

고가용성 홈 서비스를 위한 오토노믹 자가관리 유ти리티

최창열⁰ 김성수 조위덕

아주대학교

{clchoi⁰, sskim, chowd}@ajou.ac.kr

Autonomic Self-Management Utility for High-available Home Service

Changyeol Choi⁰, Sungsoo Kim, We-Duke Cho
Ajou University

요약

홈 서비스 네트워크 시스템의 복잡도가 증가하면서 고가용도 요구사항을 만족하기 위한 관리 메커니즘 개발이 더욱 어려워졌다. 더욱이, 사용자는 시스템 결함 관리와 같은 복잡한 시스템 관리를 직접 하는 걸 원하지 않는다. 그러므로 홈 네트워크로 연결된 장치 또는 가전제품으로 구성된 통합 시스템의 고가용성 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 오토노믹 자가관리 유ти리티를 설계 및 개발한다.

1. 서론

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).

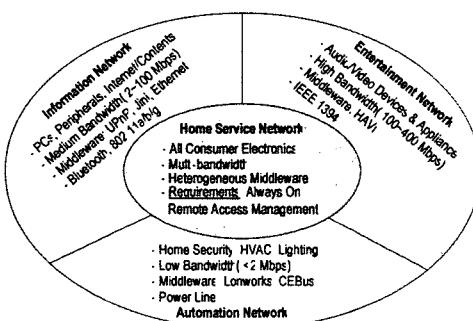


그림 1. 홈 서비스 시스템의 3대 구성 네트워크

하지만 전체 비용의 33%에서 50% 정도의 비용이 시스템에서 발생한 예기치 못한 결함에 대비하고 이를 복구하는데 소요된다[1]. 여기서 주목할 만한 점은 시스템의 예기치 못한 결함 발생 주원인이 하드웨어적인 문제에서 소프트웨어적인 문제로 변하였으며, 또한 시스템을 구축 및 유지함에 있어서 발생하는 문제보다는 운영 중 발생한 잘

못으로 인한 문제로 변하였다는 것이다[2]. 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. 따라서 본 논문에서는 홈 서비스 네트워크 시스템의 원격 자가 관리 메커니즘을 설계한 후 오토노믹 자가관리 유ти리티 형태인 AHU (autonomic healing utility)를 개발하고자 한다.

2. AHU 기능 및 구조

AHU는 홈 서비스 네트워크 시스템에 결함이 발생하기 이전에 결함 발생 요인을 관리하고 결함이 발생할 가능성이 있는 응용 서비스를 검출하기 위한 결함 예방 서비스와 더불어 프로액티브한 결함 복구 서비스를 자동화한 것으로 디자인 설계 접근 방식은 표 1과 같다.

표 1. AHU의 디자인 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

또한, AHU에 탑재된 고가용성 서비스 관리 메커니즘은 본 연구의 사전 연구결과[3]로 도출된 오토노믹 컴퓨팅

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크 원천기반기술개발 사업의 지원에 의한 것임.

기술과 오토노믹 설계/개발 방법론을 기반으로 설계 및 개발되었으며, 주요 특징은 다음과 같다.

- 흄 서비스 네트워크 시스템을 관리하기 위한 제어를 원격에서 할 수 있도록 하는 제어 리다이렉션(control redirection) 기능 지원
 - 시스템 및 응용 서비스의 헬스(health) 모니터링 지원
 - 이벤트 트리거 결함 검출 및 경량화된 결함 분석 기법에 의한 결함 진단
 - 고속 마이크로 소프트웨어 재활을 위한 부팅 경로 및 응용 서비스 실행 경로 분석
 - 즉각적인 시스템 셧다운(shutdown) 및 점차적인 셧다운 절차 지원
 - 응용 서비스 수준의 non-blocking 체크포인트 기반 시스템 운영 중 결함 처리의 자동화 기능 제공

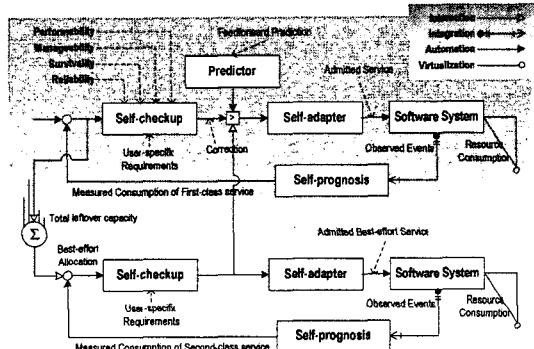


그림 2. AHU의 분산 협업을 위한 구조

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 유필리티이다[4]. 따라서 사용한 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, 1차적으로 자가 관리를 수행하였지만 서비스 수준 계약을 만족시키지 못할 경우 흔 서비스 네트워크 시스템 내 다른 장치와 협업을 통해 이를 해결할 수 있다. 또한 AHU는 시스템 성능(performability), 관리성(manageability), 신뢰성(reliability), 보안성(survivability)을 보장하기 위해 변화된 상황에 스스로 대처할 수 있는 기능을 갖고 있다. 따라서 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 항상된 관리 기능을 내포하고 있다. 또한 예후 관리(예-

사용성 평가, 서비스 수준 계약 달성을 점검)를 통해 사용자에게 만족스러운 신뢰성 및 성능을 제공할 수 있다. 그리고 데이터베이스 재구성, 자료 무결성 검증, 주요 데이터 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 데이터를 자동 백업하고 복구 할 수 있는 자가정정(self-correcting) 기능을 제공한다. 마지막으로 흔 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.

