Oprogramowanie middleware

Dariusz Wawrzyniak

Politechnika Poznańska
Instytut Informatyki
ul. Piotrowo 2 (CW, pok. 5)
60-965 Poznań

Dariusz.Wawrzyniak@cs.put.poznan.pl
Dariusz.Wawrzyniak@put.edu.pl

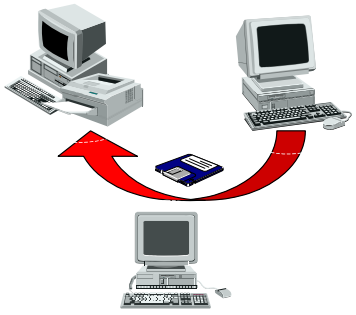
www.cs.put.poznan.pl/dwawrzyniak/Middleware

+48 (61) 665 2963

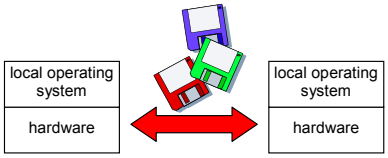
System rozproszony

- Zbiór niezależnych komputerów i zasobów komputerowych zdolnych do kooperacji (np. poprzez sieć komputerową), postrzeganych przez użytkownika jako całościowo spójny system.
- Ogólny cel projektowy systemu rozproszonego: stworzenie przezroczystego, otwartego, elastycznego, wydajnego i skalowalnego mechanizmu współdzielenia zasobów.

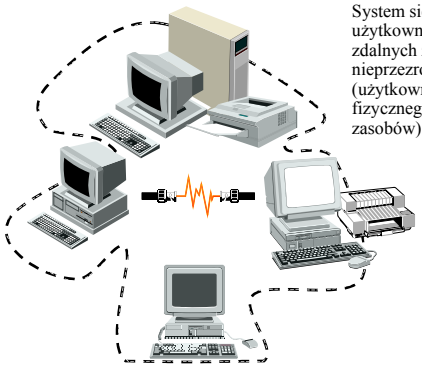
Niezależne komputery



Niezależne komputery — organizacja oprogramowania

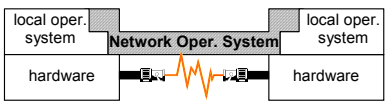


System sieciowy

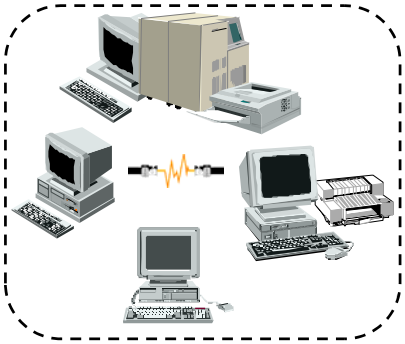


System sieciowy umożliwia użytkownikom dostęp do zdalnych zasobów w sposób nieprzezroczysty (użytkownicy są świadomi fizycznego rozproszenia zasobów)

System sieciowy — organizacja oprogramowania

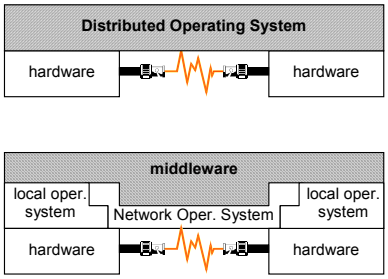


System rozproszony



Rozproszony system operacyjny umożliwia użytkownikom dostęp do zdalnych zasobów w sposób przezroczysty (jest jednolity sposób dostępu do zasobu lokalnego i zdalnego)

System rozproszony — organizacja oprogramowania



Koncepcja dostępu do współdzielonych zasobów

- użytkownik zasobu — moduł (program, proces) zgłaszający zapotrzebowanie na zasób, żądający dostępu (wykonania operacji)
- zarządca zasobu (ang. resource manager) — moduł oprogramowania odpowiedzialny za udostępnianie zasobu (koordynację i realizację operacji dostępu, żądanych przez użytkownika)

System rozproszony — podstawowe własności (1)

- współdzielenie/współużytkowanie zasobów (sprzętu, danych)
  - poprawa efektywności wykorzystania (np. drukarka)
  - możliwość interakcji (np. system plików, baza danych)
- otwartość
  - standaryzacja reguł dostępu do usług oraz kompletność interfejsów
  - interoperacyjność (zdolność do współdziałania)
  - przenośność
- rozszerzalność
  - zdolność do rozbudowy i rekonfiguracji,
  - możliwość **dodawania** nowych

System rozproszony — podstawowe własności (2)

