유비쿼터스 컴퓨팅 환경을 위한 액티브네트워크상의 문맥인식성을 고려한 자치 적응성 서비스

(Self-adaptation Service with Context-awareness on Active Network for Ubiquitous Computing Environment)

홍 성 준 † 한 선 영 ††
(Sungjune Hong) (Sunyoung Han)

요 약 최근 유비쿼터스 컴퓨팅 환경의 등장에 따라, 사용자의 자주 변화하는 제약조건에 맞는 서비스를 제공하기 위해서 망 내에서의 문맥인식성을 고려한 자치적응성(self-adaptation)이 요구되고 있다. 그러나 기존 망에 자치적응성 기능을 추가하고자 할 때 느린 망 표준화 문제와 느린 서비스 재배치 문제가 발생한다. 액티브네트워크는 이러한 문제점을 해결하기 위한 적합한 환경으로써 망에 새로운 자치 적응성을 추가하고 빠른 재배치를 할 수 있다.

그러므로 본 논문은 문맥인식성을 고려한 자치적응성 지원을 위해서 에이전트 기반 액티브네트워크와 제약조건 기반 SCE(Service Creation Environment)를 이용한 SAS(Self Adaptation Service)를 제안하였다. 본 SAS의 이점은 망 내에서 문맥인식(context-aware)을 고려한 서비스 지원과 빠른 서비스 재배치 지원이다.

키워드 : 액티브네트워크, 자치적응성

Abstract A self-adaptation with context-awareness is needed within network to meet customed services according a user's changing constraints. But the existing network has many difficulty in adding new functions because of slow standardization of network and slow deployment of new services. To solve this problem, an active network can support the suitable environment to add new function such as self-adaptation.

Therefore, this paper suggests Self Adaptation Service(SAS) using agent-based active network and the constraint-based Service Creation Environment(SCE) to support self-adaptation with context-awareness. SAS provides benefits to support the context-aware service and the fast deployment of new services.

Key words : Active Network, Self-adaptation

1. 서 론

현재 인터넷 응용의 컨텐츠 적응성은 너무 웹 서버에 의존적이어서 위치정보, QoS정책, 디바이스의 유형과 같은 정보를 지원하지 못하는 단점이 있다. 이러한 단점을 보완하기 위해서 W3C의 CC/PP[1]는 XML/RDF 등을 이용하여 확장된 웹 서버 상의 컨텐츠 적응성을 지원하기 위한 표준화가 진행 중에 있고, IETF의 OPES [2]는 웹 서버가 아닌 에지(edge)망에서 컨텐츠 적응성을 지원하려는 표준화 시도가 진행 중에 있다. 그러나 CC/PP나 OPES 등의 시도는 망 내에서 문맥인식을 고려한 자치적 적응성 고려가 부족하다. 최근 유비쿼터스 컴퓨팅 환경이 등장함에 따라 사용자의 자주 변화하는 제약조건에 맞는 서비스를 제공하기 위해서 망 내에서 문맥인식성(context-awareness)[3]을 고려한 자치적응성(self-adaptation)이 요구되고 있다. 그러나 기존 망에서 자치적응성 기능을 망에 추가하고자 할 때 느린 망 표준화 문제와 느린 서비스 재배치 문제가 발생한다.

· 본 논문은 2004년도 한국전산원 연구비지원(과제번호 : 2004-협약-위05)에 의하여 수행되었음
† 정 회 원 : 여주대학 정보통신과 교수
sjhong@mail.yeojoo.ac.kr
†† 종신회원 : 건국대학교 컴퓨터공학부 교수
(corresponding author임)
syhan@cclab.konkuk.ac.kr
논문접수 : 2003년 11월 28일
심사완료 : 2004년 8월 2일

액티브네트워크는 이러한 문제점을 해결하기 위한 적합한 환경으로써 망에 새로운 자치 적응성을 추가하고 빠른 서비스 재배치를 할 수 있다. 그러므로 본 논문에서는 액티브네트워크와 제약조건 기반 SCE(Service Creation Environment)[4]를 이용한 SAS(Self-Adaptation Service)를 제안하였다. 여기서 SAS는 사용자가 사용하는 무선 이동 단말기 등의 제약조건이 바뀌어도 빠르게 그 해당 제약조건에 맞는 서비스를 제공할 수 있는 액티브네트워크 상의 자치적응성 서비스을 의미한다. 그리고 향후 유비쿼터스 컴퓨팅 환경에서의 RFID나 무선 센서 망과 연동을 통해서 망 내의 자치적응성을 통한 문맥인식성 서비스로 발전 시켜나갈 수 있다. 여기서 문맥인식성은 사용자에게 관련된 정보나 서비스를 제공하기 위해서 문맥을 사용하는 것을 의미하며, 문맥은 어떤 사람이나 사물이나 객체의 상황적인 특성을 표현하기 위해서 사용되는 정보를 의미한다.

본 논문의 목적은 액티브네트워크상에서 자치적응성 서비스의 개발이다. 본 SAS의 이점은 첫째, 자치 적응성을 지원함으로 망 내에서 문맥인식성을 고려한 서비스를 지원할 수 있고, 둘째, 망에 자치 적응성을 빠르게 재배치(deploy)할 수 있다.

본 SAS를 위한 액티브네트워크 계층 구조는 연결계층, 제어 계층, 모델링 계층 구조로 구성된다. 연결 계층은 기존의 TCP/IP를 이용하고, 제어 계층은 액티브네트워크 중에 하나인 ALAN(Application Level Active Network) [5][6][7]을 확장하여 이용한다. 그리고 모델링 계층은 제약조건 기반 SCE(Service Creation Environment)를 이용한다.

