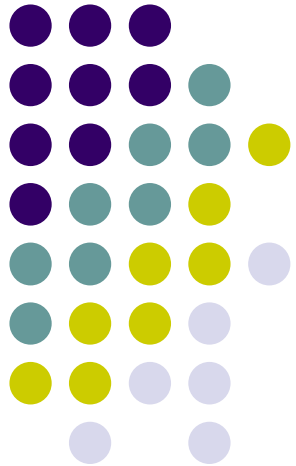
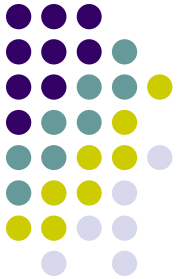
Мрежово програмиране

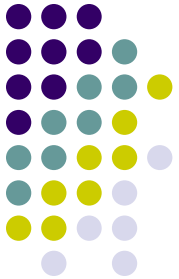
RPC и RMI





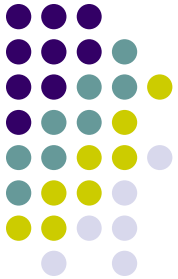
Алтернатива на сокетите е Remote Procedure Call (RPC)**.**

Вместо директната работа със сокети, програмистът работи така, както с локална процедура.



- Основната идея е създаването на фамилия от обекти, които могат да се намират върху различни машини и които могат да комуникират помежду си чрез стандартни мрежови протоколи (TCP, UDP, HTTP).

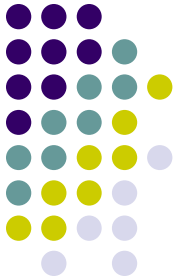
Концепция на отдалеченото извикване на процедури



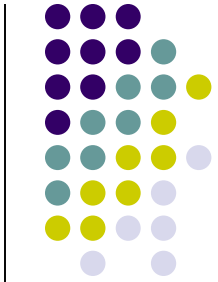
Идеята за отдалечено извикване на процедури се състои в разширяването на добре известния механизъм за предаването на управлението и на данните вътре в програмите, изпълнявани на една машина и този - за предаване на управлението и на данните по мрежата.

Характеристики на RPC:

- Асиметричност, т.е. едната от взаимодействащите страни е инициатор;
- Синхронност, т.е. изпълнението на извикващата процедура спира от момента на изпращане на заявката и се възобновява едва след възвръата (return) от извикващата процедура.

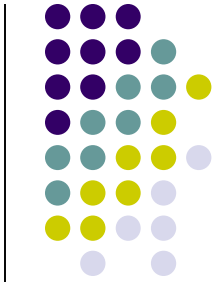


Понеже извикващата и извикваната процедури се изпълняват на различни машини, затова те имат различни адресни пространства, което създава проблеми при предаването на параметри и резултати, особено ако машините не са идентични.



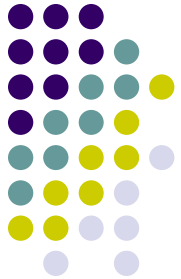
В реализацията на RPC участват минимум два процеса - по един на всяка машина. Ако единият от тях приключи аварийно, могат да възникнат следните ситуации:

- при авария на извикващата процедура процедурите на отдалеченото извикване ще "осиротеят",
- при аварийно приключване на отдалечените процедури извикващите процедури ще станат "родители с убито потомство", като ще чакат безрезултатно отговора от отдалечените процедури.



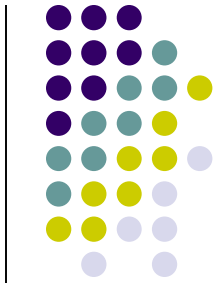
Основни недостатъци на RPC:

- Ниска защитеност от взлом, което се потвърждава от многочислените съобщения за проникванията;
- Отсътствие на някакви вградени механизми, осигуряващи надеждност или отказоустойчивост.



- Освен тези проблеми съществуват и редица други, свързани с нееднородността на езиците за програмиране и на операционните среди: структурите данни и структурите на извикването на процедурите, поддържани от конкретния език за програмиране, не се поддържат точно така, както от другите езици.

Базови операции на RPC

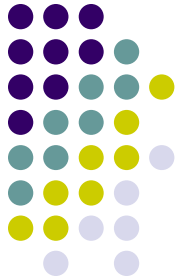


Как изглежда извикването на локална процедура? Нека като пример разгледаме системното извикване:

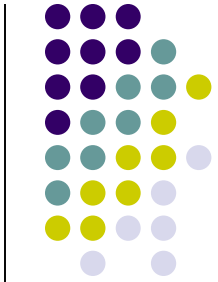
m = my_write (fd, buf, length);

За да се осъществи това извикване, извикващата процедура съхранява параметрите в стека в обратен ред. След приключването на изпълнението на извикването слага връщаната стойност в регистър, прочита адреса на възврата и връща управлението на извикващата процедура. Извличат се параметрите от стека и се възстановява изходното му състояние.