- współbieżność
  - możliwość współbieżnego (równoczesnego) ubiegania się o zasoby i ich użytkowania
- skalowalność
  - możliwość dostosowania systemu do rosnących rozmiarów problemów i wymagań użytkowników
- tolerowanie uszkodzeń
  - odporność na awarie przez redundancję (sprzętu, danych) oraz możliwość odtworzenia spójnego stanu po awarii
- przezroczystość (transparentność)
  - ukrywanie fizycznego odseparowania zasobów przed użytkownikiem

Przezroczystość (1)

- przezroczystość dostępu
  - ukrywanie różnic w reprezentacji danych,
  - zagwarantowanie jednolitego sposobu dostępu do zasobów, niezależnie od tego, czy są to zasoby lokalne, czy zdalne
- przezroczystość położenia
  - identyfikacja zasobów niezależna od ich fizycznej lokalizacji (np. dzięki usłudze nazw)
- przezroczystość replikacji
  - utrzymywanie i udostępnianie kilku egzemplarzy tego samego zasobu (kopii) w taki sposób, jak gdyby użytkownik widział i działał tylko na jednym

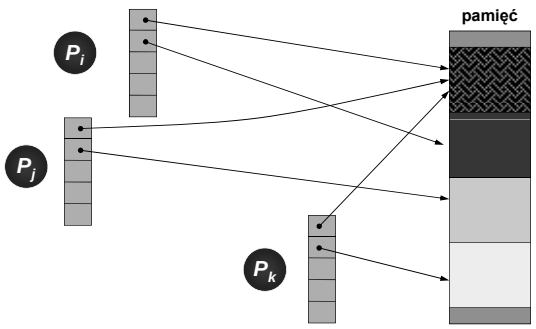
Przezroczystość (2)

- przezroczystość migracji
  - zmiana fizycznej lokalizacji zasobu nie powoduje zmian w sposobie ich identyfikacji ani w sposobie dostępu
- przezroczystość relokacji
  - fizyczna zmiana lokalizacji zasobu może być dokonana w sposób niewidoczny (nieodczuwalny) dla użytkownika w czasie realizacji dostępu
- przezroczystość współbieżności
  - realizacja współbieżnego dostępu do zasobu w taki sposób, że konkurujące procesy nie przeszkadzają sobie wzajemnie (nie są świadome rywalizacji)
- przezroczystość ukrywania awarii
  - zdolność ukrycia przed użytkownikiem faktu chwilowych nieprawidłowości funkcjonowania

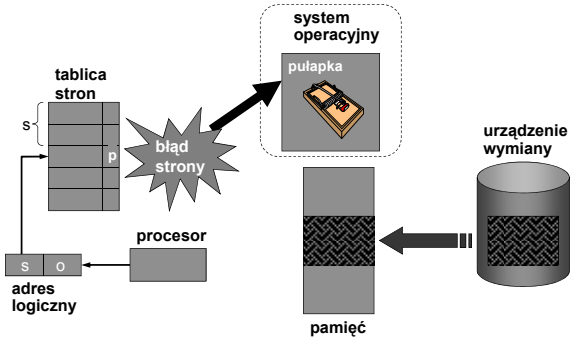
Paradygmat interakcji pomiędzy zdalnymi jednostkami

- Dostęp do wspólnej pamięć
- Przekazywanie wiadomości (komunikatów)
  - komunikacja synchroniczna lub asynchroniczna
  - komunikacja przejściowa lub nieustanna
  - komunikacja punkt-punkt lub grupowa
- Wywoływanie zdalnych procedur
- Dostęp do zdalnych obiektów
- Komunikacja strumieniowa

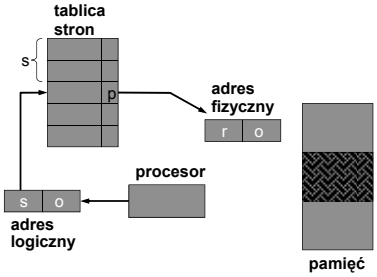
Współdzielenie pamięci (przykład w systemie ze stronicowaniem)



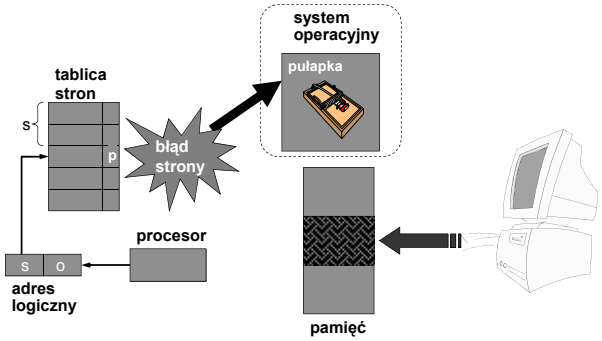
Obsługa błędu strony w systemie pamięci wirtualnej



