Henryk Rosa
Orange Labs
Zakład Platform Usługowych i Middleware

Telco 2.0 – realizacja koncepcji w technologii JAIN SLEE

Artykuł opisuje możliwości wykorzystania technologii JAIN SLEE, przy realizacji koncepcji Telco 2.0. Standard JAN SLEE definiuje uniwersalną architekturę serwerów aplikacyjnych, które łatwo można zintegrować z różnego typu systemami teleinformatycznymi oraz elementami sieci operatora telekomunikacyjnego. Platformy usługowe zaimplementowane w tym standardzie, dzięki swojej uniwersalności, mogą także być stosowane jako serwery udostępniające funkcjonalności sieci telekomunikacyjnych w Internecie (jako Web Services).
Pierwsza część artykułu krótko opisuje standard JAIN SLEE, w kolejnych rozdziałach omówiona jest możliwość jego zastosowania dla Telco oraz prototypowe rozwiązanie Web Gateway zaimplementowane w tej technologii przez Orange Labs.

1. Wprowadzenie

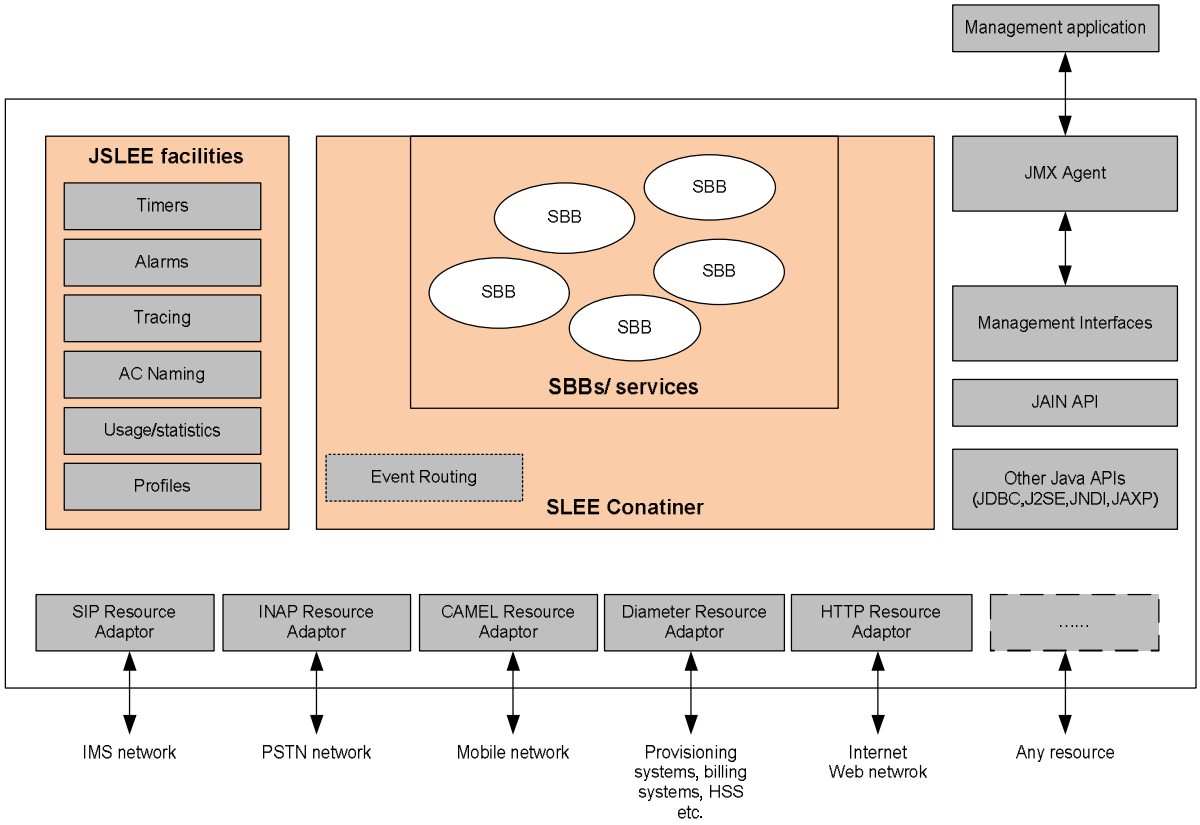
Skrót JAIN SLEE (**J**ava **API**s for **I**ntegrated **N**etworks **S**ervice **L**ogic **E**cxecution **E**nvironment często też skracany do JSLEE – Java SLEE) oznacza otwarty standard, który definiuje model tworzenia usług telekomunikacyjnych w języku programowania Java, oraz środowisko ich wykonywania. Standard opracowano w ramach organizacji JCP (Java Community Process - http://jcp.org). Pierwsza specyfikacja powstała już w marcu 2004 (JSLEE wersja 1.0 - opublikowana jako JSR 22). W lipcu 2008 ukazała się wersja 1.1 (JSR 240), która wprowadza szereg poprawek oraz dokładniejszą standaryzację komponentów Resource Adaptor (omówione dokładniej w dalszej części artykułu).

