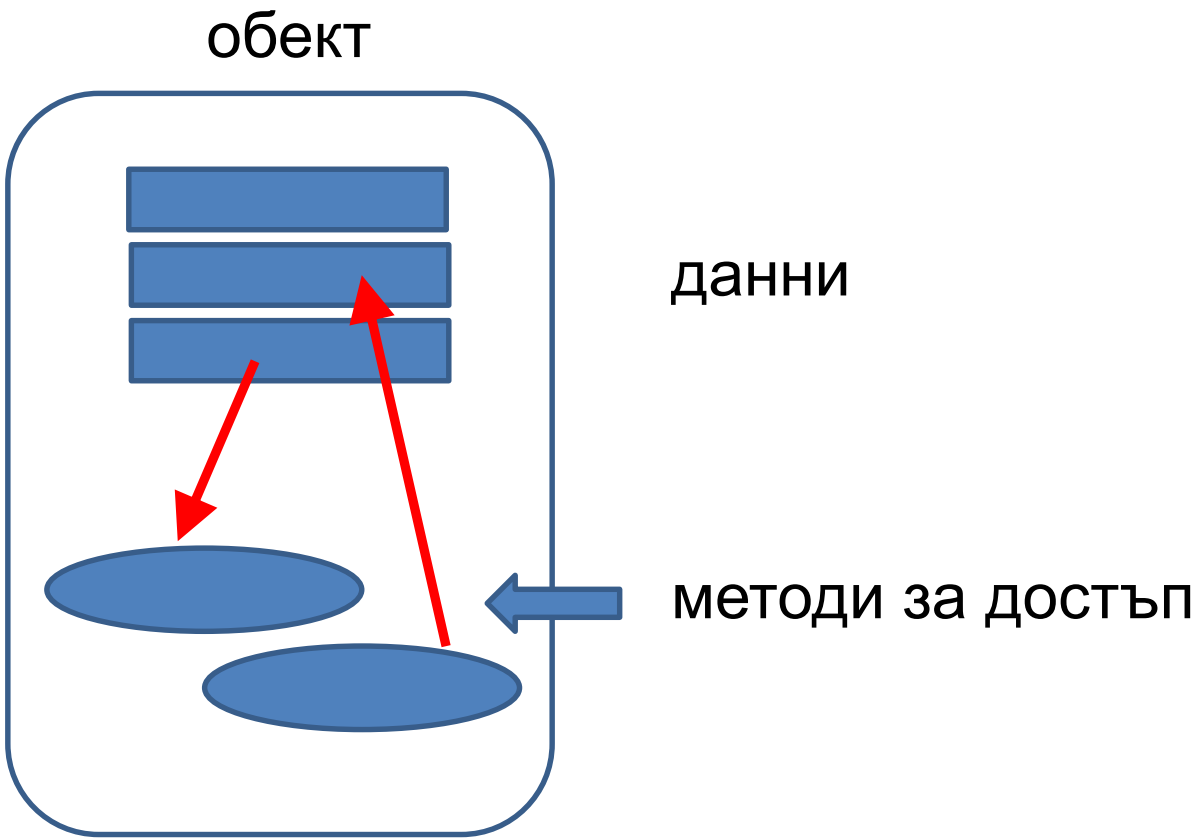
Разпределени обекти. RMI

Обекти и класове

- **Обект** – програмна единица със състояние и поведение
- **Клас** – съвкупност от обекти

- Обектът е инстанция на клас
- Класът е шаблон, от който може да се създаде клас

Обекти и класове



Пример на Java

```
public class Student {
  private String name;
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name }
  }

class Test {
  public static void main(String[] args) {
    Student s = new Student();
    s.setName(“Petar”);
    System.out.println(s.getName());
  }
}
```

Разпределени обекти

Системите с разпределени обекти използват клиент-сървър архитектурата. Обектите се управляват от сървъри и клиентите извикват техните методи чрез RMI. Заявка за извикване на метод се реализира чрез съобщение към съответния сървър. Обектите в сървърите могат да станат клиенти на обекти от други сървъри – верига извиквания на методи.