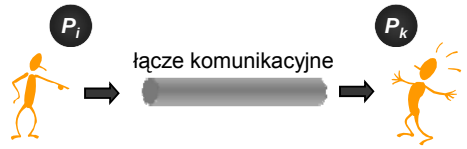
Obsługa błędu strony w systemie pamięci wirtualnej



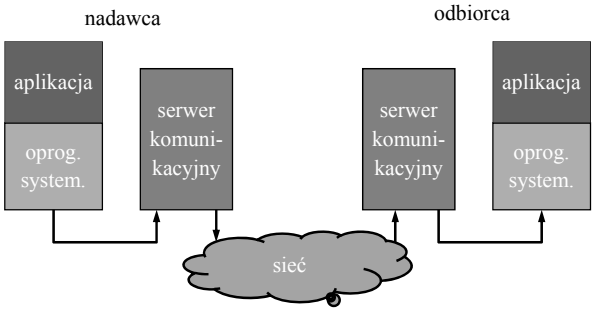
Obsługa błędu strony w systemie wirtualnej pamięci rozproszonej



Przekazywanie wiadomości (komunikatów)



Przesyłanie wiadomości — model systemu



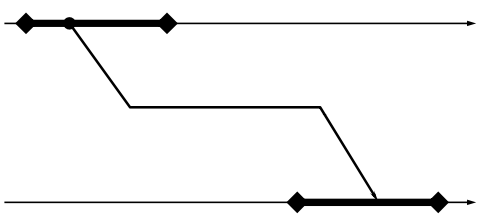
Trwałość komunikacji

- komunikacja przejściowa (ang. transient communication) — wiadomość jest przekazywana (utrzymywana w podsystemie komunikacyjnym) pod warunkiem, że działa nadawca i odbiorca tej wiadomości
- komunikacja nieustanna (ang. persistent communication) — wiadomość jest przechowywana w celu doręczenia do odbiorcy nawet, gdy odbiorca nie działa w danej chwili, a nadawca zakończył działanie po wysłaniu tej wiadomości

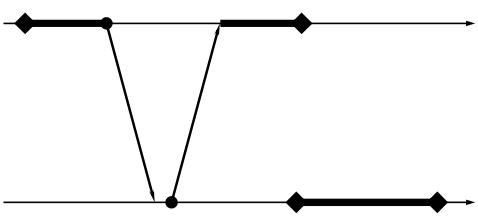
Synchroniczność komunikacji

- komunikacja synchroniczna — nadawca kontynuuje działanie dopiero, gdy wiadomość znajdzie się w buforze odbiorczym lub zostanie doręczona do adresata
- komunikacja asynchroniczna — nadawca kontynuuje działanie zaraz po przekazaniu wiadomości do podsystemu komunikacyjnego

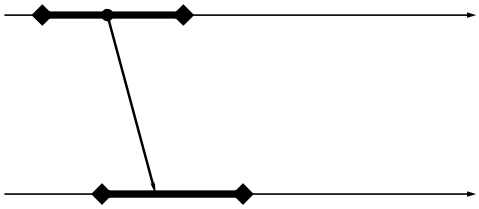
Komunikacja nieustanna asynchroniczna



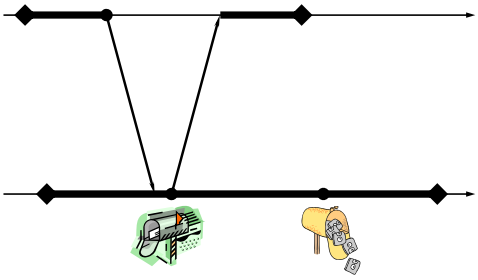
Komunikacja nieustanna synchroniczna



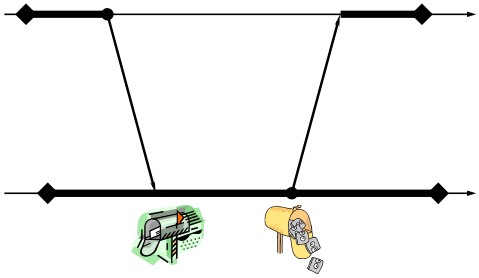
Komunikacja przejściowa asynchroniczna



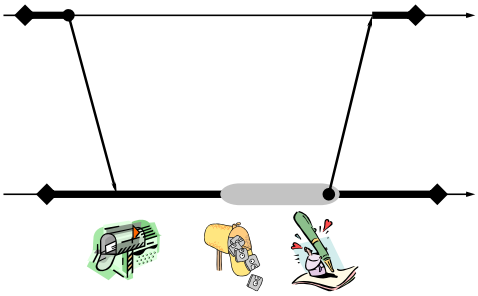
Komunikacja przejściowa synchroniczna z potwierdzeniem dotarcia