Początkowo technologia JSLEE nie cieszyła się popularnością wśród dostawców platform usługowych i operatorów telekomunikacyjnych. Wykorzystywano w większości rozwiązania platform sieci inteligentnych oparte na zamkniętych systemach usługowych, tworzonych zazwyczaj specjalnie dla konkretnej platformy sprzętowej, na której były instalowane.

Wraz z rozwojem sieci telekomunikacyjnych obserwujemy jednak coraz większą popularność języka Java oraz otwartych technologii przy dostarczaniu usług (zastosowanie technologii OSA Paraly, SIP servlet a także ostatnio JSLEE). Standardy te wprowadzają pewnego rodzaju uniwersalność oraz w pewnym stopniu przenośność usług.

2. Architektura platform usługowych JSLEE

Standardowa architektura platformy JSLEE zdefiniowana w specyfikacji została przedstawiona na rysunku nr 1 poniżej.



Rys. 1. Architektura JSLEE

Podstawowym elementem tej architektury jest środowisko wykonywania usług 'SLEE Container', w którym może być umieszczona logika usługi zdefiniowana przez programistę. Algorytm usługi opiera się zwykle na reakcji na zdarzenia (*Events*) pochodzące z różnych elementów sieci telekomunikacyjnej lub innych systemów. Takim zdarzeniem może być np. wiadomość SIP:INVITE, CAMEL:IDP, HTTP request itd. Środowisko wykonawcze zapewnia odpowiednie przesyłanie informacji o zdarzeniach pomiędzy komponentami JSLEE w formie zakodowanych obiektów języka Java (*Event Routing*). Każdy komponent umieszczany na serwerze aplikacyjnym musi spełniać wymogi SLEE ściśle określone przez specyfikację. Programista (twórca komponentów) musi zaimplementować wszystkie wymagane interfejsy. W szczególności konieczne jest zaimplementowanie '*Management Interfaces*', które dostarczają metody do kreacji instancji obiektu, jego aktywacji/dezaktywacji itd. Interfejsy te są udostępniane dla aplikacji zarządzających opartych na JMX Agent.
Poniżej przedstawiono krótko funkcję standardowych komponentów JAIN SLEE.

Komponenty Event i Activity

Jak już to zostało wspomniane zdarzenia (*Events*) są najczęściej powiązane z sygnalizacją w sieci telekomunikacyjnej (np. reprezentują pojawienie się określonej wiadomości SIP lub INAP), ale mogą również odnosić się do innych protokołów lub zasobów platformy np. zmiana danych w profilu usługi lub upłynięcie określonego czasu (*timer'a*). Źródłem *event'ów* mogą być różne komponenty JSLEE, inne usługi i samo środowisko 'SLEE Container'.

Ciąg zdarzeń powiązanych ze sobą reprezentowany jest przez kolejny obiekt Java zdefiniowany dla JSLEE – *activity*. Przykładowo połączenie telefoniczne realizowane z użyciem SIP będzie reprezentowane przez *activity*, który zostanie zainicjowany przez pierwszy *event* - pojawienie się SIP:INVITE, kolejne *event'y* mogą reprezentować następne wiadomości (180 Ringing, 200 OK, itd.).

Komponent SBB

Aplikacje usługowe JSLEE oparte są na blokach funkcjonalnych SBB (Service Building Blocks reprezentowanych również przez obiekty Java). Obiekty te przypominają elementy SIB (Service Independent Blocks) znane z zamkniętych platform IN. W każdym SBB może być zdefiniowana pewna funkcjonalność usługi. Może ona być prosta (niskopoziomowa) np. dekodowanie numerów z wiadomości SIP lub bardziej złożona np. zarządzanie obsługą przekierowania połączenia, obsługa odwołań do bazy danych itp. Wszystko zależy od koncepcji programisty - w skrajnym wypadku możliwe jest zdefiniowanie nawet całej logiki usługi telekomunikacyjnej w jednym SBB.

