Proiectarea Sistemelor Software Complexe

Curs 5 –Tehnologii Middleware Bazate pe Mesaje

Tehnologiile middleware bazate pe mesaje sunt foarte importante pentru construirea de sisteme software complexe. De cele mai multe ori aceste tehnologii sunt folosite ca liant pentru a interconecta sisteme software independente, transformându-le într-un singur sistem. Sistemele software interconectate prin tehnologii middleware bazate pe mesaje pot fi dezvoltate utilizând tehnologii diferite și pot rula pe diferite platforme hardware și/sau software. Interconectarea sistemelor software prin acest tip de middleware nu necesită rescrierea aplicațiilor existente și nici modificări substanțiale (riscante). Interconectare sistemelor software prin tehnologii middleware bazate pe mesaje presupune comunicarea prin intermediul unei cozi de mesaje. Acest tip de tehnologie middleware este de cele mai multe ori folosită pentru interconectarea sistemelor software nou dezvoltate cu cele existente sau pentru comunicarea intre sisteme software aparținând unor companii diferite. Tehnologiile middleware bazate pe mesaje pot fi împărțite în două categorii: cozi de mesaje și tehnologii middleware bazate pe modelul de comunicare publică-subscrie.

5.1 Cozi de Mesaje

Cozile de mesaje reprezintă un model de comunicare slab-cuplat și asincron. Pot fi folosite și tehnologii middleware de tip cozi de mesaje sincrone, dar chiar dacă acestea au și avantaje el pot duce la arhitecturi fragile dacă toate componentele și legăturile dintre ele trebuie să fie funcționale tot timpul pentru ca sistemul să funcționeze corect.

Infrastructura bazată pe cozi mesaje decuplează expeditorul și destinatarul prin intermediul unei cozi de mesaje. Expeditorul poate trimite mesaje către destinatar având garanția că mesajul va ajunge la destinație chiar dacă rețeaua nu funcționează sau destinatarul nu este conectat în momentul în care mesajul este transmis. Emițătorul doar spune middleware-ului să transmită mesajul după care își continuă activitatea. Emițătorul nu va ști ce proces sau aplicație va prelucra mesajul (Fig. 5.1).

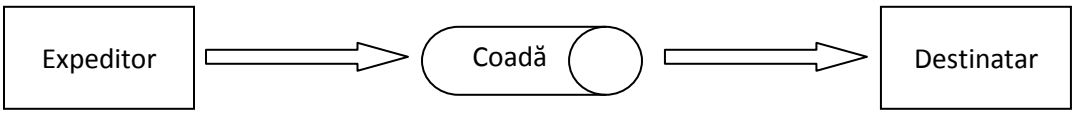

Fig. 5.1. Modelul de comunicare pentru tehnologiile middleware bazate pe coadă de mesaje.

Tehnologiile middleware bazate pe cozi mesaje sunt de cele mai multe ori implementate ca și un server care poate gestiona mai mulți clienți conectați simultan. Astfel pentru a se realiza decuplarea expeditorului și a destinatarului sunt folosite una sau mai multe cozi în care expeditorii pot pune mesaje, respectiv din care destinatari pot citi mesaje. Un astfel de server poate să gestioneze mai mult cozi simultan, respectiv mai multe mesaje transmise simultan prin intermediul firelor de execuție organizate

in pool-uri de fire de execuție. Un singur proces poate trimite mesaje către mai multe cozi, respectiv fiecare coadă poate fi interogată de unul sau mai mulți destinatari. Identificare cozilor se face pe baza numelor (Fig. 5.2).

Un server de cozi trebuie să îndeplinească câteva funcții. Trebuie să poată recepționa mesaje de la expeditori și să confirme recepționarea mesajelor. Trebuie să pună mesajul în coada specificată de