확장된 ALAN을 살펴보면, 자바 애플릿 기반의 기존 ALAN은 Jade[8]라는 이동/지능 에이전트(mobile/intelligent agent)[8]를 이용하여 확장되었다. 제어계층에서 액티브네트워크를 이동 /지능 에이전트로 통합 확장한 이유로는 액티브네트워크는 망에서 지능성을 추가하려는 접근 방법은 있으나 현재 자치 적응성과 같은 지능성 지원이 부족한 단점이 있다. 반면에 지능 에이전트는 서버와 단말기 측면에서만 지능성 지원이 고려되고 있고 망 차원 망의 지능성에 대한 고려가 부족한 단점이 있다. 이러한 문제를 상호 보완하기 위해서 기존 액티브네트워크를 이동/지능 에이전트기반의 액티브네트워크로 확장하여 사용하였다. 기존 ALAN의 접근 방법은 망의 하위 계층에 영향을 미치지 않고 더욱 빠른 서비스 배치가 이루어 질 수 있는 장점이 있다. 기존 ALAN은 액티브네트워크(Active Network)의 특성과 프락시릿(Proxylet)이라고 불리는 이동 에이전트의 특성을 동시에 가지면서 망내에서 자치 구성성(self-configuration)을 지원한다. 그러나 기존 ALAN의 프락시릿이 자바 애플릿과 같은 형태로 지원되고 있기 때문의 자신의 상태 유지 기능 및 지능성 기능이 부족하다.

액티브네트워크에 적용된 제약조건 기반 SCE를 살펴보면, 원래 SCE는 전화망 기반의 지능망에서 유래된 것으로 망의 지능성을 제공하지만 전화망에 의존적이고 비 개방적인 구조로 외부개발자(Third Party)의 참여가 어려운 단점이 있다. 또한 사용자의 변화하는 제약조건에 대한 고려가 없다. 이를 위해서 본 제약조건 SCE도구는 개방적인 구조로 UML/OCL기반의 GME(Generic Modeling Environment)[9][10]를 이용하였다. GME관련 프로젝트에서는 TAO[11]라는 미들웨어에 GME를 적용하는 시도로서 CoSMIC(Component Synthesis using Modeling Integrated Computing)[12]이라는 GME 도구를 개발하고 있다. 본 연구와 차이점으로 CoSMIC은 CORBA상에서 QoS지원을 하는데 초점을 두고 있고, 망 내의 액티브네트워크 상에서 사용자의 자주 변하는 제약조건과 ISP(Internet Service Provider) 서비스 규칙을 처리하는 자치적응성에 대한 고려가 부족하다. 반면에 본 논문은 제약조건 SCE를 액티브네트워크에 적용하였고 망내의 자치 적응성을 지원한다.

그러므로 본 논문은 SAS를 위한 액티브네트워크 망 구조와 SAS의 설계 및 구현에 관하여 언급하였다.

2. SAS 시나리오 및 기존 연구와 비교

2.1 SAS 시나리오

그림 1은 본 논문에서 전제하고 있는 시나리오를 소개하고 그 시나리오에 따라 자치 적응성을 지원하는 접근 방법을 보이고 있다. 시나리오를 살펴보면, 사용자가 온라인 게임을 유선상의 PC에서 즐기다가 밖으로 나가서 계속 온라인 게임을 즐기기 원할 때, 사용자의 디바이스가 유선상의 PC에서 무선 이동 디바이스로 변경될 필요가 있다. 이때, 사용자는 단순히 유선 PC에서 무선 이동 디바이스의 변경을 간단히 버튼을 누르는 조작을 통하여 하고, 사용자는 연속적으로 무선 이동 디바에스에서 온라인 게임을 즐긴다.

그림 1의 왼편에 사용자 측에서는 유선상의 PC와 무선 이동 디바이스에 이동/지능 에이전트가 내장되어 있다고 가정한다. SAS를 포괄적으로 살펴보면, 이동/지능 에이전트는 사용자의 변화되는 제약조건을 인식할 수 있고 그러한 변화되는 제약조건을 액티브네트워크상의 SAS에 전달한다. 액티브네트워크상의 SAS는 이러한 제약조건에 맞게 서비스를 재조합하여 사용자에게 결과를 보낸다. 사용자는 결과적으로 자신의 선호도와 제약조건에 적합한 문맥인식성 지원 서비스를 받는다. 본 SAS는 확장된 ALAN상에서 제약조건 기반의 SCE를 이용한다. 제약조건 SCE는 사용자의 제약조건과 ISP나

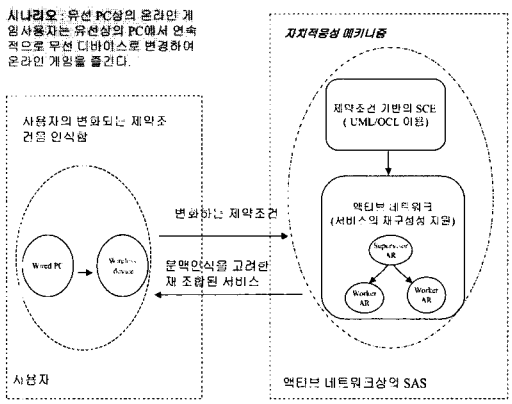

그림 1 SAS의 시나리오

네트워크 관리자의 망 정책 및 서비스 규칙 등을 UML (Unified Modeling Language)과 OCL(Object Constraint Language)[13]을 이용하여 표현하고 액티브네트워크에 반영시킨다. 액티브네트워크상의 SAS는 Supervisor AR(Active Router)와 Worker AR로 구성되어 있다. Supervisor AR은 요청라우팅 기능을 담당하고 Worker AR은 분산 기능과 자치 적응성 기능을 지원한다.