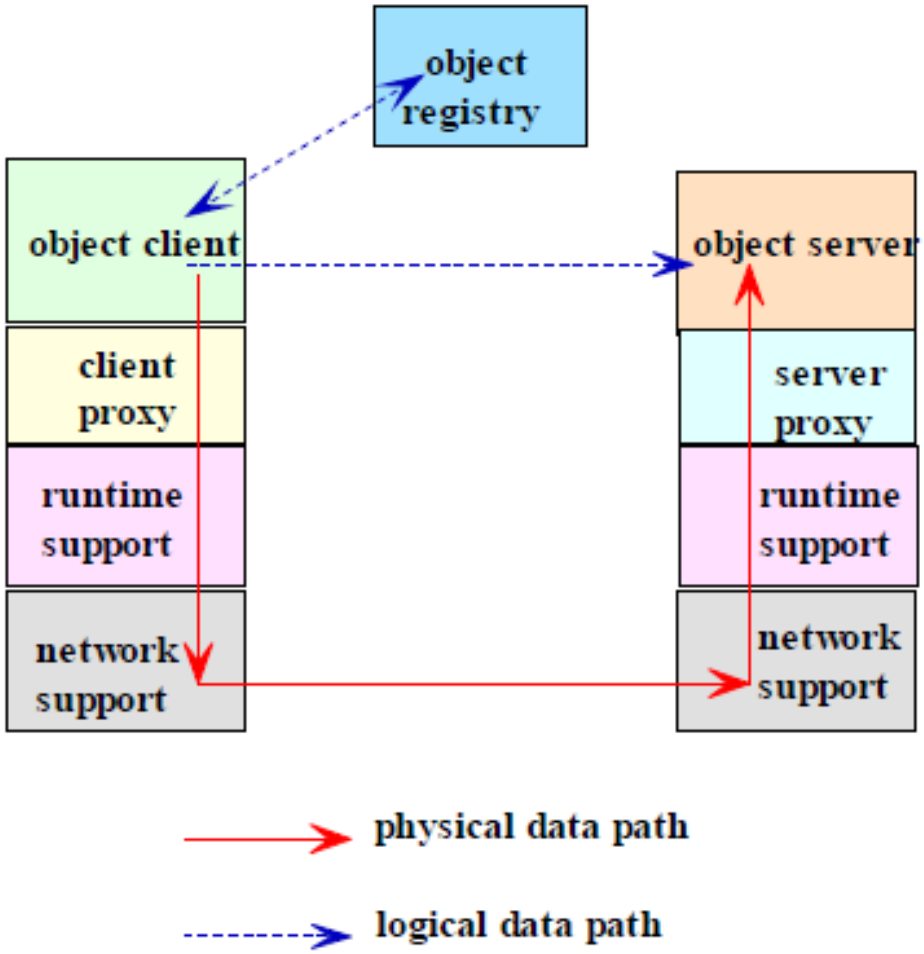
Разпределени обекти

Наличието на клиентски и сървърни обекти в различи процеси подсилва капсулирането. Достъпът до състоянието на обект е само чрез негови методи – не е възможен неоторизиран достъп.

Достъпът само чрез методи води до друго предимство – използване на хетерогенни системи с различни даннови формати без клиентът да се грижи.

Marshaling – процес на конвертиране на типовете данни на езика в подходящ за предаване по мрежата формат.

Архитектура на система с разпределени обекти



Реализация на системи с разпределени системи

- Java Remote Method Invocation (RMI)
- Common Object Request Broker Architecture (CORBA)
- Distributed Component Object Model (DCOM)
- Simple Object Access Protocol (SOAP)

RMI – разлики с RPC

- Програмистът може да използва мощта на ОО програмиране (обекти, класове, наследяемост);
- На базата на концепцията на идентичността на обектите в ОО системи, всички обекти в RMI системи има уникална референция. Такива обекти могат да се изпращат и като параметри.
- RMI позволява параметри да се пращат не само по стойност, но и като референция на обект. Чрез нея отдалечената страна може да извика методите на обект, вместо да се прехвърля целия обект през мрежата.

RMI – обектен модел

- Обектите комуникират помежду си чрез извикване на методи, пращайки аргументи и получавайки резултат.
- В разпределените системи данните на обект могат да се достъпват единствено чрез методите му (C++, Java допускат директен достъп до данни).

