STUDIA INFORMATICA 2003
Volume 24 Number 3 (55)

Marek NOWAK
Wojskowa Akademia Techniczna, Wydział Cybernetyki

ARCHITEKTURA SYSTEMU ROZPROSZONYCH KOMPONENTÓW Z MIGRUJĄCYMI USŁUGAMI

Streszczenie. Artykuł przedstawia architekturę systemu rozproszonych komponentów z migrującymi usługami. Zostały wymienione zasadnicze cechy rozpatrywanej klasy systemów. Końcowa część artykułu zawiera opis architektury systemu, na którą składają się struktura oprogramowania i przepływy danych między procesami systemu.

Słowa kluczowe: system rozproszonych komponentów, migracja kodu, migracja usług.

ARCHITECTURE OF DISTRIBUTED COMPONENTS SYSTEM WITH SERVICE MIGRATION

Summary. The purpose of this paper is to present architecture of distributed components system with service migration. There are shown major features of such system taking into consideration. The last part consists of the description of system architecture as a set of software structure and data flows between system processes.

Keywords: distributed components system, code migration, service migration.

1. Wprowadzenie

Sieci komputerowe ewoluują w szybkim tempie w kilku obszarach jednocześnie. Fenomen wzrostu *rozmiaru* sieci nie jest ograniczony tylko i wyłącznie do Internetu. Organizacje oparte na sieciach inter- i intranetowych również powiększają swoje sieci, a dzieje się to za sprawą dostępności taniejącego sprzętu i jest stymulowane potrzebą tworzenia jednorodnych, otwartych i efektywnych kanałów informacyjnych wewnątrz i pomiędzy organizacjami. Efekt uboczny w postaci wzrostu obciążenia sieci wymusza na naukowcach i przemyśle podjęcie

wysiłków zmierzających do zapewnienia zadowalającej wydajności infrastruktury komunikacyjnej. Wzrost wydajności sieci komputerowych jest jednocześnie przyczyną i skutkiem *dostępności* i *wszechobecności* sieci komputerowych. Dostępność sieci oznacza zmierzch ery rozpatrywania sieci komputerowej jako drogiego elementu dodanego i obecnie należy ją widzieć jako podstawowy element infrastruktury przetwarzania danych. Wszechobecność sieci charakteryzuje rozszerzające się możliwości korzystania z sieci poprzez stosowanie stałych łączy oraz mobilnego dostępu do sieci. Mobilni użytkownicy mogą zmieniać swoje położenie geograficzne, nie tracąc jednocześnie możliwości korzystania z sieci firmowej lub globalnej. Tym samym znikają dotychczasowe ograniczenia w dostępie do systemów informatycznych. W zgodnej opinii ekspertów na pierwszy plan wysuwa się przepustowość sieci oraz zakres dostępnych aplikacji.

Dostępność i wszechobecność sieci komputerowych stymuluje także stały rozwój technologii WWW powszechnie wykorzystywanej nie tylko przez profesjonalistów, lecz również przez tzw. naiwnych użytkowników sieci. Połączenie sieci komputerowych z serwisem WWW stworzyło nowe rodzaje aplikacji oraz nowe obszary zastosowań informatyki. Obecnie nowoczesna sieć komputerowa stanowi medium wspierające nowe formy kooperacji i komunikacji pomiędzy użytkownikami oraz użytkownikami a systemami (np. kioski multimedialne, systemy *call center*, telewizja interaktywna, aplikacje P2P).

Wielość nowoczesnych kanałów dostępu oraz przede wszystkim otwartość systemu na użytkowników sieci globalnej może prowadzić do dużego obciążenia systemu wyrażonego liczbą żądań obsługi przybyłych do systemu w jednostce czasu. W przypadku popularnych portali kształtuje się ono na poziomie nawet kilkuset tysięcy żądań na minutę [1]. Skala obciążenia systemu przy stałym wzroście liczby potencjalnych użytkowników stanowi jeden z podstawowych kłopotów inżynierów systemowych w zakresie zapewniania zadowalającej wydajności systemu. W konsekwencji dostępność i wszechobecność sieci komputerowych prowadzi do problemu *skalowalności* systemu. Skalowalność to zdolność oprogramowania lub systemu komputerowego do sprawnego działania w warunkach rosnącej liczby użytkowników, zwiększającej się objętości przetwarzania danych lub rozrostu liczby węzłów sieci komputerowej [10]. Wyróżnia się dwa rodzaje skalowalności [2]: poziomą i pionową. W zakresie tej publikacji zastosowanie znajduje przede wszystkim skalowalność pionowa, a więc migracja systemu w kierunku większych i wydajniejszych serwerów lub wieloserwerów (farm serwerów).

