고가용성 홈 서비스 네트워크 시스템을 위한 오토노믹 자가 관리 메커니즘

(An Autonomic Self-management Mechanism for High-available Home Service Networked System)

최 창 열 † 김 성 수 ††
(Changyeol Choi) (Sungsoo Kim)

요 약 홈 서비스 네트워크 시스템은 실용적이면서 프로엑티브한 결함 관리를 통한 고가용도 서비스를 필요로 한다. 하지만, 홈 서비스 시스템의 복잡도가 증가하면서 고가용도 요구사항을 만족하기 위한 서비스를 제공하는 것은 더욱 어려워졌다. 더욱이, 사용자는 시스템 결함 관리와 같은 복잡한 시스템 관리를 직접 하는걸 원하지 않는다. 그러므로 홈 네트워크로 연결된 장치 또는 가전제품으로 구성된 통합 시스템의 고가용도 요구사항을 만족하면서 사람의 개입을 최소화할 수 있는 자가 관리 기능 및 원격 결함 관리 능력을 홈 서비스 시스템은 갖춰야 한다. 따라서 본 논문에서는 홈 서비스 네트워크 시스템의 가용도를 향상시키면서 관리 비용을 최소화할 수 있는 원격 자가 관리 메커니즘을 탑재한 오토노믹 자가치유 유틸리티를 설계 및 개발한다.

키워드 : 오토노믹 자가치유, 자가관리, 고가용도, 홈 서비스 네트워크

Abstract Home service networked systems require a high-availability service with a proactive and practical fault management. However, as the system complexity grows, it is not easy to meet the requirement. Moreover, user may want to pay no attention to a sequence of complex or nervous maintenance jobs for system fault managements. Therefore, the home service networked systems must have self & remote fault management capability with a minimal human intervention for meeting high-availability requirement of the integrated systems that consist of the networked appliances or devices. In this paper, we present an autonomic healing utility equipped with a remote self-managing mechanism in order to both increase the availability of home service networked systems and decrease the maintenance cost.

Key words : Autonomic healing, Self-management, High-availability, Home Service Network

1. 서 론

홈 서비스 네트워크 시스템은 홈 환경을 구성하고 있는 모든 사용 가능한 객체로 구성된 시스템을 의미하므로 단일 컴퓨터로 구성될 수도 있으며 계산 능력을 보유한 모든 장치가 연결된 네트워크로 구성된 요소들의 집합이기도 하다[1-3]. 따라서 홈 서비스 네트워크 시스템은 정보 시스템, 엔터테인먼트 시스템, 그리고 주요 홈 가전제품을 접근하기 쉽고 제어하기 쉬울 뿐만 아니라 사용하기 쉽게 하기 위하여 다수의 장치를 하나의 단일 시스템처럼 통합 관리하기 위한 시스템이라 말할 수 있다(그림 1 참조).

하지만 홈 서비스 네트워크 시스템과 같은 통합 관리 시스템의 사용이 급증하면서 제공해야 되는 서비스의 수는 기하급수적으로 증가하고 있으며 이로 인해 관련 응용서비스/장치/시스템을 관리하기 위한 비용 또한 함께 증가하고 있다. 다시 말해서, 홈 서비스 네트워크 시스템을 보유하기 위한 비용(cost of ownership)의 약 80%가 시스템 운영, 유지 및 작은 기능 향상 등과 같은 관리를 위한 비용으로 소비된다[4,5]. 심지어는 전체 비용의 33%에서 50% 정도의 비용이 시스템에서 발생한 예기치 못한 결함에 대비하고 이를 복구하는데 소요된

· 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크 원천기반기술개발 사업의 지원에 의한 것임
† 정 회 원 : 숭실대학교 정보미디어기술연구소 연구원
cychoi@otlab.ssu.ac.kr
†† 종신회원 : 아주대학교 정보통신전문대학원 교수
sskim@ajou.ac.kr
논문접수 : 2006년 3월 22일
심사완료 : 2007년 3월 30일

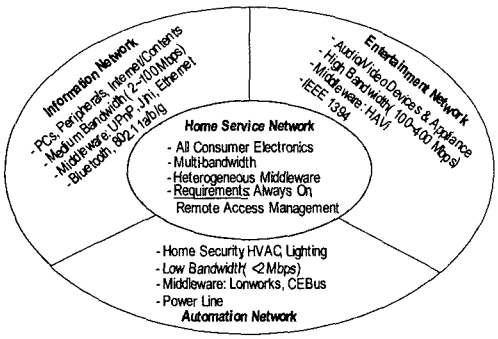


그림 1 홈 서비스 네트워크 시스템의 3대 구성요소

다. 여기서 주목할 만한 점은 시스템의 예기치 못한 결함 발생 주원인이 하드웨어적인 문제에서 점차 소프트웨어적인 문제로 변하였으며, 또한 시스템을 구축 및 유지함에 있어서 발생하는 문제보다는 운영 중 발생한 잘못으로 인한 문제로 변하였다는 것이다[6]. 따라서, 홈 서비스 네트워크 시스템의 운영 중에 발생하는 소프트웨어 결함을 효율적으로 관리할 수 있는 솔루션이 필요하며, 이는 고가용도 서비스를 제공하여 시스템 정지 시간이 거의 제로에 가깝도록 하기 위한 필수요소라는 것이다. 하지만, 시스템 정지 시간을 제로에 가깝게 하는 것은 유지보수를 위한 막대한 비용이 필요로 할 뿐만 아니라 홈 서비스 네트워크 시스템의 사용자가 결함 처리 및 복구에 대한 지식을 습득해야만 된다는 것을 내포한다. 그러나 시스템 관리에 대한 전문적인 지식이 없는 보통 사용자들은 시스템 결함 관리와 같은 복잡한 일련의 작업에 포함되지 않길 바라며, 심지어 결함이 발생하였어도 자신이 인지하지 못한 사이에 처리되기를 바란다. 이는 사람(사용자 및 전문가)의 간섭 없이 시스템 스스로 결함에 일정 부분 대응할 수 있는 자가 관리(self-management) 기능을 홈 서비스 네트워크 시스템이 보유해야만 한다는 것을 의미한다. 따라서 본 논문에서는 홈 서비스 네트워크 시스템의 원격 자가 관리 메커니즘을 설계한 후 이를 실제 적용 가능한 형태인 오토노믹 자가치유 유틸리티(autonomic healing utility, AHU)로 개발하여 프로엑티브한 소프트웨어 결함 관리 솔루션을 설계 및 구현하고자 한다. 이를 위해 2장에서는 소프트웨어 결함을 예방하기 위한 메커니즘에 대한 관련 연구에 대해 소개하고 3장에서는 오토노믹 자가치유 유틸리티의 구조 및 특징에 대해 설명한다. 그리고 4장에서는 제안한 오토노믹 자가치유 유틸리티에 탑재한 원격 자가 결함 관리 메커니즘에 대해 상세히 기술하고 5장에서는 실제 구현 사례를 통해 이의 실효성을 검증한다. 마지막으로 6장에서는 개발한 프로토타입 구현 결과를 정리하고 향후 연구주제에 대해 살펴본다.

