서비스 번들링을 위한 에이전트 기반 소프트웨어 구조

장근영, 이원석, 전진희, 강문석
KT 마케팅연구소

요약

각종 서비스와 망의 통합화 추세에 따라, 기존의 수익성 있고 인기 있는 서비스를 통합하여 새로운 부가 서비스를 창출할 수 있는 번들링 서비스에 대한 요구가 늘어나고 있다. 이러한 번들링 서비스는 다양한 서비스의 통합으로 이루어질 수 있으며 기 진행 중인 서비스에 영향을 주지 않으면서 고객이 원하는 통합 효과를 얻을 수 있는 데이터 관점에서의 통합이 효과적이다. 여기서는 각 서비스에서 발생한 데이터 변화가 서비스를 대표하는 소프트웨어 에이전트를 통하여 타 서비스의 에이전트에 영향을 주어 고객이 원하는 서비스 통합을 가능하게 하는 에이전트 기반 소프트웨어 플랫폼인 SPAS(Service Peering and Aggregation Server)에 대해서 기술한다. SPAS는 각 서비스들의 구조를 변경시키지 않고 표준적인 통신 규격에 따라 인터넷 환경에서 핵심적인 역할을 수행하고 있는 중요 서비스들(메신저, 인터넷폰, 아웃룩 일정관리)을 결합시는데 성공하였다. 현재 개발된 플랫폼은 인터넷망의 사용에 한정되어 있으나, 추후 연동 G/W를 이용하여 유선과, 무선, 그리고, 각종 멀티미디어 콘텐츠 제공서비스에까지 확대될 수 있을 것이다.

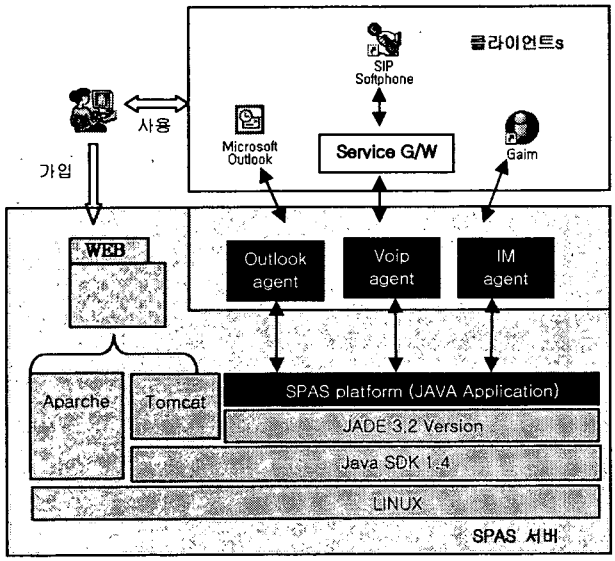
1. 서론

최근 광대역 통합망인 BCN 시험망이 구축되어감에 따라 새로운 서비스에 대한 요구가 증가하고 있다. 그러나 BCN에서 이루고자 하는 대부분의 기본적인 서비스들은 기존 지능망이나 인터넷에서 구현되어 가입자들에게 제공되고 있다. 따라서 가입자들의 시선을 유도할 수 있는 신규 서비스의 개발이 매우 어려운 상황이다. 이에 따라 기 제공중인 서비스를 통합하여 새로운 서비스를 시장에 도입하려는 움직임이 나타나고 있다. 이미 고객에게 성공적으로 제공중인 서비스를 묶어서 신규 서비스로 가입자에게 제공하면 시장분석 시간이 단축되고 성공가능도 매우 높으며 비용측면에서도 유리하다[1]. 최근 서비스 가입자들은 여러 서비스 제공자로부터 다양한 서비스를 제공받고 있다. 이메일과 메신저 서비스는 포탈 서비스 제공자로부터, 유선 서비스는 유선 통신 사업자로부터, 무선 서비스는 이동통신 사업자로부터 제공받고 있다. 이러한 환경하에서 서비스 가입자들은 PC 환경에서 제공되고 있는 메신저, 이메일 및 아웃룩 등과 같은 인터넷 서비스와 기존 유무선 통신 서비스를 하나의 관점에서 통합시킨 새로운 서비스를 원하고 있다. 이렇게 고객을 둘러싼 통신 환경이 복잡해짐에 따라 단일 서비스 제공자가 가진 자원만으로 고객들이 원하는 서비스를 제공할 수 없으며 타 서비스제공자와 연합하여 보다 완벽하고 비용이 저렴한 서비스를 제공할 필요가 있다[2]. 이러한 서비스 통합에는 각 서비스 처리에 개입하는 로직 기반 통합과 서비스의 데이터를 기반으로 하는 데이타 기반 통합으로 나뉘어 질 수 있다[3]. 서비스는 서비스 로직(제어)와 서비스 데이터로 이루어 진다. 서비스 로직 기반으로 통합된 새로운 서비스는 기존 서비스와 매우 긴밀하게 연관되어 있으며 서비스 제어에 직접적으로 영향을 미친다. 일반적으로 통신 서비스들은 서비스간 상호 통신을 염두에 두고 개발되지 않았다. 따라서 기존 서비스들을 로직 기반으로 묶는 것에는 많은 노력과 시간이 투입될 수 있다. 또한 서비스를 제공하는 사업자가 동일하지 않은 경우 로직 기반 통합은 더욱 어렵게 된다. 따라서 기 진행 중인 서비스에 영향을 주지 않으면서 고객이 원하는 통합 효과를 얻기 위해서는 데이터 관점에서의 통합이 효과적이다. 여기서는 각 서비스에서 발생한 데이터 변화가 각 서비스를 대표하는 소프트웨어 에이전트를