Състояние на стека при извикване на локална процедура



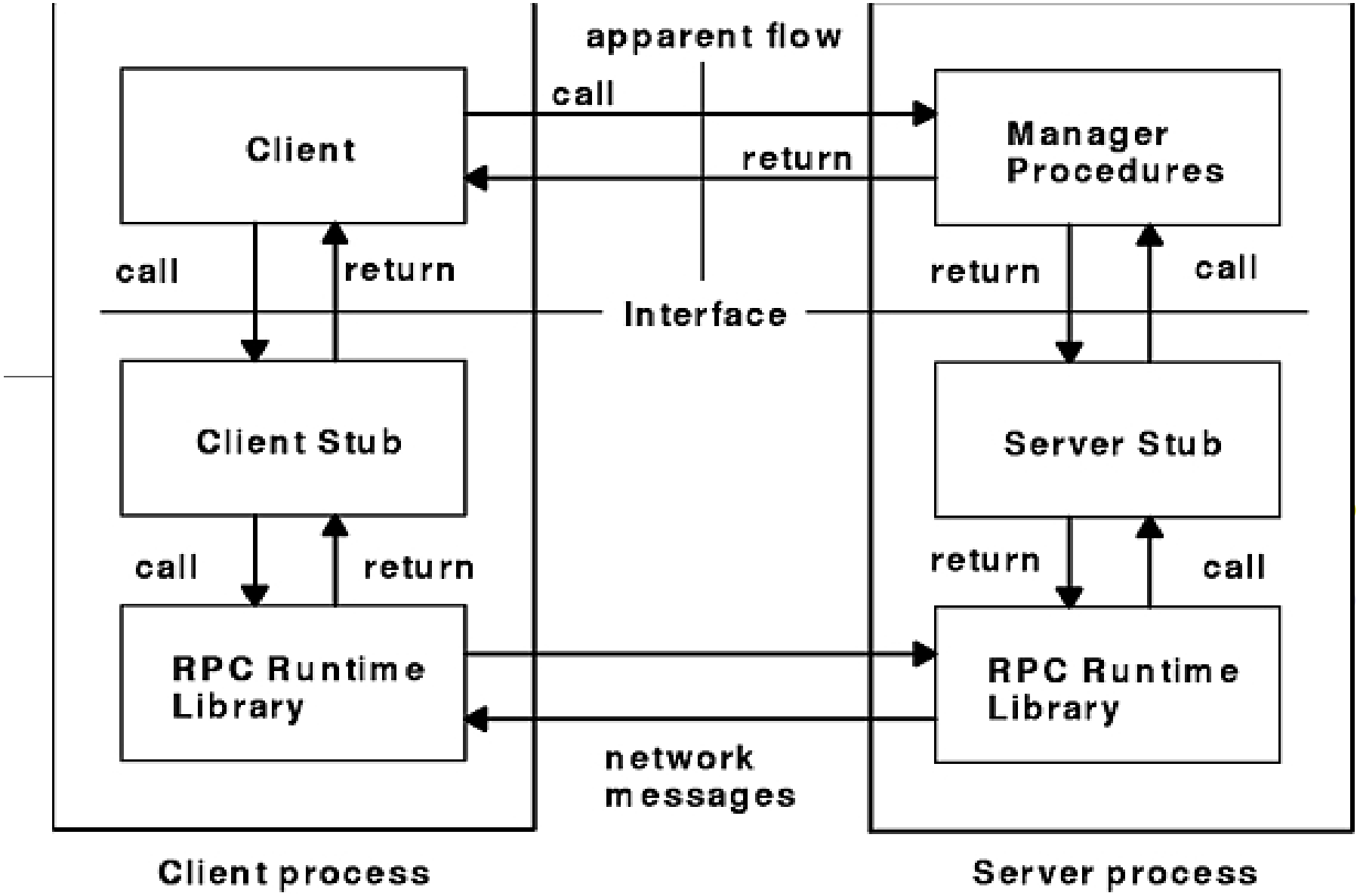
В езика С параметрите могат да се извикват или *по указател* (by name - представляващ адрес на глобална област от паметта, в която се съхранява параметъра) или по *стойност* (by value - параметърът се копира от изходната област на паметта в локалната памет на процедурата, разполагана обикновено в стековия сегмент).



