정보통신 분산 응용의 시스템 차원 시험을 위한 감시 기법

이 한 영[†] · 민 병 준[††] · 김 문 회[†††] · 서 동 선[††††] · 허 웅[††††]

요 약

차세대 정보통신 서비스 응용들이 분산처리 환경에서 객체지향 기법으로 개발되고 있다. 본 논문에서는 개발 단계 뿐 아니라 운용 단계에서 이 서비스들의 효율적인 시험을 위하여 분산 프로그램의 수행 상태를 감시하기 위한 하부구조의 감시기능을 정의하고 이를 토대로 시스템 차원의 시험 방법을 제시한다. 본 논문의 방법에서는 객체 수행을 감시하고 감시된 데이타를 모으는 것을 임무로 하는 감시 서버와 이를 분석하고 시험할 내용과 순서를 정하는 시험자와의 기능을 분리시킴으로써 감시에 따른 영향을 최소화할 수 있다. 또한, 오류가 발생된 부분의 시험을 반복함으로써 정확한 진단을 내릴 수 있게 된다. 이 기법은 실시간 결함허용 시스템 구현의 핵심적 요소로서, 정보통신 시스템 뿐 아니라 일반 분산 시스템에도 적용 가능할 것이다.

A Monitoring Mechanism for the System-Level Test of Telecommunications Distributed Applications

Han Young Lee[†] · Byung Jun Min[††] · Moon Hae Kim[†††] · Dong Sun Seo[††††] · Wong Hur[††††]

ABSTRACT

Object-oriented programming is used to develop next-generation telecommunications services running on the distributed processing environment. In order to test these services efficiently at the system-level during not only in the development phase but also in the operation phase, we define an embedded monitor service within the infrastructure to monitor the operation of the distributed programs, and describe a system-level test mechanism based on the monitor service. By separating the function of monitor server which monitors operations of objects and collects monitored data and that of tester which makes analysis and decides the sequence of test events, the invasive effect of monitoring can be minimized. At the same time, accurate diagnosis on the system can be achieve by exploiting the test mechanism. The mechanism, as a core component for the implementation of real-time fault-tolerant systems, is applicable to general-purpose distributed systems as well.

1. 서 론

다양한 정보통신 서비스를 구현하기 위한 응용 프로그램들이 객체지향 기법에 근간을 두고 분산의 투명성(transparency)을 제공하는 플랫폼 상에서 개발되는 추세가 두드러지고 있다. 이러한 분산처리 환경은

† 정 회 원:한국통신 통신망연구소
†† 정 회 원:시립 인천대학교 전자계산학과
††† 정 회 원:건국대학교 전자계산학과
†††† 정 회 원:명지대학교 전자공학과
논문접수:1995년 10월 25일, 심사완료:1996년 1월 23일

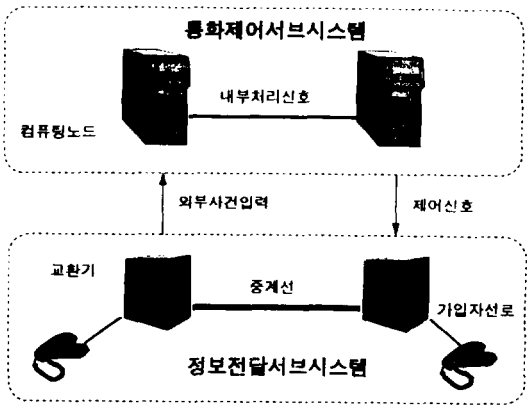
응용 프로그램에게 다양한 분산의 투명성을 제공하여 자원의 공유와 상호운용성(interoperability)을 제공한다. 또한 객체지향 소프트웨어는 재사용 가능성과 프로그램 이식성을 높여 줌으로써 소프트웨어 개발을 용이하게 하고 있다. 그러나, 이러한 분산 객체지향 프로그램들을 시험하는 문제는 일반 단일 컴퓨터에서 수행되는 전형적인 프로그램들을 시험하는 것에 비하여 매우 어렵다. 실제로 분산 객체지향 소프트웨어의 개발 및 유지 보수 단계 중디버깅 및 시험에 투입되는 비용이 전체 비용의 50% 이상을 차지하는 것으로 나타나고 있다[12]. 그 이유는 여러노드에서 수행되는 프로그램 모듈들을 동시에 제어해야 하기 때문이다. 또한 객체지향 소프트웨어의 시험은 전통적인 소프트웨어의 시험과 그 방법을 달리해야 할 필요가 있다. 전통적인 소프트웨어는 구조를 바탕으로 시험이 이루어지는 반면, 객체지향 소프트웨어는 어떤 사건(event)의 발생으로 인한 동작 상태의 변화(behavior)로 검토되어야 하기 때문이다. 전통적인 소프트웨어는 명령형의 언어로 작성되어 프로그램의 각 기능을 여러 단위로 구분하여 묘사함으로써, 개발 단계에서 단위 시험, 통합 시험, 시스템 시험의 과정을 거쳐서 결함이 없는 소프트웨어를 획득할 수 있었다. 명령형의 언어란 프로그램 소스 문장의 순서가 컴파일 결과의 목적 명령 수행의 순서를 결정하는 것으로 현재 사용되는 대부분의 언어가 이에 속한다. 이와 달리, 서술형의 언어는 그래프 등을 이용하여 엄밀하게 묘사되는데 각 노드는 문장을, 이들을 연결하는 선은 제어 흐름을 나타낸다. 객체지향 소프트웨어는 사건에 의해 구동 되는 성향을 갖는데 이로 말미암아 객체지향 소프트웨어의 시험은 서술적인 성질을 갖게 된다[10]. 시험은 개발단계 뿐 아니라 운용 또는 유지 보수 단계에서 시스템 구성 요소의 결함을 찾아내고, 분석하여, 의심되는 부분에 수정을 가하는 행위를 의미한다. 소프트웨어에 대한 시험은 정적인 시험과 동적인 시험으로 구분된다. 정적인 시험이란 프로그램 소스 코드의 각 기능 모듈에 대한 검사를 말하고, 동적인 시험이란 프로그램 수행 중에 메모리 덤프, 추적, 브레이크포인트 등의 방법을 이용하여 프로그램 오류를 발견하고 나아가서 프로그램의 전반적인 동작 상태의 분석을 통하여 프로그램의 성능 평가도 가능하게 해주는 것을 말한다. 프로그램 수행에 관한 정보 수집, 즉 사건발생을 감지하고 이들에 대한 데이타를 모으는 것을 감시라고 한다. 이러한 행위는 동적인 프로그램 시험과 고도의 서비스 품질을 지원하는 중요한 역할을 한다.

