DOI: 10.3745/KIPSTA.2009.16A.5.339

웹 서비스 동적 연동을 위한 클라이언트 에이전트 프레임워크

박 영 준[†] · 이 우 진[††]

요 약

일반적으로 웹 서비스에 접속하기 위해서는 .NET 또는 자바 런타임 등의 무거운 프레임워크를 사용하여야 한다. 이러한 프레임워크들은 기본적으로 PC 급 이상의 리소스를 가진 경우에 사용할 수 있으므로 센서 노드와 같이 제한적인 리소스를 가진 경우에는 웹 서비스를 사용할 수 없다. 이 논문에서는 이러한 클라이언트 노드에서 웹 서비스를 사용할 수 있는 클라이언트 에이전트 프레임워크를 제안한다. 클라이언트 에이전트 프레임워크는 충분한 리소스를 가진 제 3의 서버에서 관리되며, 실제 클라이언트 노드는 에이전트 서버에 접속하여 해당 클라이언트 에이전트의 웹 서비스 연동 기능을 이용한다. 클라이언트 에이전트는 클라이언트 요청 시에 WDSL 정보를 활용하여 동적으로 생성된다. 이러한 웹 서비스 연동 방법을 이용하면, 센서 노드나 모바일 단말에서 최소의 리소스로 웹 서비스를 연동할 수 있으므로 다양한 유형의 서비스를 구현할 수 있다.

키워드 : 웹 서비스, 클라이언트 에이전트, 동적 프레임워크, 모바일 단말

A Client Agent Framework for Dynamic Connection with Web Services

Young Joon Park[†] · Woo Jin Lee[††]

ABSTRACT

In order to connect web services, clients generally should use heavy frameworks such as .Net framework and Java run-time environment, which require high performance hardware resources like a personal computer. Therefore, it is impossible for sensor nodes to handle web services due to their limited resources. In this paper, a client agent framework is proposed for dynamically connecting web services in the client node with limited resources. A client agent, which is managed by the framework in other server, has full capability for connecting web services, while a real client has a simple connection module with the client agent. In this framework, a client agent is dynamically generated using the WSDL in the web service server. By using the framework, sensor nodes or mobile devices can enhance their functionalities and services by accessing web services with minimum resources.

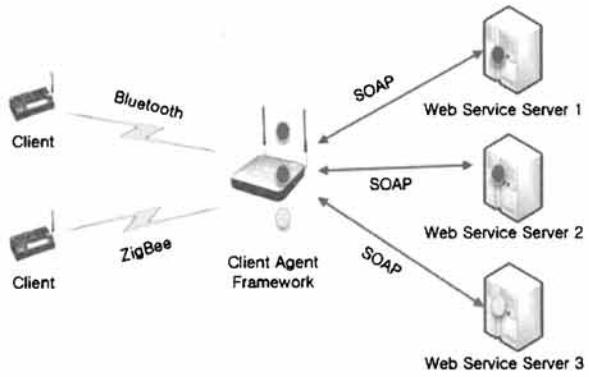
Keywords : Web Services, Client Agent, Dynamic Framework, Mobile Device

1. 서 론

웹 서비스는 표준화된 XML 기반의 인터페이스를 통하여 플랫폼 독립적이고 프로그램 언어에 중립적인 방법으로 네트워크 상에서 응용프로그램들을 연동할 수 있다. 이러한 웹 서비스는 분산 컴퓨팅 환경 하에서 동적으로 등록, 탐색되며 구동될 수 있으며 인터넷 어플리케이션 프로그램에 의한 실시간 상호작용을 촉진시켜서, 연관 기업들이 손쉽게 서비스들을 연동시킬 수 있다. 웹 서비스의 전략은 개방되고 표준화된 인터페이스를 기반으로 하드웨어, 운영시스템 및 프로그래밍 환경에 독립적인 상황을 만드는 것이다. 현재 웹 서비스는 MS가 닷넷(.NET) 기반의 윈도우 플랫폼에서 웹 서비스를 지원하고[1], IBM은 웹 서비스를 개발할 수 있는 개발 도구와 운영할 수 있는 실행환경 등 e-비즈니스의 통합환경 솔루션을 제공하며[2], SUN은 모든 소프트웨어 방식에 서비스 온 디맨드(services on demand) 개념[3]을 포함하고 있다. 이와 같이 메이저 IT 업체들이 웹 서비스 개발을 주도하고 있다.

현재 클라이언트가 웹 서비스를 받기 위해서는 소프트웨

※ 본 연구는 BK21 사업의 지원으로 수행되었습니다.
† 정 회 원 : LG전자 BS연구소 Security 연구실 연구원
†† 정 회 원 : 경북대학교 전자전기컴퓨터학부 부교수
논문접수 : 2009년 3월 27일
수 정 일 : 1차 2009년 7월 31일
심사완료 : 2009년 8월 10일



(그림 1) 웹 서비스 연동 클라이언트 에이전트 프레임워크 개념도

어적으로 MS의 닷넷 프레임워크 또는 SUN의 JAVA 런타임 환경을 지원해야 한다. 그리고 하드웨어적으로 PC와 같은 고성능을 요구한다. 그러므로 센서 네트워크 또는 유비쿼터스 컴퓨팅 노드 등과 같이 하드웨어나 소프트웨어적으로 제한적인 리소스를 갖는 단말 노드에서 웹 서비스를 받기 어렵다.