통하여 타 서비스의 에이전트에 영향으로 주어 고객이 원하는 서비스 통합이 가능하게 하는 에이전트 기반 소프트웨어 플랫폼인 SPAS(Service Peering and Aggregation Server)에 대해서 기술한다. 서비스 에이전트는 대등한 관계로서 사업자가 제공하는 서비스를 대표하게 되며 각 에이전트가 속한 서비스의 데이터가 변경되면 관련된 타 서비스의 에이전트로 통보하게 되는 구조로서 에이전트와 에이전트간 프로토콜 및 가입자 데이터로 이루어지는 하나의 소프트웨어 플랫폼이다.

2. SPAS 시스템 구조

그림 1은 2004년 하반기에 진행된 SPAS (Sevice Peering and Aggrigation Server)시스템의 구조도이다.



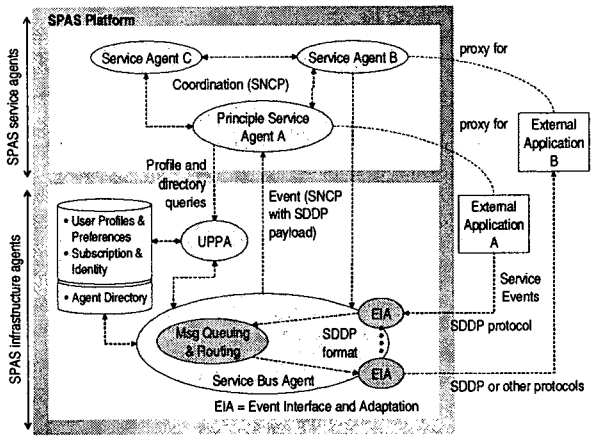
[그림1] 구현된 SPAS 시스템 구조도

번들링의 대상이 된 서비스는 메신져 서비스, 아웃룩 일정관리 서비스 및 VOIP를 이용한 인터넷 전화 서비스이다. 본 시스템의 구현 목적은 세 서비스의 프레즌스 정보를 관리하여 보다 지능적인 서비스를 제공하고자 함에 있다. 예를 들어, PC를 통해 주로 작업하는 가입자가 자리를 비워, 메신져가 부재중으로 설정되는 경우, 본 시스템은 이 가입자를 자리에 없는 것으로 판단하여, 인터넷 전화로 걸려온 전화를 자동으로 가입자의 휴대전화로 호전환시켜 주는 서비스를 제공할 수 있다. 혹은 가입자가 아웃룩을 통해 일정관리를 하는 경우, 회의중으로 설정된 시간에는 이 가입자에게 걸려온 전화에 대해, 사서함으로 바로 연결되는 서비스를 제공 할 수 있으며, 그 밖의 기타 번들된 서비스와의 연계를 통한 다양한 서비스로의 확장이 가능하다.