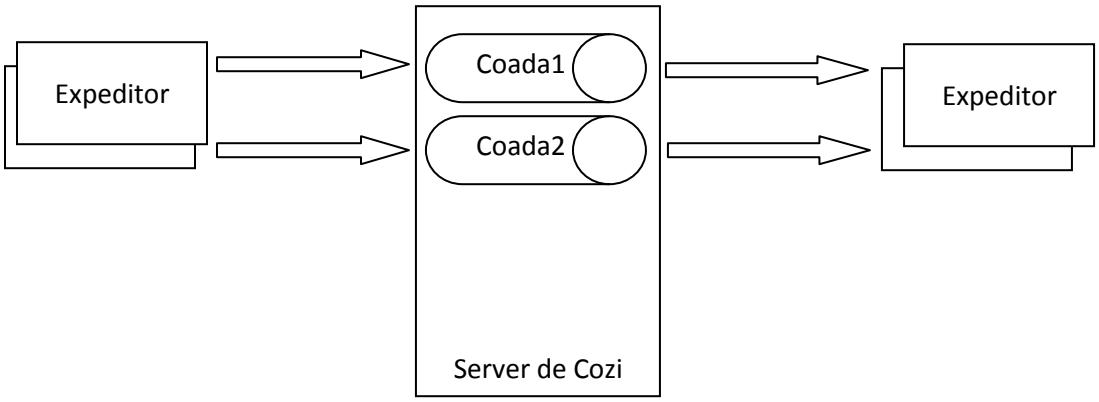

Fig. 5.2. Server de cozi de mesaje.

expeditor. Trebuie să păstreze mesajele în coadă până când un destinatar le citește (pot fi trimise mai multe mesaje până când un destinatar le citește). Mesajele sunt extrase de destinatari după principiul primul-intrat-primul-ieșit (FIFO). Atunci când un destinatar cere un mesaj, cel mai vechi mesaj va fi transmis destinatarului, iar odată ce mesajul a fost recepționat cu succes de destinatar va fi șters din coadă.

Datorită caracterului asincron și a gradului mare de decuplare tehnologiile middleware bazate pe cozi de mesaje pot fi folosite pentru a rezolva o varietate mare de probleme. În continuare vor fi enumerate câteva situații în care pot fi folosite astfel de tehnologii:

- expeditorul nu are nevoie de un răspuns; dorește doar să transmită un mesaj și să continue activitatea; acest model de comunicare se mai numește transmite și uită (send-and-forget);
- expeditorul nu are nevoie de un răspuns imediat la un mesaj cerere; se poate ca destinatarul să aibă nevoie de câteva minute pentru a procesa cererea, iar expeditorul poate realiza alte activități în acest timp;
- destinatarul sau conexiunea dintre acesta și expeditor poate să nu fie operațională tot timpul; expeditorul bazându-se în acest caz pe middleware pentru a transmite mesajul destinatarului atunci când se poate stabili comunicarea cu acesta.

De cele mai multe ori utilizarea unei simple cozi de mesaje nu este suficientă, fiind nevoie de garanții și performanțe care nu pot fi oferite de o coadă de mesaje normală. În continuare vor fi discutate astfel de caracteristici.

5.1.1 Livrarea Mesajelor

În majoritatea sistemelor software livrarea mesajelor trebuie să fie garantată. Expeditorul trebuie să fie sigur că mesajul va ajunge la destinatar. De exemplu, dacă se consideră un sistem financiar care procesează tranzacții pentru cărți de credit, atunci când posesorul unei cărți de credit face o plată se poate ca acea plată să fie pusă într-o coadă urmând să fie procesată mai târziu. Dacă echipamentul hardware pe care este găzduită coada se defectează atunci dacă informația despre tranzacție nu este

recuperată nu se va mai realiza plata către magazin. Un astfel de sistem nu poate tolera pierderea mesajelor fiind necesară garantarea mesajelor.

Creșterea fiabilității cu care sunt livrate mesajele se poate realiza doar cu scăderea performanței. De aceea de cele mai multe ori trebuie realizată o balansare între fiabilitatea cu care sunt livrate mesajele și performanță. De obicei o coadă de mesaje pune la dispoziție trei niveluri de fiabilitate:

- efort redus – mesajele sunt stocate doar în memorie existând riscul ca mesajele să se piardă dacă apare o defecțiune în sistem sau la rețeaua de comunicare;
- persistență – mesajele sunt stocate atât in memorie cât si pe disc, ele putând ajunge la destinatar chiar dacă pare o defecțiune a sistemului sau a rețelei de comunicare;
- tranzacțional – mesajele pot fi grupate în tranzacții de tipul “totul sau nimic”.

Au fost realizate o serie de studii pentru a măsura diferența în ceea ce privește performanța intre cele trei niveluri de fiabilitate. Astfel, în funcție de tehnologie se poate observa o scădere a performanței între 30% și 80% atunci când se trece de la nivelul “efort redus” la cel “persistență”. Între nivelul “persistență” și cel “tranzacțional” nu există o diferență prea mare.

5.1.2 Tranzacții

De obicei cozile de mesaje tranzacționale sunt construite peste cele care oferă suport pentru persistență. În acest caz operațiile cu mesaje sunt strâns legate de codul aplicației, astfel că mesajele care formează o tranzacție nu sunt livrate până când aplicația expeditor nu commite tranzacția. Cozile de mesaje tranzacționale permit crearea de grupuri de mesaje care sunt transmise ca și o singură unitate indivizibilă.