Rozwiązanie problemu skalowalności systemu jest uzależnione od charakteru zmian wymagań użytkowników lub oczekiwań właściciela systemu. Obejmuje ono metody sprzętowe (sieci, serwery klastry itp.) i programowe. Serwery aplikacji jako podstawowy obiekt zainteresowania w dalszych rozważaniach stanowi przykład rozwiązań programowych. Serwer

aplikacji funkcjonuje w warstwie środkowej trójwarstwowego modelu referencyjnego systemów klient-serwer.

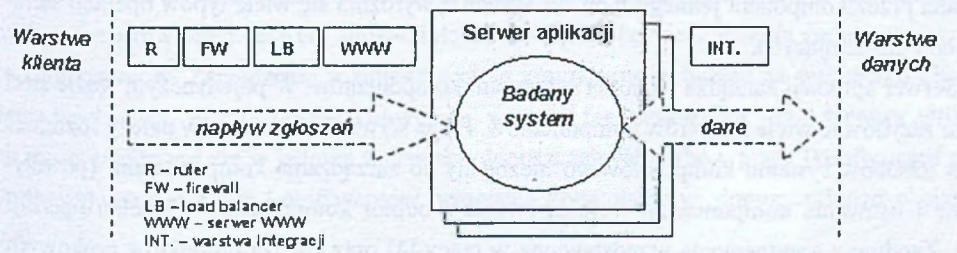
2. System rozproszonych komponentów z migrującymi usługami

Trójwarstwowe systemy klient-serwer składają się z trzech głównych warstw: klienta, środkowej i danych. Przedmiotem dalszych rozważań będzie jedynie warstwa środkowa, natomiast pozostałe warstwy stanowią jej otoczenie, gdzie żądania obsługi napływają z warstwy klienta. Warstwa danych stanowi składnicę danych niezbędnych do funkcjonowania aplikacji warstwy środkowej.

Warstwa środkowa także składa się z wielu warstw. Rysunek 1 przedstawia najważniejsze rodzaje serwerów stosowanych w tej warstwie. Należą do nich:

- serwery komunikacyjne (R),
- serwery firewall (FW),
- serwery wyrównywania obciążenia (LB),
- serwery WWW,
- serwery aplikacji,
- serwery integracji (INT).

Rozpatrywany system rozproszonych komponentów (SRK) rezyduje w serwerze aplikacji (rys. 1). Infrastruktura sprzętowa serwera aplikacji jest zazwyczaj złożona z wielu niezależnych węzłów połączonych siecią komputerową. Ze względu na wydajność, dostępność oraz niezawodność obsługi klientów oprogramowanie serwera aplikacji łączy niezależne węzły w "jeden organizm", zatem mamy do czynienia z przykładem systemu rozproszonego z wbudowanymi rozwiązaniami zapewniającymi skalowalność systemu w wymiarze pionowym i poziomym.



Rys. 1. Architektura warstwy środkowej

Fig. 1. Middle-tier architecture

Przyjmuje się założenie, że aplikacje serwera aplikacji są zaimplementowane w technologii rozproszonych komponentów. Komponent jest rozumiany jako [3]: element oprogramowania, który może być wykorzystywany w różnych sieciach, systemach operacyjnych, pakietach oprogramowania. Komponent jest obiektem, który nie jest przywiązany do określonego programu, języka programowania lub implementacji.

