유비쿼터스 컴퓨팅 환경에서 상황적응형 자가구성 시스템의 설계와 구현

A Design and Implementation of Context-Adaptive Self-Configuration System in Ubiquitous Computing Environment

이승화, 오제환, 이은석

성균관대학교 정보통신공학부 컴퓨터공학과
경기도 수원시 장안구 천천동 300, 440-746
Tel: +82-31-290-7220, Fax: +82-31-290-7211, E-mail: {jbmania, hide7674, eslee}@selab.skku.ac.kr

요 약

본 논문에서는 분산된 관리대상의 시스템자원과 사용자정보, 사용패턴을 Context로 수집하여, 구성(Configuration)을 수행하는 적응형 자가관리시스템을 제안한다. 본 시스템은 기존에 수동으로 이루어지던 Configuration작업들(Install, Reconfiguration, Update)을 자율적으로 수행하여, 사용자의 시스템관리에 대한 부담을 줄여주게 되며, 많은 비용과 오류를 감소시켜준다. 본 시스템은 수집된 Context정보를 기반으로 사용자의 환경에 맞는 구성요소를 선택하여 설치하게 되며, 사용자의 기존 애플리케이션의 환경설정과 사용패턴을 기반으로, 보다 개인화된 설정을 해준다. 설정 이후에는 사용자의 행동을 암시적 피드백으로 받아, 이를 학습하고 유사한 상황이 다시 발생할 경우, 이를 다음 행동에 반영한다. 그리고 기존에 중앙서버로부터 일률적으로 관련파일을 전송하고 관리하는 중앙집중배포방식의 여러 문제점에 대응하기 위해 Peer-to-Peer방식으로 파일을 카피하고, 이를 통해 중앙서버의 과부하를 줄이는 동시에 빠른 파일의 배포가 가능하도록 하였다. 본 시스템의 평가를 위해 프로토타입을 구현하여, 기존 수동 Configuration작업, MS-IBM과 같은 관련시스템과의 비교를 수행하였으며, 기능적 측면과 작업에 소요되는 시간에 대한 비교결과를 통해 본 시스템의 유효성을 증명하였다.

Keywords:

Context-Awareness; Ubiquitous Computing; Autonomic Computing; Self-Configuration

1. 서 론

최근 정보통신관련기술(IT)의 급속한 발전을 통해 각 디바이스들의 성능은 갈수록 높아지고 있으며, 이러한 변화에도 불구하고 가격은 지속적으로 낮아지고 있다. 이와 함께 Ubiquitous Computing의 등장으로 보다 다양한 형태의 컴퓨팅 디바이스들이 확산되고 있으며, 이와 같은 변화에 따라 관리해야 할 대상은 갈수록 증가하고 있고 복잡성도 증가하고 있다. 그동안 이러한 시스템의 관리는 대부분 인간이 직접 수행하였으며, 이에 따라 시스템의 전문관리요원도 계속 증가하고 있는 추세이다.

전문조사기관에 의하면 데스크탑, 워크스테이션, PDA, 휴대폰, 호출기 등 다양한 컴퓨팅 디바이스의 연평균 성장률은 앞으로 3년간 38%에 이를 것이라고 한다. 그리고 계속 이러한 추세로 디바이스가 확산된다면, 인터넷에 연결된 전 세계의 10억 인구와 수백만 기업을 지원하기 위해, 앞으로 10년간 2억의 관리 전문가가 필요할 것이라고 추측하고 있다. 이는 대략 미국 총 인구에 달하는 수치이다.[1]

이렇게 변화하고 있는 컴퓨팅 환경에서, 나날이 늘어나는 디바이스들을 인간이 일일이 관리하는 것은 매우 번거롭고 어려운 일이다. 구체적으로 여러 호스트를 관리-운영하는 시스템의 경우, 각 호스트에 소프트웨어를 일일이 설치하고, 업데이트 하는 일은 비용이 많이 들 뿐 아니라, 많은 시간과 노력이 필요한 작업이다. 또한 각 디바이스의 관리에 익숙하지 않은 초보적인 사용자들은 운영체제의 긴급결함패치나 백신 프로그램의 업데이트의 경우, 그 필요성이 매우 중요함에도 불구하고, 관리를 소홀히 하여 시스템에 치명적인 문제가 발생하기도 한다.

이와 같은 문제에 대응하기 위한 기존의 노력은 크게 다음과 같다. : 분산 자원들의 중앙통합관리, 작업부하를 줄이기 위한 무인설치기술, 주기적인 자동 업데이트 등.

그러나 모두 공통적으로 작업의 자동화에 대한

연구가 주 목적이며, 사용자나 시스템의 Context를 반영한 차별화된 Configuration에 대한 연구는 매우 취약한 편이다. 따라서 본 논문에서는 관련연구의 장점들을 통합함과 동시에, 분산된 관리대상의 *시스템자원과 사용자정보, 사용패턴*을 Context로 수집하여, Configuration을 수행하는 적응형 자가구성 시스템을 제안한다.

본 제안시스템은 Install작업에서는, 사용자와 시스템의 다양한 Context 정보를 이용하여 구성요소를 설치할 때 사용되는 '자동응답파일'을 개인화 하여 작성하며, 사용자 취향을 반영한 기본 환경설정을 자동으로 수행한다. 또한 Reconfiguration 작업에서는 시스템의 자원상황에 따라 설치된 구성요소의 옵션을 자동으로 변경해주며, 이를 통해 변화되는 시스템 상황에서 사용자의 작업을 안정적으로 지속시켜준다. Update 작업에서는 구성요소의 사용빈도를 파악하여 업데이트의 우선순위를 결정하며, 이를 통해 관리대상 시스템에 남은 저장공간이 부족하거나 업데이트에 필요한 시간이 부족할 경우, 효율적인 작업을 가능하게 한다. 파일의 카피는 기존의 중앙 집중배포방식의 여러 단점을 보완하고 보다 효율적인 업데이트를 수행하기 위해 Peer-to-Peer 방식을 이용하였다.