2. Background

시스템 결함은 하드웨어에 의한 결함과 소프트웨어에 의한 결함으로 구분할 수 있는데, 주로 예기치 못한 결함은 소프트웨어에 의한 것이 대부분을 차지하며, 대표적인 결함 원인으로 소프트웨어 노화현상을 들 수 있다[7,8]. 소프트웨어 노화현상이란 소프트웨어에 잠재적으로 존재하는 것으로 메모리 유출(memory leakage), 버퍼 오버플로우, 데이타 원형 손상, 수학적 에러 누적 등으로 인해 데이타의 유실, 통신 장애 및 서비스 불가능을 야기시키는 현상이다. 이에 대한 대안으로 결함 예방(fault prevention) 기술인 소프트웨어 재활 기법이 최근 연구되고 있는데, 이는 심각한 시스템 오류를 발생시킬 수 있는 상황을 분석 및 예측하여 사전에 시스템 내부 상태를 깨끗한 초기 상태와 같은 상태로 전환시켜 지속적인 서비스가 가능하도록 하는 일련의 과정을 포함하는 기술이다[9-11]. 여기서 내부 상태를 깨끗이 하는 과정이란 쓸데없는 자료 수집(garbage collection), 운영체제 커널 테이블의 재정리, 내부 자료 구조의 초기화 등이다. 하지만 소프트웨어 재활 기술의 최근 관련 연구들은 기술 적용에 따른 시스템의 가용도 변화에 대한 분석, 또는 결함에 따른 손실 비용 분석, 단일 시스템과 클러스터와 같은 동종 서비스를 제공하는 분산 시스템에 적용 가능성을 타진하는 정도로만 연구가 수행되었을 뿐 이질적인 네트워크 환경인 홈 서비스 네트워크 시스템에 접목하기 위한 장치 관리 모듈의 구조 도출 및 관리 기능을 적용함에 있어 사람의 간섭을 최소화하기 위한 연구는 수행되지 못했다. 이로 인해 시스템을 관리하는데 있어 사람의 간섭을 최소화해야 되는 유비쿼터스 컴퓨팅 환경에 기존 연구들을 직접 접목하는 것은 부적절하므로 이를 해결하기 위한 오토노믹 컴퓨팅(autonomic computing) 관련 기술 개발이 필요하다. 오토노믹 컴퓨팅 기술이란 기존에 개발된 시스템 관리 기술을 자동화하여 시스템 관리 비용을 줄이기 위한 기술로써, 기본 개발 철학은 인간의 신체가 체온이 저하되면 열의 방출을 억제하기 위해 근육을 수축하고, 어두워 보이지 않을 경우 동공을 확장하는 것과 같이 뇌의 활동에 의한 명백한 사고 없이 몸을 관리하는 자율신경계의 동작 원리를 컴퓨터 시스템 관리에 적용하자는 것이다. 본 연구의 사전연구로 수행된 [12,13]에서는 범용 시스템을 대상으로 오토노믹 컴퓨팅 관련 기술과 결함 관리 기술을 접목하기 위한 시도를 하였지만, 이는 설계 방법론 및 분석 모델 정립에 중점을 두었을 뿐 홈 서비스 네트워크 시스템과 같은 실제 적용 분야에 적용하기 위한 구체적인 구조 및 메커니즘 개발에 대한 연구가 부족하다. 따라서 본 논문에서는 [12,13]의 개발 방법론을 기반으로 홈 서비스 네트워크 시스템에서 발생 가능

한 소프트웨어적인 결함을 처리하기 위한 메커니즘을 개발하고 이를 구현하여 실효성을 검증한다.

3. 오토노믹 자가치유 유틸리티의 구조

3.1 오토노믹 자가치유 유틸리티의 설계 목표 및 특징

오토노믹 자가치유 유틸리티(이하 AHU)는 홈 서비스 네트워크 시스템에 결함이 발생하기 이전에 결함 발생 요인을 관리하고 결함이 발생할 가능성이 있는 응용 서비스를 검출하기 위한 결함 예방 서비스와 더불어 프로엑티브한 결함 복구 서비스를 자동화한 것이다. 이를 위해 AHU는 홈 서비스 네트워크 시스템 운영에 영향을 미치지 않도록 독립적으로 동작할 수 있는 Stand-alone 형태로 개발되었다. 또한 응용 서비스 관리자는 시스템의 자원, 구성 컴포넌트 및 응용 서비스를 직접 제어할 수 있는 기능을 원하기 때문에 AHU는 이를 지원한다. 더욱이 AHU에 포함된 결함 정보 수집기(fault collector)는 시스템의 컴포넌트 계층 및 응용 서비스 계층으로부터 결함 발생 원인을 분석하고 관련 정보를 수집하여 이후 해당 결함 발생을 사전에 방지할 수 있도록 결함 관련 지식 기반 자가 성장(self-growing) 기능을 지원한다. 이와 같은 AHU 목표 기능을 달성하기 위해 본 논문에서 채택한 디자인 설계 접근 방식을 정리하면 표 1과 같다.

AHU에 탑재된 고가용성 서비스 관리 메커니즘은 오토노믹 컴퓨팅 기술[14]과 오토노믹 설계/개발 방법론[12,13]을 기반으로 설계 및 개발되었다. 따라서 홈 서비스 네트워크 시스템이 스스로 결함을 관리할 수 있는 능력을 보유하여 예기치 못한 결함이 발생하였을 때 '어떤 결함이 어디서 발생하였으며 이를 어떻게 해결해야 되는지'와 같은 복잡하고 전문 지식을 습득해야 되는 사용자의 부담을 제거했다.

3.2 AHU의 주요 기능

AHU의 주요 기능은 그림 2에서 보듯 크게 3가지로 분류할 수 있는데, 시스템 스스로 자신의 상태를 관리할 수 있는 구조를 기반으로 시스템 스스로 자신의 상태를 검진(self-checkup)하고 이에 필요한 적절한 대응을 수행(self-adaptation)한 후 그 결과에 대한 예후 관리(self-prognostic analysis)까지 가능하게 하는 일련의 관련 기술이 피드백 루프 구조를 통해 자동화(self-directed automation)된 유틸리티이다[15].

따라서 가용한 시스템 및 자원의 성능을 최적화하고 서비스 계약 수준을 달성하기 위한 동적 재구성이 가능하며, 사용자가 신뢰할 수 있는 범위 내에서 끊임없는 서비스를 제공할 수 있다. 또한, 1차적으 로 자가 관리를 수행하였지만 서비스 수준 계약을 만족시키지 못할 경우 홈 서비스 네트워크 시스템 내 다른 장치와 협업

표 1 AHU의 기능 설계 목표 및 스타일

Design Goal	Design Style
Slight efforts on the application development	Out of the Box
Minimal administrator intervention	Command-line/programming interface
Application-specific high-availability service provision	Application-controlled managing
Minimal overhead on running application/middleware	Stand-alone or background