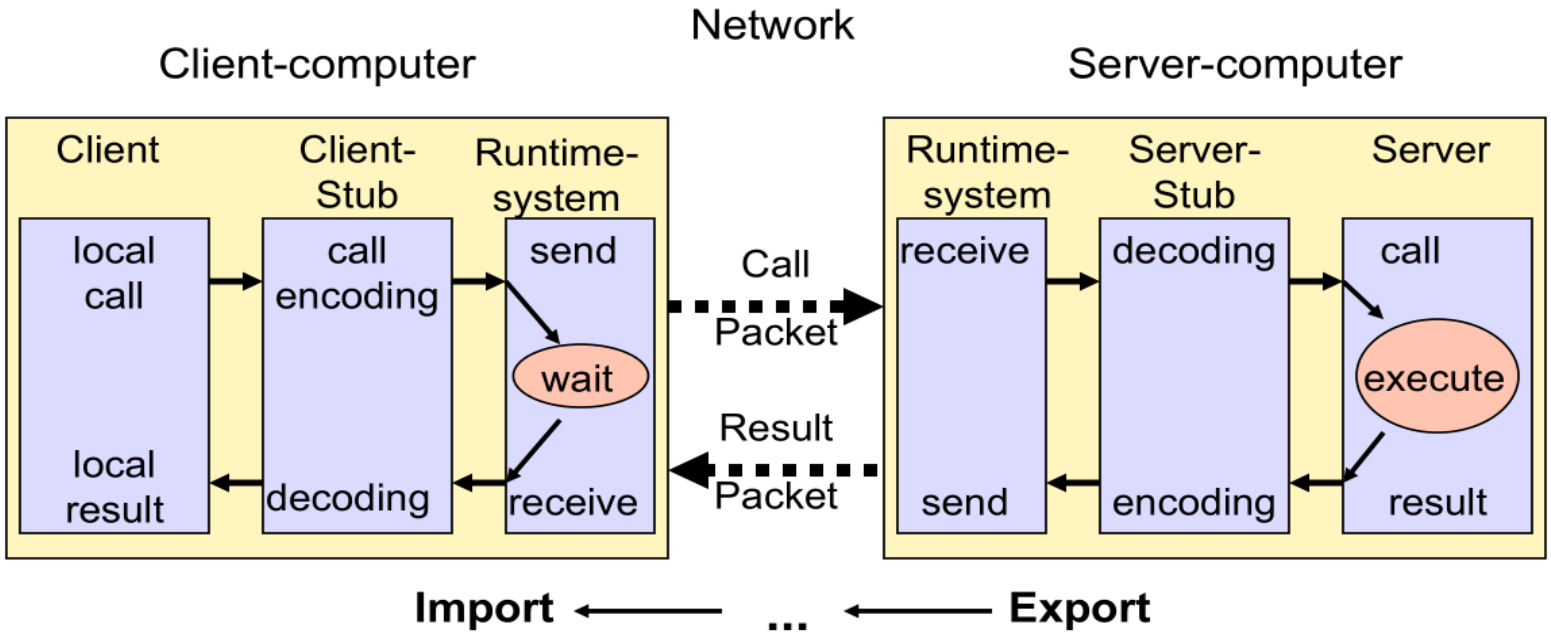
- *В първия случай извикваната процедура работи с оригиналните стойности на параметрите и техните изменения се виждат веднага от извикващата процедура.*
- *Във втория случай извикваната процедура работи с копията на стойностите на параметрите и техните изменения никак не влияят на стойностите на оригиналите на тези променливи в извикващата процедура.*

Разработчиците на програмния език решават какъв механизъм за предаването на параметрите да се използва.

Целта е параметрите да се преобразуват в такъв формат, че той да е пригоден за предаване по мрежата от една машина към друга.



Remote Procedure Call Flow



Подобно на оригиналната процедура stub-ът се извиква с използване на извикваща последователност така, както става прекъсване при обръщение към ядрото. За разлика от оригиналната процедура той не слага параметрите в регистри и не заявява данни от ядрото. Той формира съобщение за изпращане на ядрото на отдалечената машина.

	клиент	сървър
	извикване на stub процедурата ↓	изпълнение на необходимата работа
stub	подготовка на буфера на съобщението ↓	
	подреждане на параметрите в буфера	извикване на сървъра ↑
	добавяне на заглавно поле на съобщението ↓	записване на параметрите в стека
	изпълнение на прекъсване към ядрото ↓	разопаковане на параметрите ↑
ядро	превключване на контекста на ядрото	превключване на контекста на сървърния stub ↑
	копиране на съобщението в ядрото	копиране на съобщението на сървърния stub ↑
	определяне адреса на целта ↓	определяне дали нужният stub очаква пакета ↑
	записване на адреса в заглавието на съобщението ↓	определяне на кой stub да се предаде пакета ↑
	установяване на мрежовия интерфейс	проверка правилността на пакета ↑
	включване на таймер ↓	прекъсване на процеса ↑

Отдаленото извикване на процедури определя както начина на организация на взаимодействието между компонентите, така и методиката за разработването на тези компоненти

Съществуват различни реализации на RPC в различните ОС.