2.2 기존 연구와 확장된 ALAN상의 SAS의 비교

표 1은 기존 웹 서버상의 서비스와 더불어 기존 ALAN상의 네트워크 적응성 연구[6]와 확장된 ALAN상의 SAS의 특징을 비교하였다. 기존 웹 서버상의 서비스는 TCP/IP기반의 인터넷 및 HTTP를 기반으로 하는 웹상의 서비스로서, 수동구성으로 관리되고 있다. 기존 ALAN상의 네트워크 적응성 연구는 망의 수동 구성성의 문제점을 해결하기 위해서 자치 구성성이 주요 관심 분야이다. 또한 망의 자치 적응성을 지원을 위한 메카니즘이 일부 소개되었지만 단순한 제약조건 검증 도구와 수학적이고 이론적인 접근 방법에만 머무르고 있다. 자치 구성성과 자치 적응성의 차이점을 살펴보면, 자치구성성은 망 관리자 입장을 고려한 망 구성의 자동화를 의미하고 자치적응성은 사용자 입장을 고려하여 망의 구성요소가 재조합되어 사용자에게 보다 적합한 서비스를 제공하는 것을 의미한다. 기존 ALAN상의 프락시릿은 자바 애플릿으로 구성되어 있기 때문에 상태유지 기능 및 지능성 기능이 부족하고, 제약조건 기반 메카니즘은 Alcoa/Alloy[14]라고 불리는 제약조건 검증 도구를 제시하여 단지 제약조건 검증 만 가능하고 실제 구현은 어렵다. 기존 ALAN의 통신 프로토콜은 자바기반의 RMI(Remote Method Invocation)를 이용한다.

반면에 SAS는 액티브네트워크상에서 액티브어플리케이션(Active Application)이며 사용자의 변화하는 제약

표 1 기존 웹 서버상의 서비스와 기존 ALAN상의 네트워크 적응성 연구[6] 그리고 확장된 ALAN상의 SAS의 비교

	기존 웹 서버상의 서비스	기존 ALAN상의 네트워크 적응성 연구	확장된 ALAN상의 SAS
시스템 개요	인터넷 및 웹	응용수준 액티브네트워크 (ALAN)	에이전트 기반 응용수준 액티브네트워크
서비스	수동 구성성 서비스	자치구성성 있는 서비스	자치 적응성 있는 서비스
프로토콜	HTTP	RMI(Remote Method Invocation)	ACL(Agent Communication Language)
제약조건 지원 도구		Alcoa/Alloy	UML/OCL기반 GME
이동코드 (proxylet)	-	자바애플릿	이동/지능 에이전트

조건에 대해서 빠르게 대응 할 수 있는 자치 적응성 서비스를 의미한다. 또한 SAS를 위한 프락시릿은 이동/지능 에이전트 기반 액티브네트워크로 구성되었기 때문에 지능성 지원이 가능하고, 제약조건 기반 SCE를 이용하므로 모델링 및 제약조건 검증과 더불어 실제 에이전트 기반의 액티브네트워크에 적용 구현이 용이하다. 제약조건 지원 도구는 UML/OCL기반의 GME를 이용하였다. SAS의 통신 프로토콜은 이동/지능에이전트의 ACL (Agent Communication Language)[15]라는 통신 프로토콜을 사용한다.

3. SAS의 설계

3.1 SAS의 구조

그림 2는 SAS를 위한 액티브네트워크의 망 계층과 SAS의 내부구조를 보이고 있다. SAS를 위한 액티브네트워크 망 계층 구조는 연결 계층, 제어 계층, 모델링 계층으로 나뉘어 진다. 연결 계층은 기존의 TCP/IP를 이용하고, 제어계층은 모델링 계층에서 모델링 된 서비스를 실현하는 계층으로 이동/지능 에이전트로 확장된 ALAN을 이용한다. 모델링 계층의 SCE는 *Constraint*와 *Service Rule* 기능이 정의되어 있다. *Constraints*에서 UML/OCL기반의 GME를 이용하여 사용자의 변화하는 제약조건을 정의하고, *Service Rule*에서 ISP의 서비스를 정의하고 컴포넌트들을 재조합하는 서비스 규칙을 정의한다. SAS의 내부 구조는 Supervisor AR과 Worker AR로 구성되어 있다. Supervisor AR은 Select_the_Fast_AR과 Request_Routing으로 구성되어 있고 사용자와 가장 가까운 Worker AR를 찾아 요청라우팅하는 기능을 수행한다. 여기서 사용자와 가장 가

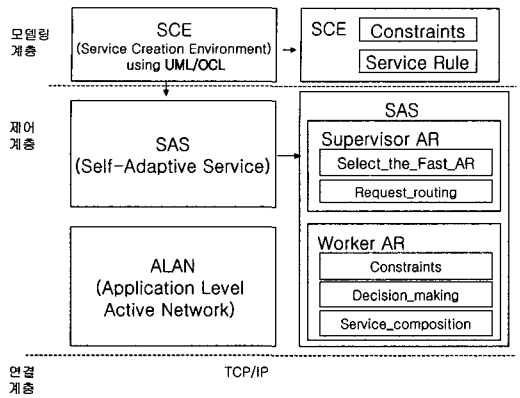

그림 2 SAS의 구조

깝다는 의미는 사용자에게 서비스를 할 수 있는 가장 적합한 대역폭을 제공할 있는 위치의 Worker AR을 찾는다는 의미이고, 이 방식은 웹 캐싱 서버가 여러 중복된 캐싱 서버 중에서 사용자와 가장 가까운 중복 서버를 찾는 방법과 유사하다. Worker AR은 Constraint, Decision_Making 그리고 Service_Composition으로 구성되어 있고 사용자의 요구사항에 적합한 자치적응성 기능을 수행한다.