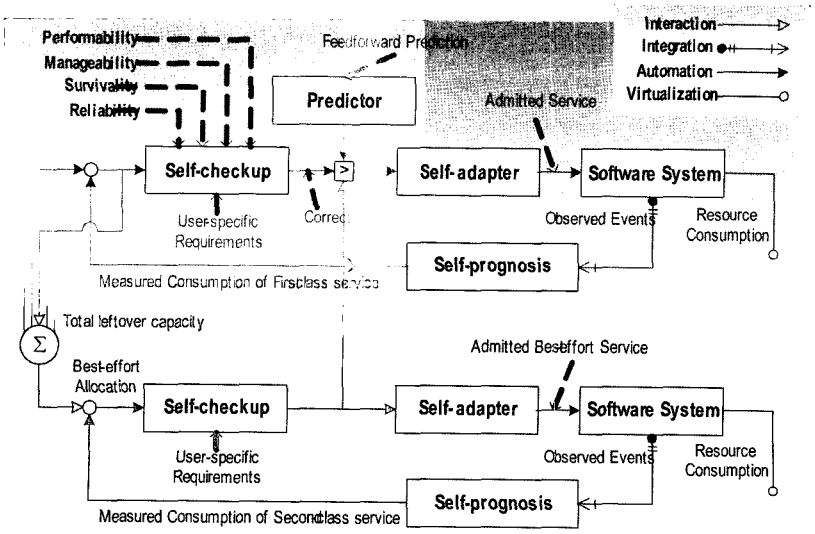


그림 2 AHU의 분산 협업을 위한 구조

을 통해 이를 해결할 수 있다. 또한 응용 프로그램과 플랫폼간 또는 기반구조 컴포넌트가 명확히 구분되어 있으며, 소프트웨어 서비스 또한 완벽히 가상화(virtualization)되어 있어 독립적 운영이 가능하다. 그리고 관리 기능의 통합 (integration)으로 인해 향상된 관리 기능을 내포하고 있다. 또한 예후 관리(예: 사용성 평가, 서비스 수준 계약 달성도 점검)를 통해 사용자에게 만족스러운 신뢰성 및 성능을 제공할 수 있다. 그리고 데이타베이스 재구성, 자료 무결성 검증, 주요 데이타 백업 등과 같이 꼭 필요하지만 빈번히 발생하는 작업이 자동화(automation)되어 있다. 또한 하드웨어 업그레이드, OS 버전 갱신 등과 같은 계획된 시스템 정지가 발생하더라도 작업 중이던 데이타를 자동 백업하고 복구할 수 있는 자가정정(self-correcting) 기능을 제공한다. 마지막으로 홈 서비스 네트워크 시스템은 단일 프로그램이나 단일 장치에서 제공받는 서비스 보다는 다수 프로그램이 연동되거나 여러 서버가 동작해야지만 이뤄지는 서비스가 많기 때문에 시스템 관리에 있어 필요한 서비스/시스템간 상호작용(interaction)을 지원하기 위한 시스템간 통신, 승인, 협업 기능을 제공한다.

그림 3은 AHU 기본 동작 원리 설명을 위해 AHU의 주요 기능을 제공하기 위한 컴포넌트간 상호작용을 보여준다. 먼저, 자가 검진 기능 모듈은 응용 프로그램, 장치 및 시스템의 컨텍스트 정보를 해당 홈 서비스 네트워크 시스템 구성요소에게 요청한다. 이때, 해당 구성요소에는 AHU의 클라이언트 기능 모듈이 탑재되어 있으며, AHU 컨텍스트 수집기로부터 정보 제공 요청을 받으면 관련 컨텍스트 정보를 수집하여 AHU 서버 기능 모듈에게 전송한다. 이때 AHU 서버 기능 모듈은 수집된 정보를 근간으로 해당 요소의 신뢰도와 가용도를 분석하고 안정적인 상태이며 해당 요소 관련 컨텍스트 정보를 메모리 및 영구 저장장치에 체크포인팅 및 저장한다. 이후 자가 예후 관리기에서 결함 발생이 예측되면 결함 알림기(fault notifier)는 프로액티브 결함 예방 조치가 필요함을 자가 수행(self-adapter) 기능 모듈에게 알리고, 자가 수행 기능 모듈은 현 상황에 적절한 결함 예방책을 적용한다. 또한 분석 결과 현재 결함이 발생되었음을 검진하면, 자가 수행 기능 모듈은 롤백(rollback) 메커니즘과 같은 리엑티브(reactive) 결함 관리를 수행한다.

4. 오토노믹 결함 관리 메커니즘

홈 서비스 네트워크 시스템에 적용하기 위해 본 논문에서 제안하는 오토노믹 결함 관리 메커니즘 설계는 다중 환경에서 객체 지향 스키마의 협업적인 개발을 지원하는 CIM(common information model)과 UML(unified modeling language)를 사용한다.