우리는 본 시스템의 평가를 위해 프로토타입을 구현하여, 기존 수동 Configuration작업, MS-IBM과 같은 관련시스템[3][4]과의 비교를 수행하였으며, 기능적 측면과 작업에 소요되는 시간에 대한 비교결과를 통해 본 시스템의 유효성을 증명하였다.

본 논문의 구성은 다음과 같다. 2장에서는 Configuration에 대한 여러 기능들(Install, Reconfiguration, Update)을 분류하고 정의하였으며, 각 기능별로 관련 연구들을 소개하여, 기존 연구들의 장단점 분석을 통해 해결해야 할 문제점을 제시하였다. 3장에서는 제안시스템에 대해 자세하게 설명하였고, 4장에서는 구현과 실험을 통해 시스템의 평가를 수행하였다. 마지막으로 결론 및 향후 과제를 5장에 기술하였다.

2. 관련연구

본 논문에서 Configuration 은 관리되어야 할 대상이 필요로 하는 구성요소를 '설치'하고, 그것을 각 상황에 맞게 재구성하는 것으로 정의한다.

Configuration작업은 다음과 같이 분류할 수 있다.

- Install : 관리대상이 필요로 하는 구성요소(OS또는 소프트웨어, etc.)를 신규 설치하는 것을 의미한다.
- Reconfiguration : 설치된 구성요소를 각각의 상황에 맞게 재구성 하는 것을 의미한다.
- Update : 애플리케이션의 버전 관리나, 결함을 치유하기 위해 구성요소를 새롭게 수정하는 것을 의미한다. 또한 바이러스나 시스템 오류등에 의해 구성파일이 일부 손상되었을 경우, 치유의 목적으로 재설치 해주는 것을 포함한다.

1) Install

시스템의 구성요소를 인간이 일일이 수동으로 설치하는 것은 많은 시간과 노력이 필요한 작업이다. 특히 관리 대상이 다수일 경우에는 매우 심각한 문제가 될 수 있다.

최근에 이러한 문제점을 해결하기 위해 개발된 툴로 IBM의 Tivoli configuration manager[3]가 있다. 이는 원격설치와 같은 Microsoft의 여러 가지 배포기술[4]을 이용한 도구로써, 관리자는 이를 이용하여, 중앙에서 네트워크에 연결된 관리대상 시스템을 원격 부팅하고, OS나 설치되어야 할 구성요소를 미리 이미지파일로 만들어 두었다가, 이를 전송하여 설치하게 된다. 이를 통해, 거대한 분산 시스템들의 통합관리가 가능해지고, 설치에 드는 많은 시간과 비용을 줄일 수 있게 된다.

그리고 기존에 인간이 일일이 수동으로 구성요소를 설치하는 경우에는 설치가 이루어지는 동안 사용자의 이름이나 사용하는 언어 등, 설치에 필요한 여러 정보를 인간이 일일이 응답하여야 했다 그러나 이 툴에서는 이러한 여러 내용들을 스크립트언어를 이용하여 '자동응답파일'을 작성하여 함께 배포함으로써 '무인설치'가 가능하도록 하였으며, 이와 같이 사용자의 정보를 스크립트 작성에 반영하여 어느 정도의 개인화 설치가 가능하도록 하였다.

그러나 이것만으로는 '개인화'라는 관점에서 부족한 점이 많으며, 사용자는 구성요소의 설치 이후에 또 추가적으로 자신에 맞는 세부적인 환경설정을 해주어야 했다. 본 제안시스템에서는 기존에 저장되어 있던 User data를 기반으로, 개인화된 자동응답파일과 세부적인 환경설정값을 자동으로 작성하여 전송하게 되며, 이를 통해 보다 개인화된 Configuration이 이루어지게 된다.

제안시스템은 이외에도 관리대상의 시스템 리소스(사용가능한 메모리용량)을 Context로 수집하여, 충분한 설치공간이 없을 경우, 구성요소의 최소설치 옵션을 자동으로 선택하여 스크립트를 작성하는 기능도 가지고 있다. 이때, 이는 사용자에게 알리고 최종결정은 사용자가 하게 되며, 에이전트는 사용자의 행동을 관찰하여 학습하고, 이후의 행동을 결정하는데 반영한다. 3절 제안시스템에서 보다 자세한 내용을 소개한다.

2) Reconfiguration

이기종의 다양한 형태의 시스템들은 그 자체적으로도 여러 옵션형태의 다양한 구성을 가지게 되며, 또한 그 안에 설치된 구성 요소들은 수백가지 환경설정 값들을 파라매터로 가지고 있다. 이러한 구성을 환경변화에 따라, 그리고 각각의 사용자에 맞게 개인화하여, 인간이 일일이 변경하는 것은 매우 어려운 일이다. 따라서 각 상황에 맞는 최적의 Setup을 규칙으로 가지고 있다가, 이를 통해 자동 재구성을 수행하는 시스템이 필요하다.

이러한 대응방법은 인공지능 분야에서 오랜 기간 연구되고 있는 규칙기반 전문가시스템의 형태이며, 최근 Context Awareness기능을 가진 여러 미들웨어 분야[5][6]에 적용되어지고 있다.