Serwer aplikacji zarządza wieloma węzłami, w których rezydują komponenty. Wszystkie dostępne komponenty stanowią zbiór rozproszonych po węzłach sieci komponentów. Każdy z komponentów ma ściśle zdefiniowany interfejs zawierający wykaz operacji, jakie może wykonywać. Zakłada się, że w systemie może rezydować wiele komponentów różnego typu. Pojedynczy węzeł sieci może obsługiwać wiele instancji komponentu tego samego typu. Typ komponentu jest jednoznacznie definiowany przez zbiór operacji przez niego implementowanych. Przydział komponentów do węzłów sieci może ulegać zmianie w czasie pracy systemu. Zmiana rozmieszczenia komponentów wprowadza także zmianę rozmieszczenia operacji (usług). Tym samym można mówić o *migracji usług*.

Napływające do systemu żądania obsługi są traktowane jako zadania do wykonania. Pojedyncze zadanie stanowi skończoną sekwencję wywołań operacji. Wykonanie każdego zadania rozpoczyna się od tzw. operacji sterującej, w algorytmie której są zaimplementowane wywołania operacji niesterujących. Zatem operacja sterująca poprzez znajomość ciągu wywołań operacji niesterujących jednoznacznie definiuje typ zadania. Zadanie dowolnego typu można przedstawić jako ciąg wywołań operacji w postaci:

(operacja sterująca, [operacja niesterująca])

Wykonanie pojedynczego zadania może odbyć się przy współudziale wielu węzłów poprzez wywoływanie rezydujących w nich operacji niesterujących. Liczba typów zadań jest skończona i równa liczności zbioru operacji sterujących. Komponenty mogą implementować wiele operacji sterujących, a tym samym w systemie można wykonywać zadania wielu typów. Niezależnie od rodzaju operacji (sterująca, niesterująca) operacja dowolnego typu jest implementowana przez komponent jednego typu. W systemie wyróżnia się wiele typów operacji sterujących i niesterujących.

Serwer aplikacji zarządza wieloma serwerami komponentów. W pojedynczym węźle sieci może rezydować wiele serwerów komponentów. Przez serwer komponentów należy rozumieć zbiór zasobów systemu komputerowego niezbędny do zarządzania komponentami (powoływanie i usuwanie komponentów, przekazywanie i odbiór komunikatów wywołań operacji itp.). Zgodnie z argumentacją przedstawioną w pracy [4] oraz analizą produktów rynkowych przyjęto założenie, iż serwer komponentów zawiera co najwyżej jedną instancję komponentu określonego typu. Serwer komponentów zarządza wieloma komponentami, lecz każdy z nich jest innego typu. Na potrzeby obsługi wywołań operacji implementowanych przez rezydujące

komponenty serwer komponentów zarządza dwiema pulami wątków: sterujących (wywołania operacji sterujących) i niesterujących (wywołania operacji niesterujących). Przykładowo, serwer komponentów można utożsamiać z wirtualną maszyną Javy, procesem systemu Windows NT 4.0/2000 z komponentami COM/COM+ itp.

Wymienione założenia na SRK zostały przedstawione w sposób formalny w języku teorii grafów i sieci, a zarys modelu formalnego zamieszczono w [5] i [6]. Spośród wielu poczynionych założeń szerokiego komentarza wymaga migracja usług.

3. Migracja usług

Zmiana rozmieszczenia komponentów wprowadza także zmianę rozmieszczenia operacji, a więc można mówić o migracji usług realizowanych przez SRK. Migracja usług należy do podstawowych założeń charakteryzujących rozpatrywaną klasę systemów. Docelowym wynikiem rozważań w zakresie SRK jest opracowanie metody zarządzania tak rozumianym środowiskiem rozproszonych komponentów. Analiza oprogramowania komercyjnego prowadzi do wniosku, że poczynione założenie dotyczące migracji usług odróżnia SRK od rozwiązań dostępnych na rynku. Podczas specyfikacji założeń funkcjonalnych SRK dokonano przeglądu wiodących produktów rynku serwerów aplikacji: Microsoft Windows 2000, IBM WebSphere, iPlanet Application Server, BEA WebLogic, Oracle Application Server. Administrator każdego z wymienionych serwerów aplikacji ma możliwość zmiany przydziału komponentów do węzłów sieci, lecz wymaga to albo ponownego uruchomienia systemu, albo implementacji własnego oprogramowania z funkcją zmiany rozmieszczenia komponentów w czasie pracy systemu, co nie jest zadaniem trywialnym.