3.2 SAS의 동작

그림 3은 SAS의 동작을 보이고 있다. 액티브네트워크 상의 SAS는 모델링 계층의 제약조건 기반 SCE에서 GME 도구를 이용하여 정의되어 있고, 제어계층의 확장된 ALAN에서 Supervisor 액티브 라우터, Worker 액티브 라우터가 각 기능을 수행한다. 모델링 계층에서는 GME를 이용하여 SAS(Self-Adaptive Service)라 명명된 서비스가 정의되었다. ①에서 정의된 SAS내의 *Constraints*와 *Service Rule*에서 사용자의 제약조건과 ISP의 서비스 규칙을 명시한다. 이 모델링된 SAS는 제어계층에서 실현된다. ②에서 무선 이동 단말기에 내장된 이동/지능 에이전트는 사용자의 변화되는 제약조건을 인식하고 그 제약조건을 Supervisor AR에 전달한다. ③에서는 Supervisor 액티브 라우터가 사용자와 가장 가까운 Worker 액티브 라우터를 검색하고 선택한다. ④에서 Supervisor 액티브 라우터가 선택한 Worker 액티브라우터로 메시지 요청을 라우팅시킨다. ⑤에서 선택된 Worker 액티브 라우터에서는 사용자의 제약조건과 ISP의 서비스 규칙에 적합한 서비스를 재조합하고 서비스 재조합된 결과를 이동 터미널 에이전트(Mobile Terminal Agent)에게 보낸다. 기존 ALAN과 비교해 보면, 기존 ALAN은 동적 프락시릿(Dynamic Proxylet) 서버, 프락시릿(Proxylet), 프락시릿(Proxylet) 서버, 프로토콜(Protocol) 서버 등으로 구성되어 있다. 동적 프락시릿 서버는 프락시릿을 로딩하여 수행하는 서버로서 EEP (Execution Environment for Proxylet)으로 이름이 변경되었다. EEP상에 로딩되어 수행되는 프락시릿은 기존에는 자바 애플릿이었으나 본 논문의 프락시릿은 이동/지능 에이전트로 확장하여 이용되었다. 프락시릿은 일종의 이동코드이다. 프로토콜 서버는 프로토콜 스택의 저장소를 의미하는데 본 논문은 Worker AR에서 이러한 기능을 담당하고 주로 WAP(Wireless Application Protocol)이나 RTP(Real-Time Protocol)와 같은 통신 프로토콜을 지원하는 프락시릿이 이러한 기능을 담당한다. 그리고 프락시릿 서버의 주 기능은 프락시릿을 공유하고 프락시릿을 검증하는 기능을 한다. 본 논문은 Worker AR에서 이 기능을 담당한다.

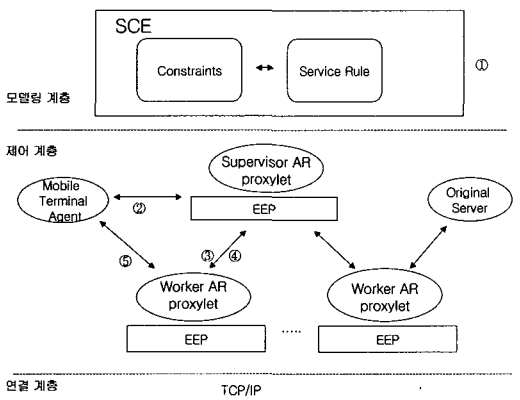

그림 3 액티브네트워크상의 SAS 동작

3.3 SAS의 메카니즘

그림 4는 자치 적응성 메카니즘에 대한 구조를 보이고 있다. 자치 적응성 메카니즘은 SAS 내의 Worker AR 프락시릿이라고 명명되었다. 본 Worker AR 프락시릿은 Constraint, Decision-Making, Service Composition 기능으로 구성되어 있다. Constraint는 사용자의 제약조건을 명시하는 도메인, 변수, 제약조건, 규칙 등을 명시한다. Decision_Making기능은 ISP의 서비스 재조합 정책을 위한 서비스 규칙을 명시한다. Service_Composition은 다양한 프락시릿들이 정의되어 있고 Decision_Making에서 결정된 정책에 따라서 프락시릿 재조합 수행 기능을 한다.

그림 5는 사용자 제약조건을 명시하기 위해서 도메인(Domain), 변수(Variable), 제약조건(Constraint), 그리고 규칙(Rule)을 정의하였다. 도메인에는 사용자를 위한 *Subscriber Profile*, ISP와 네트워크 관리자를 위한 *Service Profile* 그리고 *Service Policy Profile*이 정의되어 있다. *Subscriber Profile*은 사용자의 선호도, 사용자가 사용하는 단말기 유형, 사용자가 사용 중인 네트

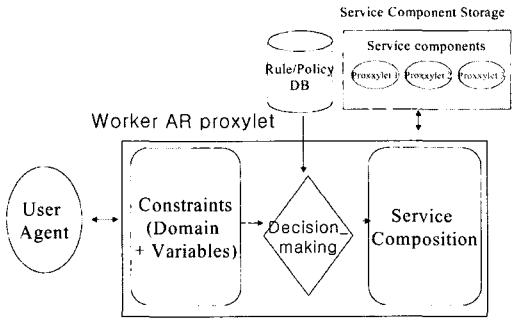

그림 4 자치적응성 메카니즘

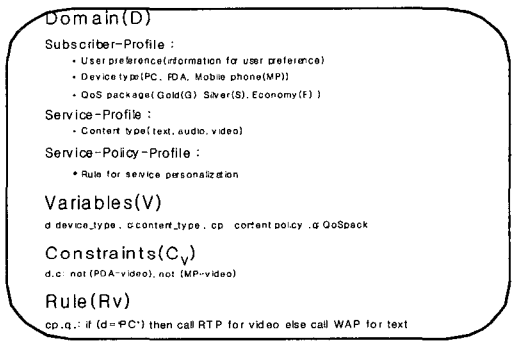

그림 5 사용자 제약조건을 위한 도메인, 변수, 제약조건, 규칙

워크의 품질 상태 등에 대한 정보를 가지고 있다. *Service Profile*은 ISP나 네트워크 관리자가 제공하는 서비스의 유형에 대한 정보를 가지고 있다. 그리고 *Service Policy Profile*은 ISP나 네트워크 관리자가 서비스 지원을 위한 정책에 대한 정보를 가지고 있다. 변수에는 디바이스 타입이 d, 컨텐츠 타입이 c, 컨텐츠 정책이 cp 등으로 정의되었다. 그리고 제약조건을 표시한 예로 *not(PDA=Video)*이 있다. 이 표현의 의미는 PDA는 비디오(video)를 지원하지 못한다는 의미를 나타낸다. 그리고 규칙은 서비스 재조합을 위한 정책(policy)을 명시하는 것으로 *if(d="PC") then call RTP for video else call WAP for text*의 의미는 디바이스 타입이 PC이면 비디오 서비스를 위한 RTP 프락시릿을 호출하고 디바이스 타입이 MP(Mobile Phone)이면 텍스트(text) 서비스를 위한 WAP 프락시릿을 호출하는 것을 의미한다. 이것은 서비스 재조합을 위한 규칙의 한 예를 보인 것이다.

