라고 하면 한번 폴링 시간은 2t가 된다. App_cnt2 > 1이므로 Poll_int*App_cnt2 > Sm_t_a이다. 즉, 제안된 방법이 더 효율적이다.

오류 유형을 분류하고 복구하는 방식에서도 제안된 방식은 에이전트의 개념을 도입하였다. 오류의 원인이 되는 결함을 발견하고 복잡하고 많이 연결되어 있는 분산 망에서 사용자가 결함이 발생한 응용 소프트에어를 수동적으로 찾기란 쉽지 않다. 그러므로 멀티 에이전트의 특성인 각 에이전트의 자치권을 보장하면서 서로간의 협력을 통해서 전체적인 망 관리, 특히 응용 소프트웨어에 대한 결함을 발견하고 오류의 유형을 PDB를 이용하여 자동적으로 분류하고 이 정보를 축적하여 지식베이스에 저장하며, 또 다른 오류가 발생 했을 때 학습 기능을 가지고 능동적으로 대처하고 자동적으로 복구하는 에이전트 기법을 사용한 특징이 있다. 제안된 논문의 결점은 도미노효과를 고려한 검사점 및 복구 알고리즘에 대한 연구가 미흡한 점이다.

5. 결 론

멀티미디어 통신의 발달로 분산 시스템의 중요성은 더해가고 있다. 분산 시스템은 하나의 노드 또는 자원이 결함이 발생하더라도 전체 시스템에 영향을 미치지 않기 때문에 결함 허용 시스템 설계를 위한 좋은 조건을 가진다. 본 논문에서는 분산 시스템 상에서 윈도우즈 95 API 함수를 사용하여 결함을 감지할 수 있고 자동으로 복구할 수 있는 에이전트의 설계를 제안하였다.

본 논문에서 제안한 FDRA는 멀티에이전트, 즉 이들 에이전트들 간의 상호 협조를 통하여 비정상적으로 종료되어 비활성화되어 있는 응용 소프트웨어를 발견한다. 이때 세션과 직접 연관된 프로세서만을 주기적으로 폴링하여 오류 감지를 수행한다. 오류가 감지되면 자동으로 오류 유형을 분류하고 그 유형에 따라 자동으로 복구 시켜주는 에이전트이다. 따라서, 본 논문

에서는 분산 멀티미디어 환경에서 응용 소프트웨어를 활용하고 운영하는 이에게 그 시스템의 결함을 미리 통보하여 예방책을 마련하는 결함 허용 소프트웨어에 대한 하나의 모델을 제안하였다. 향 후 연구 과제는 분산 멀티미디어 환경에서 다중 세션이 활성화되어 있는 경우에 도미노 효과를 고려한 검사점 설정 및 복구 알고리즘에 대한 연구 등이다.

참 고 문 헌

[1] Eung-Nam Ko, Dae-Joon Hwang, "Implementation of a Fault-Tolerant System Running on DOORAE : FTSD," In proceedings of IEEE ISCE'98, Taipei, Taiwan October 19- 21,1998.

[2] Ralf Steinmetz and Klara Nahrstedt, "Multimedia : computing, communications & Applications," Prentice Hall P T R.

[3] 박길철,황대준, "멀티미디어 원격 교육 시스템 설계", 한국 정보 처리학회 멀티미디어 시스템 연구회 학술대회 논문집, pp.54, 1994.

[4] Roy D. Pea, "Learning through multimedia", IEEE Computer Graphics & Application, pp.58-66, Jul. 1991.

[5] Matthew E. Hodges, Russel M.Sasnett. "Multimedia Computing-Case studies from MIT project Athena-.," Appison-Wtsley pub., pp.29-37, 1993.

[6] Victoria Rosenborg, "A guide to multimedia," New Riders pub., pp.187-205, 1993.

[7] 전준걸, 황대준, "상호 참여를 위한 탁상회의 시스템의 구현", 95년 한국 정보 과학지 가을 학술 발표 논문집 Vol.22, No.2, pp.1041, 1995.