Моделът

- **Референция на обект** - Достъпът до обектите е чрез референция. Обект, на който се извиква метод – *target* или *receiver*.
- **Интерфейси** – предоставя дефиниция на множеството методи (тип на аргументите, връщани резултат и изключение). Обект ще предоставя интерфейс ако неговия клас съдържа код, реализиращ методите на този интерфейс. Интерфейсите нямат конструктори.

Моделът

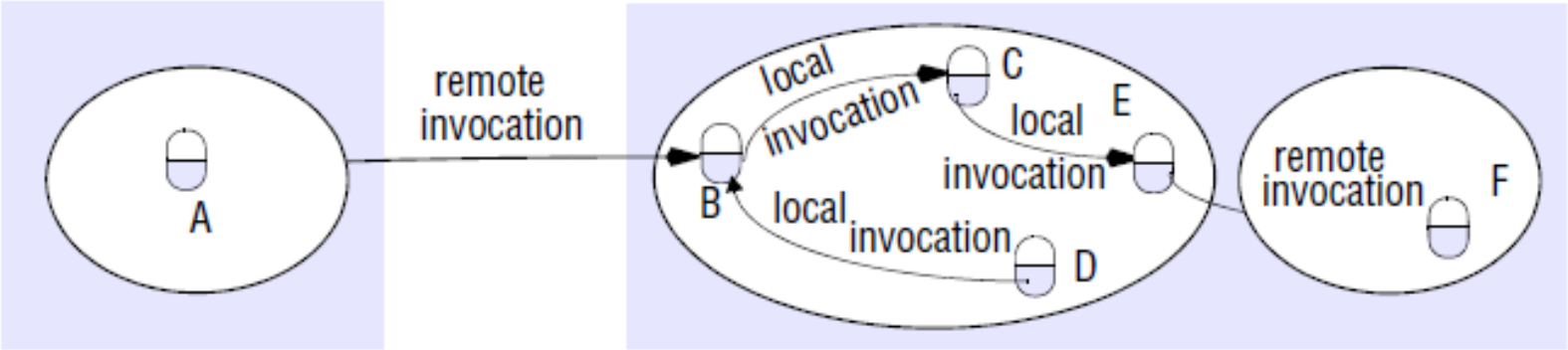
- **Actions –** действията в ОО програма се инициират от обекта, извикващ метод на друг обект. Receiver изпълнява съответния метод и връща управлението на извикващия обект като евентуално връща и резултат. Извикване на метод резултира в:

1. Промяна е състоянието на receiver
2. Създаване на нова инстанция на обект
3. Последващо извикване на методи в други обекти

Моделът

- **Exceptions –** *throw catch* генерира се изключение и контролът се предава на друг блок
- **Garbage collection** – освобождаване на паметта. Java позволява автоматично връщане на заетата памет от обект, който повече не се достъпва.

Разпределен обектен модел



Всеки процес съдържа колекция от обекти, някои могат да получават локални и отдалечени извиквания, други само локални. Извиквания между обекти в същия процес – *локални*. Извиквания между обекти в различни процеси (на една или различни машини) – *отдалечени*.

Обекти, получаващи отдалечени извиквания – отдалечени обекти - B и F.