W SRK przyjęto założenie, że zmiana rozmieszczenia komponentów będzie należała do zadań serwera aplikacji i będzie mogła zachodzić w czasie pracy systemu bez udziału administratora. W tym zakresie rola administratora będzie ograniczała się do zdefiniowania określonych parametrów pracy serwera aplikacji, których wartość będzie sterowała zmianą przydziału komponentów. Zarządzanie rozmieszczeniem komponentów bazuje na wskaźniku obciążenia węzła sieci. W zależności od chwilowej wartości tego wskaźnika węzeł serwera aplikacji może znajdować się w jednym z czterech dopuszczalnych trybów pracy (konfiguracji początkowej, powoływania komponentów, usuwania komponentów, dopuszczalnego obciążenia).

Decyzja o zmianie rozmieszczenia komponentów jest podejmowana przez scentralizowany menedżer komponentów. Spośród czterech wyróżnionych trybów pracy węzła w dwóch z nich (tryby powoływania i usuwania komponentów) może być podjęta decyzja o zmianie rozmieszczenia komponentów. Migracja usług jest wynikiem powoływania i usuwania kom-

ponentów. Każdorazowa decyzja w tym zakresie jest podejmowana osobno dla każdego z węzłów kandydujących po uprzednim uwzględnieniu globalnego stanu systemu i ograniczeń wynikających z konfiguracji węzłów (np. ograniczenie w postaci maksymalnej liczby komponentów w węźle). Operacja usunięcia komponentu może być odroczona w czasie ze względu na konieczność zakończenia oczekujących wywołań operacji. W przeciwieństwie do tradycyjnego rozumienia mobilności kodu usunięcie komponentu nie jest skorelowane z natychmiastowym powołaniem komponentu tego samego typu, lecz w innym węźle sieci. Operacje usuwania i powoływania komponentów są niezależne od siebie. Zasięg zmian rozmieszczenia usług wynikający z wykonania pojedynczej decyzji menedżera komponentów jest ograniczony do jednego węzła.

Migrację usług należy rozpatrywać w kontekście dobrze znanego w dziedzinie systemów rozproszonych zagadnienia migracji kodu [7][8]. Początkowo przedmiotem migracji były procesy (Condor, Locus, MOSIX). Wraz z rozwojem technologii obiektów rozproszonych rozszerzono tę listę także o obiekty. Obecnie bardzo szeroko dyskutowanym zagadnieniem jest oprogramowanie implementujące technologię mobilnych agentów. W przeciwieństwie do początkowych prób migracji procesów, gdzie przeniesieniu kodu z węzła do węzła podlegała cała przestrzeń adresowa procesu, obserwuje się rozwój rozwiązań umożliwiających przenoszenie także fragmentów kodu (aplety Javy, COOL, Emerald, Sumatra). Eksperci systemów rozproszonych wyróżniają dwa zasadnicze rodzaje migracji kodu [7][8]: *silną* i *słabą*. Silna mobilność kodu oznacza kontynuację obliczeń w nowej lokalizacji od miejsca ich uprzedniego zatrzymania. W przypadku słabej mobilności kodu podczas przenoszenia nie jest przekazywany stan jednostki przetwarzającej (proces, obiekt, agent, wątek itp.) co wymusza rozpoczęcie obliczeń od początku po ich przybyciu do nowej lokalizacji. Z uwagi na niezależność operacji powoływania i usuwania komponentów oraz każdorazowego tworzenia komponentu startując od stanu początkowego rozwiązanie przyjęte dla SRK można sklasyfikować jako słabą mobilność kodu.

4. Architektura systemu rozproszonych komponentów

Projekt systemu rozproszonych komponentów wymaga rozwiązania podobnych zagadnień projektowych jak w przypadku tradycyjnych systemów rozproszonych [9]: nazewnictwo, komunikacja, struktura oprogramowania, przydzielanie obciążeń i spójność systemu. Jednak nie wszystkie z wymienionych zagadnień projektowych mieszczą się w ramach prowadzonych dla SRK rozważań. Przyjmuje się założenie, że problemy nazewnictwa, komunikacji i spójności systemu są rozwiązane, a ich implementacja już znajduje się w serwerze aplikacji. W zakresie architektury SRK rozważa się problematykę zarządzania aplikacjami serwera aplika-

cji, rozdziału ruchu wchodzącego, monitorowania systemu oraz migracji usług, które mieszczą się w zakresie struktury oprogramowania i przydzielania obciążeń z pracy [9]. W związku z tym zostały zdefiniowane cztery podstawowe problemy badawcze, które muszą zostać szczegółowo rozwiązane:

P1. Początkowe rozmieszczenie komponentów

P2. Wybór węzła koordynatora wykonania zadania

P3. Zmiana rozmieszczenia komponentów

P4. Wybór miejsca wykonania kolejnej operacji zadania.