[8] 박길철, 황대준, "네트워크 환경에서 멀티미디어 객체 동기화 모델 설계", 한국 정보처리 응용학회 제 1 권 2호, pp.568-571, 1994.10.

[9] 장순주, 임종규, 정구영, 구용완, "분산 시스템에서 결함 허용성을 위한 프로세스 이주 연구", 한국 정보 과학회지 가을 학술발표 논문집 Vol.21, No2, pp.132, 1994.

[10] 김상용,장영철,장병탁,이창훈, "멀티에이전트 시스템을 이용한 교통 제어 시스템의 구현", 1995년 한국 정보 처리 학회 추계 학술 발표 논문집 제2권 제2호, pp.619-623, 1995.

[11] Michel Wooldridge & Nicholas R, Jennings, "Intelligent Agent : Theory and practice," Submitted to knowledge Engineering Review, October, 1994.

[12] Iain D. Craig Dept of computer science univ. of warwick, "A perspective on Multi-Agent systems."

[13] Ernest A.Edmonds, Linda Candy, Rachel Jones, Bassel Soufi, "Support for Collaborative Design : Agents and Emergence," Communications of the ACM, Vol.37, No.7, pp.41-44, July 1994.

[14] 김문희, "결함 허용 시스템의 설계 고려사항 및 동향", 정보과학회지, 제11권 제3호, pp. 7, 1993.

[15] Jonson, b.w. "Design and Analysis of Fault-Tolerant Digital Systems," Addison Wesley, 1989.

[16] Randell, B., "System Structure for Software fault tolerance," IEEE Trans. on Soft Engr., pp.220-232, June 1975.

[17] Stephens,LM. & Merx,M. "The effect of agent organization on the performance of DAI systems," IEEE Transactions on systems, Man and Cybernetics, 1989.

[18] 전인걸, 이은석, "인터넷 상에서의 정보 검색 을 위한 지능형 정보 검색 에이전트의 설계", 1996 한국 정보 처리학회 추계 학술 발표 논문집 제3권 제2호, pp.546-551, 1996.

[19] Wooldrige,Jennings, "Agent theories, Architectures and Languages : A survey," ECAI-94 workshop on Agent Theories, Architectures and Language, Oct. 1994.

[20] 윤재영, 김학배, "Rollback과 Roll-forward 기법을 사용한 TMR 고장의 시간 여분 복구 정책", 한국 정보처리학회 논문지 제 6권 제 1 호, pp.216-224, 1999년 1월.

[21] Matt Pietrek, "Windows 95 System Programming SECRETS," IDG Books Worldwide,Inc., pp.80, 1995.

[22] 허신, "소프트웨어 결함 허용 기법에 대한 고찰", 한국 정보과학회지 제 11권 제 3호, pp.32-39, 1993, 6월.

[23] Bernard P. Zeigler, "Object-Oriented Simulation with hierarchical, Modular Models", Academic Press, 1990.

[24] Martin T.Hagan, Howard B. Demuth, Mark Beale,

"Neural Network Design," PWS Publishing Company, pp.43, 1996.

고 응 남

e-mail : ssken@hanmail.net

1984년 연세대학교 수학과(이학사)

1991년 숭실대학교 정보과학 대학원(공학석사)

1999년 현재 성균관대학교 정보공학과(박사과정 수료)

1983년 11월~1993년 2월 대우통신 컴퓨터 개발부 선임 연구원

1993년 3월~1997년 2월 동우대학 전자계산과 교수

1997년 3월~현재 신성대학 컴퓨터 계열 교수

관심 분야 : 멀티미디어, 결함 허용, 에이전트 등

황 대 준

e-mail : djhwang@yurim.skku.ac.kr

1978년 경북대학교 전자 계산기 공학과(공학사)

1981년 서울대학교 자연과학대학 계산통계학과(이학석사)

1986년 8월 서울대학교 자연과학대학 계산통계학과(이학박사)

1981년 9월~1987년 2월 한남대학교 교수

1990년~1991년 MIT 컴퓨터과학연구소 연구교수

1987년 3월~현재 성균관대학교 정보공학과 교수

관심분야 : 멀티미디어, 병렬 처리, 원격교육 등