본 논문에서는 객체지향 기법에 기반을 두고 분산처리 환경에서 수행되는 분산통신 서비스 응용 프로그램의 시험을 위하여 분산 프로그램의 수행 상태를 감시하기 위한 하부구조(infrastructure)의 감시 기능에 대하여 정의한다. 이 기능을 바탕으로 한 시스템 차원의 시험 기법을 제시하고 이를 구현하는데 따르는 제반 문제점들에 대하여 논한다. 논의된 내용은 광대역 종합정보통신망의 네트워킹 시스템을 위한 분산처리 환경 하에서의 객체지향 응용 서비스 시험 구현 경험을 바탕으로 한 것이다. 본 논문의 구성은 다음과 같다. 정보통신 시스템의 일반적인 특성에 대하여 2절에서 논하고, 광대역 멀티미디어 정보통신 서비스를 위한 개방형 구조를 도출해 내기 위한 TINA (Telecommunications Information Networking Architecture) 컨소시움에서 제시된 분산처리 하부구조(Distributed Processing Environment)에 대하여 3절에서 요약한다. 4절에서는 기존의 프로그램 시험방법들에 대하여 살펴보고 5절에서 개방형 정보통신 시스템 응용의 시험을 위한 감시기법을 제시한다. 이를 토대로 한 시스템 차원의 시험 방법이 6절에서 논의되고, 마지막으로 7절에서 결론을 맺는다.

2. 정보통신 시스템 모델

이 절에서는 정보통신 시스템의 특성에 관하여 논한다. 전형적인 정보통신 시스템을 분산 구조의 관점으로 보면(그림 1)과 같이 정보전달 서브 시스템과 통화제어 서브 시스템으로 구분하여 생각할 수 있다. 전자는 사용자와의 상호작용을 위한 단말 장치, 교환기, 그리고 이들간을 연결하는 선로와 중계 시설 등으로 구성되며, 후자는 정보전달 서브 시스템을 제어하는 서브 시스템으로 네트워크로 연결된 컴퓨팅 노드들로 구성된다.

이들간의 상호작용은 보편적인 제어시스템의 경우와 마찬가지로 (1)외부 사건 입력, (2)처리, 그리고 (3)처리 결과에 따른 제어로 이루어진다. 외부 사건 입력은 정보 전달 서브 시스템에서 발생된 사건이 통화



(그림 1) 전형적인 정보통신 시스템 구조
(Fig. 1) Structure of a conventional telecommunication system

제어 서브 시스템으로 입력되는 것을 말한다. 통화 제어 서브 시스템에서는 이 사건에 대한 처리를 수행하고 처리 결과를 정보 전달 서브 시스템으로 반송하여 제어한다. 본 고에서 고려된 정보통신 시스템은 다음의 특성을 갖는다.