2.1 SPAS 플랫폼 구성

본 시스템을 구현함에 있어 중점을 둔 사항은, 각 서비스들이 독립적으로 동작하면서 정해진 표준 통신 규격에 의해 메시지를 SPAS 플랫폼으로 전달시키고, 다시 이 메시지에 의해 영향을 받는 서비스로 이벤트 메시지를 보내는 것이다. 즉, 각각의 서비스는 다른 서비스의 존재를 고려하지 않고 SPAS 플랫폼과 통신하며, SPAS플랫폼은 가입자에게 부가적인 가치를 제공할 수 있도록 다른 서비스들과의 협상기능을 수행한다. 각각의 서비스와 통신하기 위해 Telcodia의 제안에 따라 TILAB(telecom italia lab)에서 JAVA 로 구현한 JADE(Java Agent DEvelopment Framework)를 사용하였다. JADE는 Multi-agent 기능 및 이 agent들의 분산처리를 지원한다. Agent들과 SPAS 플랫폼 사이의 연동은 RMI통신을 따른다[4].

그림2는 SPAS 플랫폼의 내부 구성도 이다. 각 서비스를 담당하는 Service Agent와, 외부 통신을 담당하는 Service Bus Agent, 그리고 가입자의 개인 정보와 서비스 우선 순위 정보 관리를 담당하는 User profiles and preferences Agent로 구성된다[3]



[그림2] SPAS 플랫폼 구성도

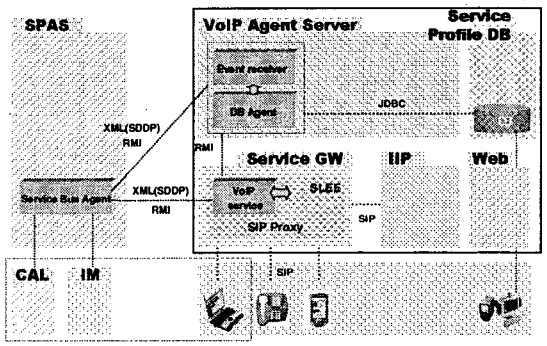
SPAS 플랫폼의 주요 기능으로는 첫째, 이벤트가 발생한 특정 서비스에서 SPAS로 보낸 메시지를 수신하는 기능, 둘째, 받은 메시지를 가공한 뒤, 가공된 메시지를 전달해야할 할 대상을 선정하는 기능, 셋째, 통합 관리 서버에서 선택한 대상 서비스로 메시지를 전송하는 기능, 넷째, 선택된 대상 서비스에서 이 메시지를 받아 처리하는 기능이다[3]. 각 서비스와의 연계를 위하여 SPAS 플랫폼에서는 서비스별 SA(Service Agent)를 둔다[3]. 그리고 이 SA들과 번들링될 서비스들의 통신을 위하여 SBA(Service Bus Agent)를 이용한다[3]. Service Bus Agent 는 번들링된 서비스들로부터 메시지를 받아, 이벤트가 발생한 서비스를 담당하는 PSA(Principle Service Agent)로 전송한다[3]. SA는 번들링된 서비스의 수만큼 생성되며 SA들 중 이벤트가 발생한 서비스에 해당하는 SA가 PSA의 역할을 맡아 SBA로부터 메시지를 받아 기능을 수행한다[3]. PSA는 UPPA를 통해 가입자 프로파일과 서비스 우선순위가 저장된 DB를 검색하여, 가입자가 미리 설정해 놓은 우선순위에 따라 이벤트가 영향을 주어야 하는 SA로 메시지를 보낸다[3]. 표 1은 서비스의 가입자 테이블이다. 한 가입자에 대해 가입한 서비스의 갯수만큼 테이블의 Row를 생성한다. 이 테이블에 저장된 서비스 우선 순위에 따라, 한 서비스의 이벤트 발생시 다른 서비스로의 영향을 결정한다. 예를 들어, 메신져 서비스에서 부재중 이벤트가 발생하였을 때, 메신져 서비스의 우선순위가 인터넷 전화 서비스의 우선순위보다 높은 경우 이벤트를 인터넷 전화 서비스로 전달하며, 그렇지 않을 경우, 이벤트는 무시된다.

Field	Type	value
USER_ID	VARCHAR2 (10)	예) wslee
SERVICE	VARCHAR2 (20)	예) Voip
SUBSCRIBER_ID	VARCHAR2 (30)	예) 021112222
UPDATE_PERMITION	VARCHAR2 (2)	예) Y
SERVICE_RANK	NUMBER	예) 1

[표 1] SPAS 가입 테이블

이러한 결정에 따라 최종적으로 이벤트를 받은 SA는 자신의 서비스에 해당하는 외부 서비스로 메시지를 전송한다. 연동된 외부 서비스들의 구조 및 동작은 다음 각 절에서 설명하고 있다.