În ceea ce privește recepționarea mesajelor care formează o tranzacție acesta se face tot într-un context tranzacțional. Mesajele care formează o tranzacție vor ajunge la destinatar împreună în ordinea în care au fost transmise, fiecare mesaj va fi transmis o singură dată, ele vor fi șterse din coadă doar dacă destinatorul commite tranzacția în care au fost recepționate mesajele. Dacă tranzacția este anulată mesajele rămân în coadă fiind disponibile pentru o procesare ulterioară.

Mesajele tranzacționale pot fi coordonate atât la recepție cât și la transmisie cu alte operații tranzacționale, de exemplu actualizarea unei baze de date. Astfel o aplicație poate inițializa o tranzacție, trimite un mesaj, actualizează o bază de date și comite tranzacția. În acest fel se asigură că mesajul nu va fi disponibil până când baza de date nu a fost actualizată.

5.1.3 Clustere

De cele mai multe ori cozile de mesaje reprezintă mecanisme de comunicare primare. De aceea atunci când o coadă de mesaje se defectează aplicațiile nu mai pot comunica. Pentru a se mări fiabilitatea unui astfel de sistem software se poate recurge la utilizarea de clustere de cozi de mesaje, ceea ce înseamnă că vor fi disponibile mai multe cozi de mesaje aflate pe mai multe mașinii.

Modul în care funcționează un cluster este dependent de implementare, dar la nivel de principii toate implementările sunt asemănătoare. Astfel un cluster este format din mai multe servere de cozi. Fiecare server conține același set de cozi, iar distribuirea acestor cozi între servere este transparentă pentru aplicația client. Clienții se comportă ca și când ar fi doar un singur server.

Când un client trimite un mesaj una din cozi este selectată, iar mesajul este stocat în acea coadă. La fel și la recepție, este selectată o coadă și se scoate din ea un mesaj. Direcționarea cererii către o coadă

anume este responsabilitatea clusterului de cozi. Această direcționare poate fi statică atunci când clientul face prima interogare sau dinamică pentru fiecare interogare (acest mod de funcționare nu este utilizabil pentru o aplicație care trebuie să primească mesajele în ordinea în care au fost transmise).

Utilizarea clusterelor are două avantaje: dacă un server de cozi se defectează atunci celelalte server vor fi disponibile, iar al doilea avantaj este faptul că se poate face o balansare a încărcări pe serverele de cozi, ceea ce duce la îmbunătățirea performanței sistemului software.

5.1.4 Cozi de Mesaje Bidirecționale

Cu toate că tehnologiile bazate pe cozi de mesaje sunt prin definiție decuplate și asincrone se pot folosi și pentru dezvoltarea de sistem software care necesită o comunicare sincronă și un grad mai ridicat de cuplare. În astfel de situații cozile de mesaje sunt folosite pentru a trimite mesaje către un destinatar. Mesajul conține informații care permit identificarea cozii în care trebuie scris răspunsul la mesaj. După ce a trimis mesajul expeditorul așteaptă primirea mesajului răspuns în coada specificată.

Câteva din motivele pentru care se recurge la folosirea cozilor de mesaje pentru comunicarea sincronă sunt enumerate în continuare:

- tehnologiile bazate pe cozi de mesaje pot fi folosite pentru interconectarea de aplicații existente cu un cost redus și un risc minim; adaptori care realizează interfațarea între diferitele tehnologii bazate pe cozi de mesaje sunt disponibili sau pot fi realizați cu un efort redus; aplicațiile nu trebuie rescrise pentru a putea fi integrate într-un sistem pe scară largă;
- tehnologiile bazate pe cozi de mesaje sunt disponibile pe majoritatea platformelor; acest lucru facilitează integrarea cu aplicațiile mai vechi sau cu cele ale unor companiilor partenere.

5.2 Modelul Publică – Subscrie (Publish-Subscribe)

Tehnologiile middleware bazate pe cozi de mesaje sunt foarte eficiente în ceea ce privește dezvoltarea sistemelor software slab cuplate, dar evident există și anumite limite. Principala limitare constă în faptul că tehnologiile middleware sunt tehnologii care permit doar comunicarea unu-la-unu. Astfel un singur expeditor trimite un singur mesaj către o singură coadă și un singur destinatar extrage mesajul din coadă. Pentru a se elimina această limitare a fost propus un alt model bazat pe mesaje si anume modelul publică-subscrie.