W celu rozwiązania tych problemów badawczych zaproponowano algorytmy ich rozwiązywania bazujące na charakterystykach obciążenia systemu udostępnianych przez podsystem monitorowania. W ramach implementacji SRK algorytmy rozwiązujące problemy P1÷P4 mieszczą się w kilku warstwach oprogramowania, które rozszerzają podstawowe moduły serwera aplikacji. Potrzeba rozwiązania problemów P1÷P4 wprowadziła konieczność wyróżnienia kilku procesów przetwarzania odpowiedzialnych za funkcjonowanie SRK w zakresie wymienionych problemów. Wprowadzono następujące rodzaje procesów:

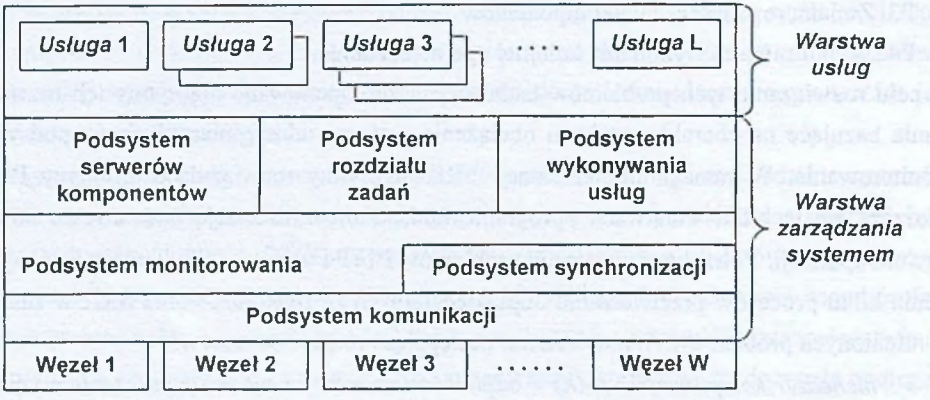
- *menedżer komponentów (MK)* – odpowiada za zarządzanie rozmieszczeniem komponentów oraz udostępnia pozostałym menedżerom dane w zakresie rozmieszczenia komponentów i obciążenia węzłów sieci,
- *menedżer rozdziału zadań (MRZ)* – odpowiada za wybór węzła koordynatora wykonania zadania (węzła wykonującego operację sterującą zadania),
- *menedżer wykonywania zadań (MWZ)* – odpowiada za realizację kolejnych wywołań operacji niesterujących zadania,
- *serwer komponentów (SK)* – odpowiada za utrzymywanie komponentów (powoływanie i usuwanie komponentów),
- *wątek obsługi* – odpowiada za wykonanie operacji; wyróżnia się dwa rodzaje wątków: na potrzeby obsługi operacji sterujących (WOS), na potrzeby obsługi operacji niesterujących (WNS).

Wszystkie wymienione rodzaje procesów składają się na architekturę SRK, którą można przedstawić jako złożenie dwóch elementów:

- struktury warstwowej SRK,
- przepływu danych pomiędzy procesami SRK.

Struktura warstwowa zawiera ramową budowę SRK określającą składowe systemu oraz zależności między nimi. Rysunek 2 przedstawia warstwy oprogramowania SRK. Zawierają one dwie podstawowe warstwy:

- *warstwę zarządzania systemem* – przede wszystkim odpowiadającą za tworzenie środowiska pracy aplikacji, które ukrywa obecność wielu niezależnych maszyn, zarządza farmą serwerów oraz dostarcza mechanizmów interakcji między aplikacjami,
- *warstwę usług* – odpowiadającą za zarządzanie powołanymi komponentami i wykonywanie żądań obsługi (zadań) przybyłych do systemu.