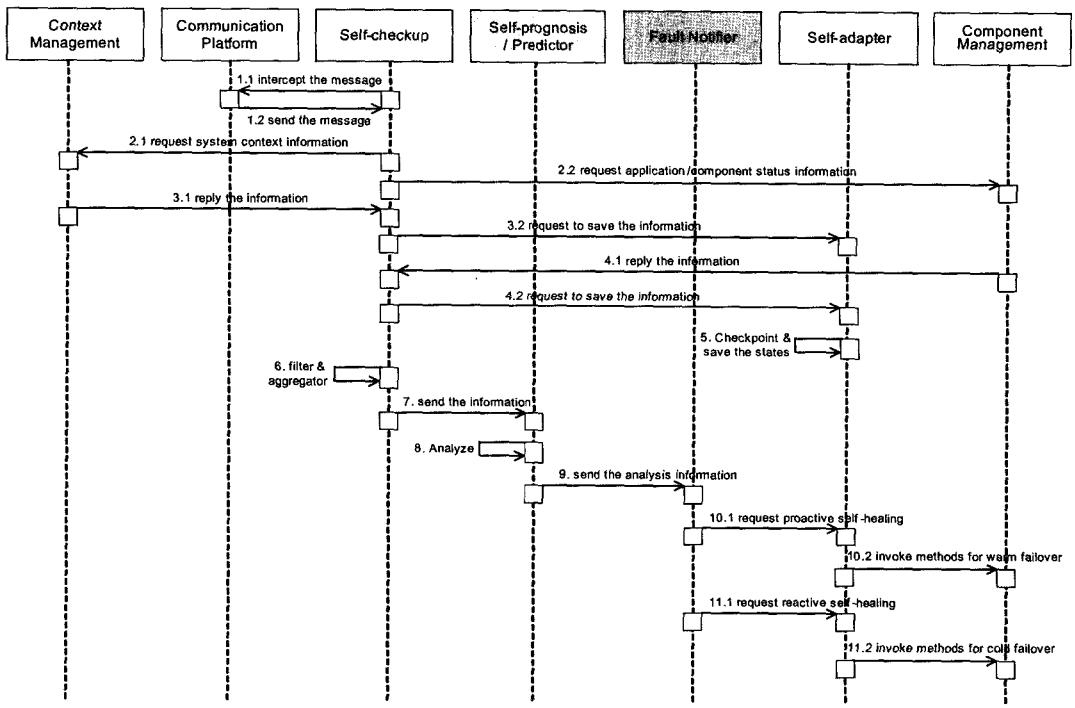


그림 3 AHU 주요 컴포넌트간 상호작용 흐름도

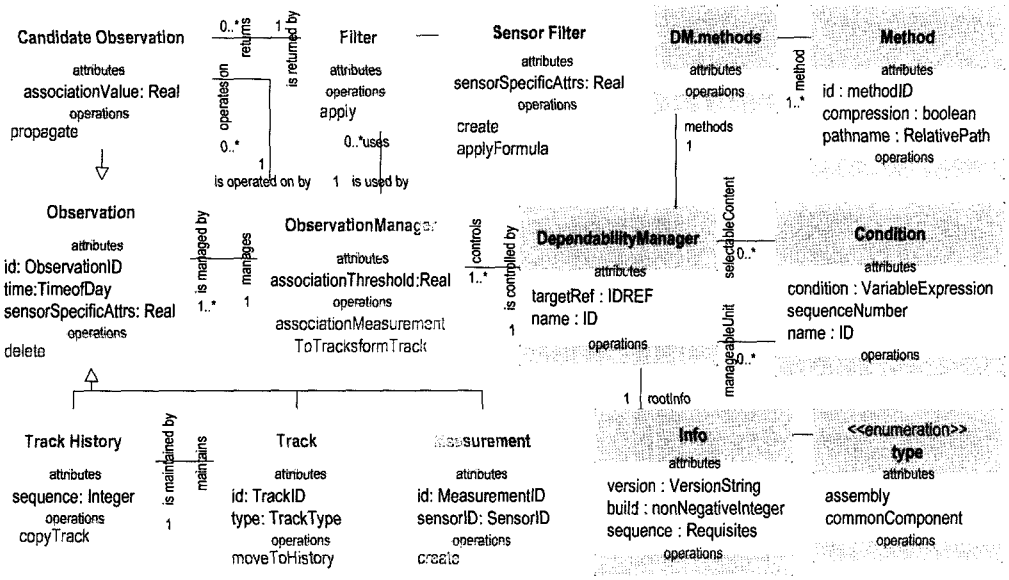


그림 4 오토노믹 결함 관리 메커니즘을 위한 UML 기반 설계도

그림 4에서 하나의 사각형으로 표현된 것은 시스템 관리를 위해 필요한 개념을 표현하기 위한 클래스이다. 기본적으로 클래스에는 클래스 이름, 정의를 위해 최소 필요한 자료구조, 각 클래스를 동작시키기 위한 연산들이 포함된다. 또한 각 클래스를 연결하는 선은 클래스간 관계를 표현하기 위한 것으로 다중성에 대한 정의도 포함되는데, 예를 들면, "하나의 의존도 관리자는 하나 이상의 다중 관찰 관리자를 제어한다. 또는 하나의 관찰 관리자는 하나의 의존도 관리자에 의해서 관리된다."와 같은 의미적인 표현을 내포하고 있다. 오토노믹 결함 관리 메커니즘은 크게 두 부분으로 나뉘는데, 하나는 관리를 위한 계획 수립과 실행을 제어하기 위한 의존도 관리자(dependability manager)이며, 다른 하나는 모니터링과 분석을 수행하기 위한 관찰 관리자(observation manager)이다(그림 4 참고).

4.1 의존도 관리자

의존도 관리자는 자율적인 결함 관리를 위한 메커니즘의 중추 역할을 담당하며, 홈 서비스 네트워크 시스템에 소프트웨어 재활 기술을 적용하기 위한 기본 모델이다. 따라서 사전에 정의된 서비스 계약 수준을 만족하기 위한 응용 프로그램과 전체 시스템의 품질을 평가하고, 필요하다면 발생한 문제를 해결하기 위한 메커니즘을 작동시킨다. 또한 계획을 수립하거나 방안을 적용시키기 위해서 관찰 관리자와 협업한다. 이 협업을 통해 자원의 상태에 대한 지시자를 생성하고 모델에 통합된 각 객체의 특성을 분석하는데, 기본적인 정보는 다음과 같다.

- **조건(condition)**: 시스템 내에 정의된 컴포넌트의 상태를 표현하기 위한 일반적인 지시자로써 이진(boolean) 표현 방식으로 표기하며 조건이 거짓일 경우 의존도 관리자는 해당하는 메커니즘을 동작시킨다.
- **기법(method)**: 고유의 인식자(identifier)를 가지며, 수행해야 하는 일련의 작업들로 주어진 요구사항을 만족하기 위해서 코드를 실행한다.
- **정보(Info)**: 시스템에 설치된 컴포넌트의 버전과 시스템 정보에 대한 특성을 묘사하고, 소프트웨어 요구사항을 만족하기 위한 플랫폼에 대한 정보 또한 포함한다.

4.2 관찰 관리자

관찰 관리자는 홈 서비스 네트워크 시스템의 상태에 대한 지시자를 생성하는 역할을 담당하며, 이를 위해 관련된 측정값을 수집, 통합, 필터링, 추적하는 기능을 수행하며, 오토노믹 컴퓨팅 기술의 모니터링과 분석 부분과 동일한 역할을 담당한다.

- **필터(filter)**: 모든 센서 필터에서 필요한 기본 정보를 제공하기 위한 클래스로 필터에 의해서 정의된 추적과 측정을 수행하여 정보를 생성한다.
- **측정(measurement)**: 관찰 관리자가 수집하게 되는 정보로써 하나의 센서 또는 센서들간의 통신 인터페이스에서 제공된 결과값으로 다양한 센서와 인터페이스가 존재하므로 범용적으로 사용할 수 있는 부분에 대해서만 정의한다.
- **관찰(observation)**: 결함 관리 영역에서 관리되는 객체 데이타로써 후보 관찰(candi-date observation)은

필터의 기준에 부합되는 각 추적과 측정을 위해 각각 생성된다.

- **추적(track)**: 추적 객체의 상태 자료로써 특정 추적과 관련된 과거 정보를 기록한다.

4.3 자동화 기법

시스템이 예기치 못하게 서비스가 불가능해지는 경우는 하드웨어 및 환경적인 요인에 의한 에러, 시스템 에러, 사람의 오작동에 의한 에러 중 하나가 발생하였지만 그것을 검출하지 못하여 그에 대한 대응책이나 복구를 수행하지 못하는 경우이다. 그런데 소프트웨어에 의한 결함 및 잘못된 자원 관리에 의해서 발생하는 결함이 홈 서비스 네트워크 시스템에서 큰 비중을 차지하고 있으므로 본 논문에서 해당 결함을 발생시키는 주요 원인인 시스템 에러를 처리하기 위한 메커니즘을 자동화하기 위한 기법을 제안한다. 이를 위해 사전연구[12,13]에서 개발한 자동적인 모니터링과 분석을 위한 기본 모델을 적절한 시스템 운영 모델과 시스템 에러를 검출하기 위한 참조모델로 확장한다. 또한 AHU는 과거 상태에서 수집된 정보를 해석할 수 있는 참조모델을 근간으로 메모리 및 프로세스와 같은 시스템 관련 객체의 변화를 추적한다. 여기서 상태 수집 정보란 소프트웨어의 장시간 작동에 의해 발생하는 소프트웨어 노화현상인 메모리 부족, 버퍼 오버플로우, 수학적 에러 누적 등을 감지하기 위해 필요한 상태 정보를 말한다. 또한 특정 시스템에서 발생할 수 있는 결함에만 국한되지 않고 각 시스템 및 장치의 관찰 관리자간 협업을 통해 전체 시스템 자원의 전반적인 상태변화를 수집한다. 이때, 수집된 정보를 근간으로 결함 발생 여부에 대해 분석하기 위해 결함 보고의 수, 유형, 심각성에 대해 테스트해야 되고 결함 보고 비율 및 밀도(단위 크기당 문제 보고 수)를 계산하여 결함에 대한 추세를 분석한다. 이를 위해서 객체지향 패러다임에서 사용하고 있는 구조적인 기술과 개념화 기술을 적용하여 필요로 하는 정보를 구성하고 이에 대한 해석을 수행함으로써 결함 발생 근원지를 찾아낼 수 있다. 예를 들면 그림 5와 같이 메모리 유출 현상에 의한 결함 발생 여부를 분석하기 위한 자원모델링으로 메모리의 사용가능 용량(Total byte available)이 한 프로세스가 실행되기 위해 필요한 메모리 용량(Minimum available)보다 작으면 결함이 발생할 수 있는 상황으로 규정하는 것이다.

