경량 지니 룩업서비스 구현

장준[O], 오명환, 최훈
충남대학교 컴퓨터공학과
{jjang[O], mhoh, hchoi}@ce.cnu.ac.kr

Implementation of the Light Weight Lookup Service of Jini

June Jang[O], Myounghwan OhHoon Choi
Department of Computer Engineering, Chungnam National University

요 약

과거 하나의 컴퓨터에 집중화되어 있던 컴퓨팅 환경이 디바이스 기술의 발달로 인하여 분산 환경으로 변화하고 있다. 분산된 디바이스 간에 통신이나 자원의 이용을 지원하는 네트워크 컴퓨팅 기술이 지니이다. 1999 년에 썬 마이크로 시스템즈에서는 지니 룩업서비스를 발표하였는데, 이 지니는 소형 디바이스에서 사용하기에는 부적합하다. 본 논문에서는 기존의 룩업서비스를 모듈별로 나누어 모듈이 필요할 때만 다운로드하게하여 실행시 메모리 요구량을 줄여 경량 미들웨어로서 최적화된 룩업서비스를 구현하였다.

1. 서 론

과거 하나의 컴퓨터에 집중화되어 있던 컴퓨팅 환경이 모바일 통신 기술과 디바이스 기술의 발달로 인하여 분산 환경으로 변화하고 있다. 이들 디바이스들은 거의 개별적으로 사용되지만, 가끔 디바이스들 간의 협력이 필요할 때가 있다. 네트워크 프린터를 여러 종류의 디바이스가 이용하는 경우가 대표적인 예이다.

지니는 이러한 일을 사용자의 설정없이 자동적으로 실행되도록 썬 마이크로 시스템즈에서 고안한 기술이다 [1,2]. 지니는 UPnP [3]와 비슷하게 네트워크 환경에서 분산 컴퓨팅을 가능하게 해주는 기술로서 필요한 디바이스만 존재하면 사용자는 지니의 배경지식 없이도 사용할 수 있다.

썬 마이크로 시스템즈에서 개발한 지니 룩업서비스는 네트워크 내의 서버에서 실행되는 것을 가정하여 설계된 모델이다. 즉, 고정 서버에 이동 클라이언들이 접속하여 필요한 서비스를 검색하여 사용하도록 설계되었다. 하지만, 무선 환경에서 유선망 내 룩업서비스를 접속할 수 없는 환경이나, ad-hoc 네트워크 환경에서 서로 서비스를 제공할 수 있다면 더욱 효과적일 것이다.

현재의 지니 룩업서비스는 아직 경량화된 컴퓨팅 환경에서 사용하기에는 부적합하다 [4,5]. 본 연구에서는 썬 마이크로 시스템즈에서 개발한 룩업서비스의 실행 시 메모리 요구량을 조사하고, 룩업서비스 구현 객체의 경량화하여 모바일 디바이스에서도 탑재할 수 있는 방안을 구현하였다.

본 논문의 구성은 다음과 같다. 2 장에서 지니란 무엇인지를 살펴보고, 3 장에서는 썬 마이크로 시스템즈의 지니 룩업서비스와 그 문제점을, 4 장에서는 본 논문에서 구현한 모듈화된 지니 룩업서비스를, 5 장에서는 결론 및 향후 방안에 대해 제시한다.

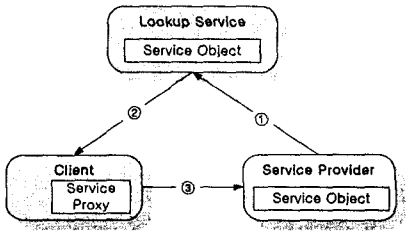
* 본 논문은 한국과학재단이 지정한 지역협력연구센터(RRC)인 충남대학교 소프트웨어연구센터의 지원으로 수행된 과제의 결과입니다.

2. 지니(Jini)

지니(Jini : Java intelligent network infra-structure)는 네트워크에서 가용한 자원들이 관리자의 개입없이 자동으로 검색되어 이용되게 한다. 지니 네트워크에 참여한 모든 서비스나 디바이스들이 service provider 나 service client 를 수행하면서, 서비스나 디바이스 등이 " plug and play" 에 의해 자동적으로 네트워크에 참여/탈퇴가 가능하도록 설계되어 있다. 지니에서는 스캐너, 프린터, 전화와 같은 디바이스들이 서비스로 이루어져있어 사용하기 위한 조건이 단순하고, 서비스들의 자유로운 참여/탈퇴와 감시로 새로운 버전의 수정 및 오류로 인한 오작동을 막을 수 있다. 또한, 일정 수의 지니 서비스들은 서로 검색할 수 있는 서비스를 공유하여 group 을 형성함으로써 확장 및 연합이 매우 용이하다.