본 제안시스템은 지속적으로 변화하는 메모리 용량을 Context로 수집하여, 미리 정의된 규칙에 따라 애플리케이션의 설정을 조정하고, 서비스의 질을 조정하는 기능을 갖고 있다. 이를 통해, 모바일 디바이스와 같은 한정된 자원을 가진 환경에서 보다 안정적인 성능을 유지할 수 있게 된다. 또한 불필요하게 리소스를 차지하는 프로세스를 자동으로 종료 시켜주고, 이후에 리소스가 확보되었을 때 이를 다시 재실행해주는 기능도 갖고 있다.

3) Update

각 디바이스의 관리에 익숙하지 않은 초보적인 사용자들은 운영체제의 긴급결함패치나 백신 프로그램의 업데이트의 경우, 그 필요성이 매우 중요함에도 불구하고, 관리를 소홀히 하여 시스템에 치명적인 문제가 발생하기도 한다. 이와 같이 업데이트는 시스템관리에 중요한 역할을 한다. 기존 애플리케이션들의 업데이트방식을 살펴보면, Microsoft사의 Windows update와 같은 다양한 애플리케이션에 대한 통합 업데이트의 경우, 자동 업데이트를 수행하는 에이전트가 주기적으로 작동하여 서버와 통신을 수행하거나, 사용자가 직접 웹사이트를 방문하여 업데이트를 수행한다. 이때 사용자가 각각의 업데이트에 대한 선택을 직접 하는 경우도 있지만, 대부분 통합 업데이트를 수행하게 된다. 따라서 자주 사용하지 않는 애플리케이션에 대한, 불필요한 업데이트도 함께 수행되고, 이에 따라 업데이트에 많은 시간이 소요되기도 한다.

본 제안시스템에서는 사용자의 디바이스에서 작동하는 에이전트가 주기적으로(지속적으로)서버와 통신을 통해, 자동 업데이트를 수행하게 되며, 이때 사용자의 소프트웨어 사용빈도를 Context로 지속적으로 수집하여 저장하고, 이 History를 기반으로 업데이트의 우선순위를 결정한다. 따라서 관리대상 시스템에 남은 저장공간이 부족하거나, 업데이트에 필요한 시간이 부족할 경우, 반드시 필요한 업데이트를 위주로 수행하여 효율적인 업데이트가 가능해진다.

그리고 기존 Configuration시스템들의 파일배포는 대부분 중앙관리서버에 의해 이루어졌기 때문에, 중앙 서버의 과부하가 크고, 서버의 고장에 취약하다는 단점을 가지고 있었다. 이러한 중앙 집중적인 배포방식의 문제점을 해결하기 위한 연구로 Cornell Univ.의 Astrolabe[2]가 있다. Astrolabe는 분산된 관리대상들을 DNS개념을 도입하여 계층적 구조로 구성하고, 해당 존에 존재하는 각 호스트들이 가지고 있는 관리 파일 리스트를 존을 대표하는 호스트가 수집한다. 다음, 어떤 특정한 파일을 필요로 하는 호스트는 이 대표호스트에게 파일을 가지고 있는 가까운 호스트를 질의하고, 위치를 파악하여, Peer-to-Peer형태로 파일을 전송 받는다. 이를 통해, 중앙 집중적인 배포방식의 여러 가지 문제점을 해결하였으며, 본 제안시스템에서는 이러한 Astrolabe의 구조를 반영하여, 이 시스템이 가지고 있는 여러 장점들을 통합하였다.

3. 제안시스템

3.1 설계목표

기존에는 관리자가 다양한 형태의 분산 시스템에 일일이 개별적인 Configuration (Install, Reconfiguration, update)작업을 수행해야 했다. 하지만 이러한 작업은 많은 시간과 비용이 소비되는 일이고, 인간의 많은 노력을 필요로 한다. 이러한 문제점을 해결하기 위해 중앙에서 관리 대상들을 통합 관리하는 연구가 많이 진행되었다. 그러나 아직 많은 부분을 관리자에 의존하고 있고, 관리 대상에 대한 종합적인 Context를 반영한 개인화된 Configuration이 이루어지지 않고 있다.

본 논문에서는 이러한 문제점들을 해결하기 위해, 멀티 에이전트간 협동 작업을 통해, 시스템자원과 사용자의 시스템 사용패턴에 대한 Context를 종합적으로 수집하고, 이를 분석하여 각 상황에 맞는 Configuration 작업을 자율적으로 수행하는 상황-적응형 자가구성 시스템을 제안한다. 이를 통해 기존에 수동으로 이루어지던 여러 작업들의 자동화뿐만 아니라, 보다 개인화된 Configuration 작업이 가능해진다.

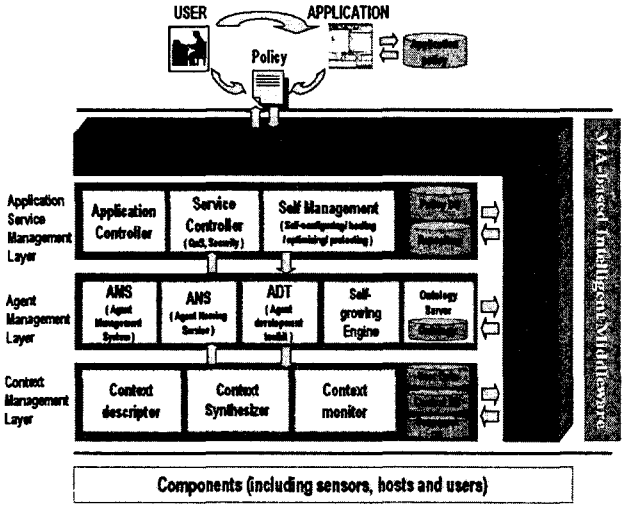
또한 기존의 중앙 집중 배포방식의 여러 단점을 보완한 Cornell Univ.의 Astrolabe 시스템의 장점을 통합하여, 업데이트시 Peer-to-Peer 방식으로 파일을 카피하고, 이를 통해, 중앙서버의 과부하를 줄이고 빠른 파일의 배포가 가능하도록 하였다.