또한 홈 서비스 네트워크 시스템은 사용자 요구에 따라 시스템 구성에 적용된 토폴로지가 변화할 수 있으며 새로운 컴포넌트가 추가되거나 제공받는 응용 서비스가 변경될 수 있다. 따라서 이와 같은 시스템 변화를 위한 시스템 매핑(mapping)과정에서 예기치 못한 결함이 발생할 수 있으므로 새로운 컴포넌트를 개발하거나 기반 구조의 서비스를 추가할 경우에는 기존 시스템에 미칠 수 있는 영향에 대한 분석을 수행해야 한다. 이를 위해서 전통적인 결함주입기법(fault injection technique)이 사용될 수 있다. 다시 말해서 AHU에 탑재된 결함주입 기법에 의해서 한 컴포넌트 및 서비스에 결함이 발생하였을 경우 미칠 수 있는 영향에 대해 분석하고 이를 근

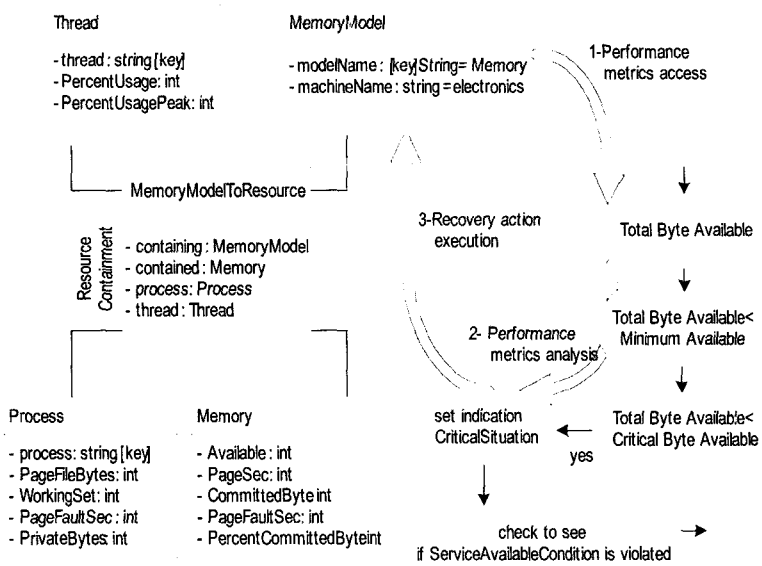


그림 5 메모리 부족 현상을 감지하기 위한 객체 모델링

간으로 결함이 발생하였을 경우 복구를 수행해야 되는 부분을 결정할 수 있는 정책을 결함 트리 형태로 정의할 수 있으며, 각 서비스를 제공하기 위해 필요한 최소한의 자원에 대해서도 결정한다. 예를 들어, 홈 서비스 네트워크 시스템의 대표적인 응용 서비스로 개발되고 있는 홈 보안 서비스 제공을 위해 필요한 컴포넌트(HomeSafetyComponent)를 컴포넌트간 상관 관계 규명을 통해 결함에 대한 영향을 분석하고 각 컴포넌트 및 서비스에서 필요로 하는 최소자원 요구량을 결함 트리 형태로 도식화하면 그림 6과 같이 표현된다.

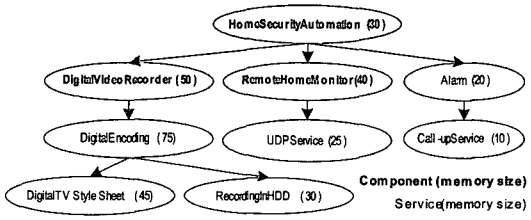


그림 6 결함 트리 및 최소자원 요구량

5. Implementation

본 장에서는 3,4장에서 설명한 AHU의 구조 및 설계를 근간으로 구현한 AHU를 설명한다. 우선 기본적인 구현 환경은 홈 서비스 네트워크 시스템의 최근 상태를 확인하거나 쓰레드 정보를 추출하기 위해 JDI(java debugging interface)를 사용하였으며, 대상 응용 프로그램의 주요 메소드(method)를 호출하거나 시스템 자원 정보 수집을 위해 JNI(java native interface)를 사용하였다.

5.1 클래스 정의

AHU의 컨텍스트 수집기(context collector)는 홈 서비스 제공을 위해 사용되는 장치나 가전제품의 자원 및 응용 프로그램 상태 정보를 수집하고 확인한다. 이를 위해 AHU 클라이언트 모듈은 결함 관리에 활용되기 위한 정보를 기반으로 사용자 장치, 가전 제품 또는 시스템의 스냅샷(snapshot)을 생성한다. 예를 들면, 그림 7은 홈 서비스에 사용된 휴대 장치의 스냅샷의 예로 시스템 및 대상 응용 프로그램의 상태 모니터링 및 수집된 관련 정보로 구성되어 있다. AHU 관리 모듈에서는 해당 스냅샷을 수신하면 기호(@, delimiter)를 기반으로 정보를 구별하고 분석을 위해 필요한 정보를 추출한다. 예를 들어 스냅샷에서 처음으로 얻을 수 있는 정보는 응용 서비스의 정상적인 종료 여부를 확인할 수 있는 플래그(flag) 정보이다. AHU 클라이언트 모듈은 특정 응용 서비스 제공을 위해 사용된 응용 프로그램이나 시스템 소프트웨어가 정상적으로 동작한 후 서비스가 완료되면 플래그를 1로 셋팅한다. 이와 반대로 시스템 오류(예: 네트워크 비연결, 계획되지 않은 재시작 등)나 사용자의 실수로 인해 비정상 종료가 되었을 경우 플래그를 2로 셋팅한다. 따라서 AHU 관리 모듈은 플래그 정보를 근간으로 비정상 종료가 되었음을 확인하고 이에 리엑티브 결함 관리가 필요함을 인식하게 된다. 이후 결함 발생 요인이 '자원 관리 오류인지, 사용자 실수인지, 쓰레드 정보를 분석한 결과를 근간으로 대상 응용 프로그램의 오동작 때문인지'와 같은 원인 분석을 나머지 스냅샷 정보를 근간으로 파악한다.

그림 8은 시스템 내 한 장치 및 해당 장치에서 동작하고 있는 응용 프로그램의 상태 정보를 수집하기 위한 AHU의 클라이언트 모듈과 AHU 서버 모듈간 절차를 도식화한 것이다. 그림 7의 휴대 장치의 스냅샷을 얻기 위해 사용된 대상 객체는 monlib.dll, batlib.dll, and ThreadMXBean이다. 다시 말해서 AHU 클라이언트 모듈은 JNI 및 ThreadMXBean 을 이용하여 대상 응용 프로그램에 대한 리소스 정보를 추출해 AHU 모니터링