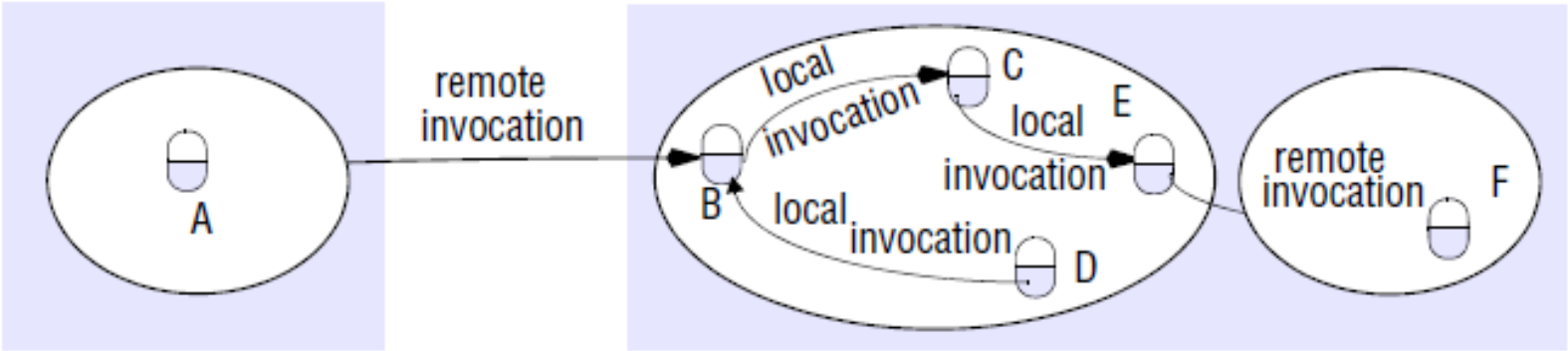
Всички обекти могат да получават локални извиквания – извикващия трябва да има референция към тях (C има на E)

Разпределен обектен модел

Основата на модела – две фундаментални концепции:

Remote object reference (ROR) – Обекти може да извика метод на отдалечен обект само ако има достъп до неговата ROR (ROR за B трябва да бъде налична за A)

Remote interface (RI) - Всеки отдалечен обект има RI, Описващ кои негови методи могат да се извикват отдалечено (B и F трябва да имат такива)



Референция към отдалечен обект

Това е идентификатор, използван в разпределена система за указване на уникален отдалечен обект.

Например:

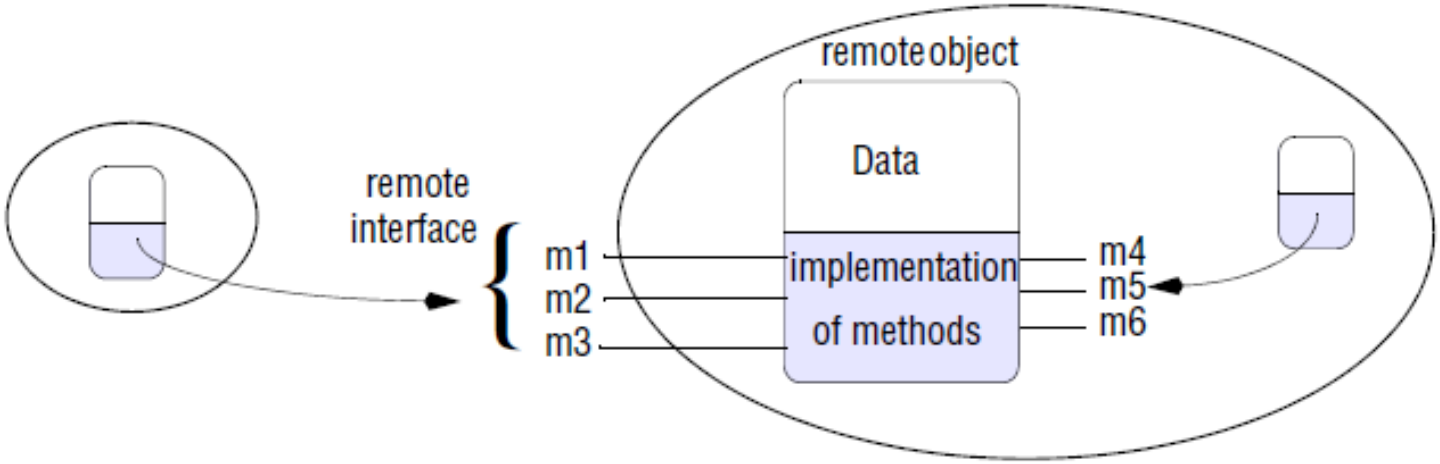
Време на създаване на обекта.

Номерът се увеличава при всяко създаване на обект в процеса.

Информация зз интерфейса на отдалечения обект (името на интерфейса)

32 bits	*32 bits*	*32 bits*	*32 bits*	
Internet address	port number	time	object number	interface of remote object

Remote Interfaces



Класът на отдалечения обект реализира методите на неговия RI. Обекти в други процеси могат да извикват само методите, принадлежащи на неговия RI. Локалните обекти могат да извикват нормално тези методи. RI няма конструктор.