3.2 시스템구성

3.2.1 전체적인 미들웨어

본 연구를 수행하면서 우리는 Ubiquitous Computing환경에 적합한 전체적인 미들웨어 구조부터 설계하였으며, 이중 제안시스템은 Application Service Management 계층에서 지원하는 Self-management, Self-configuring system에 해당하는 부분이다.

전체적인 미들웨어의 구조는 [그림 1]과 같다.



[그림1] 멀티에이전트 기반 지능형 미들웨어의 구조

제안 미들웨어는 전체적으로 3계층으로 나눌 수 있으며, 각 모듈은 지능형 에이전트화 되어있다. 우선 Context Management Layer는 여러 Component로부터 Context정보를 수집하여 분석하고 처리하는 작업을 수행하는 모듈들로 구성되어 있다.

- Context Descriptor(CD) : 각 Component들의 속성, 기능 등 각종 주요 정보를 기술하기 위한 에디터
- Context Monitor(CM) : 각 Component들을 관찰하고 개별적으로 정보를 수집
- Context Synthesizer(CS) : CM으로부터 수집된 각 정보를 통합하고, 추론을 통해 상황을 인식
- User Data : 사용자들에 대한 개인정보나 취향, 사용이력 등이 저장
- Context DB : Context history와 Context Ontology가 저장
- Component DB : CD를 통해 기술된 Component들에 대한 정보를 저장

중간 계층인 Agent Management Layer는 에이전트의 생성과 삭제, 변경, 진화 등 에이전트의 생명주기에 관계하는 모듈들로 구성되어 있다.

- Agent Management System(AMS) : 에이전트가 생성되면 관리대상으로 등록되어, 에이전트의 이동, 삭제 등 생명주기와 기능에 대한 관리가 수행
- Agent Naming Service(ANS) : 생성된 에이전트들에 대한 lookup service를 제공하며, Directory Facilitator(DF)와 동일한 기능을 가짐
- Agent Development Toolkit(ADT) : 에이전트작성을 지원하는 Toolkit으로, JADE, ABLE, DeCAF [7][8][9]등과 유사한 기능을 제공.
- Self-Growing Engine : 에이전트의 자가성장을 위한 추론, 학습기능을 제공.
- Ontology Server : 에이전트간의 상호통신과 협조를 위한 어휘체계를 구축하고 관리.

그리고 Application/Service Management Layer는 애플리케이션과 서비스의 관리에 관계하는 모듈들로 구성되어 있다.

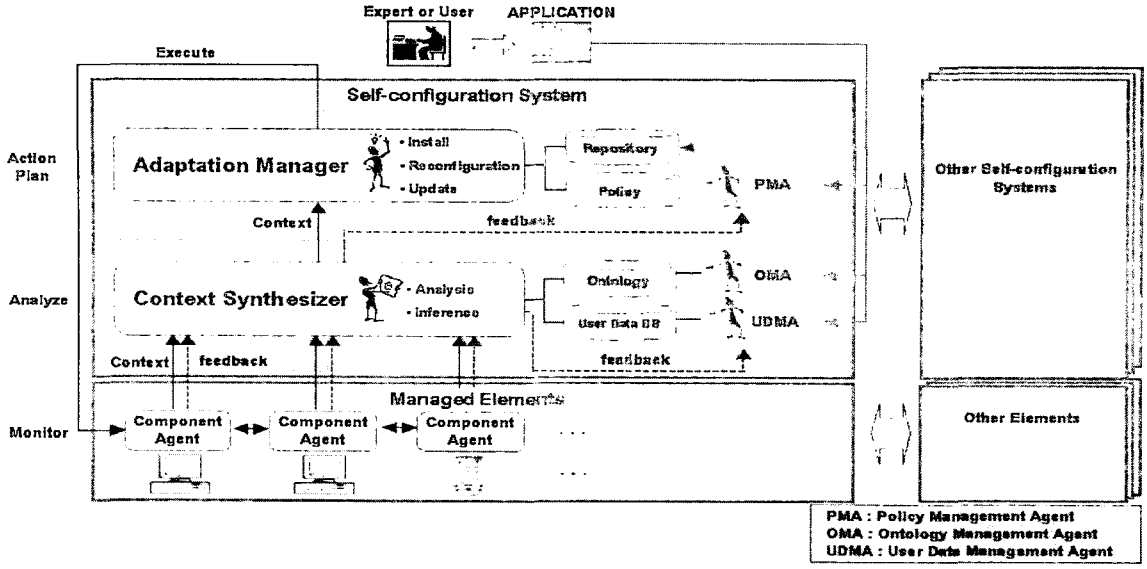
- Application Controller(AC) : 사용자가 사용하는 애플리케이션에 대해, 사용자를 포함한 상황정보에 따라 최적의 사용방안을 제공.
- Service Controller(SC) : QoS나 Security등 시스템에서 제공하는 서비스를 상황을 인식하여 최적으로 제공.
- Self-management(SM) : Self-configuring, healing, protecting, optimizing등의 기능에 기반한 시스템 자가 관리기능의 제공
- Policy Database : 애플리케이션 제공자나 사용자에 의해 작성된 정책을 저장한다.
- Repository Database : 각종 상황별 시스템의 최적버전이나 최신버전 등 시스템구성을 위한 (메타)정보를 저장한다.

3.2.2 상황-적응형 자가구성시스템

앞에서 설명한 것과 같은 전체 미들웨어 구조를 기반으로 설계된 자가구성시스템의 전체구조는 [그림 2]와 같다.