를 통해서 제공되는 VoIP 서비스는 크게 SLEE(Service Logic Execution Environment)와 SIP 응용 서버 역할을 수행하는 SIP Service GW에 탑재되어 SIP기반의 VoIP호를 처리하는 VoIP 서비스와 SPAS의 Service Bus Agent로부터 IM 서비스나 OutLook서비스등 외부서비스의 Event와 관련된 SDDP메시지를 수신하여 VoIP서비스프로파일 DB의 정보를 갱신하거나 VoIP 서비스의 DB Query요청을 처리하는 VoIP Agent server 그리고, 이용자의 VoIP 서비스에 대한 프레즌스 상태에 따른 처리옵션 및 착신번호등을 담고 있는 서비스프로파일 데이터베이스로 구성되어 있다. [그림 3]은 VoIP 서비스의 전체 구성 및 연동을 나타낸 것이다.



[그림 3] VoIP 서비스 구조

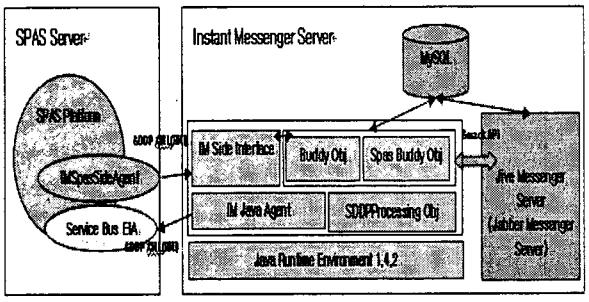
VoIP 서비스는 SIP Service GW의 SLEE에 로딩되어 SIP 단말들에서의 호처리 서비스요청을 수신하여 프로파일 데이터베이스에 따른 착신 단말로의 호설정 및 IP IVR의 일종인 IIP(Internet Intelligent Peripheral)와 IETF RFC 2897을 확장한 Audio Package 를 통한 부재중 안내멘트 송출, 착신번호 수집 등을 담당하며 호처리의 결과를 SPAS서버로 전달하는 역할을 한다.

VoIP서비스는 JRE1.4.2기반의 Java로 구현되어 있으며 SPAS서버 및 CallAgent서버와 RMI를 통해서 연동되어 있다. VoIP Call Agent는 SPAS로부터 UPDATE, COMMIT등의 SDDP메시지를 수신하는 RMI Server역할을 하는 Event Receiver와 프로파일 DB의 검색 및 갱신을 처리하는 DB Agent로 구성되어 있으며 DB Agent는 JDBC를 통해 서비스프로파일 DB와 연동한다.

서비스 프로파일 DB는 Oracle DB를 사용하며 SPAS서버와 공유하는 가입자 ID, 프레즌스 정보, 프레즌스 정보에 따른 호처리 옵션, 그리고 가입자의 단말번호 및 부재중 안내메시지 ID등의 정보로 구성되어 있다.

2.3 IM

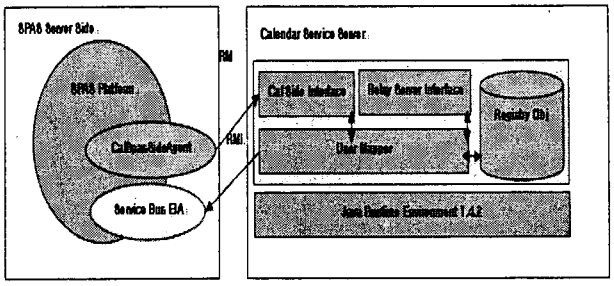
그림 4는 메신저 서비스 쪽 구성도이다. IM 서버와 SPAS쪽과 통신은 IMSpasSideAgent라는 에이전트 및 Service Bus EIA가 담당한다. 메신저 서버는 MySQL을 DB로 사용하였고, Jive 메신저라는 Jabber 프로토콜을 사용하는 메신저 서버를 사용하였다[6]. 이 메신저 서버는 Smack API를 지원하며 이 API를 이용하여 SPAS에서의 이벤트 변화를 메신저 서버에 반영하였고, 메신저 서버의 이벤트를 수집하여 SPAS 서버 쪽으로 보내는 구조이다. Jive 메신저와 SPAS와 통신 처리 모듈은 모두 자바로 구현하였다. 인스턴트 메신저 클라이언트로 Gaim 메신저를 사용하였다[8].