본 논문에서는 센서 노드와 같이 리소스가 부족한 클라이언트에서 웹 서비스에 연동할 수 있도록 웹 서비스의 클라이언트 에이전트를 자동으로 생성하고 관리하는 에이전트 프레임워크를 제안한다. 즉, 클라이언트 에이전트는 리소스가 충분한 게이트웨이나 PC와 같은 독립적인 사이트에서 배치 및 운용되면서 실제 클라이언트가 요구하는 웹 서비스를 대신 처리해 전달함으로써 실제 클라이언트의 리소스 제약사항 문제를 해결한다. 클라이언트 에이전트 프레임워크에서는 실제 클라이언트의 웹 서비스 요청이 들어오면, 웹 서비스 서버가 제공하는 WSDL(Web Services Description Language) 정보를 참조하여 클라이언트 에이전트 코드를 자동 생성한다. 그리고 클라이언트의 통신 인터페이스를 고려하며 접속된 클라이언트들을 관리한다.

본 논문의 구성은 다음과 같다. 제 2 절에서는 웹 서비스와 관련 연구에 대해 설명한다. 제 3 절에서는 센서 노드와 같이 제한된 자원을 갖는 클라이언트의 웹 서비스 지원을 위한 클라이언트 에이전트 구조에 대해 기술한다. 제 4 절에서는 클라이언트 에이전트의 구현 및 성능평가에 대해 설명하며 마지막으로 제 5 절에서 결론을 맺는다.

2. 연구 배경

무선 센서 네트워크 기술은 온도, 습도, 산불관리, 동물생태 등의 다양한 실세계 정보를 수집할 수 있는 센서와 수집된 정보를 처리할 수 있는 최소한의 컴퓨팅 리소스 그리고 데이터 전송을 위한 무선 네트워크 기능을 포함하고 있다[5]. 센서 네트워크와 같은 유비쿼터스 컴퓨팅 기술이 산업 전반에 널리 사용됨에 따라 좀 더 유연한 구조로 다양한 서비스 요구가 증가하고 있다.

〈표 1〉 PC와 센서 노드 하드웨어 비교

	PC	센서 노드(MICA2)
CPU	1 GHz	22 MHz(8051)
저장소	512 MB RAM 20 GB HDD	128 KB Flash 8 KB SRAM
데이터 전송율	LAN	ZigBee
	11 Mbps	250 kbps(2.4GHz) 40 Kbps(915Mhz) 20 Kbps(868Mhz)
응용분야	고성능 작업	제어 & 모니터링

일반적으로 유비쿼터스 환경에서 사용되는 센서 네트워크 노드들은 <표 1>과 같이 제한된 자원을 가진다. MICA2 센서 노드의 경우, 22 MHz 정도의 CPU, 128KB 플래시 메모리, 전송 속도가 낮은 저전력 근거리 무선 통신 지그비(ZigBee)를 사용한다. 일반적으로 웹 서비스를 이용하기 위해서는 클라이언트는 적어도 PC급 이상의 하드웨어 리소스를 필요로 하므로 센서 노드에서는 웹 서비스의 사용이 현실적으로 불가능하다. 또한 소프트웨어적으로도 <표 2>와 같이 닷넷 프레임워크, 자바 가상 머신, gSOAP 프레임워크 등의 소프트웨어 환경을 사용하여야 한다. <표 2>에서 알 수 있듯이, gSOAP은 다른 웹 서비스 프레임워크에 비해 상대적으로 경량 환경을 제공한다.

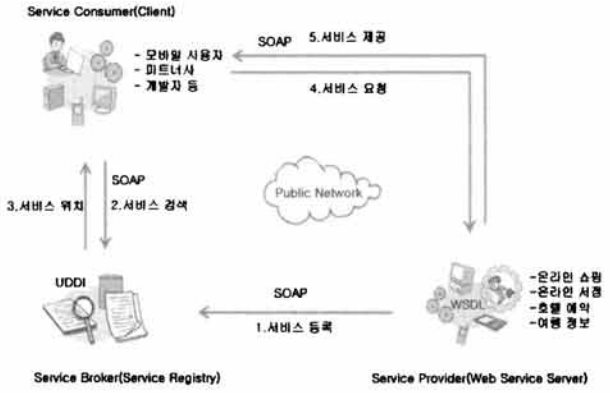
〈표 2〉 웹 서비스 지원 프레임워크의 크기 비교

	Microsoft .NET Framework 1.1	SUN JAVA VM	gSOAP
지원 OS	Windows 98/ME/NT/ 2000/XP/2003	Windows 2000(SP3+), Linux, Solaris	Windows, Linux, Solaris
크기	23.34 MB	18.32 MB	1.4 MB

2.1 웹 서비스 기술

웹 서비스 기술은 플랫폼과 언어에 관계없이 표준 인터넷 프로토콜에 기반하여 응용 프로그램간의 원활한 통합 및 협력을 지원하는 분산 컴퓨팅을 지원하는 기술로 SOAP(Simple Object Access Protocol), UDDI(Universal Discovery Description and Integration), WSDL(Web Services Description Language) 등의 국제 표준 기술이 있다[6].

(그림 2)는 웹 서비스의 구성 요소와 이용 과정을 나타낸다. 서비스 요청자는 웹 서비스의 사용자이고 웹 서비스 제공자는 비즈니스 고객의 서버에서 실행되고 있는 프로그램인 경우가 보통이다. 서비스 요청자가 서비스를 사용하기 위해서는 서비스 제공자의 인터페이스 호출 방법을 먼저 알아야 한다. 먼저 서비스 제공자는 자신의 서비스를 사용할 수 있도록 인터페이스의 호출 방법을 설명하는 문서를 UDDI 저장소에 저장하여 공개한다. 이 문서는 XML로 되어 있으며 WSDL라 불린다. 문서 안에는 서비스 요청자와 서비스 제공자 양자 간에 전달되는 파라미터의 이름들, 등록



(그림 2) 웹 서비스의 수행과정

된 웹 서비스가 실제 존재하는 URL 등 그 외에도 여러 가지 정보들이 기술되어 있다. 그러나 UDDI 저장소를 꼭 거쳐야만 되는 것은 아니고 서비스 제공자의 URL을 이미 알고 있다면 서비스 요청자가 바로 서비스 제공자를 호출할 수도 있다. 서비스 요청자 프로그램은 서비스 중개자에 저장된 WSDL 파일을 우선 참조하여 사용하고자 하는 서비스에 대한 상세한 정보를 얻는다. 그 다음에는 서비스 제공자에게 서비스를 요청하는 SOAP 메시지를 생성하여 서비스를 받는다.