1) Component Agent

관리대상에 설치되어 메모리나 상태변화를 모니터하고, 각 사용자의 애플리케이션의 환경설정 정보, 사용빈도를 체크하여 주기적으로(지속적으로) 서버에 전송하는 역할을 한다. 그리고 install 이나 reconfiguration, update등을 위해 configuration system과 상호작용을 하게 되며, reconfiguration 수행 시에는 각 상황에 맞게 애플리케이션의 옵션을 조정해주는 역할을 한다. 각 애플리케이션의 옵션조정에 대한 규칙과 지식은 처음 애플리케이션이 설치될 때, 애플리케이션과 함께 관리대상에 전송되며 이는 Component Agent가 관리하게 된다.



[그림 2] 제안 시스템의 전체적인 구조

이외에도 Configuration system 또는 같은 존의 호스트로부터 전송 받은 Package나 구성파일의 자동설치를 돕는 역할을 하며, 설치 이후의 사용자의 행동을 관찰하여, 다시 Context Synthesizer에 전송해주는 역할을 한다.

2) Context Synthesizer

Component Agent로부터 전송 받은 Context를 취합하는 역할을 하며, 사용자의 기본정보와 애플리케이션의 환경설정 정보 및 사용빈도를 User Data DB에 저장한다. 이후에 사용자정보와 Ontology(다양한 애플리케이션들이 사용하는 환경설정 정보에 대한 여러 가지 표현들이 저장된다)를 기반으로 현재의 상황을 분석하고 추론하게 되며, 이 결과는 Adaptation Manager 에게 전송된다.

3) Adaptation Manager

Context Synthesizer로부터 받은 현재의 상황에 맞게, 미리 정의된 정책을 기반으로 적절한 행동을 결정하고 수행하게 되며, Repository에 저장된 컴포넌트들을 Install 해주거나, Reconfiguration, Update등의 Configuration 작업을 하게 된다.
이때, 사용자 정보를 기반으로 배포될 설치이미지와 설치에 이용되는 개인화된 '자동응답파일'이 자동 생성된다.

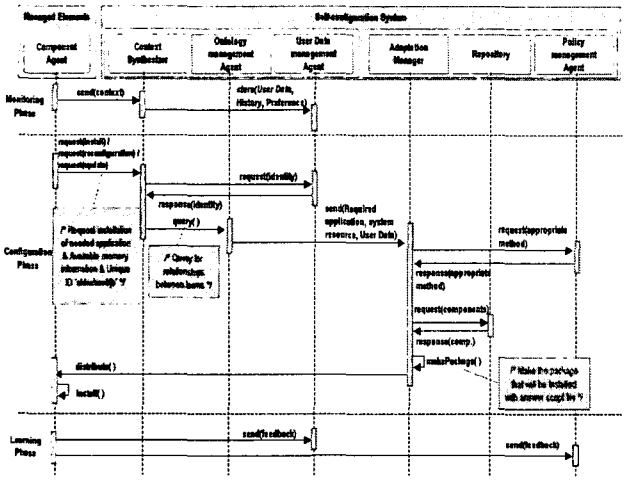
관리자나 사용자는 관리 툴을 이용하여 Repository에 저장되는 컴포넌트와 정책, Ontology, User Data를 조정하고 관리할 수 있다. 그리고 본 제안 시스템은 외부의 다른 Configuration Server들과의 통신을 수행하며, 정보를 공유하게 된다.

3.3 시스템 동작

1) Install

여러 대의 호스트를 중앙에서 관리하는 분산시스템에서 새로운 호스트를 추가하려고 할 때, Microsoft 의 원격설치 기술들을 이용하여 호스트를 추가하는 것만으로 OS의 '무인설치'가 가능해진다. 추가된 호스트는 PXE를 통해 원격부트가 가능하며, 부팅 이후에는 DHCP서버로부터 IP를 부여받고, DNS서버와 Active directory 서버로부터 관리서버를 찾아, 설치될 OS 이미지를 전송받는다.[4]
제안시스템은 여기에 기존에 저장되어 있던 User data(사용자정보, 기존 프로그램 환경설정 정보와 사용패턴을 통해 추론된 사용자 Preference)를 기반으로, 개인화된 자동응답파일과 세부적인 환경설정값을 자동으로 작성하게 된다.
예를 들어 새로운 문서작성 애플리케이션을 설치하는 경우, 기존에 사용자가 다른 유사한 애플리케이션에서 자주 사용했던 폰트를 환경설정의 기본 폰트로 설정해주는 것이다.(Ontology에는 각 애플리케이션마다 같은 의미를 가진 다른 표현들이 저장되며, 이를 이용하여 다른 애플리케이션의 환경설정 내용을 파악한다.) 또한 다른 애플리케이션에서 자동백업주기를 설정한 내용을 파악하여, 새로 설치되는 문서작성 애플리케이션의 환경설정을 해주거나, Windows와 같은 GUI운영체제를 이용하는 사용자의 경우, 최초 설치 시에 바탕화면에 아이콘을 생성하지 않는 등의 행동도 자동스크립트작성에 반영하게 된다.
이와 같이 다른 유사한 애플리케이션의 환경설정값 분석을 통해 사용자의 취향을 파악하고, 이를 반영한 개인화 설정을 수행하게 되며, 이는 앞에서

말한 것과 같이 '자동응답 스크립트' 또는 '환경설정 값'을 직접 작성하여 배포함으로써 수행한다.
이후, 사용자의 행동을 지속적으로 관찰하여, 이러한 설정을 다시 직접 변경하는 경우, 이를 부정적 피드백으로 파악하고, 정책을 조정하는 Reinforcement learning[10]을 수행한다.
그리고 관리대상의 시스템 리소스(사용가능한 메모리 용량)를 Context로 수집하여, 충분한 설치공간이 없을 경우, 구성요소의 최소설치 옵션을 자동으로 선택하여 스크립트를 작성하는 기능도 갖고 있다. 이때, 이는 사용자에게 알리고, 최종결정은 사용자가 하게 되며, 에이전트는 사용자의 행동을 관찰하여 학습하고, 이후의 행동을 결정하는데 반영한다.
이후, 사용가능한 메모리가 확보되었을 때, 이를 파악하여, 전체설치를 제안하고, 이후의 사용자 행동도 관찰하여 학습을 수행한다.

