TLS-Technologie
email office@tls-technologies.com
www http://www.tls-technologies.com/

Zaawansowane narzędzia programowania rozproszonego

Karol Gołąb
karol.golab@tls-technologies.com

28 listopada 2001

[1]

Streszczenie

Omówienie i porównanie popularnych standardów mechanizmów komunikacyjnych: CORBA, JavaSpaces, Java Message Service, JXTA.

1 Wstęp

Wraz ze wzrostem powszechności korzystania z sieci komputerowych (Internetu w szczególności) rośnie gwałtownie znaczenie oprogramowania rozproszonego – złożonego z wielu programów wykonywanych na osobnych komputerach połączonych przez sieć.

Do niedawna model budowy oprogramowania pracującego w sieci był oparty na architekturze klient-serwer. Do tworzenia takiego oprogramowania wystarczały stosunkowo proste technologie i narzędzia.

Obecnie sytuacja ulega zmianie. Prosty model klient-serwer coraz mniej wystarcza. Rosnąca złożoność oprogramowania rozproszonego wymaga użycia coraz bardziej skomplikowanych mechanizmów komunikacyjnych.

[1] Współzałożyciel firmy TLS-Technologie(s). Doktorant w Instytucie Informatyki UW. Zajmuję się wszelkimi aspektami sieci komputerowych, przede wszystkim problemami algorytmicznymi.

2 Założenia

Jednym z powodów powstania nowych modeli komunikacji była chęć usunięcia wielu niekorzystnych założeń wymuszanych na twórcach oprogramowania przez architekturę klient-serwer.

Do podstawowych założeń stojących za omawianymi tu systemami a służących rozwiązaniu problemów istniejących w środowisku aplikacji rozproszonych należą:

- Współdziałanie (ang. *interoperability*) — zapewnienie możliwości współdziałania pomiędzy różnymi aplikacjami używającymi tego samego systemu. W szczególności możliwość komunikowania się niezależnie napisanych programów.
- Skalowalność — łatwe zwiększanie wielkości i wydajności systemów.
- Przenośność — jak największa (czy wręcz całkowita) niezależność od platformy programowej czy sprzętowej.
- Jednorodność — zapewnienie jednolitych interfejsów dostępu do informacji na różnych platformach, prezentacja informacji we wspólnym formacie.
- Powszechność (ang. *ubiquity*) — udostępnienie i rozpowszechnienie na możliwie wielu platformach, niezależnie od mocy obliczeniowej.
- Asynchroniczność — rozdzielenie w czasie operacji wysyłania i odbierania komunikatu. Docelowo możliwość skomunikowania się dwóch aplikacji nawet w sytuacji gdy ich czas życia nie pokrywa się.
- Wyszukiwanie informacji — udostępnienie metod znajdowania nieznanego *a priori* źródła informacji.

3 Systemy i produkty

Obecnie na rynku istnieje wiele konkurencyjnych rozwiązań, opartych na różnych architekturach. Większość architektur posiada jednak wiele cech wspólnych wynikających z wymienionych wcześniej założeń.

- Warstwa pośrednicząca (ang. *middle-tier*) – zajmuje się tłumaczeniem danych z/do różnych formatów, ukrywa implementację i niższe warstwy (np. system operacyjny). Zazwyczaj zapewnia także większą skalowalność systemu.
- Modularność – pozwala na instalację aktualnie potrzebnej części funkcjonalności a także na stopniowe rozszerzanie aplikacji poprzez dodawanie kolejnych podsystemów.

3.1 CORBA

Pierwszy standard CORBA'y (*Common Object Request Broker Architecture*) został opublikowany przez OMG w 1991 roku. CORBA to zarówno architektura jak i cała infrastruktura (język specyfikacji interfejsów IDL, oprogramowanie pośredniczące ORB, szereg dodatkowych usług takich jak przestrzeń nazw czy wsparcie dla transakcji) pozwalająca na łatwe tworzenie aplikacji rozproszonych działających w modelu obiektowym.

Do CORBA'y należą także narzędzia wspomagające proces tworzenia oprogramowania – np. kompilatory tworzące ze specyfikacji szkielet implementacji obiektu w danym języku.