2.2 웹 서비스 지원 플랫폼

주요 벤더들의 웹 서비스 지원 플랫폼으로는 MS의 닷넷, IBM의 WebSphere 어플리케이션 서버, SUN의 SunONE, 오픈 소스 기반의 gSOAP 등이 있다. MS 닷넷은 XML 기반 웹 서비스 플랫폼이다. MS는 닷넷 플랫폼 위에 더 많은 프로그램을 만들어 낼 수 있도록 개발 도구로 Visual Studio 2005 .NET을 제공한다. IBM WebSphere 어플리케이션 서버는 e-비즈니스 세계와 더불어 기업 데이터 및 트랜잭션을 통합하는 자바 기반의 웹 어플리케이션 서버이다. 트랜잭션 관리, 보안, 클러스터링, 기능성, 가용성, 연결성 및 확장성에 이르는 완전한 응용프로그램 서비스 세트를 구비하고, 개방형 기술과 API들을 활용하는 동시에 기업 전반의 어플리케이션에 대한 관리와 통합을 지원한다. SUN의 SunONE은 네트워크에 기대할 수 있는 최상의 개념인 서비스 온 디멘드를 포함시켰다. 서비스 온 디멘드의 핵심은 기업 이익을 위해 서비스 형태로 정보 자산을 완벽히 활용하는 것으로, 웹 어플리케이션 또는 웹 서비스 제공을 위해 실시간으로 조립되며 중앙 디렉터리에 저장된 어플리케이션 컴포넌트 형태가 될 수 있다.

경량화된 SOAP 툴킷 중 하나인 gSOAP은 미국 연방 정부의 지원으로 개발된 오픈 소스의 웹 서비스 개발 툴킷이며 주요 성능은 C/C++을 이용한 웹 서비스 구현을 간단하게 하고, 분산된 SOAP/ XML 웹 서비스 오퍼레이션에서 C/C++ 오퍼레이션과 데이터 타입을 자동으로 맵핑하는 컴파일러 기술을 제공한다.

〈표 3〉 SOAP 개발 도구 기본 특성 비교

	Axis	Axis2	XFire	SOAP::Lite	PHP	gSOAP	Visual Studio 2005
Language	Java	Java	Java	Perl	PHP5	C++	Yes
SOAP 1.1	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SOAP 1.2	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SOAP with attachment	Yes	Yes	Not Yet	Yes	Yes	Yes	Yes
WSDL client code generation	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WSDL server code generation	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Stand-alone server	No	Yes	Yes	Yes	No	Yes	Yes
Support for document/literal	Good	Good	Good	Average	Average	Good	Good
Runtime requirements	JVM	JVM	JVM	Perl interpreter	PHP engine	Nothing	.NET framework
Documentation	Good	Little	Good	Average	Average	Good	Good
Stable version since	>1year	<6months	<6months	>1year	>1year	>1year	>1year
Support tools in the distribution	Yes	Yes, including Eclipse plug-in	Yes,including Eclipse plug-in	No	No	No	Extensive

<표 3>은 프로그램 언어별 SOAP 개발 도구 특징을 나타내고 있다[8]. SOAP 개발 도구로는 JAVA 기반의 Axis/Axis2, XFire, PHP와 Perl 웹 스크립트 언어, MS의 Visual Studio .NET, C/C++언어 기반의 gSOAP 등이 있다. 이중에서 C/C++ 언어 기반의 gSOAP 개발 툴킷만이 런타임 플랫폼을 필요치 않는다. 런타임 플랫폼이 필요치 않으므로 다양한 OS에서 특별한 제약사항 없이 사용할 수 있으며 웹 서비스 어플리케이션 크기를 경량화시킬 수 있는 장점이 있다.

<표 4>는 단말과 단말간에 배열 크기에 따른 초 단위 응답 시간 성능 측정한 표이다[9]. Axis, gSOAP, .NET, XSOPA4 중에서 gSOAP의 응답 속도가 제일 좋음을 알 수가 있다.

이 논문에서는 리소스 규모, 런타임 요구사항, 수행 성능, 웹 서비스 개발환경 등을 고려하여 gSOAP 플랫폼을 이용하여 웹 서비스 연동 클라이언트 에이전트 프레임워크를 구현한다.

〈표 4〉 SOAP 개발 도구의 단말 간 성능 비교

(단위:초)

Tool Kit	배열 크기	10	100	1000
Axis	ECHO DOUBLE Arrays	0.0031	0.0057	0.0366
	Deserialization Double	0.0028	0.0047	0.0215
gSOAP	ECHO DOUBLE Arrays	0.0015	0.0029	0.0204
	Deserialization Double	0.0013	0.0019	0.0111
.NET	ECHO String	0.0052	0.1904	1.0836
	receive Ints	0.0046	0.1085	0.5467
XSOAP4	ECHO String	0.0045	0.2738	1.3549
	Receive Ints	0.0032	0.1207	0.6061