Komunikacja przejściowa synchroniczna z potwierdzeniem odebrania



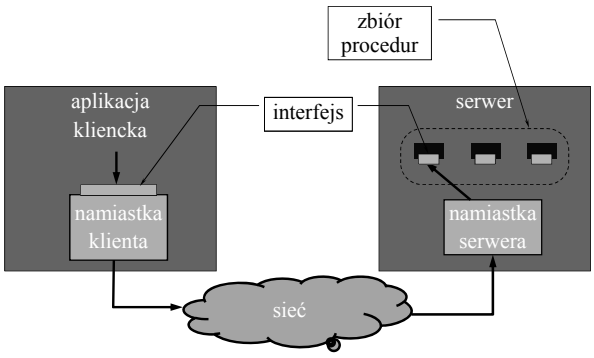
Komunikacja przejściowa synchroniczna z oczekiwaniem na odpowiedź



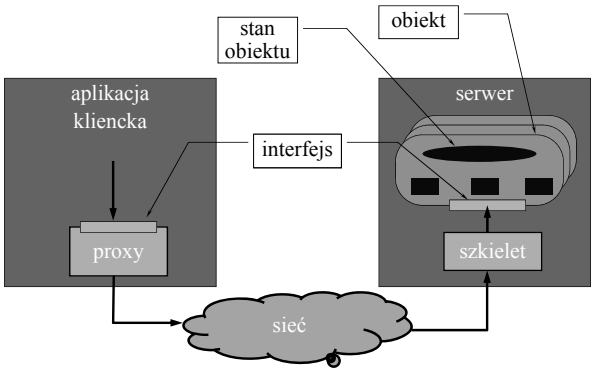
Zdalne wywoływanie procedur

- przeźroczystość dostępu — ukrycie komunikacji sieciowej przed aplikacją przez odpowiednie opakowanie funkcji komunikacyjnych namiastką klienta oraz serwera.
- gwarancja wykonania — ukrywanie błędów komunikacyjnych
- specyfikacja interfejsu — sposób opisu sygnatur procedur zdalnych (nazwy, typy parametrów)
- obsługa sytuacji wyjątkowych

Zdalne wywoływanie procedur — model systemu



Zdalne wywoływanie metod — model systemu



Istota podejścia obiektowego

- Obiekt jest jednostką integrującą w sobie dane (stan obiektu) oraz odpowiednio zdefiniowane operacje (metody).
- Jedyna forma dostępu do danych polega na wywoływaniu metod wyspecyfikowanych w publicznym interfejsie obiektu.
- Obiekt może implementować wiele interfejsów
- Ten sam interfejs może być implementowany przez wiele obiektów.

Podejście obiektowe do budowy systemów rozproszonych

- Kluczowe dla mechanizmu wywoływania metod zdalnych jest oddzielenie definicji interfejsu (specyfikacji) od jego implementacji w obiekcie.
- W językach definicji interfejsu (IDL) interfejs traktowany jest jak typ danych.
- W języku implementacji każdy obiekt implementujący dany interfejs jest instancją tego typu.

Przykład opisu interfejsu w podejściu obiektowym (CORBA IDL)

```
interface Konto {
     float stan();
     float wplac(in float value);
     float pobierz(in float value);
};

interface Bank {
     Konto otworzKonto(in int numer);
     float zamknijKonto(in Konto k);
};
```

Zdalne wywoływanie metod ☛ zdalny dostęp do obiektów

- obiekty zdalne (ang. remote objects) i rozproszone (ang. distributed objects)
- obiekty dostępne w czasie kompilacji (ang. compile-time objects) oraz w czasie wykonania (ang. runtime objects)
- wiązanie obiektu w aplikacji klienckiej
- obiekty trwałe (ang. persistent) i przejściowe (ang. transient)
- statyczne i dynamiczne wywoływanie metod
- przekazywanie parametrów

Obiekty zdalne i rozproszone

- Obiekt zdalny (ang. remote object) — stan obiektu utrzymywany jest przez jeden serwer (obiekt istnieje na jednej maszynie, jest więc scentralizowany), ale jest dostępny dla klientów działających na innych maszynach.
- Obiekt rozproszony (ang. distributed object) — stan obiektu utrzymywany i współtworzony jest przez serwery działające na różnych maszynach w taki jednak sposób, że klient postrzega obiekt jako jedną całość (przezroczystość położenie, być może również replikacji).