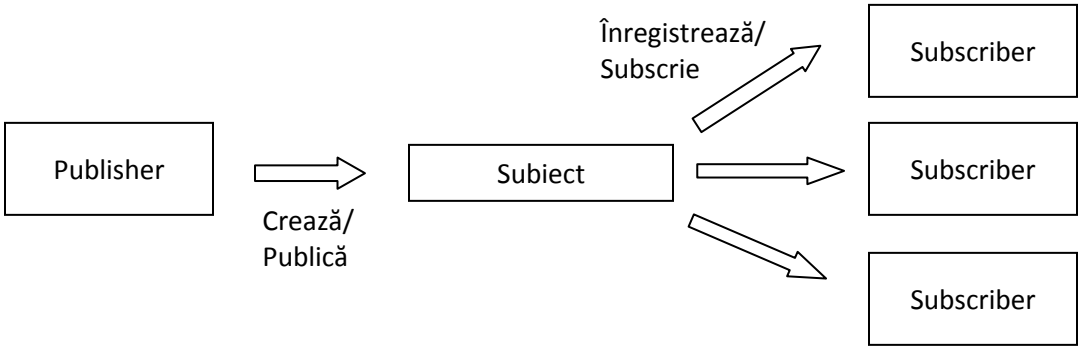

Fig. 5.3. Modelul publică-subscrie.

Modelul de comunicare publică-subscrie extinde modelul de comunicare bazat pe cozi pentru a permite implementarea tuturor modelelor de comunicare: 1 la mai mulți, mai mulți la mai mulți și mai mulți la 1. Un *publisher* trimite o singură copie a unui mesaj către un *subiect*. Subiectul reprezintă un nume logic pentru modelul publică-subscrie, el fiind echivalentul cozii din cazul modelului de comunicare bazat pe cozi de mesaje. *Subscriber-ii* ascultă mesajele care sunt trimise pentru un topic și care îi interesează. Ca și în cazul cozilor de mesaje și modelul publică-subscrie presupune existența unui server al cărui rol este acela de a distribui mesajele primite pentru un anumit topic către toți clienți care sau înregistrat pentru acel topic (Fig. 5.3).

Din punctul de vedere al decuplării, modelul publică-subscrie prezintă două proprietăți importante. Expeditorul și destinatarul sunt decuplații, nici unul din ei neavând informații despre celălalt. Cea de a două proprietate se referă la faptul că pentru fiecare topic pot fi mai multe aplicații care publică, respectiv care au subscris pentru a primii mesaje. Astfel o aplicație care publică poate să alterneze perioadele când este online cu cele în care este offline. Pe de altă parte o aplicație care s-a înregistrat pentru un anumit subiect poate să fie înregistrată și pentru alte subiecte sau să modifice dinamic lista de subiecte pentru care este înregistrată.

În cazul modelului de comunicare publică-subscrie, tehnologia middleware are rolul de a gestiona subiectele, de a ține evidența aplicațiilor care au subscris pentru un anumit subiect și de a transmite mesajele recepționate pentru un anumit subiect către toate aplicațiile care s-au înregistrat pentru acel subiect. La fel ca și în cazul cozilor, mesajele pot fi stocate doar în memorie sau pot fi stocate și pe disc. Se poate de asemenea să se specifice durata de viața a unui mesaj (time to live). Această durată specifică intervalul de timp în care se va încerca trimiterea mesajului către toate aplicațiile care s-au înregistrat pentru subiect și care sunt active. După expirarea acestui interval de timp mesajul fiind șters.

Implementarea unui middleware bazat pe modelul publică-subscrie se poate face fie folosind comunicare de tipul punct-la-punct sau comunicare de tipul multicast. Evident utilizarea comunicării multicast va duce la obținerea unor performanțe superioare.

Pentru tehnologiile middleware de tipul publică-subscrie subiectul este echivalentul unei cozi. Numele unui subiect este o valoare de tipul string care poate fi definită fie prin program fie într-o consolă de configurare. Orice subiect are un nume logic care trebuie cunoscut de toate aplicațiile care publică pentru acel subiect sau care s-au înregistrat pentru a primi mesaje de la acel subiect. Majoritatea tehnologiilor existente oferă suport pentru construirea unor structuri arborescente de nume unele oferă suport și pentru folosirea așa numitor “wildcards” (ex.: upt/ac/is/pssc, upt/*/pssc etc.).

Bibliografie

[1] Ian Gorton. Essential Software Architecture. Editura Springer. 2006.
[2] http://java.sun.com/products/jms/tutorial/
[3] http://java.sun.com/javaee/5/docs/tutorial/doc/