(1) 연속적인 운용 상태 유지

시스템 내부에 일부 결함이 발생되어도 결함이 발생된 부분과 관련되지 않은 시스템의 나머지 부분은 계속적으로 작업을 수행한다. 시스템 내에 제공되는 여러 응용 서비스 중 일부 서비스의 시험이나 교체를 위하여 전체 시스템의 운용을 중단할 수 없다.

(2) 시간에 따른 제약

시스템 외부로부터 요구되는 사건의 입력은 많은 경우에 시간적 제약을 갖는다. 즉, 일정시간 내에 처리되지 않는 작업은 오류로 처리된다.

(3) 비동기적 프로세스들 간의 상호작용

시스템 내에는 여러 개의 프로세스들이 비동기적으로 진행된다.

(4) 불확정적이고 재현 불가능한 결과

처리가 요구되는 신호의 전송 지연 시간이 일정하지 않아 같은 입력 조건에 대해서도 여러 사건이 일어난 결과가 바뀌어질 수 있다. 어떤 객체들은 여러 서비스 인스턴스들에 의하여 불확정적인 순서로 반복적으로 사용될 수 있다.

(5) 글로발 클록 유지의 어려움

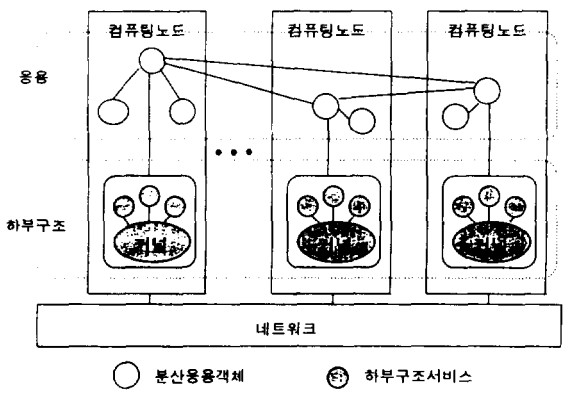
시스템 전체를 정확히 동기화할 수 있는 클록의 유지가 어려워 많은 사건들이 동시에 일어나는 경우 이들에 대한 해독이 어렵다.

3. 분산처리 환경

3.1 분산처리 하부구조[1, 2, 3, 4]

2절에서 모델링된 하부구조인 분산처리 환경의 주요 기능은 다음과 같다. 응용을 이루는 논리적 구성요소들은 연산객체로 간주되며, 이들 객체들은 메시지를 교환하여 상호 통신한다.

연산 객체로 구성되는 응용들은 객체의 배치, 수행, 및 상호작용을 지원하는 하부구조에 그 수행을 의존한다. 이 하부구조를 제공하는 각 컴퓨팅 노드들은 연산장치, 메모리, 통신을 위한 자원들로 구성되는데, 이러한 연산 자원들을 다른 노드와 독립적으로 관리하는 기능은 커널이라는 하나의 객체로 이루어진다. 커널은 또한 연산객체 인스턴스의 수행과 이들 간의 통신을 지원하는 메카니즘을 갖고 있다. 개방형 시스템의 관점에서 볼 때 컴퓨팅 노드에는 최소한 하나의 네트워크 접근 메카니즘이 있다.



(그림 2) 분산처리 하부구조
(Fig. 2) Infrastructure for distributed processing

하부구조의 속성은 응용에게는 그들이 서로 다른 컴퓨팅노드에서 수행되고 있다는 사실을 숨긴 채, 객체들이 배치되어 있는 노드에 무관하게 객체의 수행과 상호 작용을 가능하게 한다. 즉, 하부구조를 지원

하는 어떠한 노드에서도 객체설계의 재사용을 보장하는 이식성과 한 객체가 다른 컴퓨팅 노드에 위치한 다른 객체의 동작을 유발시킬 수 있는 상호 운용성을 제공하며, 접근, 위치, 이주, 트랜잭션, 복제, 장애 등의 다양한 분산투명성을 제공할 수 있다. 이러한 투명성은 분산 프로그램 작성시 복잡성을 감소시키기 위한 것으로서 특수한 응용, 예를 들면, 실시간, 고가용도 특성이 요구되는 경우는 몇 가지의 투명성을 선택적으로 적용해야 한다. 응용 객체에 공통으로 제공되는 하부구조의 서비스들로는 트레이딩 서비스, 통보 서비스, 트랜잭션 서비스, 저장 서비스, 형상 서비스, 보안 서비스 등이 고려되고 있다. 이 서비스들은 객체 상호작용 메카니즘을 지원하기 위한 기능이거나 하부구조 자체의 기능으로 간주되기도 하며, 상호 연관되어 동작한다. 5절에서는 이상에서 언급된 서비스들 외에 객체 외부 상태를 감시하고 효율적인 시스템 시험을 지원하기 위한 새로운 하부구조 공동 서비스로 감시 서비스를 제안하고 그 기능을 정의한다.