3. 웹서비스 연동 클라이언트 에이전트

3.1 클라이언트 에이전트 개념

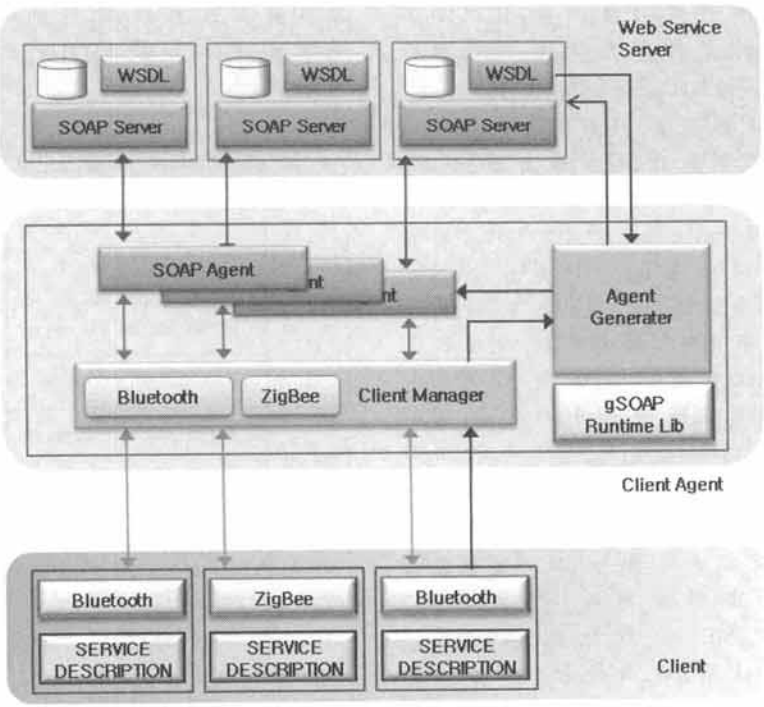
실제 클라이언트가 웹 서비스와 연동되기 위해서는 닷넷, 자바 가상 머신 등의 미들웨어가 필요하며 또한 하드웨어적으로도 PC급 이상의 리소스를 필요로 한다. 클라이언트 에이전트 기술은 소프트웨어 및 하드웨어의 제한적인 리소스를 가지고 있는 클라이언트가 웹 서비스를 받고자 할 때 클라이언트의 제한적인 리소스 단점을 보완하여 클라이언트가 웹 서비스를 원활히 받을 수 있도록 하는 것이다. 즉, 클라이언트의 웹 서비스 요청 시에 제 3의 사이트에서 동적으로 SOAP 에이전트를 생성하여 실제 클라이언트의 웹 서비스 요청을 대신 처리한다. 또한 실제 클라이언트의 하드웨어 및 소프트웨어 제한 사항을 고려하여 생성된 SOAP 에이전트들을 관리한다. 웹 서비스 기반 클라이언트 에이전트 프레임워크는 이러한 클라이언트 에이전트 개념을 지원하는 프레임워크로 다음과 같은 요구사항을 가진다.

- 실제 클라이언트의 요청에 따라 웹 서비스를 대신 연동해주는 SOAP 에이전트를 동적으로 생성하여 웹 서비스를 제공할 수 있어야 한다. SOAP 에이전트는 웹 서비스 서버의 WSDL 파일을 참고하여 웹 서비스 정보를 분석하여 해당 웹 서비스의 클라이언트 에이전트 코드를 자동으로 생성하여야 한다.
- 다양한 유형의 네트워크에 연결된 클라이언트들에게 웹 서비스 에이전트 역할을 담당할 수 있어야 한다. 클라이언트 에이전트 프레임워크는 지그비, 블루투스 등의 다양한 네트워크 기술에 적용될 수 있어야 한다.
- 생성된 클라이언트 에이전트들은 접속된 실제 클라이언트의 상태 정보와 함께 효율적으로 관리되어야 한다. 클라이언트의 웹 서비스 요청 또는 접속 종료에 따라 웹 서비스를 제공하는 SOAP 에이전트를 생성 또는 소멸하여 클라이언트 에이전트들의 소요 자원을 효율적으로 관리하여야 한다.