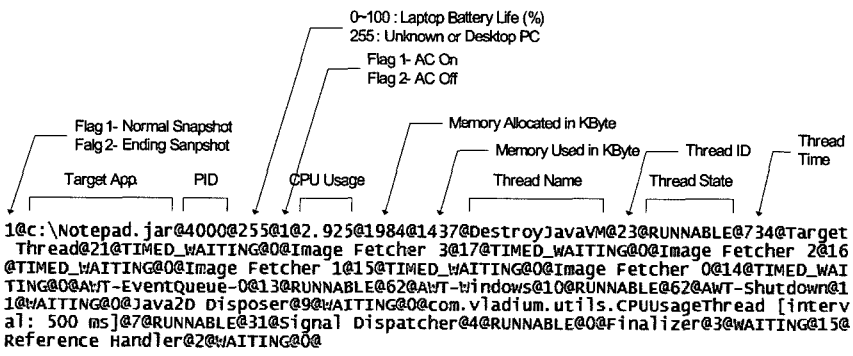


그림 7 휴대장치의 시스템 및 응용 프로그램 정보 획득을 위한 스냅샷

서버에 전송한다. 이때, AHU 클라이언트 모듈에서 추출한 정보는 장치별로 또는 대상 응용 프로그램 개발 환경별로 정보 표현 방식이 다르기 때문에 이를 범용적으로 인식할 수 있는 정보 변환이 필요하다. 따라서 AHU에서는 primitive-to-value 변환을 위해 JDI에서 제공하는 mirrorOf() 메소드를 이용하여 구현하여, 홈 서비스 네트워크 시스템의 구성 요소에서 제공되는 원시 데이타를 결합 관리에 활용할 수 있는 정보로 변환시킨다. 자바 가상 머신 내에서 활동 중인 쓰레드의 지역 변수, 배열의 내용을 가져오거나 재설정 하는 것이 가능하다면, 해당 쓰레드가 재시작 되었을 때, 예전의 상태로 복구하는 것이 가능하다. 가상 머신은 여러 개의 쓰레드로 구성되어 있으며, 각 트레드는 여러 개의 프레임으로 구성된 스택을 갖는다. 또한 스택 프레임을 조정하기 위해선 JDI에서는 각 프레임을 구성하는 지역 변수를 setValue(value), getValue(value) 메소드를 통해 설정하거나 얻어오는 것이 가능하도록 지원한다.

그림 9는 응용 프로그램별 상태 정보를 추출하기 위해 AHU 내에 구현된 일부 기능을 발췌한 것으로, 응용

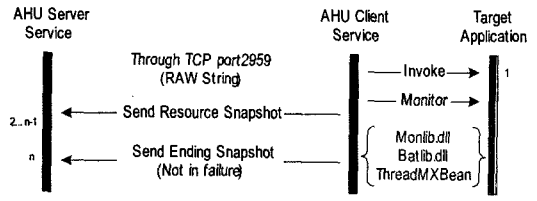


그림 8 상태 정보 수집을 위한 AHU의 상호작용 흐름도

프로그램별로 장치 및 시스템에서 할당한 자원 중 스택 프레임 내 지역 변수 값을 콘솔 창에 출력해주는 코드이다. 크게 3개의 For 반복문으로 구성되어 있으며, 첫 번째 반복문에서는 JDI에서 제공하는 메소드 중 allThreads()를 사용해, 가상 머신 내 동작하고 있는 모든 쓰레드를 탐색(traverse)하기 위한 것이다. 두 번째 반복문은 Thread-Reference.frames() 메소드를 통해 반환된 각 쓰레드의 프레임들을 탐색하여 프레임 정보를 추출하기 위한 것이고, 마지막 반복문은 각 스택 프레임의 할당된 지역 변수의 값에 접근하기 위해 Stack-Frame.visibleVariables() 메소드를 사용하게 된다. 이와 같은 과정으로 장치별, 응용 프로그램 상태 정보를 추출할 수 있다.

고유 아이디와 메인 클래스 이름을 가지고 AHU 클라이언트 모듈에서는 rData 객체를 생성하여 AHU 서버 모듈에 전송해 준다. rData 객체란 이질적인 네트워크 환경을 지원하기 위한 XStream 라이브러리를 이용하여 XML(extensible markup language)로 변환된 형태의 데이타를 의미하며, 또한 홈 서비스 네트워크 시스템의 구성 요소의 상태 정보를 포함하고 있어 체크포인팅을 위한 기본 데이타로 사용된다. 따라서, 현 장치의 응용 서비스가 안정적일 경우 rData를 신뢰할 수 있는 저장장치에 보관하게 된다. rData 수신 측은 그림 10과 같이 java.io.BufferedReader.readLine() 메소드를 이용해 </rData> 태그를 수신할 때까지 버퍼에 쌓아두었다가 한꺼번에 rData 객체로 변환해 주는 작업이 필요하다. 그러므로 이 때 버퍼의 역할을 하는 것이 String

```
// VMmon read stack frames of vm.allThreads();
List threads = vm.allThreads();
for (int i = 0; i < threads.size(); i++)
{
    // read all stack frames
    List frames = thread.frames();
    for (Iterator j = frames.iterator(); j.hasNext();)
    {
        StackFrame frame = (StackFrame)j.next();

        // read all local variables and output their values
        List locals = frame.visibleVariables();
        for (Iterator k = locals.iterator(); k.hasNext();)
        {
            LocalVariable var = (LocalVariable)k.next();

            System.out.println(var.typeName() + " " + var.name() + " = " + frame.getValue(var));

            . // 1. Recover 'var' until rDataList becomes empty
            . // 2. Sending new rData to repository service
        }
    }
    thread.resume();
}
```

그림 9 쓰레드 정보 추출을 위한 함수

```
String xml = "", xmlPiece = "";
while ((xmlPiece = inputStream.readLine()) != null)
{
    if (xmlPiece.compareTo("EOF") == 0) break;

    // Getting the piece of the xml
    xml += xmlPiece + "\r\n";
    if (xmlPiece.compareTo("</rData>") != 0) continue;

    // Extracting rData from xml
    try {
        XStream xStream = new XStream();
        rData NEW = (rData)xStream.fromXML(xml);
        rDataList.add(NEW);
    } catch (Exception e) {
        continue;
    }

    // Clearing the xml
    xml = "";
}
```

그림 10 XML 기반 자료 전송을 위한 버퍼링 함수

xml 변수이며, readLine() 메소드의 결과값을 저장하는 임시 장소는 String xmlPiece 이다.

5.2 AHU의 스크린샷

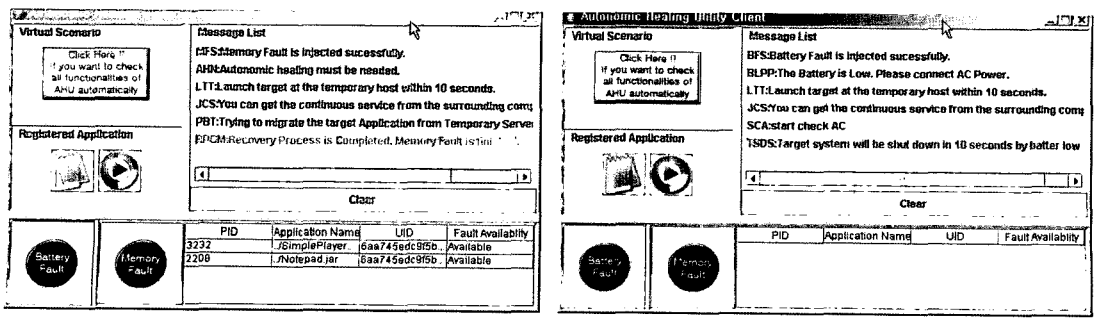
본 절에서는 AHU의 큰 3개 구성 요소인 AHU의 클라이언트 모듈, 서버 모듈, 백업 모듈의 실행 동작 화면을 통해 구현된 AHU의 동작모습을 설명한다.