4. 감시 접근 방법들

이 절에서는 시스템 시험 및 디버깅과 관련된 기존의 감시 방법들을 알아보고 2절에 언급된 정보통신 시스템에의 적용 한계에 대하여 논한다. 기존의 감시 방법들은 감시 활동의 주체의 위치에 따라 다음과 같이 나눌 수 있다[6, 11, 12].

(1) 프로그램의 변수 값이나 특정 주소의 메모리 상태를 저장하기 위한 코드를 감시대상 프로그램 내에 삽입하는 방법

(2) (1)에서와 같은 감시 목적의 코드를 시스템 커널에 삽입하여 감시대상 프로그램 수행과 분리시키는 방법

(3) 감시 대상 프로그램의 수행을 감시하는 별도의 감시 프로그램(또는 감시자)을 수행시키는 방법

또한 감시활동은 사건발생감지와 이에관한 데이타 수집의 두 가지 단계로 나누어지는데, 이를 위의 분류에 적용하여 가장 일반적으로 사용될 수 있는 감시 기법으로 다시 분류하면 다음과 같다.

(1) 감시대상 프로그램이 사건 발생 감지와 이에 관한 데이타 수집을 위한 작업을 모두 수행하는 방법

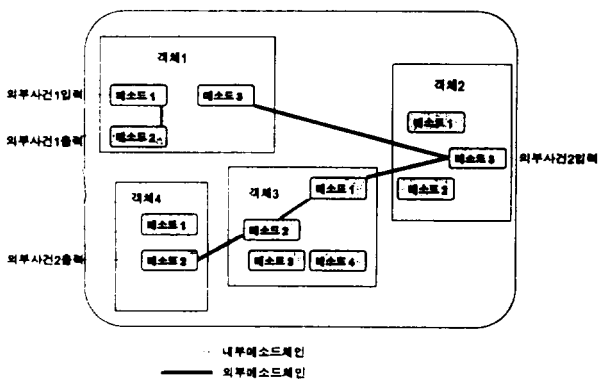
(2) 감시대상 프로그램이 사건 발생을 감지하고 감시자가 사건 데이타를 수집하는 방법

(3) 커널이 사건 발생을 감지하고 감시자가 사건 데이타를 수집하는 방법

(1)의 방법은 감시 대상 프로그램 내에 감시 코드들을 삽입시킴으로써 매우 간단하게 이루어질 수 있으나, 감시대상 프로그램 수행 시간에 직접적인 방해효과를 미친다. 즉, 감시 활동을 위해 추가로 삽입된 코드들로 인하여 사건 발생 순서나 시간이 바뀌어질 수 있다. 이와 같은 형태의 감시를 방해감시(intrusive monitoring)라 하는데 이와 같은 방법은 일반적으로 시간 제약이 엄격한 응용에는 적합하지 않다. 감시에 따르는 방해 요인으로는 감시코드 수행으로 인한 수행시간 증가, 통신량의 증가, 메모리 접근시간 및 메모리 공간의 증가, 감시자와 감시대상 프로그램간의 통신 발생 등이 있다. (2)의 방법은 감시 대상 프로그램 내에 감시 코드를 삽입시켜서 프로그램 수행 중 수행 상태를 시스템 내의 일정 장소에 저장하게 하고 프로그램 수행 후 감시자 또는 시험자로 하여금 이 데이타를 분석하여 프로그램 수행 상태를 판단하게 하는 방법이다. 전통적으로 프로그램 디버깅에 많이 사용되어 온 방법이다. 이 방법이 개발 단계에서는 효과적일 수 있으나, 시스템의 연속적인 운용 상태를 유지하기 위해서 사용자와의 상호작용(브레이크 포인트 이용 등)이 허용되지 않는 정보통신 시스템의 경우는 곤란하다. 또한 시스템 운용 단계에서 발생가능한 오류를 검출하고 복구하기 위한 목적으로는 사용될 수 없다. (3)의 방법은 감시대상 프로그램 내에는 최소한의 자체시험(self test)을 위한 코드 만을 두고 감시대상 프로그램에게 투명하게 감시를 수행하여 (1)에서와 같은 방해 효과를 최소화하는 방법이다[7]. 대부분의 정보통신 서비스의 경우, 시험을 목적으로 하는 프로그램의 일부 내용을 추가하거나 변경하는 것이 곤란하다. 즉, 소스 프로그램의 직접 사용이 어렵다. 또한, 특정 서비스 인스턴스 만을 위한 시험이 가능해야 한다. 서비스 인스턴스란 서비스를 제공하기 위한 특정 서비스 처리이다. 예를 들면 멀티미디어 화상 회의 서비스의 경우 특정 대화자 간의 특정 미디어간 서비스를 시험할 수 있어야 한다. 따라서, 소스 프로그램에 대한 수정없이 하부구조 상에서 관찰되는 정보 만으로 객체의 수행 상태를 시험할 수 있어야 한다. 결과적으로 정보통신 서비스의 시험을

위해서는 위에서 언급한 세가지 감시 방법중 (3)만이 유력한 접근방법이다.