4. SAS의 구현

SAS의 구현환경은 윈도우 2000서버, 무선랜 환경의 802.11 LAN카드를 장착한 노트북 그리고 Vanderbilt 대학의 GME소프트웨어와 Jade라 불리는 이동/지능 에이전트 소프트웨어로 구성되었다. 본 논문의 구현은 Supervisor AR, Worker AR, 이동 터미널 에이전트로 구성된다. 본 논문의 초점은 자치적응성 이므로 Supervisor AR과 이동 터미널 에이전트 구현 상세 내역을 생략하고 Worker AR의 구현 부분을 중점적으로 서술하였다.

4.1 GME를 이용하는 SAS

그림 6은 GME 도구에 의해서 새롭게 정의된 자치적 적응성을 위한 SAS를 보이고 있다. SAS 정의는 모델

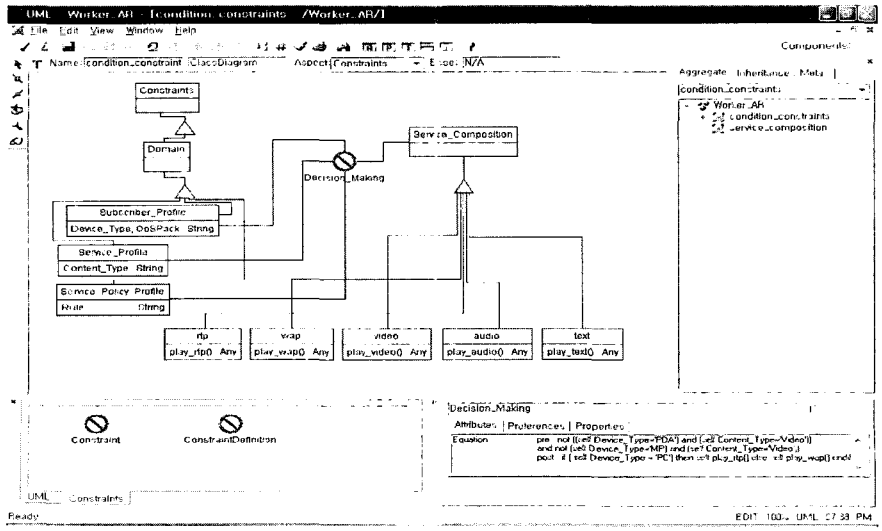

그림 6 GME도구에 의한 SAS

링 계층에서 UML과 OCL로 정의된다. GME를 이용한 Supervisor AR의 *Select_The_Fast_AR*과 *Request_Routing*의 기능이 있고, 그림 6은 GME를 이용하여 Worker AR의 *Constraint, Decision_making* 그리고 *Service_Composition* 기능 정의를 보이고 있다. *Constraint*는 사용자의 제약조건을 명시하기 위해서 도메인을 정의하였다. 도메인의 예는 *Subscriber Profile, Service Profile, Service Policy Profile*이 있다. 도메인은 UML로 표현되었다. 그리고 *Decision_Making*은 제약 심벌로 표시되었다. 이 제약 심벌의 내용은 그림 6의 하단 오른편의 OCL로 표현되어 있다. OCL은 UML의 의미를 표현하기 위한 수단으로 사용되며, 주로 *Decision_Making*기능에서 ISP의 서비스 규칙을 표현하는 수단으로 사용된다. *Service_Composition*은 RTP 프락시릿, WAP 프락시릿 등과 같은 많은 프락시릿 후보를 가지고 있고 *Decision_Making*의 정책에 따라 서비스 재조합을 수행한다.

4.2 SAS의 구현 코드

본 논문에서 사용한 GME는 UML/OCL을 C 언어로 자동 매핑시키는 기능을 지원하지만 아직 자바 언어의 자동 매핑을 지원하지 못한다. 그래서 본 논문에서는 수작업으로 UML/OCL을 자바 기반 이동/지능 에이전트 프로그램으로 매핑하였다.

그림 7은 OCL을 사용한 제약조건 및 서비스 규칙의 표현을 보이고 있다. OCL은 pre-condition과 post-condition을 구성되어 있다. 본 논문에서는 pre-condition을 사용자의 제약조건을 표시하는데 사용하였다. 그

```
pre : not
((self.Device_Type='PDA') and
(self.Content_Type='Video'))
and not (self.Device_Type='MP)
and (self.Content_Type='Video'))

post : if ( self.Device_Type = 'PC')
then self.play_rtp() else
self.play_wap() endif
```

그림 7 OCL 표현

리고 post-condition을 서비스 재조합을 위한 정책으로 사용하였다. *pre: not((self.device_type='PDA') and (self.Content_type='video')) and not(self.device_type ='MP') and (self.content_type='video')*의 의미는 디바이스 타입이 PDA면 비디오 서비스를 지원하기 못하고 MP(Mobile Phone)도 비디오 서비스를 지원하지 못한다는 것을 의미한다. *post : if(self.device_type ='PC') then self.play_rtp() else self_wap() endif*의 의미는 디바이스 타입이 PC이면 비디오 서비스를 위한 RTP 프락시릿를 호출하라는 의미이고 반면에 디바이스 타입이 MP이면 텍스트 서비스를 위한 WAP 프락시릿을 호출하라는 의미이다.