AHU의 클라이언트 모듈은 그림 11과 같으며, 역할은 홈 서비스 네트워크 시스템의 자원, 장치, 실행 중인 응용 프로그램 등의 상태 정보를 감시 및 추출하고, 또한 사용자에게 현 결함 관리 진행 상황을 알려주는 역할을 담당한다. 그림 11(a)는 현재 대상 응용 프로그램 메모장(Notepad.jar)과 미디어 플레이어(SimplePlayer.jar)가 실행되었으며, 미디어 플레이어가 동작 중 메모리 관리와 관련하여 문제가 발생하여 오동작을 발생시킬 확률이 존재함을 사용자에게 알려준다. 또한 이를 해결하기 위해 AHU의 오토노믹 결함 관리가 필요함을 알려준 후 치료 동안 서비스 중단을 방지하기 위해 대체 장치(temporary server)에서 지속적인 서비스가 가능함을 알린다. 그리고 최종적으로 메모리 결함이 치료되어 정상적인 서비스가 가능하다는 것을 다시 사용자에게 알려준다. 그림 11(b)는 휴대장치의 전력 잔여량이 부족하므로 서비스 중단이 불가능함을 사용자에게 알려주고 이를 해결하기 위해선 AC 전원을 통해 충전을 할 것을 지시한다. 하지만 일정 시간 이후까지 사용자가 AC 전원을 연결하지 않으면 현 휴대장치에서 실행 중인 응용 서비스를 대체 장치로 이동시켜 휴대장치가 셧다운된 후에도 전체 홈 서비스에 영향이 미치지 않도록 한다.

그림 12는 AHU의 서버 모듈의 실행 모습을 보여주는 것으로 홈 서비스 네트워크 시스템에서 동작 중인 응용 서비스를 확인할 수 있으며 더불어 특정 응용 서비스에서 사용 중인 자원 상태, 프로그램 동작에 영향을 주고 있는 트레드 실행 상태 등을 감시할 수 있다. 그림 12(a)는 그림 11(a)에서 메모리 결함이 발생한 경우 AHU 서버 모듈에서 감지하는 것과 이 경우 결함 복구를 위한 동작을 보여준다. 먼저 메모리 사용량 그래프를 보면 일정 수준을 유지하던 그래프가 비정상적으로 급증하는 것을 확인할 수 있으며 이때 AHU 서버 모듈에서는 이와 같은 상황이 응용 서비스의 정상적인 동작 모습인지 아니면 결함이 발생한 것인지를 쓰레드 정보를 기반으로 판단하게 된다. 또한 결함이 발생되거나 예측되면 이를 치료하는 동안 야기되는 서비스 중단을 막기 위해 대체 장치(IP: 210.107.197.105)로 대상 응용 서비스를 인계시킨다. 그림 12(b)는 결함 처리가 완료된 후 본 장치(IP: 210. 107.197.174)로 해당 응용 서비스를 다시 인계시키는 과정을 보여준다.

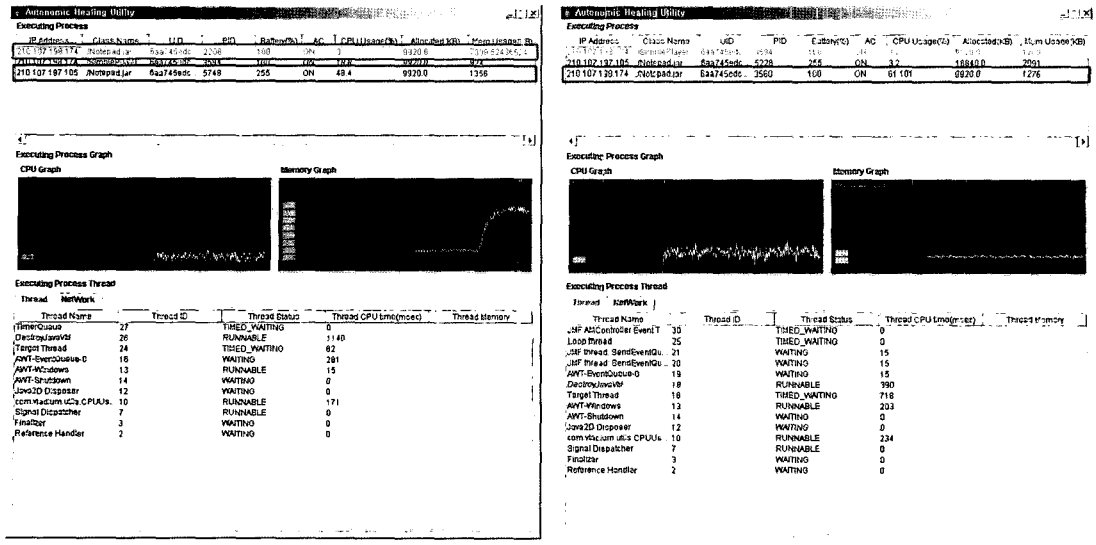
5.3 AHU의 성능분석

본 절에서는 본 논문에서 개발한 오토노믹 자가 관리 메커니즘의 실효성을 검증하기 위한 성능 분석을 수행한다. 기본 환경은 Fujitsu LifeBook S6240 P-M 1.60GHz 장치를 사용하였으며, 배터리는 Lithium Ion Battery 10.8v 4800mAh를 사용하고, 평균 체크포인트 데이타 크기는 100.5 bytes이라 가정한다.



(a) 메모리 결함 발생 이후 복구가 완료된 상태 (b) 휴대장치의 전력 결함을 감지한 상태

그림 11 AHU의 클라이언트 및 백업 모듈의 실행 화면



(a) 메모리 결함이 발생한 경우 (b) 결함이 해결된 후 시스템 상태

그림 12 AHU의 서버 모듈의 실행 화면

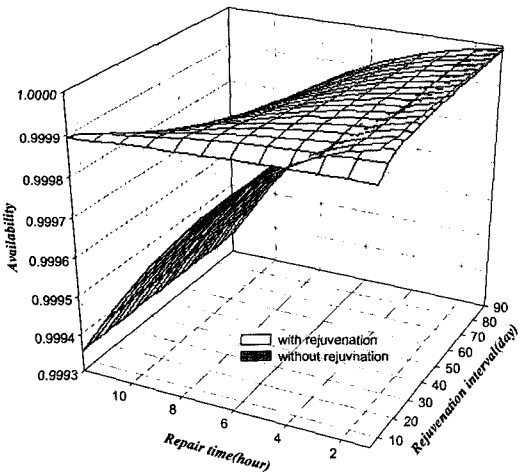


그림 13 소프트웨어 재활이 가용도에 미치는 영향

그림 13은 소프트웨어 재활 메커니즘의 장착 여부에 따라 시스템 가용도에 미치는 영향을 분석한 것이다. 결함이 발생한 서버를 수리하는데 필요한 시간이 적으면 적을수록 두 시스템 모두 가용도가 증가하지만, 소프트웨어 재활이 가능한 시스템은 긴 수리 시간이 필요한 경우에도 일정 수준의 가용도를 유지하는 것을 볼 수 있다. 그러나 재활을 수행하는 시스템일지라도 수리시간이 매우 길고 재활 주기가 길면 결함허용 시스템을 위해서 만족해야 하는 가용도 기준(99.99%)을 만족하지 못한다. 따라서 초기에 시스템을 구성하여 제공하고자 했던 성능을 유지하기 위해선 소프트웨어 재활 메커니즘과 같은 가용도 향상을 위한 메커니즘을 갖추고 있어야 한다는 것을 의미하며, AHU가 이를 제공해주고 있음을 파악할 수 있다.