Rys. 2. Podsystemy i warstwy serwera rozproszonych komponentów

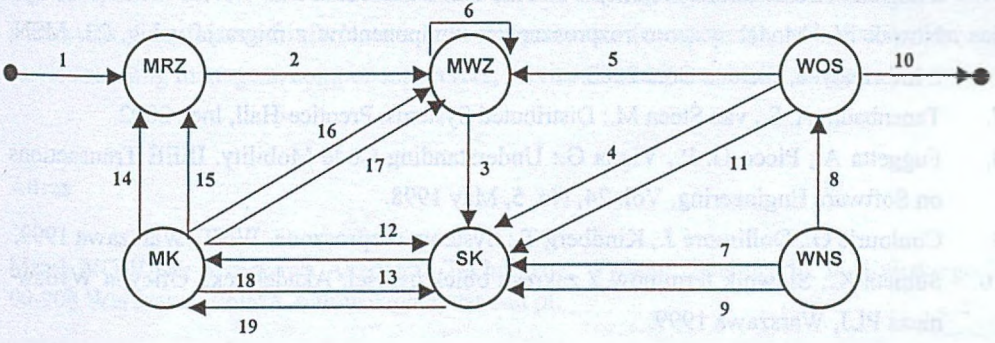
Fig. 2. Subsystems and layers of distributed components system

Warstwa zarządzania systemem stanowi bardzo rozbudowany element architektury SRK, stąd jest ona złożona z podwarstw odpowiadających kilku wyróżnionym podsystemom. Podstawowy dla funkcjonowania całego SRK jest podsystem komunikacji. Z racji jego umiejscowienia bezpośrednio nad węzłami sieci spełnia on oprócz dostarczania mechanizmów komunikacji także rolę warstwy oprogramowania zapewniającą przezroczystość systemu. Podsystemy komunikacji i synchronizacji wspólnie tworzą fundament dla podsystemów wyższych warstw. Zadaniem podsystemu synchronizacji jest przede wszystkim zapewnienie niezbędnej w tej klasie systemów rozproszonych spójności czasu, co oznacza konieczność implementacji algorytmów synchronizacji zegarów poszczególnych węzłów sieci. W tym zakresie przyjęto rozwiązanie propagacji czasu węzła wzorcowego.

Podsystem monitorowania jest implementowany przez agentów menedżera komponentów i zajmuje się zbieraniem charakterystyk obciążenia węzłów sieci. Podsystem serwerów komponentów grupuje wszystkie serwery komponentów rezydujące w węzłach SRK. Zadania tego podsystemu są tożsame z wymienionymi powyżej zadaniami SK. Podsystem rozdziału zadań odpowiada MRZ i rozwiązuje problem P2. Natomiast podsystem wykonywania usług grupuje wszystkie MWZ rezydujące po jednym w każdym węźle sieci i rozwiązuje problem P4.

Najwyżej położona w hierarchii warstwa usług odpowiada za zarządzanie rozmieszczeniem komponentów i w tym zakresie jest implementowana przez MK. W tej warstwie jest rozwiązywany problem P3. Warstwę usług można także postrzegać jako zbiór wszystkich usług realizowanych przez system. Z perspektywy klienta żądania obsługi napływające do systemu trafiają do tej warstwy i w niej są obsługiwane. Oznacza to, że warstwa zarządzania systemem jest niewidoczna dla klientów systemu, jej istnienia jest jedynie świadomy dostawca usług, czyli właściciel systemu.

Diagram przepływu danych zamieszczony na rysunku 3 przedstawia wzajemne interakcje i przepływy informacji między składowymi SRK. Odpowiada on diagramowi zerowego poziomu konstruowanego w ramach strukturalnych metodyk tworzenia systemów informatycznych. Diagram na rysunku 3 zawiera podzbiór wszystkich przepływów danych. Dla uproszczenia i osiągnięcia większej czytelności nie umieszczono przepływów danych odpowiadających pobieraniu z magazynu danych przez MK, MRZ i MWZ konfiguracji początkowej systemu.