그림 8은 UML/OCL이 자바기반 이동/지능 에이전트로 매핑된 코드의 예를 보이고 있다. Worker_AR 클래스는 이동/지능 에이전트의 *Agent*라는 기본 클래스로부터 상속을 받는다. OCL의 pre-condition 부분이 *constraint_proxylet()* 메소드에 정의되어 있고 post-condition 부분이 *decision_making()* 메소드에 정의되어 있다. *service_composition()* 메소드는 OCL의 정책

```
Class worker_AR extends Agent implements SAS{
// to specify the user's constraint
public String constraint_proxylet(String constraint1, String contraint2) {
      if ((this.device_type=="PDA") && (this.content_type=="video"))
         result = "NOTOK";
      if ((this.device_type=="MP") && (this.content_type=="video"))
         result = "NOTOK";                                    <= Pre-condition of OCL
       return result;
}

// determine how to re-composite
public void decision_making_proxylet(String rule) {
     service_composition self = new service_composition();   <= Post-condition of OCL
     if ( this.device_type == "PC")  self.play_RTP_proxylet();
     else self.play_WAP_proxylet();
}
// re-composite proxylets and deployment
public void service_composition_proxylet(String result) {

 // find the suitable proxylets for service composition and deployment
return result
}}
```

그림 8 자바기반 이동/지능 에이전트로 매핑된 SAS의 구현 pseudo 코드

에 따라서 프락시릿을 재조립하는 기능을 수행한 것으로 필요한 프락시릿을 다시 재조합하여 각 정의된 프락시릿 메소드를 호출한다.

4.3 SAS의 실행

4.3.1 SAS의 실행 결과

그림 9의 왼편 상단은 이동 터미널 에이전트를 보이고 있고 오른편 상단은 Supervisor AR과 Worker AR의 실행 예를 보이고 있다. 그리고 하단은 Worker AR의 실행 결과를 보이고 있다. 본 논문의 이동 터미널 에이전트는 J2ME를 지원하는 디바이스상의 LEAP (Lightweight Extensible Agent Platform)을 이용하여 구현 중에 있기 때문에, 현재 Jade의 더미(dummy) 에이전트를 이용하여 실험하였다. PC에서 무선 이동 단말기로 변경된 경우, PC에 내장된 에이전트는 SAS에게 get a.html PC라는 메시지를 ACL을 이용하여 보내다가 무선 이동 단말기에 내장된 에이전트가 get a.html MP라는 메시지를 ACL을 이용하여 SAS에게 보내면, ACL메시지를 받은 Supervisor AR은 이 요청 메시지를 사용자와 가장 가까운 *Container 1*에 있는Worker AR에게 우회 시킨다. 선택된 *Container-1*의 Worker AR은 SCE에서 정의된 제약조건 및 서비스 규칙에 따라 기존에 RTP상의 비디오 서비스를 전송하던 것을 WAP상의 오디오 서비스로 변경하여 서비스를 시도한다. 그림 9 하단과 같이 *Container-1*의 Worker AR이 RTP상의 비디오 서비스를 개시한다는 메시지가 출력되다가 WAP 상의 텍스트 서비스를 개시한다는 메시지로 변경되는 것을 이용하여 실험하였다.

5. 성능 비교 평가

5.1 특성 비교

표 2는 기존 웹 서버상의 서비스와 기존 ALAN상의 네트워크 적응성 연구 및 본 SAS의 주된 특성 비교를 보이고 있다. 기존 웹 서버상의 서비스는 수동 구성만 가능하기 때문에 자치구성성 및 자치 적응성에 대한 고려가 부족하다. 기존 ALAN상의 네트워크 적응성 연구에서는 디바이스 유형과 QoS를 지원하지만 망의 문맥인식성 지원 및 고려가 부족하다. 반면에 본 SAS는 자치 구성성, 디바이스 유형, QoS지원과 더불어 망의 문맥인식성지원이 가능하다. 망의 문맥인식성 예는 PC에

표 2 주된 특성의 비교

		기존 웹 서버상의 서비스	기존 ALAN상의 네트워크 적응성 연구	확장된 ALAN상의 SAS
자치구성성			X	X
자치 적응성	디바이스 유형		X	X
	QoS 지원		X	X
	망의 문맥인식성 (context awareness)			X

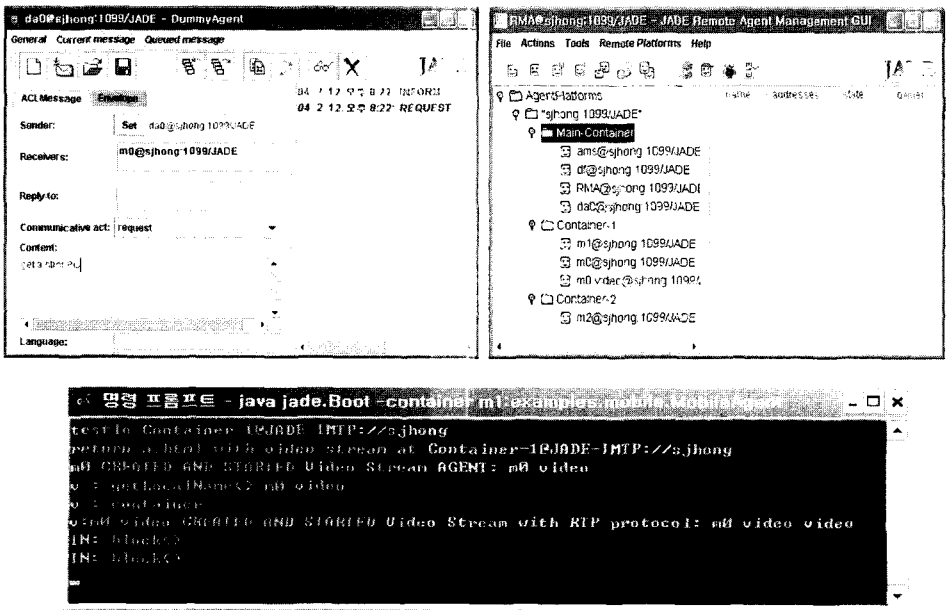

그림 9 SAS 실행 결과

서 무선 이동 단말기로 디바이스 유형이 변경 시 QoS 조정 및 연속적인 서비스 지원 그리고 위치 정보 변화에 따른 차별화된 서비스 등이다.