Zalecane jest jednak kodowanie powtarzających się funkcjonalności usług w oddzielnych blokach funkcjonalnych. Specjalne mechanizmy JSLEE umożliwiają użycie raz zdefiniowanego SBB w wielu usługach.

Resorce adaptor

Komunikacja bloków SBB z siecią telekomunikacyjną i innymi systemami odbywa się za pomocą *Resource Adaptor'ów* (RA), które to odbierają wiadomości z sieci, tworzą obiekty JAVA – *event'y* oraz inicjują *activity*. Dla każdego RA konieczne jest zdefiniowanie ściśle określonego modelu zdarzeń, które może wysyłać i odbierać od SLEE (model ten jest określany w specyfikacji jako *RA type*). May tutaj również pewną dowolność. Możliwe jest tworzenie RA niskopoziomowych gdzie zdarzenia odpowiadają poszczególnym wiadomościom protokołów, ale również wysoko-poziomowych. Możliwe jest np. stworzenie jednego RA współpracującego jednocześnie z protokołami SIP, INAP, CAMEL, który wyśle zdarzenie oznaczające pojawienie się połączenia (niezależnie, z jakiej sieci ono pochodzi). *Resorce Aadaptor* umożliwia też wysłanie przez usługę odpowiedzi na określony *event*. Aplikacja usługowa korzystając z RA API może również zainicjować nowe *activity* (np. w sytuacji, gdy to usługa ma zainicjować połączenie). *Resorce adaptor* zazwyczaj współpracuje z interfejsem API określonego zasobu dostępnego na platformie JSLEE. Np. dla SIP RA może być to API udostępnione przez system operacyjny dla protokołu TCP/IP, z kolei INAP RA może być oparty na API jakiejś karty SS7 lub oprogramowania do obsługi SIGTRAN.

RA jest komponentem JSLEE, który może być zdefiniowany dla dowolnego protokołu. Koncepcja *Resource Adaptor'a* jest również wykorzystywana do komunikacji z różnego typu systemami informatycznymi, np. z bazami danych, aplikacjami internetowymi opartymi na http itd.

Dzięki temu uzyskujemy bardzo uniwersalne i elastyczne środowisko do realizacji usług.

JAIN SLEE facilities

Innymi elementami wprowadzonymi w JSLEE są standardowe funkcjonalności serwera aplikacyjnego możliwe do wykorzystania przez programistę w kodzie swojej usługi.
Są to:

- *Timers* – możliwość zainicjowania generowania określonego *event'u* o zaprogramowanym czasie lub cyklicznie – co określony czas
- *Alarms* – usługa może wygenerować standardowy alarm, który zostanie zaraportowany przez serwer usługowy do systemu zarządzania
- *Tracing* – uporządkowane zapisywanie informacji w logach platformy usługowej, możliwa jest zmiana poziomu śledzenia dla każdego komponentu np. zapisywanie wszystkich szczegółów i komunikatów na etapie uruchamiania usługi, a w momencie uruchomienia komercyjnego (w warunkach dużego obciążenia) zapisywanie tylko wiadomości typu *Warning*.
- *Activity naming* – przydatne, gdy aplikacja musi zainicjować własną aktywność kontrolowaną później przez SLEE.
- *Ussage/Statistics* – specjalne obiekty do zapisywania informacji statystycznej, liczników wywołań itp.

- *Profiles* – obiekty zapisywane w wewnętrznej bazie danych serwera, umożliwiają przechowywanie informacji konfiguracyjnych usługi (*provisioning*). Profile są dostępne z poziomu interfejsu zarządzającego, zmiana profilu może skutkować też wygenerowaniem *eventu* dla usługi (zależnie od definicji twórcy usługi).

Przy definiowaniu kodu usługi programista może również używać innych standardowych API i standardów języka Java (JNDI, JDBC, J2SE itd.).