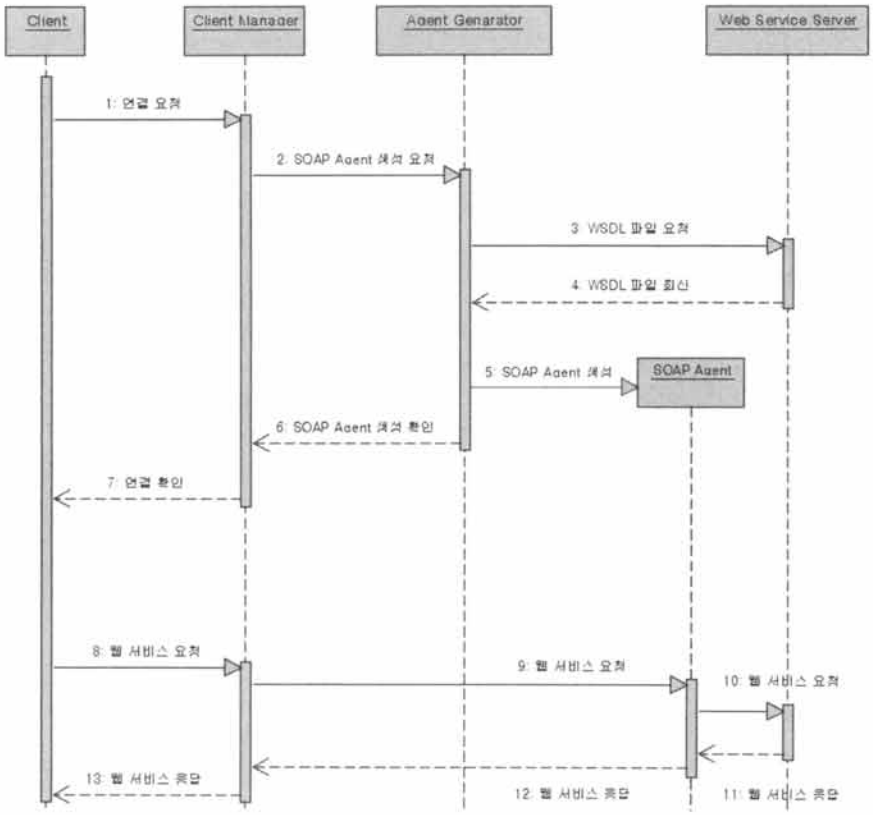
3.2 웹 서비스 연동 클라이언트 에이전트 프레임워크의 구조

웹 서비스 연동 클라이언트 에이전트 프레임워크는 실제 클라이언트 노드에 배치되는 것이 아니라 충분한 리소스를 가지는 제 3의 사이트에 배치된다. 클라이언트 에이전트 프레임워크는 (그림 3)과 같이 크게 클라이언트 매니저, SOAP 에이전트, 에이전트 생성기로 구성되어 있다. 클라이언트 매니저는 실제 클라이언트들과의 연동을 담당하는 모듈로 클라이언트의 네트워크 특성에 맞게 적당한 네트워크 접속 모듈들을 제공한다. SOAP 에이전트는 웹 서비스를 연동하는 클라이언트 기능을 제공한다. 그리고 에이전트 생성기는 gSOAP 플랫폼상에서 동적으로 SOAP 에이전트들을 생성한다.

클라이언트 에이전트 프레임워크의 수행과정은 다음과 같다. 클라이언트 매니저는 데몬처럼 항상 클라이언트 접속과 서비스 요청을 모니터링한다. 블루투스 모듈의 경우는 클라이언트 매니저의 블루투스 모듈이 항상 연결 대기 상태를 유지하고 있다가 클라이언트 노드가 감지되면 페어링 과정을 거쳐 서로 통신 가능한 상태를 만든다. 클라이언트 매니저의 지그비 모듈은 코디네이터 노드로 되어 있어 유사한



(그림 3) 웹 서비스 연동 클라이언트 에이전트 프레임워크의 구조



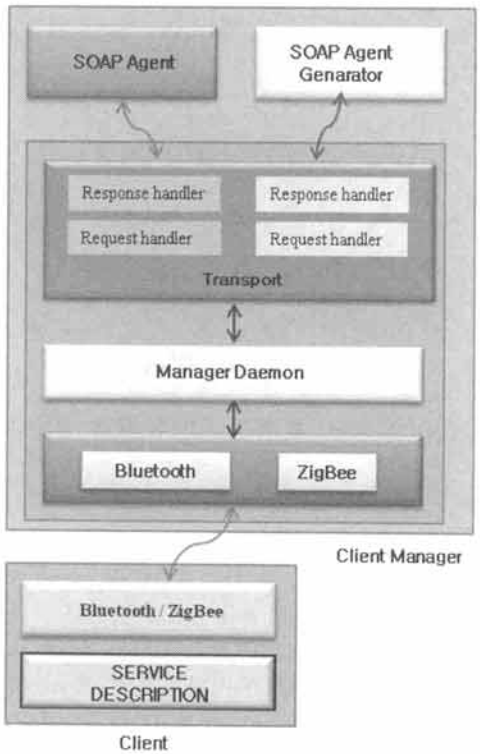
(그림 4) 클라이언트 에이전트 생성 및 서비스 요청 시나리오

연결 과정을 거쳐 통신 가능한 상태를 유지한다. 이 상태에서, 클라이언트가 웹 서비스 받고자 하는 웹 서비스 서버의 주소를 클라이언트 매니저에게 전달한다. 실제 클라이언트의 접속 요청은 에이전트 생성기에게 전달되며 에이전트 생성기는 웹 서비스 서버의 WSDL파일을 참조하여 클라이언트의 웹 서비스를 담당하는 SOAP 에이전트를 동적으로 자동 생성한다. 웹 서비스 접속이 완료되면, 클라이언트 프로그램은 SOAP 에이전트에게 서비스 요청 메시지를 보내고, 클라이언트의 요청을 받은 SOAP 에이전트는 원격으로 웹 서비스 시스템에 접속하여 웹 서비스를 처리하고 결과를 리턴한다. (그림 4)는 이러한 클라이언트 에이전트 프레임워크의 수행 시나리오를 순차도로 보여준다.

웹 서비스 연동 클라이언트 에이전트 프레임워크의 핵심 구성 요소인 클라이언트 매니저, 에이전트 생성기, SOAP 에이전트에 대해서는 다음 절부터 상세히 다루도록 한다.

3.2.1 클라이언트 매니저

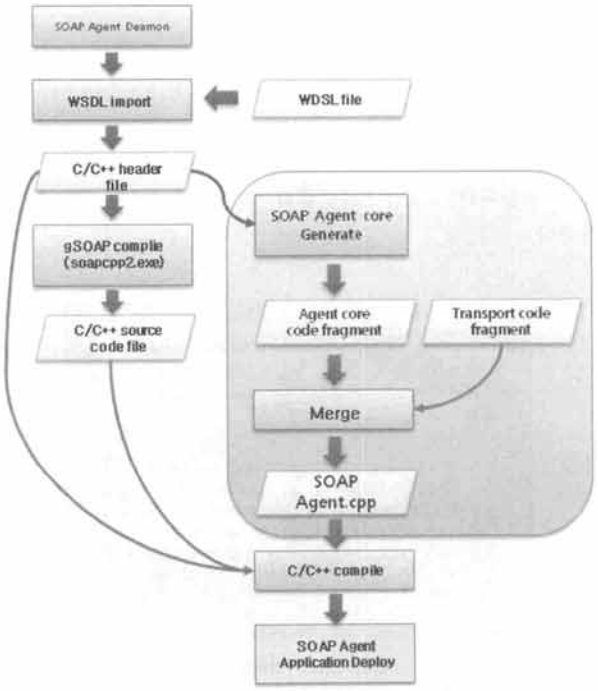
실제 클라이언트가 웹 서비스를 받기 위하여 클라이언트 에이전트에 접속하려면 클라이언트 매니저에 먼저 접속하여야 한다. 이 때 클라이언트 매니저는 지그비, 블루투스 등의 클라이언트 통신 인터페이스를 고려해서 연결한다. 매니저 데몬의 역할은 클라이언트 매니저에 접속하는 실제 클라이언트의 연결을 관리한다. 전송모듈(transport)은 클라이언트가 SOAP 에이전트 생성기에게 SOAP 에이전트 생성을 요청하고 SOAP 에이전트의 웹 서비스 요청 및 응답을 처리하는 모듈이다. (그림 5)는 클라이언트 매니저의 구조를 보여준다.