본 고에서는 하부구조 상에서 객체의 수행 상태를 감시하기 위한 방안으로 [5]에서 정의된 객체(메소드) 간 메시지 흐름(message flow)의 모델을 이용한다. 객체 내 메소드는 메시지 통신에 의해서 상호작용하는데, 한 메소드에서 발생된 메시지와 이것에 연결된 메소드의 연속된 고리를 메소드 체인이라고 한다. 즉, 일련의 연속된 수행에 관련된 메소드와 메시지의 집합이다. 하나의 메소드 체인은 한 메소드에서 시작하여 한 개 또는 그 이상의 다른 메소드로 연결될 수 있으며 더 이상 메시지를 발생시키지 않는 메소드에 이르게 되면 멈춘다. 이 메소드를 그 메소드 체인의 종점이라 하고 메소드 체인이 시작된 메소드를 시발점이라 한다. 같은 객체 내의 메소드 간 연결은 내부 메소드 체인이라 하고 다른 객체의 메소드 간 연결은 외부 메소드 체인이라 한다. 한 객체 내의 내부 메소드 체인은 하부구조에 의해 감지되지 않는다. (그림3)는 객체들간 메시지 흐름의 예를 나타낸다. 각 객체에는 고유 번호가 부여되어 있고, 객체들은 같은 컴퓨팅 노드에 또는 서로 다른 컴퓨팅 노드에 배치될 수 있다. 객체간의 통신은 메시지를 통해서만 이루어진다. 메시지는 목적지 객체와 객체 내 해당 메소드의 이름과 메소드 입출력 변수 등을 포함한다. 객체들을 2절에서 설명된 하부구조에 의하여 관리, 제어된다. 하부구조는 새로 생성된 객체에게 고유 번호를 부여하고, 객체 수행을 스케줄링하며, 객체들간의 메시지 통신을 지원한다. 하부구조는 원래 요청된 메시지에 제어 정보를 부가하여 얻어진 확장메시지를 목적지 객체에 전송한다. 하부구조에 의해 부가되는 제어 정보에 대하여는 다음 절에서 논하기로 한다.



(그림 3) 객체간의 메시지 흐름
(Fig. 3) Message flows between objects

5. 감시 서버

이 절에서는 하부구조에 포함되는 감시 서버의 기능을 새로이 정의한다. 이 서버는 3절에 요약된 서비스들을 제공하는 다른 서버들과 마찬가지로 자체도 객체로 간주되며 커널의 지원을 받는다. 감시 서버의 기능은 다음과 같다.

(1) 감시제어정보 검사 및 수정

확장메시지 내에는 그 메시지의 감시 상태를 제어하는 정보가 들어 있다. 감시 서버는 이 정보를 검사하여 어떤 외부 메소드 체인이 추적되어야 하는 가를 확인할 수 있다. 또한 이 제어 정보는 감시 서버에 의해 수정될 수 있다.

(2) 객체의 외부 수행 감시 및 기록

메시지 전달이나 객체 스케쥴링과 같은 외부 수행을 감시한다. 감시 중인 메소드 체인의 경우, 각 메소드 수행 처리 시간과 입출력 값의 범위를 확인하고 이들 데이타를 저장한다. 위급한 상태가 아닌 경우, 저장된 데이타를 일정량 모아서 시험 노드에 전달한다.

(3) 시험 프로시져 유지

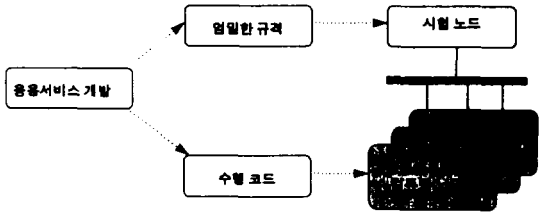
추가되는 비용으로 인하여 모든 외부 객체 수행을 감시할 수는 없다. 감시해야 할 메소드 체인의 유형과 감시해야 할 시간을 알고 있어야 한다. 이러한 감시 조건을 명시한 것을 시험 프로시져라 하는데 이를 저장 유지하여야 한다.

(4) 위급 및 비상 사태 처리

해당 노드 내의 일부가 정상적으로 동작하지 않으나 대체 가능한 요소가 있는 경우는 신속히 조치한다. 이것이 불가능하여 전체 시스템의 정상 운용에 영향을 미치게 될 비상 사태의 경우, 즉시 시험 노드에 이를 통보한다.