- В UNIX се използва процедура със същото наименование (Remote Procedure Call - RPC). Тя е внедрена в ядрото на системата. Изпълнението й се осигурява от RPC протокол.
- В MSWindows RPC е стартирал развитието си на базата на OLE механизмите, които постепенно са се развили в технологията DCOM (Distributed Component Object Model). Тази технология позволява създаването на достатъчно мощни разпределени мрежови изчислителни среди. Използват се фирмените протоколи на Microsoft.

Отдалечено извикване на методи (RMI - Remote Method Invocation)

- Технологията RMI позволява обръщението към методите на обектите, работещи на друга виртуална Java машина, различна от тази, на която работи програмата, извикваща тези методи.

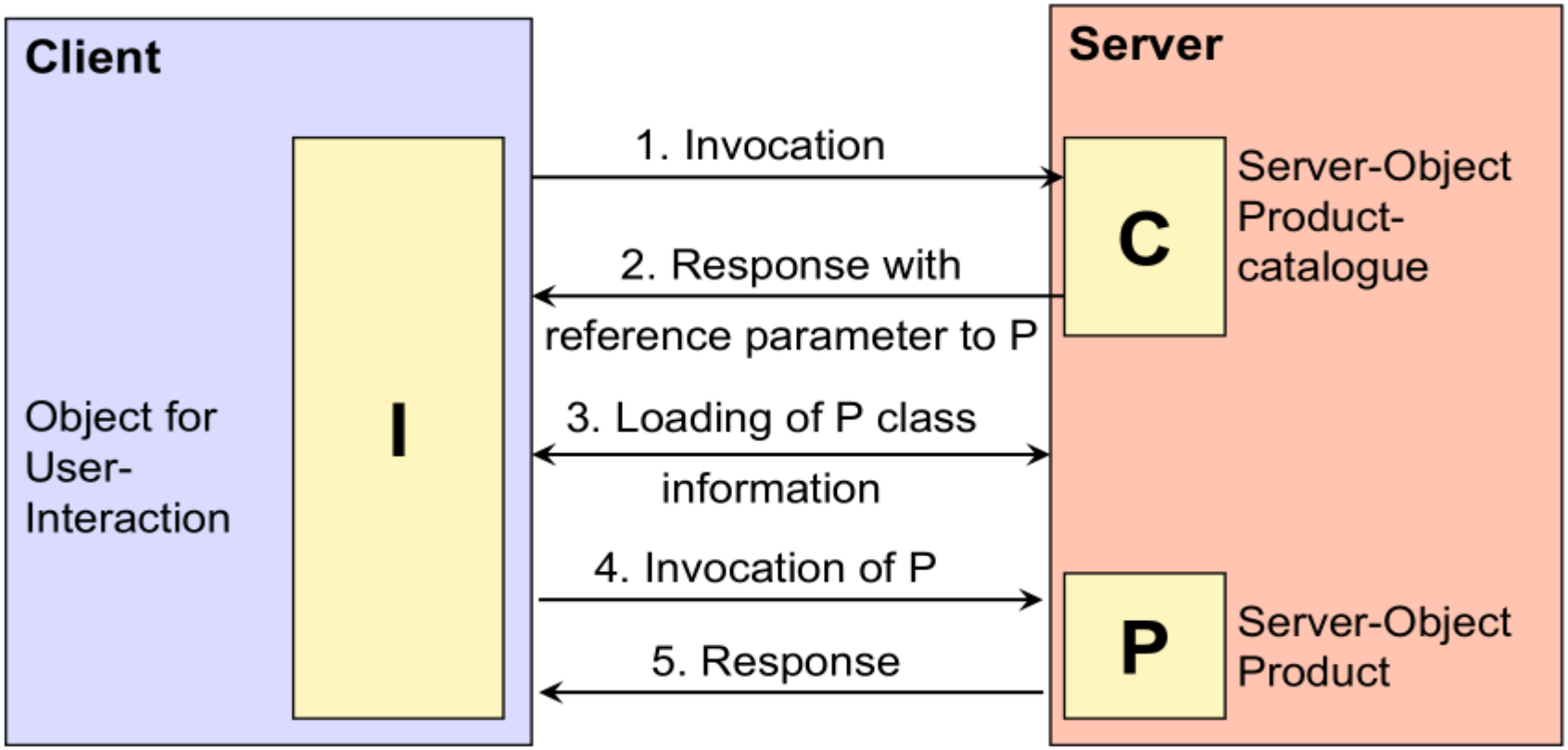
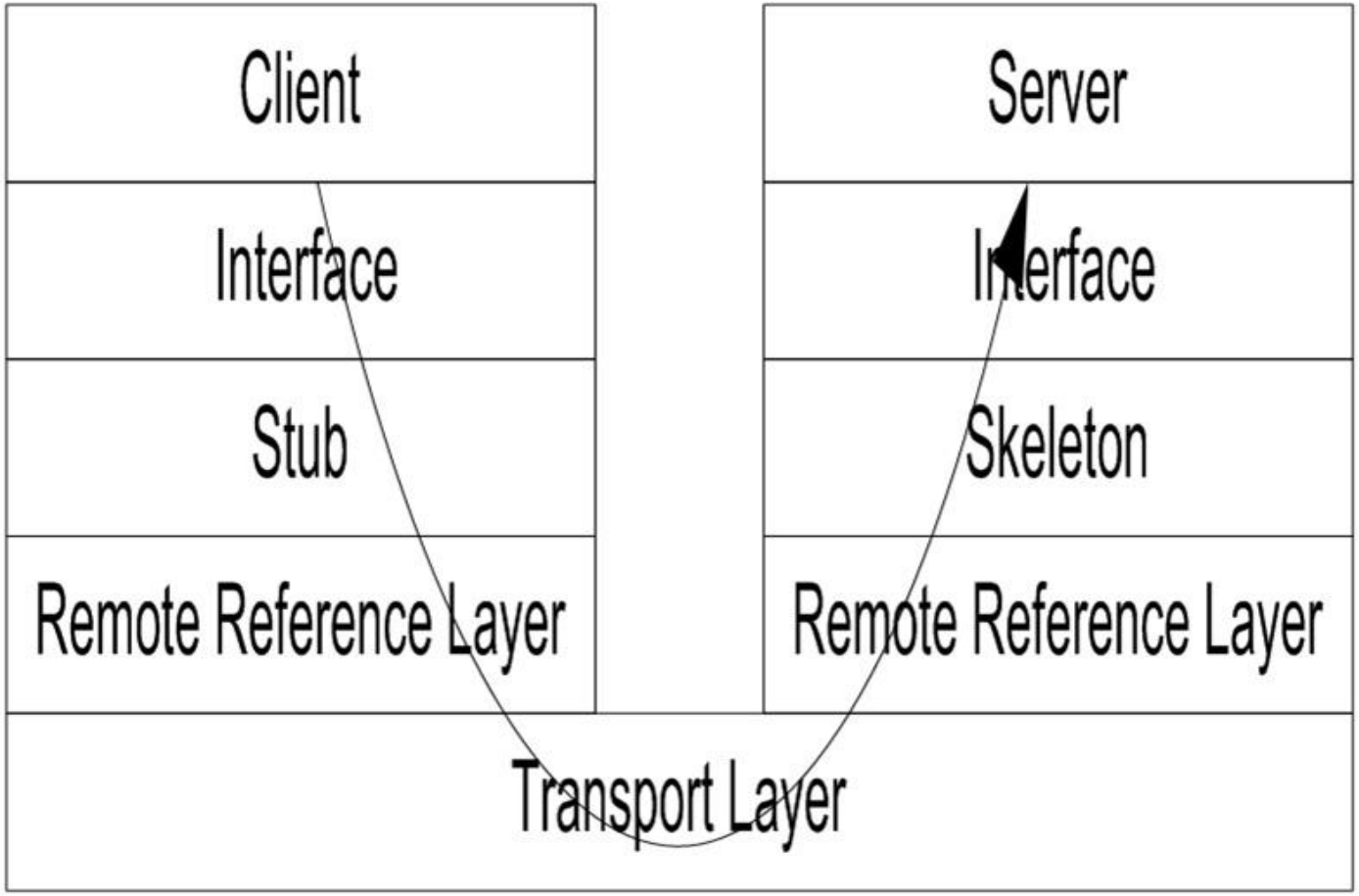


Схема на взаимодействие в RMI модела



Архитектурата на RMI се състои от три слоя:
• Слой на стъб/скелетон (Stub/Skeleton);
• Слой на отдалеченото извикване (Remote Reference Layer);
• Транспортен слой.

RRL интерпретира и управлява препратките, направени от клиентите към отдалечения обект на сървъра.

Този слой е представен както на клиента, така и на сървъра.

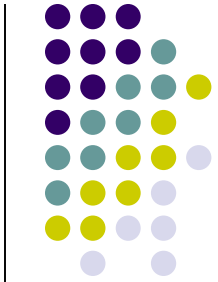
RRL на клиентската страна приема запитванията за методите от стъба, които се предават, като опакован поток от данни на RRL към сървърната страна.