5.2 성능 비교

본 논문에서는 ns(network simulator)-2를 이용한 성능 평가를 하였다. 기존 ALAN상의 네트워크 적응성 연구는 수학적 접근 방법의 결과 만 있고 비교할 만한 공개된 구현 소프트웨어 및 성능평가 자료가 없어서 본 SAS와 성능 비교를 하지 못했고, 기존의 웹 서버 상의 서비스와 본 SAS의 성능 비교를 수행하였다. 성능 평가를 위한 기본 시나리오는 디바이스가 PC에서 무선 이동 단말기로 변경된 경우에 RTP상의 비디오 전송이 WAP상의 텍스트 전송으로 변경되면서 전송되는 패킷 크기가 자동 조정되고 연속적인 서비스가 되는 것에 대한 성능평가로서 망 내에서 제약조건의 변화에 대해서 능동적으로 데이타 포맷이 변경되거나 패킷 크기가 조정되는 경우를 성능 비교하였다.

그림 10은 ns-2를 이용한 tcl 주요 코드와 C++ 클래스의 헤더를 보이고 있다. 그림 10의 좌측은 tcl로 표현된 소스로서 각 3개의 노드의 구성을 지정하고 채널 용량은 2M byte로 하였다. 본 논문에서 Application/SAS라고 불리는 응용서비스 데이타 형식을 정의하였다. 그리고 디바이스 유형이 변경된 경우의 시나리오 중에는 *ns at 0.9 '$sas_s change 500'*라는 tcl명령을 정의하고 이용하여 이동/지능에이전트가 중간에 디바이스 유형에 맞도록 패킷을 500byte로 변화시키는 것을 보이고 있다. 그림 10의 우측은 SAS를 표현하기 위한 C++ 클래스 헤더를 보이고 있다. 데이타가 흐르는 중간에 패킷의 크기 등을 조정하기 위한 *change()*메소드가 포함되어 있다. *change()*메소드는 본 SAS의 기능을 성능평가를 위해서 정의된 메소드이며, 사용자의 디바이스 유형이 변화된 경우 패킷 크기를 중간에 변경할 수 있는 기능을 지원한다.

그림 11은 ns-2를 이용한 Nam(Network Animator)의 결과로서 성능 실험을 위한 노드 설정을 보이고 있다. 0번 노드는 데이타 제공 서버를 의미하고 1번 노드는 액티브 라우터를 의미하고 2번 노드는 사용자 단말기를 의미한다. 특히 2번 노드는 한 사용자의 단말기가 PC에서 무선 디바이스로 이동 변경된다고 가정하였다. 성능 실험을 위해서 2번 노드의 디바이스 유형 변경 때문에 1번 노드의 액티브라우터 내에 있는 데이타 패킷 크기가 변경되는 경우를 실험하였다.

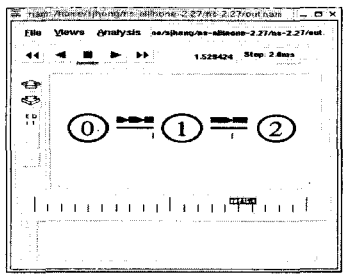

그림 11 노드 구성

```
set ns [new Simulator]
.....
set node_(n0) [$ns node]
set node_(n1) [$ns node]
set node_(n2) [$ns node]

$ns duplex-link $node_(n0) $node_(n1) 2Mb 10ms
DropTail
$ns duplex-link $node_(n1) $node_(n2) 2Mb 10ms
DropTail
.....
#Setup a SAS Application
set maapp_s [new Application/SAS]
set maapp_r [new Application/SAS]
$sas_s attach-agent $udp_s
$sas_r attach-agent $udp_r
$sas_s set pktsize_ 1040
$sas_s set random_ false
#Simulation Scenario
$ns at 0.0 "$sas_s start"
$ns at 0.9 "$sas_s stop "
$ns at 0.9 "$sas_s change 500 "
$ns at 7.0 "finish"
$ns run
```

```
// SAS Application Class Definition
    class SASApp : public Application {
      public:
       SASApp();
       void send_sas_pkt();

       void send_ack_pkt();
      protected:
       int command
            (int argc, const char*const* argv);
       void start();
       void change(int num);
       void stop(int num);
      private:
         void init();
         double rate[5];
         double interval_;
          int pktsize_;
      };
```

그림 10 ns-2의 tcl과 C++ 클래스

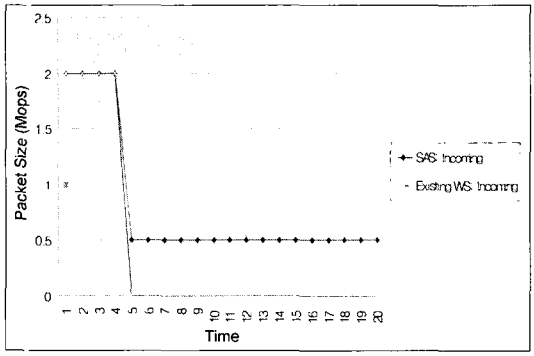

그림 12 QoS 조정 및 연속적인 서비스 성능비교

그림 12는 디바이스 유형 변경 시 QoS조정 및 연속적인 서비스를 실험한 경우를 보이고 있다. 그래프는 *SAS incoming*, *Existing WS incoming*으로 구성되어 있다. *SAS incoming*은 SAS를 이용한 경우 2번 노드에서 패킷 크기 변화를 측정한 것이고, *Existing WS incoming*은 기존 웹 서버상의 서비스인 경우 2번 노드에서 패킷 크기 변화를 측정한 것이다. 액티브 라우터를 의미하는 1번 노드가 패킷 크기를 2048byte에서 500byte로 조정하면서 서비스를 진행하는 경우에 *SAS incoming* 그래프는 사용자 단말기를 의미하는 2번 노드의 패킷 크기 변화를 보이고 있다. 반면에 2번 노드의 Existing WS incoming 그래프는 패킷의 크기가 조정되지 않으므로 중간에 서비스가 중단되는 그래프를 보이고 있다. 그러므로 디바이스 유형의 변경 등과 같은 사용자의 제약조건이 자주 변화는 경우는 기존의 웹 서버 기반의 컨텐츠 적응성 서비스 보다 SAS의 성능이 향상되는 것을 보이고 있다.