하나의 특정 서비스 인스턴스에 대한 메소드 체인 감시를 위하여 감시 서버는 이 상태를 유지 관리한다. 각 메소드 체인과 객체는 감시 상태 이거나 비 감시 상태이다. 어떤 사건으로 인하여 한 객체의 메소

드가 감시 상태로 들어가게 되면, 여기서 시작되는 감시 상태의 메소드 체인에 고유 번호가 부여되고 이 고유 번호는 확장 메시지 내에 실린다. 하부구조에서는 객체의 외부 상태 만이 감지되므로 객체 내의 메소드들 간의 상호작용은 정확히 알기 어렵다. 따라서, 감지 상태에 있는 메소드를 포함하는 객체는 감시 상태에 있는 것으로 간주된다. 감시 상태의 객체로부터 나온 메시지들은 감시 상태로 된다. 메시지를 전달하는 동안 감시 서버는 각 메시지의 감시 상태와 메소드 체인 고유 번호를 검사한다. 그 메시지가 감지 상태에 있으면 감시 서버는 감시 데이타를 메소드 체인 고유 번호와 함께 기록하고 이를 운용 시스템에 전달한다. 감시 상태의 객체는 후에 비감시 상태의 메시지를 받으면 비 감시 상태로 바뀐다. 각 객체의 오퍼레이션이 감시 서버에 의해 감시 기록되는 경우는 암시적 전달과 직접적 전달의 경우가 있다. 전자는 한 객체가 감시 상태의 메소드 체인을 받아 감시 상태로 된 경우이며 후자는 메소드체인과 무관하게 한 객체로부터 직접 감시 서버에 감시 요청 메시지가 전달될 경우인데, 이는 객체 내의 예외 사건 발생 등으로 인하여 그 객체가 감시 서버에게 위급 사태 처리를 요청한 때이다.



(그림 4) 시스템 차원의 시험을 위한 체계
(Fig. 4) Inputs for system-level test

6. 시스템 시험

이 절에서는 5절에서 정의된 하부구조의 감시 기능을 이용한 시스템 차원의 시험 방법에 대하여 논한다. 통화제어 서브시스템 내에는 일반 컴퓨팅 노드와 네트워크로 연결된 시험 노드가 있다. 이는 일반 통신시스템 관리를 위한 운용관리 시스템 노드를 겸할 수 있다.

6.1 시험노드의 주요 기능

시험 노드의 주요 기능은 다음과 같다.

(1) 응용 서비스 제공자로부터 제시된 응용 프로그램의 엄밀한 규격을 바탕으로 유효한 입력 조건과 유효한 사건 발생 순서 등을 확인한다.

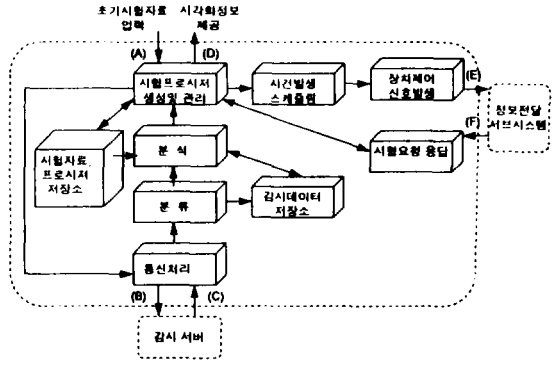
(2) 감시해야 할 객체와 메소드 체인의 유형과 감시해야 할 시간 등의 감시 조건을 명시한 시험 프로시져를 생성하고 관리한다.

(3) 각 컴퓨팅 노드의 감시 서버와의 메시지 통신을 통하여 시험 프로시져를 제공하고 감시 서버들에 의해 수집된 감시 데이타를 저장 분석한다.

(4) 분석 결과를 시각화하고 시험 프로시져를 수정, 보완한다.

(5) 자동 시험을 위하여 시험 사건을 발생시키고 동시에 시험 프로시져를 진행시킨다.

감시 데이타를 분석하여 4절에 정의된 메소드체인을 확인하여 고유 번호를 붙여서 관리하고 분석 결과를 시각화하는 데 이용한다. (그림 5)은 시험노드의 기능 구조와 외부와의 상관관계를 나타낸다.

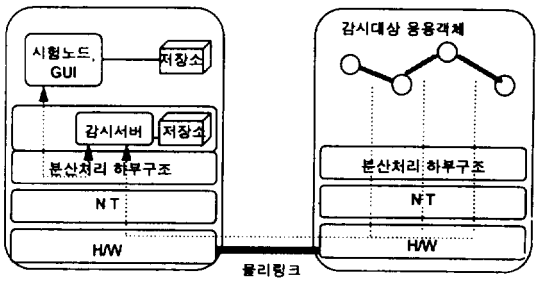


(그림 5) 시험노드의 기능 구조 및 상관관계
(Fig. 5) Functional structure of test node and relationship