Опаковането (marshalling) представлява процес, при който параметрите, предавани от клиента, се преобразуват във формат, който може да се предава по мрежата. RRL на сървърната стана изпълнява разопаковане (unmarshaling) на параметрите, които са изпратени на отдалечения метод през скелетона.

Разопаковането е процес, при който по-рано опакованите параметри, предадени от клиента на RRL от клиентската страна, се преобразуват във формат, който е разбираем за скелетона.

По време на връщане на стойностите от скелетона, данните отново се обработват чрез маршалинг и се предават към клиента чрез RRL от страната на сървъра.

Концепция

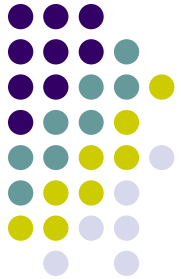


- Механизъм, позволяващ на обект от една Java машина да извика методи на друга Java машина
- RMI работи в мрежа
- Пакет java.rmi
  - Пакетът java.rmi обявява интерфейса Remote и класовете Naming и RMISecurityManager. Също така съдържа класове за изключителни ситуации, които се ползват заедно с RMI.
  - Всички отдалечени обекти трябва да реализират интерфейса Remote. Този интерфейс не съдържа методи и се използва за идентификация на отдалечените обекти.

Пакетът java.rmi се състои от следните класове:

- Клас Naming: Съдържа статични методи за достъп до отдалечени обекти чрез URL. Този клас поддържа следните методи:
  - rebind(): Свързва името на отдалечения обект със зададен URL и обикновено се използва от обекта на сървъра.
  - unbind(): Унищожава връзката между името на обекта и URL.
  - lookup(): Връща отдалечен обект, зададен с URL и обикновено се използва от обекта на клиента.
  - list(): Връща списък от URL-и, които са известни на RMI регистъра.
- Клас RMISecurityManager: Определя политика за безопасност по премълчаване за отдалечения обект стъб. Политиката се прилага само за приложенията.