[그림 4] IM 서비스 구조

2.4 아웃룩

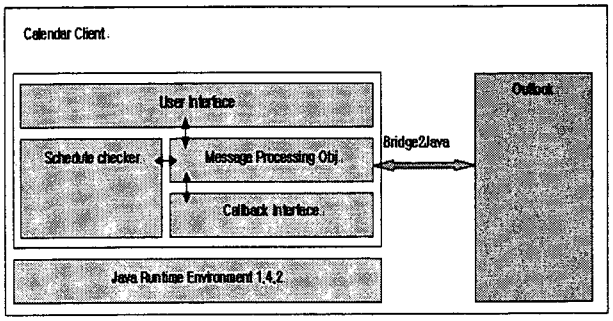
아웃룩쪽 서비스는 아웃룩 클라이언트 에이전트로부터 이 아웃룩의 상태 정보 및 이벤트를 받아서 SDDP메시지를 만들어서 SPAS를 보내주는 기능을 한다. 그림5는 아웃룩 서버쪽 구조이다. 아웃룩 클라이언트 에이전트는 실행 시에 서버에 등록하게 되고 이벤트가 발생할 경우 등록된 정보를 보고 아웃룩 클라이언트와 정보를 주고 받게 된다.



[그림 5] Outlook server architecture

아웃룩 클라이언트는 실제 사용자의 PC에 설치 된다. 아웃룩 클라이언트는 실행 후에 백그라운드로 동작하면서 아웃룩에서 일어나는 이벤트들, 그 중에서 일정 부분에서 일어나는 이벤트를 탐지하여 아웃룩 서버쪽으로 보낸다. 그림 6은 아웃룩 클라이언트 에이전트의 구조이다.

아웃룩 클라이언트 에서 아웃룩의 이벤트를 탐지하는 API로서 IBM에서 개발된 Bridge2Java를 사용하였다. 이는 COM 객체와 자바와 통신을 할 수 있도록 만든 API이다[7].



[그림 6] 아웃룩 클라이언트 에이전트 구조

3. 결론

통방 융합등 각종 서비스와 망의 통합화 추세에 따라, 기존의 수익성 있고 인기 있는 서비스를 통합하여 새로운 부가 서비스를 창출할 수 있는 번들링 서비스에 대한 요구가 늘어나고 있다. 본 논문에서 설명한 SPAS 플랫폼은 각 서비스들의 구조를 변경시키지 않고 표준적인 통신 규격에 따라 인터넷 환경에서 핵심적인 역할을 수행하고 있는 중

요 서비스들(메신져, 인터넷폰, 아웃룩 일정관리)을 결합시는데 성공하였다. 현재 개발된 플랫폼은 인터넷망의 사용에 한정되어 있으나, 추후 연동 G/W를 이용하여 유선과, 무선, 그리고, 각종 멀티미디어 콘텐츠 제공 서비스에까지 확대될 수 있을 것이다. 지속하여 연구해야할 사항은 제한되지 않은 수의 신규 서비스를 기존의 번들링 상품에 결합시키고자 할 때, 각 서비스간의 관련성에 따라 달라지는 새로운 서비스 로직을 기존의 서비스에 결정하고 반영하는 알고리즘의 추가 개발이다. 또한 여러 서비스가 결합된 만큼, 사용자들이 만족할 만한 합리적이고 안정적인 과금정책이 필요할 것이며, 그 과금정책의 복잡도 역시 증가할 것이다. 이러한 과금 기능을 수행하는 과금 에이젼트의 연구 개발이 추가로 이루어져야 한다.

[참고문헌]

1. Telcordia: Examples, Scenarios and Architecture for Bundling Services Service Peering and Aggregation Server (SPAS), 2003
2. Jean-Philippe Josep: Convergence Wireline-Wireless Network Evolution: Opportunities and Challenges, Bel Labs Journel 10, 2005
3. Telcordia: Roadmap for KT OCTAVE Initiative Deliverable 4 Convergence Services, 2005
4. Telcordia: Design Specification for the SPAS prototype Bundled Services, 2003
5. Java Agent Development Framework http://jade.tilab.com/
6. http://www.jivesoftware.com
7. http://www.alphaworks.ibm.com/tec/bridge2java
8. http://gaim.sourceforge.net