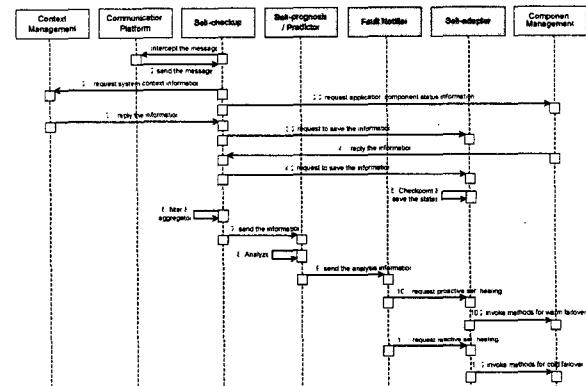


그림 3. AHU 주요 컴포넌트간 상호작용 흐름도

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 풀 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 채크포인팅 및 저장한다. 이후 자가 예후 관리 기에서 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 블랙 메커니즘과 같은 리액티브 결함 관리를 수행한다.

3. AHU 개발 및 구현

본 장에서는 앞서 설명한 AHU의 구조 및 설계를 근간으로 구현한 AHU를 설명한다. 우선 기본적인 구현 환경은 흠 서비스 네트워크 시스템의 최근 상태를 확인하거나 트래드 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. 자바 가상 머신 내에서 활동 중인 트래드의 지역 변수, 배열의 내용을 가져오거나 재설정 하는 것이 가능하다면, 해당 트래드가 재시작 되었을 때, 예전의 상태로 복구하는 것이 가능하다. 가상 머신은 여러 개의 트래드로 구성되어 있으며, 각 트래드는 여러 개의 프레임으로 구성된 스택을 갖는다. 또한 스택 프레임을 조정하기 위해서 JDI는 각 프레임을 구성하는 지역 변수를 setValue(value), getValue(value) 메소드를 통해 설정하거나 얻어오는 것이 가능하도록 지원한다. 또한 JDI에서 제공하는 메소드 중 allThreads()를 사용해, 가상 머신 내 동작하고 있는 모든 트래드를 탐색할 수 있다. 그리고 Thread-Reference.frames() 메소드를 통해 반환된 각 트래드의 프레임들을 탐색하여 프레임 정보를 추출할 수 있으며, 스택 프레임의 할당된 지역 변수의 값에 접근하기 위해 StackFrame.visibleVariables() 메소드를 사용하게 된다.

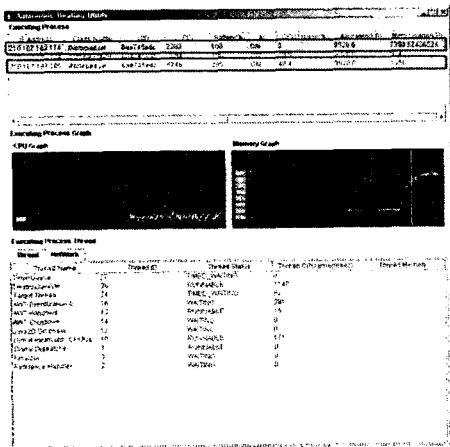


그림 4. AHU의 서버 모듈의 실행 화면

그림 4는 AHU의 서버 모듈의 실행 모습을 보여주는 것으로 흠 서비스 네트워크 시스템에서 동작 중인 응용 서비스를 확인할 수 있으며 더불어 특정 응용 서비스에서 사용 중인 자원 상태, 프로그램 동작에 영향을 주고 있는 트래드 실행 상태 등을 감시할 수 있다. 또한 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 비정상

적으로 급증하는 것을 확인할 수 있으며 이때 AHU 서버 모듈에서는 이와 같은 상황이 응용 서비스의 정상적인 동작 모습인지 아니면 결함이 발생한 것인지를 트래드 정보를 기반으로 판단하게 된다. 또한 결함이 발생되거나 예측되면 이를 치료하는 동안 야기되는 서비스 중단을 막기 위해 대체 장치로 대상 응용 서비스를 인계시킨 후 복구 작업을 수행함으로 끊임없는 서비스가 가능하다.

4. 결론

흡 서비스 네트워크 시스템은 이질적인 네트워크 시스템의 통합이며, 다양한 사용자 요구에 따라 개발된 다수 응용 서비스를 관리하기 위한 시스템이다. 따라서 언제든지 사용 가능해야 하며 고가용도 요구 사항을 만족해야만 한다. 더불어 사용자가 시스템 관리에 직접 참여되지 않도록 하여야 하며, 결함이 발생하기 이전에 이를 예방할 수 있는 솔루션이 필요하다. 따라서 본 논문에서는 흠 서비스 네트워크 시스템의 구성 요소간 결함 발생 원인을 규명하고 진단하여 이를 리액티브 뿐만 아니라 프로액티브하게 처리할 수 있는 AHU를 설계 및 개발하였다. 또한 AHU의 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 흠 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.

- [1] 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.
- [2] 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.
- [3] 최창열, 김성수, "고가용성 유비쿼터스 컴퓨팅 시스템을 위한 오토노믹 자가 관리 메커니즘," 한국정보과학회 (시스템및이론), Vol. 32, No.11, pp. 566–577, Dec. 2005.
- [4] 김성수, 조위덕, "Ubiquitous smart space," 유비쿼터스 지능공간 백서, (재)유비쿼터스컴퓨팅사업단, 정보통신부, Nov. 2005.