Пакет java.rmi.registry

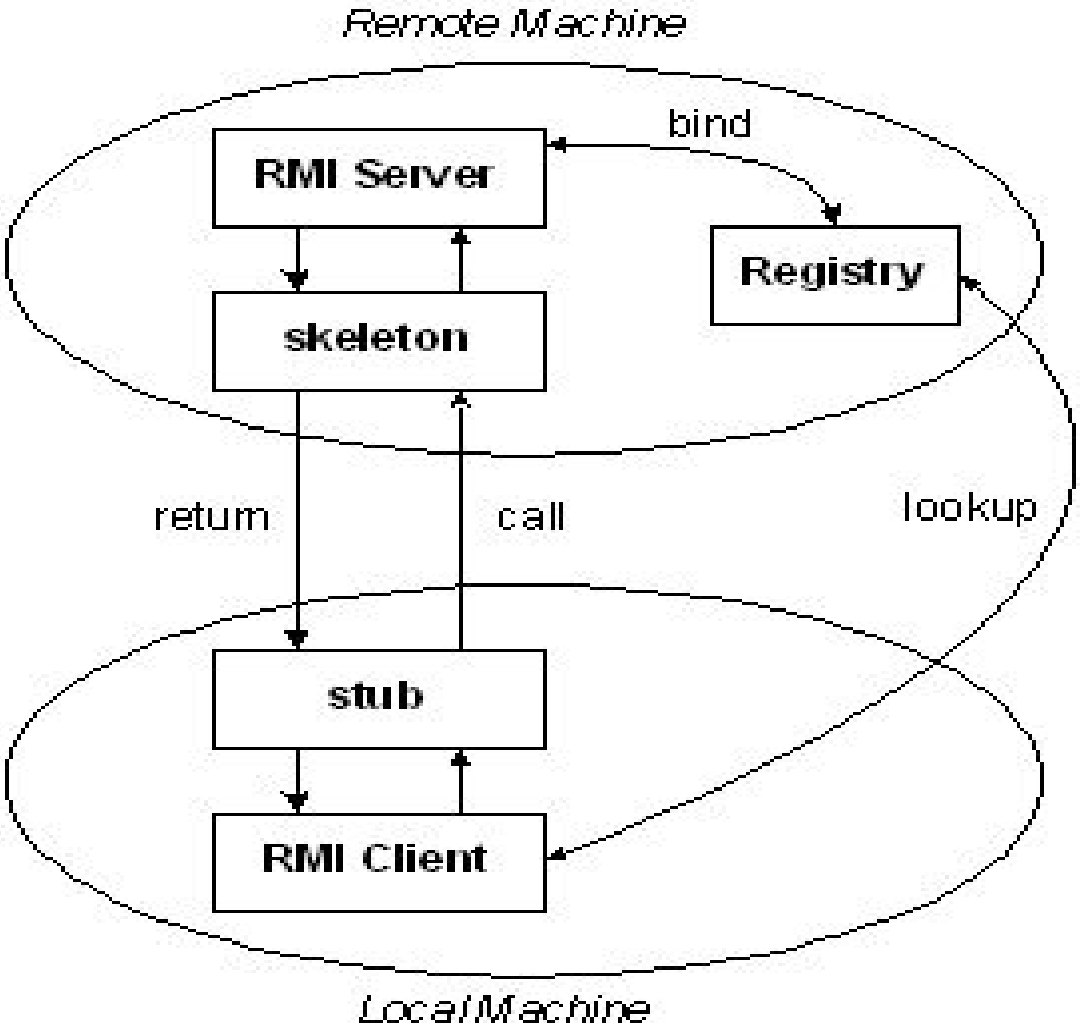


- Пакетът java.rmi.registry съдържа интерфейсите Registry и RegistryHandler и се разполага в регистъра. Тези интерфейси се използват за регистрация и достъп до отдалечените обекти по име.
- Remote обектите се регистрират, когато те се свързват с процеса на регистрация на хоста.
- Процесът на регистрация се задава с изпълнението на командата start rmiregistry.
  - Тази команда определя методите rebind(), unbind(), list() и lookup(), които се използват от класа Naming за свързване на имената и URL препратките в RMI.

Пакетът java.rmi.server реализира интерфейсите и класовете, които поддържат клиентската и сървърната част на RMI и се състои от следните класове и интерфейси:

- Клас RemoteObject: Реализира интерфейса Remote и осигурява отдалечената реализация на класа Object. Всички обекти, които реализират отдалечени обекти, разширяват класа RemoteObject.
- Клас RemoteServer: Разширява класа RemoteObject и е общ клас, който е подклас за конкретни типове реализации на отдалечен обект.
- Клас UnicastRemoteObject: Разширява класа RemoteServer и осигурява реализацията на RemoteObject по премълчаване. Класовете, които реализират RemoteObject обикновено са подкласове на обекта UnicastRemoteObject.
- Клас RemoteStub: Осигурява абстрактна реализация на стъба на клиентската страна. Статичният метод setRef() се използва, за да се свърже стъба на клиентската страна със съответните отдалечени обекти.
- Интерфейс RemoteCall: Осигурява методи, които се използват от всички стъбове и скелетони в RMI.
- Интерфейс Skeleton: Осигурява метод за получаване на достъп до методи на отдалечен обект и този интерфейс се реализира от отдалечени скелетони.
- Интерфейс Unreferenced: Дава възможност да се определи, кога клиентът повече не се обръща към отдалечения обект. Този интерфейс се реализира от класа Remote.

Схема на взаимодействие



Стъбът се свързва с отдалечения метод на обекта чрез скелетон, който е реализиран на сървъра

Скелетонът е реализиран като прокси на страната на сървъра, който продължава комуникацията със стъба като:

- Прочита параметрите за извиквания метод;
- Извиква от отдалечената услуга обекта на реализацията;
- Получава връщаната стойност;
- Записва връщаната стойност обратно в стъба.

В JVM на Java 2 са интегрирани допълнителни stub протоколи за да се избегне нуждата от skeleton.

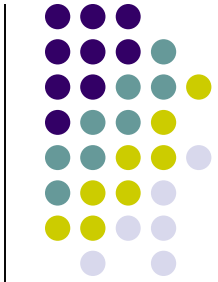
Търсене на отдалечени обекти

- Указатели на отдалечени обекти се публикуват в RMI registry
- Отдалечените обекти се търсят по URL от вида: rmi://<host>:<port>/<object>, където
  - host:port – местоположение на RMI registry
  - object – регистрирано име на обекта

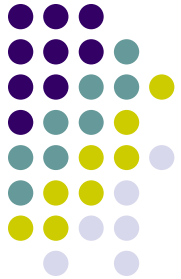
Експорт на обектите

- Преди обектът да се предаде на друга машина той трябва да бъде експортиран
- Метод
  - UnicastRemoteObject.exportObject(object)
- Наследниците UnicastRemoteObject се експортират автоматично

Създаване на приложения с използването на RMI



1. Разработка и имплементация на компонентите на разпределеното приложение.
2. Компилиране на кода, получаване на съответстващите класове stub.
3. Регистрация на класовете, за да станат класовете достъпни за отдалечените програми.
4. Стартиране на приложенията.



Пример:
Нека n е число, което ще проверяваме дали е просто, като определяме всички прости делители на числото n от 2 до sqrt(n).