6.2 시스템 시험 사례

감시 서버와 시험 노드의 기능들이 시제품 제작 형태로 구현되었다. 구현 환경으로는 Pentium PC에 WindowsNT를 설치하고 분산처리를 지원하기 위하여 CORBA(Common Object Broker)버젼 1.1의 IDL (Interface Definition Language)를 사용하고 객체지향적 구현과 GUI를 위해 C++언어를 사용하였다. 시

스팀전체구성은 (그림 6)과 같다. 여기에서 보는 것과 같이 감시대상인 분산객체들에 대한 감시 서버의 감시 및 감시 서버와 시험 노드 간의 감시정보 교환은 분산처리 하부구조를 통하여 이루어지며, 시험 노드와 감시 서버가 관리하는 저장데이터는 사용자 인터페이스인 GUI를 통하여 레코드 별로 추가, 삭제, 수정하게 되어있다. 한 노드에 구현되어 있는 시험노드와 감시 서버는 추후 다른 노드로 분리될 것이며 감시 서버의 기능은 분산처리 하부구조의 일부 기능으로 보완될 것이다.



(그림 6) 프로토타입 시스템 구성
(Fig. 6) Configuration of a prototype system

시험 프로시져는 시험 사건의 종류와 이를 보내는 순서를 정의한다. 순차 프로그램을 위한 일반 시험 소프트웨어는 시험자에 의해 제어되는 프로시져로 충분하다. 그러나, 자치적으로 수행되는 다수의 분산 객체의 경우 시험자는 여러 시험 사건들을 보내서 이러한 객체들을 제어해야 한다. 어떤 시험 사건들은 순차적으로 또 다른 것들은 동시에 비동기적으로 보내져야 한다. 시험자의 운용 오류를 방지하기 위하여 시험은 자동으로 수행될 필요가 있다.

개발단계와 운용단계에서의 각 시험과정을 간단한 호 연결 요청에 대한 시험을 예로 들면 다음과 같다.

(1) 개발단계의 시험

- 초기상태 : 시험자로부터 경로(A)를 통하여 초기 시험 자료가 입력, 저장된다. 초기시험 프로시져가 생성된다. 생성된 시험 프로시져는 경로(B)를 통하여 해당 감시 서버에 전달된다. 시험 프로시져는 어떤 호의 연결 요청에 대한 결과를 감시할 것인지에 관한 조건이 명시되어 있다.

- 시험상태 : 경로(E)를 통하여 시험하고자 하는 호의 연결 요청을 제어한다. 경로(C)를 통하여 입력된 감시데이타는 서비스 별로 분류되어 감시데이타 저장소에 저장된다. 이들 중 호 연결 요청과 관련된 자료를 분석한다. 다른 시험을 위하여 일련의 시험 프로시져들을 생성 저장한다. 경로(E)를 통하여 사건 발생을 제어함으로써 오류가 발생된 부분의 시험을 반복, 정확한 위치를 찾을 수 있다.

(2) 운용 단계의 시험

- 일상적인 감시상태 : 일상적인 감시데이타는 서비스 별로 분류되어 감시데이타 저장소에 저장된다.

- 비상사태 발생 : 비상사태 처리를 위하여 각 감시 서버에서 전달되는 데이타에는 우선순위가 적용된다. 특정한 호의 연결 장애를 유발시키는 오류가 검출되면 이 오류 상태의 호가산을 막기 위하여 그 부분을 단절시키고 복구를 위한 조치를 취한다.

- 특정 서비스 인스턴스 시험 : 경로(F)를 통하거나 시스템 운용자의 선택(경로(A))에 의해 특정 호에 대한 시험이 이루어진다. 감시해야 할 메소드 체인의 유형과 감시해야 할 시간 등이 정해진 시험 프로시져가 생성되고 이것이 경로(B)를 통하여 해당 감시 서버에 전달된다. 동시에, 시험하고자 하는 사건이 발생되도록 준비한다. 경로(E)를 통하여 해당 장치를 제어하여 정해진 감시 시간에 시험하고자 하는 사건이 발생되도록 한다. 이 결과는 경로(C)를 통하여 입력된다. 상태를 분석하여 그 결과를 경로(D)를 통하여 시험자 또는 운용자에게 알린다.