Uniwersalna architektura JSLEE oparta na otwartym standardzie posiada wiele zalet z punktu widzenia operatora telekomunikacyjnego.
Są to przede wszystkim:

- **otwartość** – możliwość integracji na jednej platformie standardowych komponentów zaimplementowanych przez różnych dostawców lub wykorzystanie rozwiązań open source. Założeniem jest również przenośność komponentów pomiędzy serwerami aplikacji spełniającymi standard JSLEE. Należy jednak pamiętać, że są one oparte na określonych zasobach dostępnych na platformie i nie zawsze będzie to możliwe.
- **elastyczność** w poziomie reprezentacji zasobów sieci – możliwość budowania usług opartych na niskim poziome interakcji (np. wiadomości poszczególnych protokołów telekomunikacyjnych) lub wysoko-poziomowej (np. udostępnienie twórcom usługi Parlay API poprzez Parlay Resource Adaptor)
- **konwergencja i uniwersalność architektury** – serwer aplikacji oparty na JSLEE może być wykorzystywany, jako klasyczna platforma IN (SCP poprzez INAP RA), jako SIP AS, jako Parlay Applicaion Serwer, jako Web Serwer dla ParalyX itd. Łatwa integracja z różnego typu zasobami sieci i protokołami telekomunikacyjnymi umożliwia zastosowanie JSLEE również w rozwiązaniach typu Service Broker, translator protokołów lub wspomniany na wstępie WebGateway, który udostępnia usługi w Internecie oparte na zasobach sieci - zgodnie z koncepcją Telco 2.0.

3. Możliwość wykorzystania JSLEE w koncepcji Telco 2.0

Decyzja o udostępnieniu zasobów sieci telekomunikacyjnej zgodnie z koncepcją Telco 2.0 wiąże się z koniecznością wdrożenia przez operatora całej infrastruktury opartej na modelu Service Delivery Framework umożliwiającego współpracę różnych platform usługowych i rozwiązań teleinformatycznych.

Konieczne jest:

- Zapewnienie dobrego środowiska wykonywania usług udostępniającego, które udostępnia Web Services, ale także umożliwia współistnienie różnych usług (np. service brokering – zastosowanie kilku usług dla jednego połączenia).
- Dostarczenie elementów wspierających realizację modelu biznesowego (rozliczanie z zewnętrznymi dostawcami), administrację i utrzymanie interfejsu do sieci (zarządzanie i kontrolowanie dostępu)
- Udostępnianie informacji o możliwościach interfejsu, dostarczenie przykładowych aplikacji, dokumentacji oraz narzędzi do tworzenia usług, zapewnienie wymiany doświadczeń pomiędzy użytkownikami
- Zapewnienie bezpieczeństwa dostępu do sieci telekomunikacyjnych.

Platforma usługowa wykorzystywana jako WebGateway powinna więc umożliwiać łatwą integrację różnego typu zasobów ze świata IT oraz telekomunikacji. Biorąc pod uwagę elastyczność w integracji z różnego typu zasobami oraz uniwersalność technologii środowisko wykonawcze oparte na JAIN SLEE wydaje się tutaj dobrym rozwiązaniem. Koncepcja Resource Adaptorów umożliwi szybką integrację i współpracę ze wszystkimi systemami operatora, które są

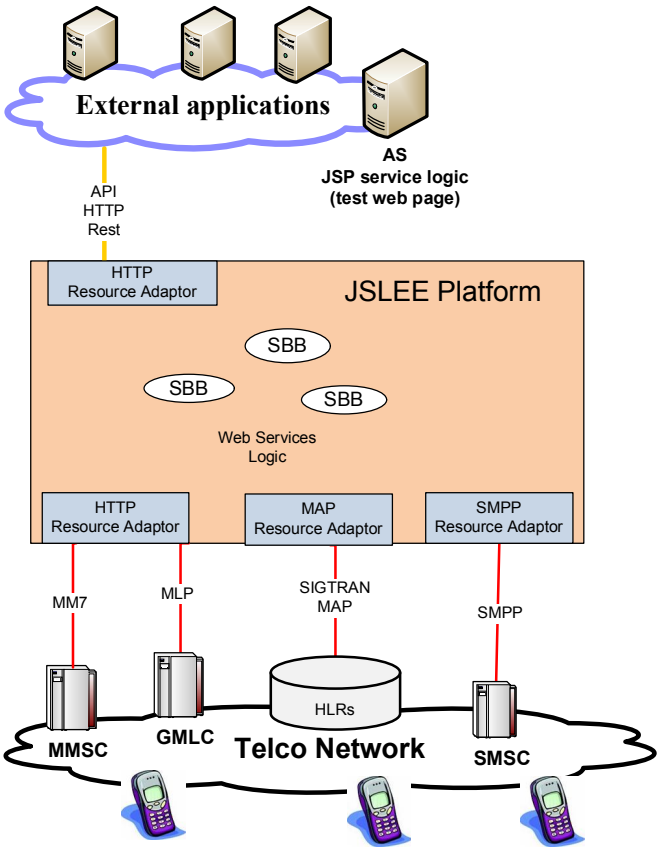
niezbędne do realizacji całości rozwiązania.