Решаваме задачата така:

сървърът ще подава числа за проверка от регистрирани клиенти,

взаимодействието ще работи в двете посоки (клиент->сървър - регистрация, сървър->клиент - число за проверка).

За целта описваме два интерфейса:

```
public interface ClientRegister extends Remote {
  public void register (PrimeChecker checker) throws RemoteException;
}
public interface PrimeChecker extends Remote {
  public boolean check (BigDecimal number) throws RemoteException;
}
```

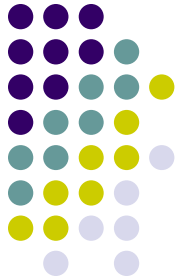
Интерфейсът ClientRegister се използва от клиента за да се регистрира на сървъра като PrimeChecker.

Сървърът използва PrimeChecker за предаване числото на клиента, което ще се проверява.

Отдаленият интерфейс трябва да се разширява пряко или косвено - интерфейс Remote.

Определяме изключението RemoteException.

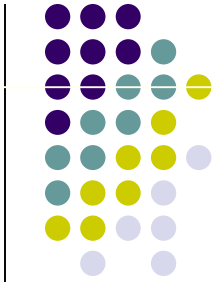
Реализация на сървъра:



```
public class PrimeNumbersSearchServer implements ClientRegister {
   ...
   public static void main(String[] args) {
    PrimeNumbersSearchServer server = new PrimeNumbersSearchServer();
     try {
     ClientRegister stub = (ClientRegister)UnicastRemoteObject.exportObject(server, 0);
      Registry registry = LocateRegistry.createRegistry(12345);
       registry.bind("ClientRegister", stub);
       server.startSearch();
    } catch (Exception e) {
       System.out.println ("Error occured: " + e.getMessage());
       System.exit (1);
    }
  }
}
```

Експортираме отдалечен обект и получаваме стъб.
Посредством този стъб клиентът ще извиква методите на нашия обект.
Вторият параметър exportObject – определя номера на порта.
Стъбът трябва да се предаде на клиента.
Създаваме регистратор и свързваме нашия стъб с името ClientRegister.
Регистраторът ще приема съединения на порт 12345.

Клиентът:

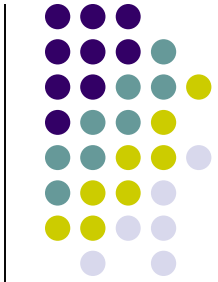


```
public class PrimeNumbersSearchClient implements PrimeChecker {
   ...
   public static void main(String[] args) {
    PrimeNumbersSearchClient client = new PrimeNumbersSearchClient();
     try {
      Registry registry = LocateRegistry.getRegistry(null, 12345);
      ClientRegister server = (ClientRegister)registry.lookup("ClientRegister");
       PrimeChecker stub = (PrimeChecker)UnicastRemoteObject.exportObject(client, 0);
      server.register(stub);
     } catch (Exception e) {
      System.out.println ("Error occured: " + e.getMessage());
      System.exit (1);
    }
  }
}
```

Клиентът трябва да получи сървърния стъб за да се регистрира.

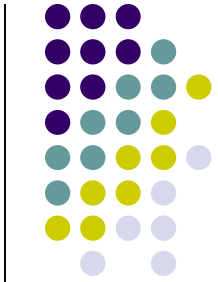
Намираме отдалечения регистратор и запитваме от него стъба, свързан с името “ClientRegister”. Първият параметър е хост (null - localhost), вторият е номер на порт.

След това експортираме клиентския отдалечен обект и предаваме на сървъра стъба (вече клиентски) - така се регистрираме. Сървърът ще добави клиента в опашката на достъпните checker-и и ще започне да му предава числа за проверка. След проверката, ако е приключила без грешки, клиентът отново попада в опашката и т.н.



Преди изпълнението на приложението е необходимо да се регистрират обектите на сървъра в RMI регистъра.
Така клиентът ще може да извиква обектите, регистрирани на сървъра.
За изпълнението на RMI приложението е необходимо да се изпълнят следните стъпки:

1. Да се генерират стъба и скелетона на отдалечения обслужващ клас (с командата rmic).
2. Да се стартира RMI регистъра (start rmiregistry).
3. Да се стартира RMI сървъра на разпределеното приложение (java NameServer).
4. Да се стартира RMI клиента на разпределеното приложение (java NameClient).



- Технологията Remote Method Invocation за първи път е представена в JDK 1.1
- Тя е вдигнала нивото на мрежовото програмиране
- Въпреки, че RMI е относително лесна за използване и не е лишена от недостатъци, тя е необикновено мощна технология и разкрива пред обикновения Java програмист напълно нова парадигма – света на разпределените обектни изчисления.