(그림 5) 클라이언트 매니저의 구조

3.2.2 에이전트 생성기

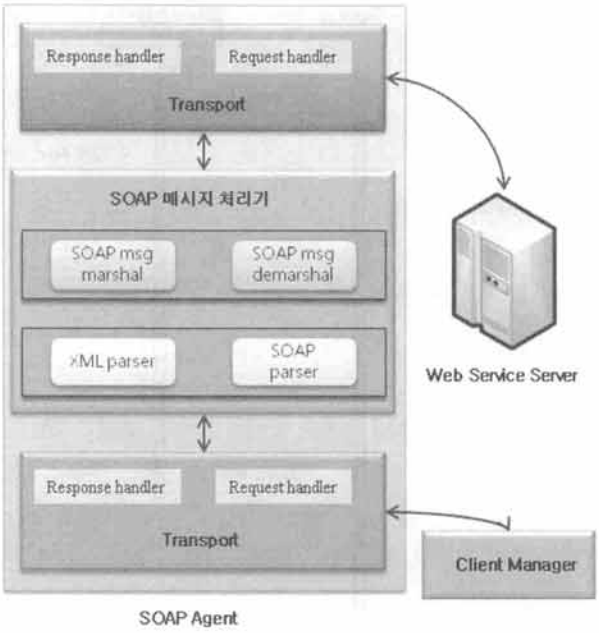
에이전트 생성기는 클라이언트의 요청에 의해 동적으로 웹 서비스를 제공하는 SOAP 에이전트를 생성 및 관리하는 기능을 수행한다. (그림 6)은 클라이언트의 요청에 따라 웹 서비스를 제공하는 SOAP 에이전트를 생성하기 위한 SOAP 에이전트 생성과정을 나타내고 있다. SOAP 에이전트를 생성하기 위해서는 서버 측으로부터 웹 서비스의 메타 정보인 WSDL 파일을 받은 다음, WSDL 파서로 메타정보를 분석하여 웹 서비스가 포함하는 오퍼레이션 정보를 포함하는 헤더파일을 얻는다. gSOAP 컴파일러는 헤더파일에 있는 함수 원형을 참조하여 원격 함수를 연결할 수 있는 스텁(stup) 코드를 생성한다. SOAP 에이전트 코어 생성기는 WSDL 파서로부터 생성된 헤더 파일을 참조하여 RPC 코드 조각을 생성한다. 웹 서비스 서버 및 클라이언트 매니저와 통신 담당 TCP/IP 소켓 통신 코드인 전송모듈의 코드 조각을 합쳐 SOAP 에이전트 파일을 만든다. 최종적으로 생성된 C/C++ 헤더파일과 소스 파일을 컴파일하여 SOAP 에이전트 코드를 동적으로 생성한다.



(그림 6) 웹 서비스를 위한 SOAP 에이전트 생성 과정

3.2.3 SOAP 에이전트

SOAP 에이전트는 클라이언트로부터 요청받은 웹 서비스 서버의 원격 메소드를 호출하며 리턴받은 SOAP 메시지를 클라이언트 응용 메시지로 변환하여 클라이언트 매니저에게 전달한다. (그림 7)은 SOAP 에이전트의 구조를 보여준다. SOAP 에이전트는 웹 서비스 서버의 원격 프로시저명과 인자를 요청 SOAP 메시지로 변환시킨다. 이것을 마샬링(marshalling)이라고 부른다. 요청 SOAP메시지는 전송 프로토콜에 의해서 웹 서비스 시스템으로 전달된다. 웹 서비스



(그림 7) SOAP 에이전트의 구조

시스템은 도착한 SOAP 메시지를 원래의 원격 프로시저명과 인자로 복원한다. 이것을 디마샬링(unmarshalling)이라고 부른다. 응용 메시지 변환기의 역할은 클라이언트 매니저의 요청 또는 웹 서비스 서버의 결과를 클라이언트 매니저 응용 메시지로 변환하는 것이다.

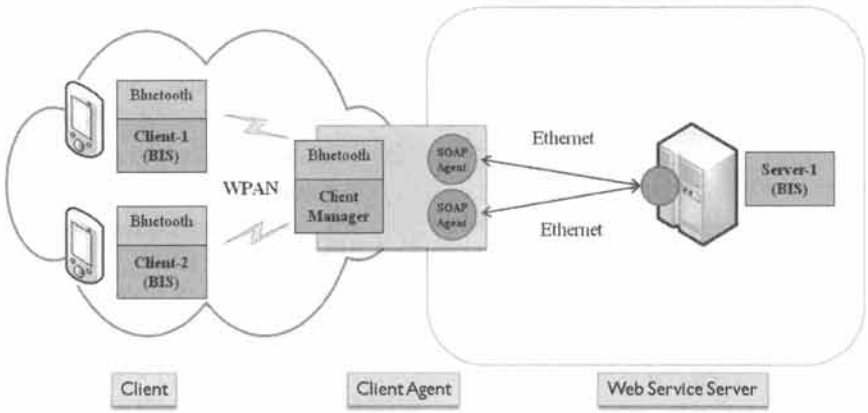
4. 구현 및 테스트

본 절에서는 웹 서비스 기반 클라이언트 에이전트를 통해 실제 클라이언트에 버스 도착 정보를 서비스하는 웹 서비스 구현 환경 및 시험 결과에 대해서 기술한다.