Legenda:
1. Obiór żądania obsługi; 2. Wybór węzła koordynatora; 3. Wybór SK; 4. Start zadania; 5. Wywołanie oper. niester.; 6. Wybór węzła wykonania oper. niester.; 7. Start oper. niester.; 8. Odp. oper. niester.; 9. Koniec oper. niester.; 10. Odp. zadania; 11. Koniec zadania; 12. Zmiana rozmieszczenia komponentów; 13. Potwierdzenie zmiany rozm. komp.; 14. Stan systemu - char. dynamiczne; 15. Stan systemu - rozm. komponentów; 16. Stan systemu - char. dynamiczne; 17. Stan systemu - rozm. komponentów; 18. Przekaż charakterystyki SK; 19. Charakterystyki SK (dynamiczne);

Rys. 3. Diagram przepływu danych SRK

Fig. 3. Data flow diagram of SRK

Przedstawione założenia funkcjonalne SRK wraz z jego architekturą nie wyczerpują całości zagadnień rozpatrywanych dla wyróżnionej klasy systemów. Zasadniczym elementem, który wykracza poza temat niniejszej publikacji, jest metoda zarządzania migrującymi usługami. Zdefiniowano także szereg miar wydajności i efektywności systemu [5]. Osobnym zagadnieniem jest przeprowadzenie serii eksperymentów symulacyjnych w celu wyznaczenia wartości zdefiniowanych miar oraz sprawdzenia ich zmienności w zależności od różnorodnej konfiguracji systemu. Wykonanie tego przedsięwzięcia wymaga implementacji symulatora

SRK. Jako platformę implementacji symulatora wybrano sieć lokalną pod kontrolą systemu operacyjnego QNX w wersji 4.25.

LITERATURA

1. IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher. *http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg245858.pdf.*
2. Orfali R., Harkey D., Edwards J.: The Essential Client/Server Survival Guide. John Wiley & Sons, Inc., 1996.
3. Orfali R., Harkey D., Edwards J.: The Essential Distributed Objects Survival Guide. John Wiley & Sons, Inc., 1996.
4. Szyperski C.: Oprogramowanie komponentowe. WNT, Warszawa 2001.
5. Nowak M.: Miary wydajności systemów rozproszonych komponentów z migrującymi usługami. Archiwum Konferencji PTETiS, Vol. 14, 2002.
6. Nowak M.: Model systemu rozproszonych komponentów z migracją usług. 23. MSN, t. Informatyka, Zielona Góra 2002.
7. Tanenbaum A. S., van Steen M.: Distributed Systems. Prentice-Hall, Inc., 2002.
8. Fuggetta A., Picco G. P., Vigna G.: Understanding Code Mobility. IEEE Transactions on Software Engineering, Vol. 24, No. 5, May 1998.
9. Coulouris G., Dollimore J., Kindberg T.: Systemy rozproszone. WNT, Warszawa 1999.
10. Subieta K.: Słownik terminów z zakresu obiektowości. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.

Recenzent: Dr inż. Dariusz R. Augustyn

Wpłynęło do Redakcji 17 kwietnia 2003 r.

Abstract

The 3-tier client-server systems have experienced phenomenal growth over the past few years, placing heavy load on middle tier. Today's application servers also process an increasing number of requests for dynamic pages generated by software components working in middle tier. The server load and their performance characteristics became even more critical.

The first part presents an introduction into 3-tier client-server systems. It is centered around a place of distributed components system in multi-tier software environment. The second and third parts contain a framework for understanding system functionality. The framework is centered around major features of server application that include two elements: software structure and service mobility.

The research environment consists of independent software components. Each component has several operations (services). Component manager can create, destroy or destroy and create components in other host that makes service migration. Service mobility is the most important feature taking into consideration. The working task consists of sequence of operation providing by components. Tasks share system resources one another. The main problem is component's management in the way in which system works effective.

Finally, this paper presents architecture of distributed components system with service migration. The architecture consists of several software layers and data flows between them. Figure 2 shows relationship between subsystems (communication, synchronization, monitoring, component server, task distribution and service running subsystems) as multi-layer software stack. The flows of information between processes (components, task distribution and service running managers, component servers, service threads) are shown in figure 3.

Adres

Marek NOWAK: Wojskowa Akademia Techniczna, Wydział Cybernetyki, ul. Kaliskiego 2, 00-908 Warszawa, Polska, mnowak@isi.wat.edu.pl.