6. 결론 및 향후 연구

본 논문은 액티브네트워크를 이용한 망의 자치적응성 지원을 목적으로 하고 있다. 본 SAS는 망의 자치 적응성 지원을 위해서 UML/OCL을 이용하는 제약조건 기반 SCE를 액티브네트워크에 적용하는 시도를 하였고 액티브네트워크상의 액티브 어플리케이션으로서 이동/지능 에이전트를 이용하여 자치 적응성을 지원하는 메카니즘을 제시하였다.

유비쿼터스 컴퓨팅 환경의 대부분의 연구가 센서 및 무선 이동 단말기 그리고 서버를 중심으로 문맥인식성(context-awareness)을 지원하고 있는데 본 SAS는 망 내에서 자치적응성을 통해서 문맥인식성 지원이 가능하고, 더 나아가 상황인식(situation-awareness)성으로 확장 가능한 메카니즘이며, OPES 표준을 향상시키는데 반영될 것이라고 기대된다.

향후 연구로는 유비쿼터스 환경에서 RFID나 무선 센서 망과의 통합 차원에서 상황 인식(situation awareness)에 관한 연구가 필요하다. 더불어 유비워터스 환경에서 망에서 이루어지는 자동적인 변화를 사용자에게 얼마나 알려줄 지에 대한 정책적인 문제와 보안에 대한 연구가 필요하다.

참 고 문 헌

[1] Graham Klyne, et al., "Composite Capability/Preference Profiles (CC/PP):Structure and Vocabularies," W3C Working Draft 25, March 2003.

[2] Tomlinson, G., Chen, R. and M. Hofmann, "A Model for Open Pluggable Edge Services," draft-tomlinson-opes-model-00.txt, work in progress, June 2001.

[3] W. Y. Lum, and F. C. M. Lau, "A Context-Aware Decision Engine for Content Adaptation," IEEE Pervasive Computing, vol.1, no.3, pp.41-49, Jul.-Sep., 2002.

[4] B. Steffen, T. Margaria, A. Claben, V. Braun, and M. Reitenspieb, "A constraint-oriented service creation environment," In PACT'96, 2nd International Conference on Practical Application of Constraint Technology, London, UK, 1996.

[5] I.W.Marshall, et, al, "Application-level Programmable Network Environment," BT Technology Journal, Vol. 17, No. 2, April 1999.

[6] K. T. Krishnakumar, M. Sloman, "Constraint-based Network Adaptation for Ubiquitous Applications,"Proceedings of the 6th International EDOC Conference, Sep. 2002, pp. 258-271, Lausanne, Switzerland.

[7] K. T. Krishnakumar, M. Sloman, "Towards Constraint-Based Configuration (CBC) of proxylets for Policy Implementation," Proceedings of PGNet 2001, Liverpool John Moores University, UK, 18th-19th June 2001.

[8] Bellifemine, F., Poggi, A., and Rimassa, G.," Developing mulit-agent system with a FIPA-compliant agent framework," Software Practice and Experience 31(2), pp.103-128, 2001.

[9] Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason IV C., Nordstrom G., Sprinkle J., Volgyesi P, "The Generic Modeling Environment," Workshop on Intelligent Signal Processing, accepted, Budapest, Hungary, May 17, 2001.

[10] B. Steffen, T. Margaria, A. Claen, V. Braun, and M. Reitenspie, "A constraint-oriented service creation environment," In PACT'96, 2nd International Conference on Practical Application of Constraint Technology, London, UK, 1996.

[11] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design and Performance of Real-Time

Object Request Brokers," Computer Communications, vol.21, pp.294-324, Apr. 1998.
[12] Aniruddha Gokhale et al. "CosMIC: An MDA Generative Tool for Distributed Real-time and Embedded Component Middleware and Applications," Proceedings of the OOPSLA 2002 Workshop on Generative Techniques in the Content of Model Driven Architecture, Seattle, WA, November 2002.
[13] J. Wing, and Kleppe, A., "OCL : The Constraint Language of the UML," JOOP, May, 1999.
[14] D. Jacson, I. Scehchter and I. Shlyakhter," Alcoa: the Alloy Constraint Analyzer," In Proceedings International Conference on Software Engineering, Limerick, Ireland, June 2000.
[15] Jian Kuo-Di, Ken Nygard, "Agent-Java: An Integration of Java and KQML," Mid-continent Information and Database Systems Conference, Fargo, ND, USA, pp. 60-69, August, 1996.

홍 성 준

1991년 경원대학교 전자계산학과 졸업(공학). 1993년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사). 1998년 건국대학교 대학원 컴퓨터공학과 졸업(공학박사). 1993년~1999년 한국통신 연구소 전임연구원. 1999년~현재 여주대학 정보통신과 조교수

한 선 영

1977년 서울대학교 계산통계학과 학사 1979년 한국과학기술원 전산학 석사 1988년 한국과학기술원 전산학 박사 1981년~현재 건국대학교 컴퓨터 정보통신공학과 교수. 1995년 6월~1997년 12월 건국대학교 산업기술연구소 정보통신연구센타 소장. 1997년 1월~현재 한국 인터넷 협회 기술위원회 위원 1998년 1월~1999년 1월 미국 Maryland대학교 컴퓨터 과학과 객원교수. 2000년 3월~2002년 8월 건국대학교 정보통신원 원장. 2003년 1월~현재 개방형 컴퓨터통신연구회 회장. 관심분야는 Real-Time CORBA, Internet Caching, 차세대 인터넷 프로토콜, Mobile IP