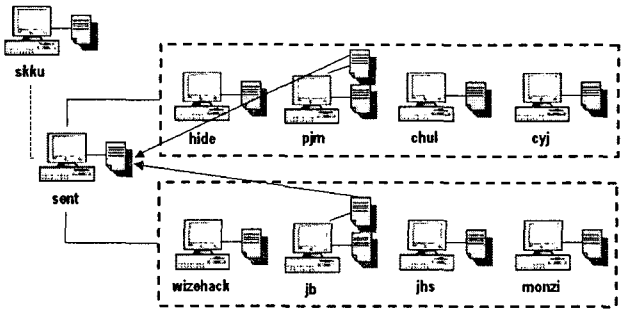


[그림 3] 제안시스템의 시퀀스 다이어그램

2) Reconfiguration
제안시스템은 지속적으로 변화하는 메모리 용량을 Context로 수집하여, 미리 정의된 규칙에 따라 애플리케이션의 설정을 조정하고, 서비스의 질을 조정한다. 이를 통해 모바일 디바이스와 같은 한정된 성능을 가진 환경에서 보다 안정적인 성능을 유지할 수 있게 된다.
각 애플리케이션의 옵션조정에 대한 규칙과 지식은 처음 애플리케이션이 설치될 때, 애플리케이션과 함께 관리대상에 전송되며, 이는 Component Agent가 관리하게 된다.
또한 Component Agent는 리소스 관리를 위해, 불필요하게 리소스를 차지하는 프로세스를 자동으로 종료시켜주고, 이후에 리소스가 확보되었을 때, 이를 다시 재실행해주는 기능도 갖고 있다. 이때 종료시키는 프로세스의 우선순위는 리스트형태로 가지고 있으며, 이는 사용자의 애플리케이션 사용빈도와 프로세스들의 관련성을 파악하고, Configuration system과 주기적인 상호작용을 통해 보다 최적화된 규칙으로 항상 업데이트 된다.

3) Update
Component Agent는 사용자의 각 애플리케이션 사용빈도를 Context로 지속적으로 수집하여 User Data DB에 저장한다. 그리고 주기적으로(지속적으로) 서버와의 통신을 통해, 자동 업데이트를 수행하게 되는데, 이때 이 History를 기반으로 업데이트의 우선순위를 결정하여 수행하게 된다. 따라서 관리대상 시스템에 남은 저장 공간이 부족하거나, 업데이트에 필요한 시간이 부족할 경우, 효율적인 작업이 가능하게 된다.

제안시스템은 업데이트의 또 하나의 기능으로, 바이러스나 시스템 오류등에 의해 구성파일이 손상되었을 경우, 치유의 목적으로 일부 파일을 재설치 해주는 기능을 갖고 있다. 손상된 구성파일은 이벤트 메시지의 분석에 의해 파악하며, 파일의 카피는 Astrolabe 시스템[2]의 구조를 반영하여, 가까운 호스트에서 Peer-to-Peer 형태로 전송받는다.
관리대상들은 [그림 4]와 같이 계층적 구조로 구성되며, 각각 유니크한 URL형태의 ID값을 갖게 된다. 그리고 해당 존에 존재하는 호스트들이 가지고 있는 관리 대상 파일들의 리스트를 존을 대표하는 호스트가 수집한다. 다음, 파일을 필요로 하는 호스트는 대표 호스트에게 파일을 가지고 있는 가까운 호스트의 위치를 질의하게 되고, 위치를 파악하여 Peer-to-Peer 형태로 파일을 전송 받는다. 이를 통해, 중앙관리서버의 부하를 줄이고, 고장에 강한 특성을 가지며, 보다 빠른 카피가 가능하다.



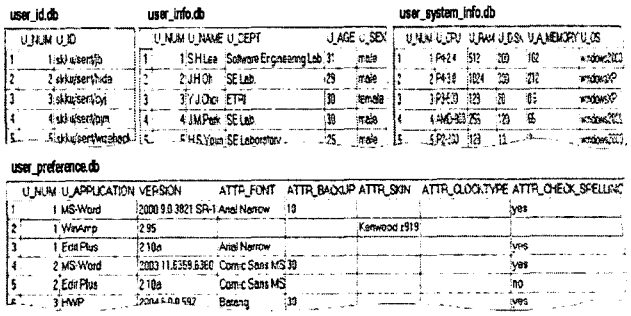
[그림 4] Peer-to-Peer전송을 위한 관리 대상들의 계층적 분류

4. 시스템 구현 및 평가

4.1 시스템 구현
본 시스템의 평가를 위해 개발된 프로토타입의 각 모듈은 주로 자바를 이용하여 개발되었으며, DB는 Oracle 9i를 이용하였다. 각 관리 대상은 Microsoft의 대표적인 OS인 Windows 2000과 XP를 운영체제로

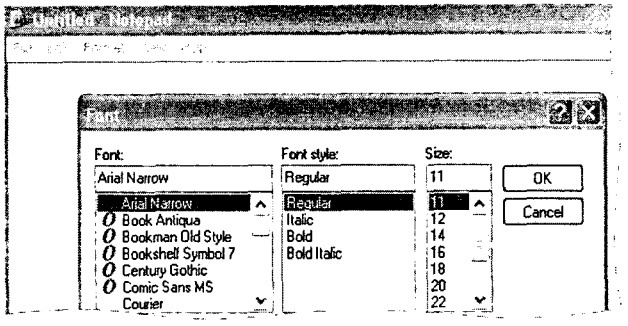
사용하는 PC를 대상으로 하였으며, Context는 jdk1.4의 JNI를 이용하여, 주로 Windows의 registry와 각각의 애플리케이션들이 정보를 담기 위해 독자적으로 생성하는 *.ini 파일을 통해 수집하였다. 자동응답 스크립트는 Adaptation manager를 통해 VBScript로 생성되어 Install 될 구성요소와 함께 전송된다.