표 2는 본 논문에서 제안한 메커니즘의 실효성 검증을 위해 Young[16]의 결과에 따른 Optimal 체크포인트와 AHU에 탑재한 Aggressive 체크포인트 기법의 성능을 비교, 분석한 것이다. Saving Time(T_s)이란 체크포인트 수행에 소모된 시간을 의미하며, System. current-TimeMillis() 메소드를 체크포인트 수행 코드 앞, 뒤에 삽입하여 그 차이를 기록한 시간이다. 또한 응용 프로그램 종료 시점까지 계속 누적되어 어플리케이션 종료 후 Rerun Time(T_r)과 함께 Loss Time(T_l)을 구할 수 있게 된다. T_r은 시스템에 결함이 발생하여 최근 저장된 체크포인트 상태로 돌아가 다시 재동작하기 위해 필요한 시간이며, T_l은 체크포인트 시점과 결함 발생 시점 사이 저장되지 못한 시스템 운영시간을 의미한다. 그리고 CPU 사용량 및 체크포인트 횟수 역시 기록하도록 하여 Optimal 체크포인트 기법과 Aggressive 체크포인트 기법간의 수치를 비교한 결과이다. 표 2에서 볼 수 있듯이 응용 프로그램의 실행 시간 자체에는 큰 차이가 없음을 알 수 있다. 또한 수행된 체크포인트 수 역시 예상대로 비슷한 수치를 보였다. 하지만 Total Saving Time은 Aggressive 체크포인트 기법이 약간 높은 수치를 보였으나, Rerun Time은 2703msec로 Optimal 체크포인트 결과의 30분의 1수준으로 낮게 나왔다. 따라서 AHU에 탑재한 체크포인트 기법이 시스템의 가장 최근 상태를 저장할 수 있으며, 결함이 발생하였을 경우에 보다 빠른

표 2 체크포인트 기법이 시스템 성능에 미치는 영향

	Aggressive Checkpoint	Optimal Checkpoint
Total Run Time	3020297msec.	3010109msec.
Num. of Checkpoints	32	34
Checkpoint Interval, T_c	6764msec. (556029msec. in normal)	78441msec.
Rerun Time, T_r	2703msec.	82381msec.
Total Saving Time, ΔT_s	18296msec.	12593msec.
Loss Time T_l	20999msec.	95874msec.

시간에 복구할 수 있으므로 고가용성 홈 서비스 네트워크 시스템에 적합하다고 하겠다.

6. Conclusion

홈 서비스 네트워크 시스템은 이질적인 네트워크 시스템의 통합이며, 다양한 사용자 요구에 따라 개발된 다수 응용 서비스를 관리하기 위한 시스템이다. 따라서 언제든지 사용 가능해야 하며 고가용도 요구사항을 만족해야만 한다. 더불어 사용자가 시스템 관리에 직접 참여되지 않도록 하여야 하며, 결함이 발생하기 이전에 이를 예방할 수 있는 솔루션이 필요하다. 따라서 본 논문에서는 홈 서비스 네트워크 시스템의 구성 요소간 결함 발생 원인을 규명하고 진단하여 이를 리엑티브 뿐만 아니라 프로엑티브하게 처리할 수 있는 오노토믹 자가치유 유틸리티를 설계 및 개발하였다. 또한 오토노믹 자가치유 유틸리티의 프로토타입 구현을 통해 이의 동작 및 실행 결과를 분석하여 그 실효성을 검증하였다. 이는 시스템 운영 중에도 예기치 못한 결함을 감지 및 예측하여 서비스 중단 요인을 제거하여 전체 홈 서비스 네트워크 시스템의 안정적인 동작이 가능하게 하였으며, 결함 발생시에도 이를 복구함에 있어 야기될 수 있는 서비스 중단을 대체 장치에서 수행하도록 하여 전체 시스템의 가용도를 향상시켰다. 향후 연구에서는 보다 다양한 응용 서비스 및 장치에 AHU를 탑재할 수 있도록 AHU의 관리 범위를 확장할 계획이다.

참 고 문 헌

[1] I. Marshall, et al., "Active management of multi-service networks," Proceedings of IEEE/IFIP Network Operations and Management Symposium, pp. 981-982, Apr. 2000.

[2] S. Lee, J. Song, and D. Lee, "Hierarchical network architecture for efficient home service management," Proceedings of IEEE Conference on Local Computer Networks, pp. 519-520, Nov. 2005.

[3] T. Saito, et al., "Home gateway architecture and its implementation," IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp. 1161-1166, Nov. 2000.

[4] A. Ganek, et al., "The response to IT complexity: autonomic computing," Proceedings of 3rd IEEE International Symposium on Network Computing and Application, pp. 151-157, Aug. 2004.

[5] D. Patterson, et al., "Recovery-oriented computing: motivation, definition, techniques, and case studies," UC Berkeley, UCB/CSD-02-1175, Mar. 2002.

[6] A. Ganapathi, "Why PCs are fragile and what we can do about it: a study of windows registry problems," Proceedings of IEEE International Conference on Dependable Systems and Networks, pp. 561-566, June 2004.

[7] S. Garg, A. Moorsel, K. Vaidyanathan, and K. Trivedi, "A methodology for detection and estimation of software aging," Proceedings of 9th IEEE International Symposium on Software Reliability Engineering, pp. 282-292, Nov. 1998.

[8] D. Scott, "Making smart investments to reduce unplanned downtime," Tactical Guidelines Research Note TG-07-4033, Gartner Group, 1999.

[9] Y. Huang, C. Kintala, N. Kolettis, and N. Fultion, "Software rejuvenation: analysis, module and applications," Proceedings of 25th IEEE International Symposium on Fault-Tolerant Computing, pp. 318-390, June 1995.

[10] K. Trivedi, K. Vaidyanathan, and K. Popstojanova, "Modeling and analysis of software aging and rejuvenation," Proceedings of IEEE 33rd Annual Simulation Symposium, pp. 270-279, Apr. 2000.

[11] C. Choi and S. Kim, "Self-configuring algorithm for software fault tolerance in (n, k)-way cluster systems," Lecture Notes in Computer Science, Springer, Vol. 2667, pp. 742-751, 2003.

[12] C. Choi and S. Kim, "A dependability management mechanism for ubiquitous computing systems," Proceedings of IFIP/IEEE International Workshops on Trusted and Autonomic Ubiquitous and Embedded Systems, pp. 1293-1302, Dec. 2005.

[13] 최창열, 김성수, "고가용성 유비쿼터스 컴퓨팅 시스템을 위한 오토노믹 자가 관리 메커니즘", 한국정보과학회(시스템및이론), Vol. 32, No.11, pp. 566-577, Dec. 2005.

[14] G. Lanfranchi, et al., "Toward a new landscape of systems management in an autonomic computing environment," IBM System Journal, Vol. 42, No. 1,

pp. 119-128, 2003.
[15] 김성수, 조위덕, "Ubiquitous smart space", 유비쿼터스 지능공간 백서, (재)유비쿼터스컴퓨팅사업단, 정보통신부, Nov. 2005.
[16] J. Young, "A First Order Approximation to the Optimum Checkpoint Interval," Communication of the ACM, vol. 17, No. 9, pp. 530-531, Sep. 1974.

최 창 열

1999년 아주대학교 정보통신공학과 졸업(공학사). 2002년 아주대학교 정보통신전문대학원 졸업(공학석사). 2007년 아주대학교 정보통신전문대학원 정보통신공학과(공학박사). 2007년~현재 숭실대학교 정보미디어기술연구소 전임연구원. 관심분야는 오토노믹 컴퓨팅, 홈 네트워크 시스템, SOA, 고가용성 시스템 등

김 성 수

1982년 서강대학교 전자공학과 졸업(공학사). 1984년 서강대학교 전자공학과 졸업(공학석사). 1995년 Texas A&M University 전산학과(공학박사). 1983년~1996년 삼성전자 수석연구원. 2002년~2003년 Texas A&M University 교환교수 1996년~현재 아주대학교 정교수. 2006년~현재 아주대학교 정보통신전문대학원 원장. 관심분야는 Dependable System & Networks, Autonomic Computing, Ubiquitous Computing & Networks 등