지니 디바이스들은 지니 연합 내의 멤버가 제공하는 서비스를 검색하고 사용할 수 있다. 지니 시스템을 실행하는 구성요소는 크게 서비스, 클라이언트, 룩업서비스로 나눌수 있으며, 이 구성요소들간에는 디스커버리(discovery), 조인(join), 리징(leasing), 이벤트(event)에 의해 유지 관리 된다.

지니를 사용하는 디바이스들은 디스커버리와 조인 프로토콜을 사용하여 자율적으로 연합을 만들 수 있다. [그림 1]은 지니의 동작 과정을 나타낸다.

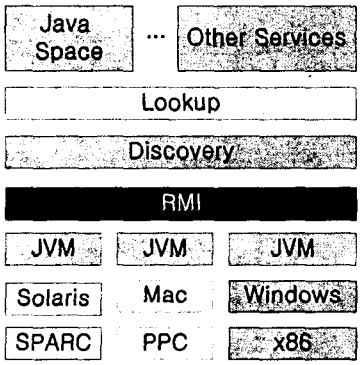


[그림 1] Service Architecture

① 서비스 제공자는 디스커버리 프로토콜을 사용하여 룩업서비스를 찾는다. 그후 자신의 서비스를 룩업서비스에 등록한다. 즉, 자신의 서비스 프록시(proxy)를 룩업서비스에 저장한다.
② 클라이언트는 디스커버리 프로토콜을 사용해서 룩업서비스를 찾는다. 그 후 클라이언트는 룩업서비스에 연결해서 어떤 서비스가 사용 가능한지를 물어본다. 자신이 필요한 서비스를 검색한 후 등록된 서비스의 프록시를 다운로드 한다.
③ 클라이언트가 서비스에 직접 연결하여 서비스를 사용한다.

3. 썬 마이크로 시스템즈의 지니 룩업서비스

1999 년 1 월 25 일 썬 마이크로시스템즈는 지니를 공식 발표하였다. 지니는 35,000 줄의 자바 코드로 구성되어 있으며, 가전제품의 내장 시스템, 홈네트워킹 기술 등에서 개발 중이다. [그림 2]는 썬 마이크로 시스템즈에서 개발한 룩업서비스의 구조도이다.



[그림 2] Jini Architecture Overview

앞서 언급한 바와 같이 룩업서비스는 서비스 즉, 서비스 제공자와 클라이언트, 즉 서비스 사용자 사이의 중계자 역할을 한다. [그림 2]에서 보는 바와 같이 썬 마이크로 시스템즈에서 발표한 룩업서비스의 경우 자바 VM(Virtual Machine) 위에 RMI 를 사용한다. [표 1]은 Boland OptimizeIt Suite 를 사용하여 자바 런타임 메모리 양을 측정한 결과이다.

[표 1] 썬의 지니 룩업서비스 실행시 메모리요구량

측정 부분	런타임시 메모리 요구량
JavaVM(1.4.1)	1.3M~1.7M
RMI	0.8M~1.2M
JavaVM+ RMI+ Jini	3.4M~3.8M

[표 1]의 결과처럼 썬 마이크로 시스템즈의 룩업서비스는 PC 와 같이 일정 수준 이상의 컴퓨팅 능력을 지닌 기기에서 작동하기 때문에, 모바일 기기만 모여있는 상황에서는 지니의 기능을 수행할 수 없다.

썬의 룩업서비스는 실행후 일정 시간후에는 메모리 요구량이 급속히 변화한다. 실행시 메모리 요구량이 변화하는 부분은 다음과 같다.

① 룩업서비스 초기화시에 로딩되는 부분.
② 다른 서비스가 룩업서비스에 등록하려 할 때.
③ 클라이언트에서 룩업서비스에 필요한 디바이스를 검색할 때.
④ 이벤트를 발생할 때.

룩업서비스의 구현 객체의 크기를 줄이는 방법으로서는 첫째, 룩업서비스의 기능을 축소해서 미니 버전 룩업서비스를 정의하고 구현하는 방법과 둘째, 부하가 많이 걸리는 부분을 대행하여 실행하게하는 방법 셋째, 룩업서비스를 실행할 때 필요한 부분만을 로드해서 실행하여 결과적으로 메모리의 양을 줄이는 방법, 이렇게 세가지 방법을 고려할 수 있다.