User data DB에 저장되는 정보는 [그림 5]과 같으며, 사용자를 구별하는 user_id.db와 사용자의 신상정보를 수집한 user_info.db가 있다(이는 애플리케이션의 설치 시에 주로 입력하는 사용자 정보이다). 그리고 사용자의 시스템리소스를 담고 있는 user_system_info.db와 각종 환경설정정보를 수집한 user_preference.db가 있다. 현재 user_preference.db의 정보는 유사한 애플리케이션의 환경설정을 해주는데 그치지만, 향후에는 사용자의 취향을 보다 폭 넓게 추론하는데 이용할 수 있을 것이다.



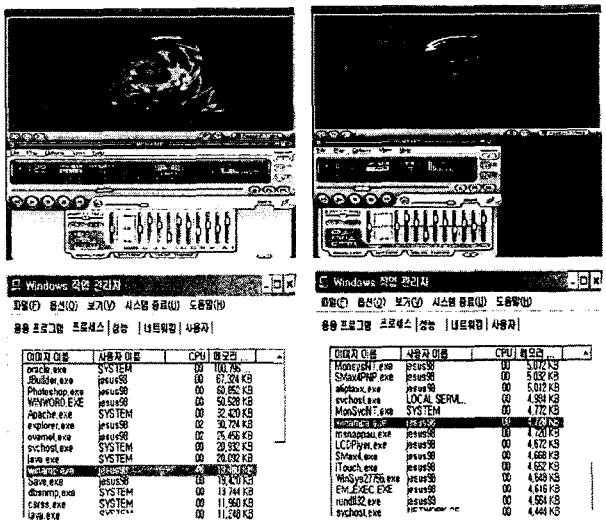
[그림 5] User Data DB에 수집된 정보

위와 같이, 수집된 '사용자에 대한 Context'와 '시스템에 대한 Context'를 기반으로 install을 수행한 예제가 [그림 6]에 나와 있다. 새로 설치되는 Notepad를 User Data DB에 저장된 사용자의 Preference(ex.기존 유사한 애플리케이션에서 주로 사용하는 폰트설정과 폰트사이즈설정 정보)정보를 기반으로 폰트와 사이즈를 변경하는 간단한 환경설정을 수행하였으며, 이는 레지스트리의 IfFaceName와 iPointSize라는 값의 변경을 통해 이루어졌다.



[그림 6] Install과 동시에 사용자 Context를 반영한 환경설정이 이루어진 애플리케이션

Reconfiguration은 지속적으로 변화하는 System context(사용가능한 메모리 용량)를 기반으로, 미리 정의된 임계값과 규칙에 따라 Component Agent가 실행되는 애플리케이션의 옵션을 조정하게 된다. 각 애플리케이션의 옵션조정에 대한 규칙과 지식은 처음 애플리케이션이 설치될 때 관리대상에 전송되며, 이는 Component Agent가 관리하고 실행하게 된다. 미리 정의된 메모리 임계값에 따라 애플리케이션의 옵션을 조정하여 메모리를 확보하는 그림이 [그림 7]에 나와있다.



[그림 7] 시스템 Context (사용가능한 메모리 용량)을 인식하여 구성요소의 옵션을 조정하고 메모리를 확보하는 Reconfiguration 기능의 예제

그리고 업데이트의 효율성을 테스트하기 위해, 우리는 임의로 업데이트 파일을 10개 지정하고, 각각의 사용빈도를 입력하였다. Component Agent는 주기적으로 서버와 해당 업데이트 파일의 버전을 비교하고, 자신이 속한 Zone에서 해당 파일을 가지고 있는 호스트를 찾아 Peer-to-Peer 방식으로 파일의 카피를 수행하였으며, 이때 사용빈도에 따라 업데이트의 순서를 조정하였다.
Zone내에서 카피되는 업데이트 파일의 위조여부는 사이즈 비교를 통해 수행하였다.

4.2 시스템 평가

시스템의 평가는 A 방식(분산된 각 관리대상에 구성요소를 사람이 일일이 수동으로 설치하는 경우)과 B방식(MS-IBM과 같은 네트워크를 통한 원격관리 시스템), 그리고 C방식(Context awareness 기능을 가진 본 논문의 제안시스템)을 기능적 측면과 작업에 소요되는 시간에 대해 비교하여 수행하였다.
먼저 각 방식에 대한 기능적 비교가 <표 1>에 정리되어 있다.

<표 1> 세가지 Configuration 방식에 대한 기능적비교

	Function	A	B	C
Characteristics	Install	Customized Install	Δ	X	O
	reconfiguration	QoS, Resource management	Δ	X	O
	update	Update by priority	Δ	X	O
		P2P	X	X	O

	Function	A	B	C
Admin'e burden	Unattended Install	X	O	O
	Central management	X	O	O

다음, install에 걸리는 작업시간의 비교를 통해 각각의 방식을 비교하였다.
설치될 애플리케이션은 510Mbyte의 MS Office이며, (스크립트 및 환경설정파일은 1Mbyte이하) 설치시간은 관리대상의 성능에 따라 평균 8분이 소요되었다. 일대일 전송시간은 LAN을 이용하여 회당 평균 80초가 소요되었다.
평가에 필요한 요소는 다음과 같으며, 평균 설치시간은 식 (1) 과 같다.