4. Testowa implementacja WebGW z wykorzystaniem JSLEE

W ramach ewaluacji platform usługowych JAIN SLEE zaimplementowano w polskim Orange Labs testowe rozwiązanie WebGateway, które umożliwia udostępnianie w Internecie szeregu funkcjonalności sieci telekomunikacyjnej:

- Wysyłanie/odbieranie SMS
- Wysyłanie/odbieranie MMS
- Wysyłanie/odbieranie wiadomości USSD
- Lokalizacja terminala abonenckiego
- Status terminala
- Pobranie informacji o aktualnym czasie z WebGW.

Architektura rozwiązania została przedstawiona na rysunku nr 2.



Rys. 2. Testowe rozwiązanie Web GW z wykorzystaniem JSLEE

W celu udostępnienia tych funkcjonalności platforma JSLEE została zintegrowana z następującymi elementami sieci telekomunikacyjnej:

- SMS Center z wykorzystaniem protokołu Short Message Per-to-per Protocol
- MMS Center poprzez interfejs Mutimedia Messaging 7
- Gatewaly Mobile Location Centre z wykorzystaniem Mobile Location Protocol
- HLR z wykorzystaniem MAP (SS7: Mobile Application Part) w celu odpytania o aktualny status terminala abonenckiego lub przesłania USSD.

Wszystkie funkcjonalności udostępniane są dla aplikacji w sieci Web poprzez HTTP resource adaptor. Aplikacja zewnętrznego dostawcy wywołuje Web Service wysyłając specjalnie zdefiniowany http request GET a w odpowiedzi (w 200OK) otrzymuje żądane informacje od logiki usługi JSLEE (lokalizacja, status terminala), bądź też potwierdzanie wysłania SMS, MMS, USSD.

W przypadku WebServices inicjowanych od strony sieci (np. odbieranie SMS) aplikacja z sieci Web musi najpierw zgłosić chęć otrzymywania informacji - wysłać odpowiedni request do logiki JSLEE. Na SMSC i MMSC zostały uruchomione specjalne numery testowe i wszystkie wiadomości adresowane na te numery są dostarczane do logiki usługi JSLEE. W przypadku, gdy są zarejestrowane jakieś zewnętrzne aplikacje zainteresowane otrzymywaniem wiadomości logika usługi na WebGW przesyła ją do nich w postaci HTTP request z zakodowaną treścią.

Prototypowe rozwiązanie uruchomiono na komercyjnej platformie JSLEE Rhino™ dostarczanej przez firmę OpenCloud™ oraz na implementacji JSLEE open source pobranej z www.mobicents.org.
Jak widać nie jest to kompleksowy system wspierający wszystkie procesy biznesowe związane z Telco 2.0. Jednak już ten prototyp pokazuje przydatność otwartej technologii JSLEE w budowie tego typu rozwiązań.

5. Wnioski

Stosowanie otwartych standardów opartych na JAIN SLEE przy budowie infrastruktury usługowej operatora umożliwia szybsze i łatwiejsze wdrożenie koncepcji Telco 2.0. Uniwersalność technologii sprawia, że łatwiejsza jest integracja rozwiązań telekomunikacyjnych z różnego typu zasobami i systemami IT, co ma bardzo duże znaczenie dla realizacji całości systemu udostępniającego usługi dla zewnętrznych dostawców.

Otwartość standardu przekłada się również na szybkość rozwoju platformy usługowej związaną z łatwością łączenia różnych komponentów pochodzących od różnych dostawców, rozwiązań open source, oraz możliwością rozwijania nowych elementów przez operatora we własnym zakresie (np. w ramach prac R&D).

Literatura

1. Stephen Hall, NSN, *Evolving the Service Creation Environment*, 2010.
2. http://jcp.org/en/jsr/summary?id=240 JSR 240: JAIN™ SLEE (JSLEE) v1.1
3. http://jcp.org/en/jsr/summary?id=22 JSR 22: JAIN™ SLEE API Specification
4. www.mobicents.org Mobicents JAIN SLEE Open Source implementation