7. 결 론

본 고에서는 감시 서버의 기능을 정의하고 이를 토대로 한 시스템 차원의 시험 방법에 대하여 논하였다. 별도의 응용 객체가 아닌 커널에 가까운 하부구조 내에 위치하는 감시 서버로 하여금 노드 내의 객체 수행 상태를 감지하고 감지된 데이타를 위급 상태가 아닌 경우 모아서 독립된 시험 노드에 보냄으로써 감시 활동으로 인한 방해효과를 최소화시킬 수 있다. 이러한 하부구조 내의 감시 서버의 역할은 필수적이다. 시스템 시험자는 여러 시험 사건들을 보내서 해당 객체들의 수행 상태를 감지해야 하는데 어떤 시험 사건들은 동시에 비동기적으로 보내져야 한다. 본 논

문에서 제시된 시험 방법은 이를 자동으로 처리하여 시험자의 운용 오류를 방지함은 물론 사건 발생을 제어, 시험을 반복함으로써 오류가 발생된 부분의 위치를 정확히 찾아낼 수 있도록 해준다. 그러나 실제 시스템에 적용되기 위해서는 이 시험방법에 대한 평가 방법이 앞으로 더 보완되어야 하고, 이 방법을 뒷받침해주고 있는 하부구조를 정착시키기 위한 연구가 계속되어야 할 것이다. 이러한 기술은 실시간 결함허용 시스템 구축을 위한 핵심적인 기술로, 정보통신 시스템 뿐만 아니라 일반 분산시스템에 적용가능할 것으로 생각된다.

참 고 문 헌

[1] W. J. Barr, T.Boyd, and Y.Inoue, "The TINA Initiative," IEEE Communications, March 1993.

[2] ISO 10746-2 RM-ODP Part 2, "Descriptive Model," Part 3, "Prescriptive Model," June 1993.

[3] H.Tokuda, M.Kotera, and C.W. Mercer, "A Real-time Monitor for a Distributed Real-time Operating System," Proc. ACM Workshop Parallel and Distributed Debugging, ACM Press, pp. 68-77, 1988.

[4] ISO 10165-7, "Information Technology-Open Systems Interconnection-Structure of Management Information," Jan. 1993.

[5] P.C.Jorgensen and C.Erickson, "Objected-Oriented Integration Testing," Comm. of the ACM, Vol.37, No.9, pp.30-38, Sept. 1994.

[6] J.Joyce, G.Lomow, K.Slind, and B.Unger, "Monitoring Distributed Systems," ACM Trans. Computer Systems, Vol.5, No.2, pp.121-150, May 1987.

[7] M.H.Kim and S.M.Yang, "Environment for Development of Reliable Real-Time Distirbuted Applications," IEEE Workshop on Future Trends of Distributed Computing Systems, pp.120-126, Aug. 1995.

[8] M.Kuboat and K.Maruyama, "Testbed for Distributed Object-Oriented Telecommunication Service Software," JCCNSS, pp.210-215, 1994.

[9] C.E. McDowell and D.P. Helmbold, "Debugging Concurrent Programs," ACM Computing Surveys, Vol.21, No.4, pp.593-622, Dec. 1989.

[10] G.C.Murphy, P. Townsend, and P.S. Wong, "Experiences with Cluster and Class Testing," Comm. of the ACM, Vol.37, No.9, pp.39-47, Sept. 1994.

[11] N. Natarajan and G.M. Slawsky, "A Framework Architecture for Multimedia Information Networks," IEEE Communications, Feb. 1992.

[12] J.J.P Tsai and S.J.H. Yang, 'Monitoring and Debugging of Distributed Real-Time Systems,' IEEE Computer Society Tutorial, pp.2-17, 1995.

이 한 영

1980년 연세대학교 전자공학(학사)

1984년 연세대학교 전자공학(석사)

1985년~현재 한국통신 통신망연구소 선임연구원

관심분야:B-ISDN구조, 통신망서비스관리, 실시간분산, 무방해감시, 객체지향

민 병 준

1983년 연세대학교 전자공학과 졸업(학사)

1985년 연세대학교 대학원 전자공학과 졸업(석사)

1991년 미국 캘리포니아주립대(UC Irvine) 전기컴퓨터공학과(박사)

1984년~1986년 삼성전자 종합연구소

1992년~1993년 한국통신 연구개발원

1994년 감사원 전산담당관실

1995년~현재 시립인천대학교 전자계산학과 재직중

관심분야:분산실시간 시스템, 결함허용, 통신망관리

김 문 희

1979년 2월 서울대학교 전기공학과 학사
1981년 2월 서울대학교 전기공학과 석사
1985년 5월 University of South Florida, MSCS
1991년 5월 University of California, Berkeley, Ph.D
1991년 3월~현재 건국대학교 컴퓨터공학과 부교수
관심분야: 소프트웨어공학, 운영체제, 실시간 분산처리시스템, 통신망관리

서 동 선

1980년 연세대학교 전자공학(학사)
1985년 연세대학교 전자공학(석사)
1989년 미국 뉴멕시코대학 전기공학(박사)
1990년~현재 명지대학교 전자공학과 교수
관심분야: 초고속정보통신기술, 객체지향 등

허 웅

1973년 인하대학교 전자공학(학사)
1975년 인하대학교 전자공학(석사)
1980년 인하대학교 전자공학(박사)
1980년~현재 명지대학교 전자공학과 교수
관심분야: 초고속정보통신망, 객체지향 등