- install time : *install_time*
- transport time : *transport_time*
- the number of managed element : *N*

$$Average\_install\_time = \frac{Install\_time_1 + Install\_time_2 + \cdots Install\_time_n}{N} = \frac{\sum_{i=1}^{N} Install\_time_i}{N} \quad (1)$$

A 방식 : 각 관리대상에 관리자가 일일이 애플리케이션을 설치하는 방식을 말하며, 관리대상이 늘어날수록 더 많은 시간과 비용, 관리자의 노력이 요구된다. 작업시간은 식 (2)을 이용하여 계산한다.

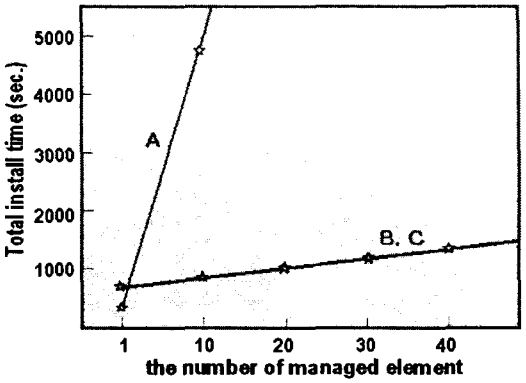
$$Total\_install\_time = Average\_install\_time \times N \quad (2)$$

B방식, C방식 : 중앙에서 네트워크를 이용하여 원격으로 애플리케이션을 전송하는 방식을 말한다. 식 (1) 에 전송시간(transport_time)과 네트워크 과부하에 대한 상수 값 α가 추가되어 계산되며, 관리대상이 적을 경우는 오히려 수동방식이 작업시간이 더 적게 걸린다. 그러나 관리대상이 많을 경우에는 시간이 줄어들고, 멀티캐스트 방식을 이용하여, 빠른 배포가 가능하다.

$$Total\_install\_time = Average\_install\_time + \alpha \cdot transport\_time \quad (3)$$

세 방식의 install_time 비교를 수행한 그래프가 [그림 8]에 나와 있다.

이와 같이 제안시스템은 다수의 분산 관리대상을 관리함에 있어서, Context를 반영한 세부 개인화설정을 수행하면서도, 중앙에서 분산 관리대상을 통합 관리하는 관련시스템과 속도면에서 같은 성능을 보인다. 또한 Update의 경우에는, 중앙 집중 배포방식이 아닌 Peer-to-peer 방식의 카피를 수행함으로써, 서버의 과부하에 따른 속도지연이 줄어들어, A, B방식에 비해 훨씬 빠른 업데이트가 가능하다.



[그림 8] 세 방식의 Install time 비교 그래프

5. 결론 및 향후과제

제안시스템은 기존에 수동으로 이루어지던 Configuration작업을 멀티에이전트간의 작업을 통해, 자율적으로 이루어지도록 하였으며, 인간의 개입을 최소화하여 관리자의 작업부하를 줄이게 된다.

그리고 기존의 중앙 집중 배포방식의 여러 단점을 보완한 Cornell Univ.의 Astrolabe 시스템의 장점을 통합하여, Update시 Peer-to-Peer 방식으로 파일을 카피하고, 이를 통해, 중앙서버의 과부하를 줄이고 빠른 파일의 배포가 가능하도록 하였다.

이러한 자동화와 함께 System context와 User context를 반영하여 차별화된 Configuration 작업을 수행하는 본 제안시스템의 특징은 사용자의 usability와 만족도를 높이고, 유비쿼터스 환경에서 인간의 시스템 관리에 대한 부담을 줄여주어 보다 편리한 컴퓨팅 환경이 가능하게 해줄 것으로 기대된다.

향후 연구 방향과 연구과제로는 다음과 같은 내용이 필요하다.

- 유비쿼터스 환경에서 일어나는 다양한 Context를 활용한, 보다 Adaptive한 자동시스템관리
- Context의 수집과 적용, 관리를 위한 보다 효율적인 알고리즘에 관한 연구
- 에이전트간에 전송되는 Context에 대한 보안과 사용자의 프라이버시 보호

- 유사한 사용자들의 Clustering을 통해 존을 구성하고, 업데이트나 애플리케이션에 대한 피드백을 공유, 이를 통한 보다 차별화된 Configuration 작업

참고문헌

[1] Paul Horn, (2001). "Autonomic Computing : IBM's Perspective on the State of Information Technology", IBM White paper

[2] Robbert van Renesse, Kenneth Birman and Werner Vogels, (2003). "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining", ACM Transactions on Computer Systems, Vol.21, No.2, pp.164-206

[3] http://www-306.ibm.com/software/tivoli

[4] http://www.microsoft.com/technet/prodtechnol/winxppro/deploy/default.mspx

[5] John Keeney, Vinny Cahill, (2003). "Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework", In Proceedings of the Fourth IEEE International Workshop on Policies for Distributed Systems and Networks

[6] Anand Ranganathan, Roy H. Campbell, (2004). "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments", In ACM/IFIP/USENIX International Middleware Conference 2004

[7] J.P.Bigus, D.A. Schlosnagle, J.R.Pilgrim, W.N.Mills III, and Y.Diao, (2002). "ABLE: A toolkit for building multiagent autonomic systems", IBM Systems Journal Artificial Intelligence, Vol.41, No.3

[8] http://jade.tilab.com/doc/index.html

[9] http://www.eecis.udel.edu/~decaf

[10] Richard S. Sutton, Andrew G. Barto, (1998). : Reinforcement Learning : An Introduction (Adaptive Computation and Machine Learning), The MIT Press