Jednym z głównych założeń jest rozdzielenie interfejsu obiektu (specyfikowanego w języku IDL) od jego implementacji, która może zostać wykonana w jednym z wielu wspieranych języków (C, C++, Java, SmallTalk, Lisp, Python i inne). Aplikacja rozproszona to zbiór współpracujących ze sobą obiektów, być może zaimplementowanych w różnych językach.

Do najpopularniejszych implementacji Open Source należą szybki, lecz wciąż niekompletny *omniORB* i większy, ale wolniejszy *mico*.

3.2 JavaSpaces

Technologia JavaSpaces, będąca częścią platformy J2EE, została oparta na systemie krotek Linda. Model zbudowano wokół współdzielonych przez sieć przestrzeni (ang. *spaces*), w których aplikacje mogą umieszczać bądź wyszukiwać dane.

JavaSpaces zapewnia, że dane przechowywane w przestrzeniach są trwałe (ang. *persistent*), w szczególności nie znikają wraz z procesem aplikacji, która je tam umieściła. Dane są dostępne na zasadzie asocjacji – można je wyszukiwać na podstawie zawartości, co umożliwia łatwe współdziałanie niezależnie napisanych aplikacji. Ponadto implementacja JavaSpaces troszczy się o transakcyjność i bezpieczeństwo wszystkich operacji.

Ponieważ w przestrzeniach przechowuje się obiekty Java'y, to przy użyciu JavaSpaces można wymieniać kod wykonywalny (a dokładniej |ByteCode| Javy) na równi z wymianą danych. W szczególności pozwala to na dynamiczne rozszerzanie aplikacji o dodatkową funkcjonalność zdefiniowaną już po utworzeniu aplikacji.

3.3 JMS

Technologia JMS (Java Message Service) jest integralną częścią platformy J2EE. Pozwala ona na wymianę komunikatów pomiędzy luźno powiązanymi (ang. *loosely coupled*) aplikacjami.

JMS zapewnia asynchroniczną, pewną i (opcjonalnie) transakcyjną wymianę informacji w dwóch modelach:

- Komunikacja producent – konsument (ang. *publish–subscribe*). Producenci wysyłają dane do systemu, wiążąc je z zarejestrowanym wcześniej tematem. Konsumenci od-

bierają dane dotyczące wybranych tematów. Każdy komunikat może zostać odebrany przez dowolną ilość konsumentów.

- Komunikacja punkt do punktu (ang. *point–to–point*), zbudowana na kolejkach. Aplikacja wysyła dane do wybranej kolejki, która z kolei dostarcza je do konkretnego odbiorcy.

Dodatkowo technologia JMS pozwala na dynamiczną konfigurację trasowania, umożliwiając w szczególności określenie odbiorcy dla danej kolejki czy tematu podczas uruchamiania systemu.

Najpopularniejsze implementacje Open Source to *JBossMQ*, *JORAM* i *OpenJMS*.

3.4 JXTA

System JXTA, zainicjowany przez Sun Microsystems przy okazji tworzenia platformy J2EE, to zestaw specyfikacji i protokołów umożliwiających wymianę dowolnego typu informacji, niezależnie od typu transportu.

JXTA rozwija model partnerski (ang. *peer-to-peer*) wraz z infrastrukturą potrzebną dla zapewnienia uniwersalności. Aktualnie w skład sytemu wchodzą następujące protokoły:

- Protokół znajdowania ogłoszeń (ang. *advertisement*) zgłaszanych przez partnerów (ang. *peer*) bądź ich grupy.
- Protokół znajdowania partnerów bądź ich grup dla zadanych kryteriów wyszukiwania.
- Protokół informacyjny służący do dynamicznego poznawania możliwości partnera.
- Protokół grup pozwalający na przyłączenie się do grupy czy też poznania jej składu i wymagań.
- Protokół znajdowania trasy do partnera pozwalający na ustanowienie połączenia nawet w przypadku, gdy partnerzy nie widzą się nawzajem bądź używają różnych transportów.
- Protokół wiązania ogłoszeń i transportów.

3.5 Przydatne adresy

- http://www.omg.org
- http://java.sun.com/products/javaspaces/
- http://developer.java.sun.com/developer/Books/JavaSpaces/introduction.html
- http://java.sun.com/products/jms/

- http://developer.java.sun.com/developer/technicalArticles/Networking/messaging
- http://www.jxta.org