4.1 구현 환경

클라이언트 에이전트 프레임워크의 동작여부를 파악하기 위해 버스 정보시스템의 일부 기능인 버스 도착 알림 서비스를 웹 서비스로 (그림 8)과 같은 구현 환경에서 시범 구현하였다. 웹 서비스 서버와 클라이언트 에이전트 프레임워크는 각각 개인용 컴퓨터(2.8 GHz, 4 GB)에 배치하였다. 웹 서비스 서버와 클라이언트 에이전트 서버간에는 유선(ethernet)으로 연결하였다. 실제 클라이언트 프로그램은 HP iPAQ (Rx4240)에 운영체제 Window Mobile 2005, 메모리 64M에서 구현하였다. 그리고 클라이언트 에이전트 서버와 클라이언트 간에는 블루투스를 통해 연결된다.

클라이언트 에이전트와 실제 클라이언트 프로그램은 C 언어로 구현하였으며, 웹 서비스 플랫폼은 gSOAP 2.7.11을 사용하였다. 시스템 환경 구성은 <표 5>와 같다. 시험에 사용한 무선 모듈은 블루투스를 사용하였으며 프로토콜 버전은 2.0 Class 2 이다, 무선 모듈의 최대 전송 속도는 1 Mbps이다.



(그림 8) 클라이언트 에이전트 프레임워크의 배치도

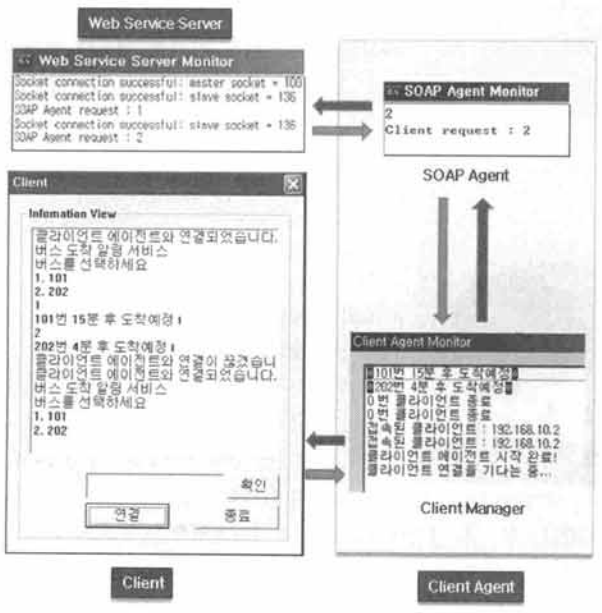
〈표 5〉 프레임워크의 개발 환경

운영체제	Windows XP
구현 언어	C언어
라이브러리	gSOAP 라이브러리
블루투스 모듈	Class 2, Version 2.0

4.2 버스 도착 알림 서비스 구축

웹 서비스를 제공하는 웹 서비스 서버를 먼저 실행시킨다. 두 번째로 실제 클라이언트 접속 관리 및 웹 서비스 제공 역할을 하는 클라이언트 에이전트를 실행시킨다. 마지막으로 클라이언트 에이전트와 클라이언트간의 블루투스 통신 연결을 한 후 실제 클라이언트 프로그램을 실행시킨다.

클라이언트가 실행되면 클라이언트는 클라이언트 에이전트에게 서비스 받고자 하는 웹 서비스 서버 주소를 보낸 후, 화면에 버스 도착 정보 서비스 메뉴를 표시한다. 클라이언트 에이전트는 클라이언트의 접속을 모니터링하고 웹 서비스를 제공하는 SOAP 에이전트 생성을 위해 에이전트 생성기에게 SOAP 에이전트 생성을 요청한다. 에이전트 생성기가 SOAP 에이전트를 동적으로 자동 생성함으로 클라이언트에 웹 서비스를 제공할 수 있는 상태가 된다.



(그림 9) 버스 도착 알림 서비스의 구현 화면

클라이언트에서 연결과 종료 버튼을 눌러 클라이언트 매니저에 접속이 원활함을 클라이언트 매니저 모니터와 클라이언트 화면을 통해 확인한다. 클라이언트 연결 버튼을 눌러 클라이언트에 접속하여 버스 도착 정보를 받기 위해 1번 또는 2번을 눌러 확인한다. 이 때 클라이언트 에이전트 모니터, SOAP 에이전트 모니터, 웹 서비스 서버의 모니터를 확인하여 클라이언트 명령에 따른 클라이언트 에이전트 요청 메시지 및 서버로부터 응답 메시지가 올바른지 확인한다.

4.3 클라이언트 에이전트의 프로그램 크기 및 수행 속도

버스 정보 서비스 예제에서 클라이언트 프로그램을 각 언어별로 구현하여 파일 사이즈를 비교하였다. 각각의 기술들은 고유의 장단점을 갖고 있음으로 어느 것이 다른 것에 우월하다고 할 수 없다. 하지만 센서 네트워크 노드의 하드웨어 리소스를 고려할 때 초소형 메모리 내에 적용이 가능해야 한다. <표 6>를 보면 .NET 프레임워크 기반 클라이언트 프로그램 사이즈는 23.43 MB, 자바 가상 머신 기반 클라이언트 프로그램 크기는 18.34 MB, gSOAP 프레임워크는 1.5 MB로 상대적으로 작기는 하지만 초소형 센서 노드에 적용하기에 적합하지 않다.

본 논문에서 gSOAP의 웹 서비스 연동의 핵심 부분들을 포함하는 클라이언트 에이전트 프레임워크는 1.41 MB 규모로 제3의 서버에 배치되므로 실제 노드의 클라이언트 프로그램의 크기는 20 KB 정도로 센서 네트워크에 적합함을 알 수가 있다. 즉, 클라이언트 에이전트 프레임워크로 인해 초경량 센서 노드에서도 웹 서비스를 연동할 수 있으므로 실생활과 산업 전반에 다양한 웹 서비스 응용이 가능할 것으로 예상된다.