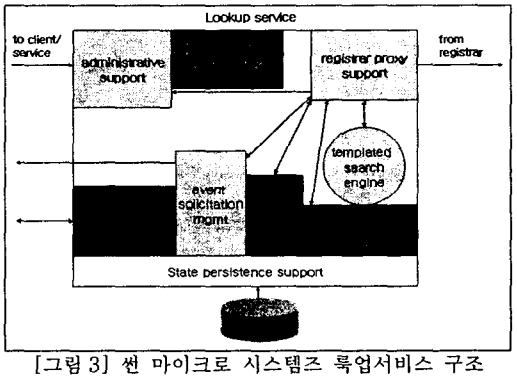
PsiNaptic 사의 Jmatos [6]는 첫번째 방식을 사용하였다. Jmatos 는 지니에서 많은 부분을 차지하는 RMI [7]를 제거하여 소형 기기에서 실행할 수 있도록 하였다. 그리고 룩업서비스 자신의 객체를 완전히 클라이언트로 넘기기 때문에 훨씬 작은 크기로 실행할 수 있다는 장점이 있다. 그러나 대부분의 지니 서비스가 RMI 를 사용해서 구현되고 있고, 룩업서비스의 실제적인 실행이 클라이언트에서 이루어지므로 클라이언트의 부담이 크다는 단점과 호환성에 문제가 있다.

Rochester Institute of Technology 의 JiniME [8]는 두번째 방식으로 설계되었다. JiniME 는 모바일 장비의 한계를 surrogate host 를 두어 Jini 의 서비스를 대행하게 하였다. 그러나 surrogate host 또한 고정된 장비이므로 모바일 환경과 같이 사용하기에는 제약이 있다.

본 연구에서는 세번째 방법으로 개발된 지니 룩업서비스를 구현한 것으로서 [9]의 후속 연구 결과이다. 추가로 두번째 방법을 사용하여 개발 중에 있다.

4. 모듈화하여 구현한 지니 룩업서비스

썬 마이크로 시스템즈에서 구현하여 배포하는 지니 룩업서비스의 구조는 [그림 3]와 같다. [그림 3]에서 표시된 모든 블록의 기능을 하나의 객체에서 지원하기 때문에 프록시의 Backend 서버에 있는 Registrar [1] 구현 객체의 크기는 143Kbyte 로서 제한된 자원을 지닌 모바일 디바이스에서 실행하기는 무리이다.



[그림 3] 썬 마이크로 시스템즈 룩업서비스 구조

본 연구에서는 썬 마이크로 시스템즈의 룩업서비스를 필수 모듈인 core 모듈과 선택가능한 모듈인 event, log, lease, log, lookup, service management, ID generator 모듈로 나누어 필요한 경우에만 선택적으로 다운로드하여 실행되도록 구현하였다.

Core 모듈은 룩업서비스가 동작하기 위해서 최소한으로 가

[1] Registrar : 룩업서비스에서 실질적으로 통신되는 객체

져야할 기능들이 포함되어 있다. 초소형 룩업서비스의 최소한의 기능은 이 모바일 디바이스가 네트워크 내의 일반 룩업서비스를 찾을 때 필요한 기능인 멀티캐스트(multicast), 유니캐스트(unicast), 아나운스먼트(announcement)하는 기능과 자신의 Registrar 프록시를 전달하는 기능이다. Lease 모듈은 서비스가 룩업서비스에 저장되어 있을 수 있는 lease 시간을 관리하는 기능을, service management 모듈은 룩업서비스에 저장될 서비스를 추가하거나 삭제하는 기능을 담당한다.

룩업서비스를 7 가지 모듈로 나눈 이유는 룩업서비스의 특성상 한 순간에 사용되는 기능들이 한정되어 있기 때문이다. 예를 들어, 룩업서비스가 클라이언트에 서비스만을 제공하는 경우 룩업서비스는 서비스를 등록받을 필요가 없기 때문에 service management 모듈과 ID generator 모듈이 필요없고, 또한, 추가되거나 삭제되는 서비스가 없기 때문에 event 모듈 역시 필요가 없게 된다. 이런 경우에 필요없는 모듈이 제외되어 있다면, 룩업서비스를 훨씬 적은 자원으로도 사용할 수 있다. 만약, 완전한 기능을 갖춘 룩업서비스를 구성하고자 한다면 모든 모듈을 선택하면 된다.

최초의 룩업서비스는 Registrar 프록시와 간단한 기능만을 제공하는 간단한 서비스이다. 이 서비스는 다른 룩업서비스에 등록되어 있는 이벤트 관리 모듈이나 리즈 모듈 등과 같은 룩업서비스의 구성요소를 필요한 경우에만 동적으로 다운 받아서 완전한 룩업서비스를 제공한다. 클라인언트는 룩업서비스의 Registrar 프록시를 통해서 룩업서비스를 사용하므로 클라이언트 입장에서 보면 완전한 룩업서비스의 형태로 보인다.

초기의 동적으로 룩업서비스를 구성하는 과정은 다음과 같다.