Dostępność obiektu

- Obiekt dostępny w czasie kompilacji (ang. compile-time object) — obiekt jest instancją jakieś klasy zdefiniowanej w konkretnym języku programowania, w którym napisany jest również program korzystający z danego obiektu. interakcja pomiędzy klientem a obiektem jest zatem widziana już na etapie kompilacji programu źródłowego.
- Obiekt dostępny w czasie wykonania (ang. runtime object) — interakcja pomiędzy klientem a serwerem może być dostrzeżona dopiero na etapie wykonania, gdyż program klienta i implementacja klasy obiektu powstają zupełnie niezależnie i mogą być tworzone zupełnie innych językach

Wiązanie obiektu w aplikacji klienckiej

- wiązanie obiektu (ang. binding) — uzyskanie proxy do wykonania operacji na obiekcie na podstawie identyfikatora obiektu
- wiązanie niejawne (ang. implicit binding) — operacja wiązania wykonywana jest niejawnie przy pierwszym odwołaniu
- wiązanie jawne (ang. explicit binding) — operacja wiązania wykonywana jest jawnie przez klienta
- PROBLEM: jak zaimplementować identyfikator (referencję) obiektu?

Istnienie obiektu

- Obiekt trwały (ang. persistent object) — z punktu widzenia klienta obiekt istnieje nawet wówczas, gdy jego stan nie jest w danej chwili utrzymywany przez żaden serwer (obiekt nie jest aktywny).
  - W momencie uruchomienia serwera stan obiektu może zostać odtworzony i tym samym obiekt może zostać udostępniony (aktywacja obiektu)
- Obiekt przejściowy (ang. transient object) — obiekt istnieje tylko tak długo, jak długo działa udostępniający go serwer.

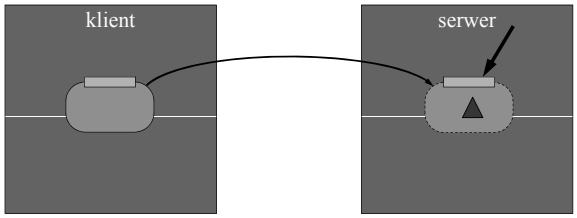
Statyczne i dynamiczne wywoływanie metod

- wywołanie statyczne — użycie do wywołania definicji interfejsu wygenerowanej wcześniej na podstawie specyfikacji przez odpowiednie narzędzie
- wywołanie dynamiczne — użycie odpowiedniej funkcji pośredniczącej w wywołaniu metody zdalnej, zgodnie z identyfikacją obiektu i metody, przekazaną jako parametr funkcji pośredniczącej

Przekazywanie obiektów jako parametrów

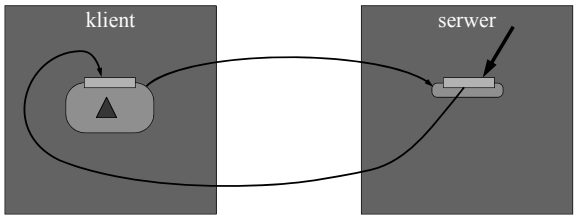
- Przekazywanie przez kopię — po stronie serwera tworzona jest kopia przekazanego obiektu.
- Przekazywanie przez referencję — przekazywany jest proxy do obiektu zdalnego (o ile obiekt jest dostępny zdalnie).
- Przekazywanie przez kopiowanie i odtwarzanie — po stronie serwera tworzona jest kopia przekazanego obiektu, a ewentualne zmiany tej kopii po zakończeniu operacji odtwarzane są w stanie obiektu u klienta.

Przekazywanie obiektów-parametrów przez kopię



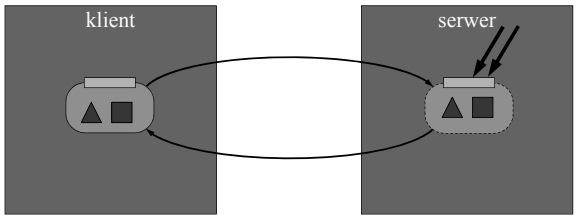
zmiany stanu obiektu (kopii) po stronie serwera mają charakter lokalny

Przekazywanie obiektów-parametrów przez referencję (zdalną)



zmiany stanu obiektu następują po stronie klienta jako wynik wykonania metody zdalnej

Przekazywanie obiektów-parametrów przez kopiowanie i odtwarzanie



zmiany stanu obiektu na serwerze odtwarzane są u klienta po zakończeniu zdalnej metody