클라이언트 에이전트 프레임워크의 성능을 평가하기 위해 버스 정보 서비스 예제의 수행시간을 측정하였다. 먼저, 클라이언트 에이전트의 생성 시간을 측정해 보았다. 생성 시

〈표 6〉 클라이언트 에이전트 프레임워크와 클라이언트 크기

	클라이언트 에이전트 프레임워크 사이즈	클라이언트 프로그램 사이즈
.NET 플랫폼	-	23.43 MB
자바 가상머신	-	18.34 MB
gSOAP 플랫폼	-	1.5 MB
클라이언트 에이전트 접근방법	1.41 MB	20 KB

간은 총 20회의 평균 784 msec 정도의 시간이 소요되는데, 이러한 시간을 대부분 gSOAP 컴파일러와 C 컴파일러를 이용하여 코드를 생성하고 컴파일하는데 소요되는 시간이었다. 그리고 클라이언트 에이전트를 거쳐 웹 서비스를 받는 데까지 소요시간은 총 20회에 평균 245 msec이 측정되었다.

5. 결 론

본 논문에서는 웹 서비스를 받기 위한 플랫폼은 주로 PC급 이상의 하드웨어 성능과 소프트웨어적으로 닷넷 또는 자바 가상머신을 지원해야 웹 서비스의 연동이 가능하다. 하지만 센서 네트워크 또는 유비쿼터스 컴퓨팅 노드 등과 같이 하드웨어나 소프트웨어적으로 제한적인 리소스를 갖는 단말 노드에서는 웹 서비스를 받기 어려운 환경에 있다. 이러한 상황에서 유비쿼터스 컴퓨팅 노드와 같은 실제 클라이언트에 웹 서비스를 제공 가능한 클라이언트 에이전트 개념을 제안하고, 이를 지원하는 프레임워크를 설계 및 구현하였다. 그리고 시범 사례에 적용하여 프레임워크의 적용 가능성을 검토하였다.

향후 웹 서비스 기반 클라이언트 에이전트는 초소형 센서 노드와 같은 클라이언트에 웹 서비스를 제공함으로써 산업 전반과 가정에서의 사용자의 요구에 부합하는 여러 형태의 웹 서비스를 제공할 것으로 예측된다. 따라서 본 논문은 향후 초소형 유비쿼터스 센서 노드와 같은 클라이언트에 웹 서비스를 제공하는 좋은 예가 될 것이다.

참 고 문 헌

[1] 오세영, "NET 2기 비전", *정보처리학회지*, 제9-A권, 제2호, 2002년 6월.

[2] 정대성, "IBM의 웹 서비스 솔루션", *정보처리학회지*, 제9-A권, 제2호, 2002년 6월.

[3] 양희정, "웹 서비스 구성을 위한 Sun ONE 전략", *정보처리학회지*, 제9-A권, 제2호, 2002년 6월.

[4] 이원준, 이춘화, *저속 WPAN*, 홍릉과학출판사, 2005.

[5] 이재용, "유비쿼터스 센서 네트워킹 기술", *TAA 저널*, 제 95호, 2004년 10월.

[6] W3C, "Web services architecture requirements," http://www.w3.org/TR/wsa-reqs

[7] ETRI 정보화기술연구소, "모바일 웹 서비스를 위한 응용 프로토콜 기술 현황", *주간 기술동향*, 1106호, 2003년 11월.

[8] Panagiotis Louridas, "SOAP AND Web Services," *IEEE Computer Society*, Vol.23, No.6, pp.62-67, Nov., 2006.

[9] Madhusudhan Govindaraju, "Toward Characterizing the Performance of SOAP Toolkits,"*Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, pp.365-372, Nov., 2004.

[10] R. van Engelen, G. Gupta, and S. Pant, "Developing Web Services for C and C++," *IEEE Internet Computing*, Vol.7, No.2, pp.53-61, Mar., 2003.

[11] OSGi Alliance, http://www.osgi.org

[12] Robert van Engelen, "Code Generation Techniques for Developing Lightweight XML Web Services for Embedded Devices," *ACM SIGAPP SAC Conference*, pp.854-861, 2004.

[13] Box D, Ehnebuske D, Kakivaya G, Layman A, Mendelsohn N, Nielsen H, Thatte S and Winer D, "Simple object access protocol 1.1," *Technical report*, W3C, http://www.w3c.org/TR/SOAP, May, 2000.

[14] Callaway, E. Gorday, P. Hester, L. Gutierrez, J.A. Naeve, M. Heile, and B. Bahl, V. "Home Networking with IEEE 802.15.4: A Developing Standard for Low-Rate Wireless Personal Area Networks," *IEEE Communications Magazine*, Vol.40, No.8, pp.70-77, Aug., 2002.

박 영 준

e-mail : koroot@gmail.com

2004년 경일대학교 컴퓨터공학과(학사)

2008년 경북대학교 정보통신학과(공학석사)

2008년~현 재 LG전자 BS연구소 Security 연구실 연구원

관심분야 : 임베디드 실시간 시스템 모델링 및 분석, Security Solution, 웹 서비스 기술 등

이 우 진

e-mail : woojin@knu.ac.kr

1992년 경북대학교 컴퓨터과학과(학사)

1994년 한국과학기술원 전산학과(공학석사)

1999년 한국과학기술원 전산학과(공학박사)

1999년~2002년 한국전자통신연구원 S/W 공학연구부 선임연구원

2002년~현 재 경북대학교 전자전기컴퓨터학부 부교수

관심분야 : 임베디드 실시간 시스템 모델링 및 분석, Requirements Engineering, Petri nets, 웹 서비스 기술 등