① 룩업서비스를 제공할 모바일 디바이스는 로컬 네트워크의 룩업서비스에서 자신의 룩업서비스를 구성하는데 먼저 디스커버리와 룩업서비스 Registrar 프록시를 제공하는 모듈을 로드한다.

② 로드된 모듈은 사용자가 원하는 서비스 제공 범위에 따라 그 서비스를 실행하는 모듈을 다운로드하고 모바일 디바이스의 Jini를 재구성한다.

위 방식은 모든 모듈은 일반 서비스와 같이 처리되어 클라이언트와 비슷하게 룩업서비스의 부담을 덜어준다.

[표 2] 실행시 메모리 요구량

단위 : Kbyte

Module	Source Size	Memory Requirement
썬사의 룩업서비스	220	3694
구현한 모듈의 총합	236.4	2102
core 모듈	143.0	1746
event 모듈	10.9	60
log 모듈	31.1	120
lease 모듈	8.7	18
lookup 모듈	25.7	110
service managèment 모듈	15.0	30
ID generator 모듈	2.0	45

[표 2]는 룩업서비스 구성 모듈별 소스코드의 클래스 크기와 실행시 메모리 할당양을 보여준다. 썬 마이크로시스템즈의 룩업서비스는 실행시에 3694Kbyte 의 메모리가 요구되었다. 본 구현에서는 평소의 코어 모듈의 경우 1746Kbyte 이고, 모든 모듈을 적재한경우 2102Kbyte 의 메모리가 요구되었다.

모든 모듈을 로드하면 결과적으로 썬 마이크로시스템즈의 룩업서비스보다 현저하게 메모리의 요구량이 줄어들었음을 볼 수 있다. 각 모듈의 메모리양의 합보다 구현 룩업서비스의 메모리가 적은 이유는 동시에 로드되는 모듈에 따라서 그 크기가 바뀌기 때문이다.

[표 2]의 메모리 요구량은 평균적인 값이다. 본 연구에서 구현한 룩업서비스의 경우에도 초기화시의 메모리의 많은 요구(약 3M 정도)의 문제점은 아직 가지고 있다. 또한 서비스를 저장하는 경우 ID, time, type, attribute 등으로 나누어 hashmap에 중복 저장되는 문제점을 가지고 있다. 본 연구는 이를 해결하기 위해 룩업서비스의 기동시에 초기화를 다른 장비에서 대행하여 실행하는 방법과, 나누어서 저장되는 서비스들과 이벤트들을 하나의 리스트에 저장하여 관리하는 방법을 구현 중이다. 이외에 여러가지 지니 라이브러리중에서 다른 룩업서비스와 호환을 위해 필요한 최소한의 라이브러리만을 사용하도록 구현하고 있다.

현재 구현된 룩업서비스는 J2SE 상에서 구현되었다. 하지만, J2ME 상에서 구현될 경우 더욱 메모리 크기를 줄일 수 있을 것으로 예상된다.

5. 결론 및 향후 방안

본 논문에서는 썬 마이크로시스템즈에서 개발한 지니 룩업서비스의 실행시 메모리 요구량을 조사하고, 문제점과 그 해결방안을 제시하고 구현하였다. 본 연구에서 구현한 룩업서비스의 경우 성능향상 보다는 모바일 디바이스에서 사용가능하도록 메모리의 요구량을 줄이는데 중점을 두었다. 완전한 기능의 룩업서비스를 제공하면서, 모듈을 필요할 때만 다운로드하여 메모리에 로드하는 방법을 사용하여 메모리의 요구량을 줄였다.

본 연구는 현재 초기화시의 메모리 요구량을 줄이는 부분과, J2ME 에서 동작 가능하도록 연구 중이다. 이러한 연구는 PDA 같은 경량 미들웨어 환경에서도 룩업서비스가 가능하게 될 것이다.

참고문헌

[1] Ken Arnold, Bill Joy, The Jini Specification (2nd Ed.)
[2] www.sun.com/jini/specs
[3] www.upnp.org/resources.htm
[4] W.K. Edwards, Core Jini(2nd Ed), Upper Sad-dle River, NJ:Prentice-Hall, 2001.
[5] Sun Microsystems Inc., The Remote Method Invocation Specification, 2001.
[6] Steve Hashman, Steven Knudsen, The Application of Jini Technology to Enhance the Delivery of Mobile Services, PsiNaptic Inc., 2001.
[7] Jan Newmarch, Aprogrammer' s Guide to Jini Tecnology, Apress, 2000.
[8] Alan Kamisky, Jini Connection Technology for Mobile Devices, 2000.
[9] 김창회, 모바일 디바이스를 위한 지니 룩업서비스 구현, 석사학위 논문, 충남대학교, 2003. 2.