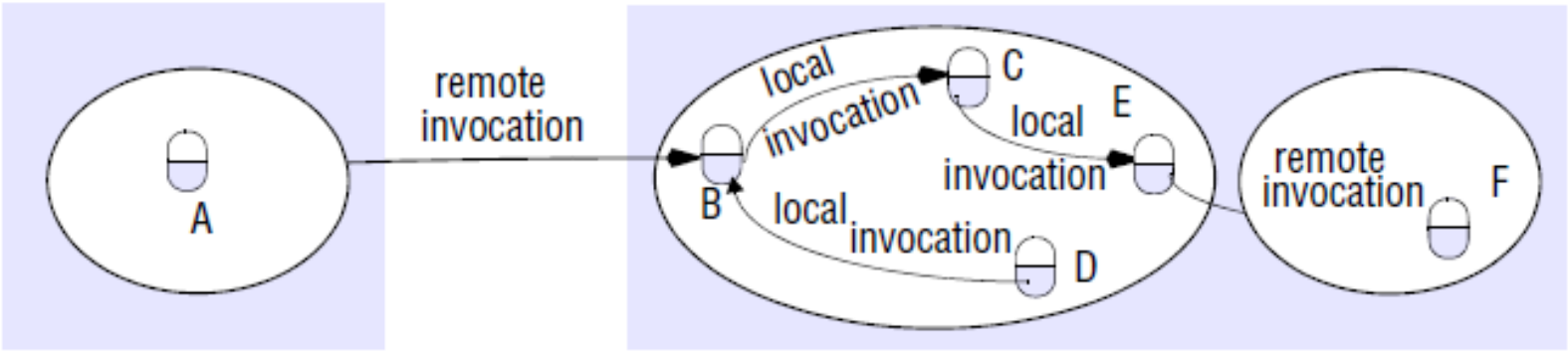
CORBA предоставя IDL за дефиниране на интерфейси.

В Java RI се дефинира аналогично на други интерфейси, като разширение на интерфейса именуван *Remote*.

Действия в системи с разпределени обекти

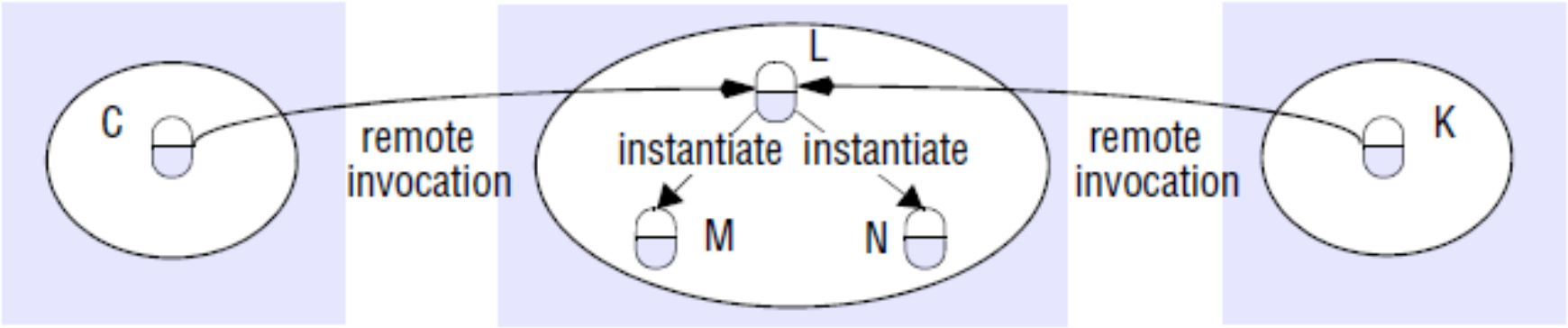
Аналогично на неразпределените с-ми, действие се инициира с извикване на метод, което може после да резултира в последващи извиквания на методи на други обекти.

Но тука тази верига от обекти може да е в различни процеси на различни машини. Ако обект е на друга машина, трябва да е налична отдалечена референция за извикващия (A трябва да знае ROR на B). Такава може да се получи и в резултат на извикване на метод (A получава за F от B).

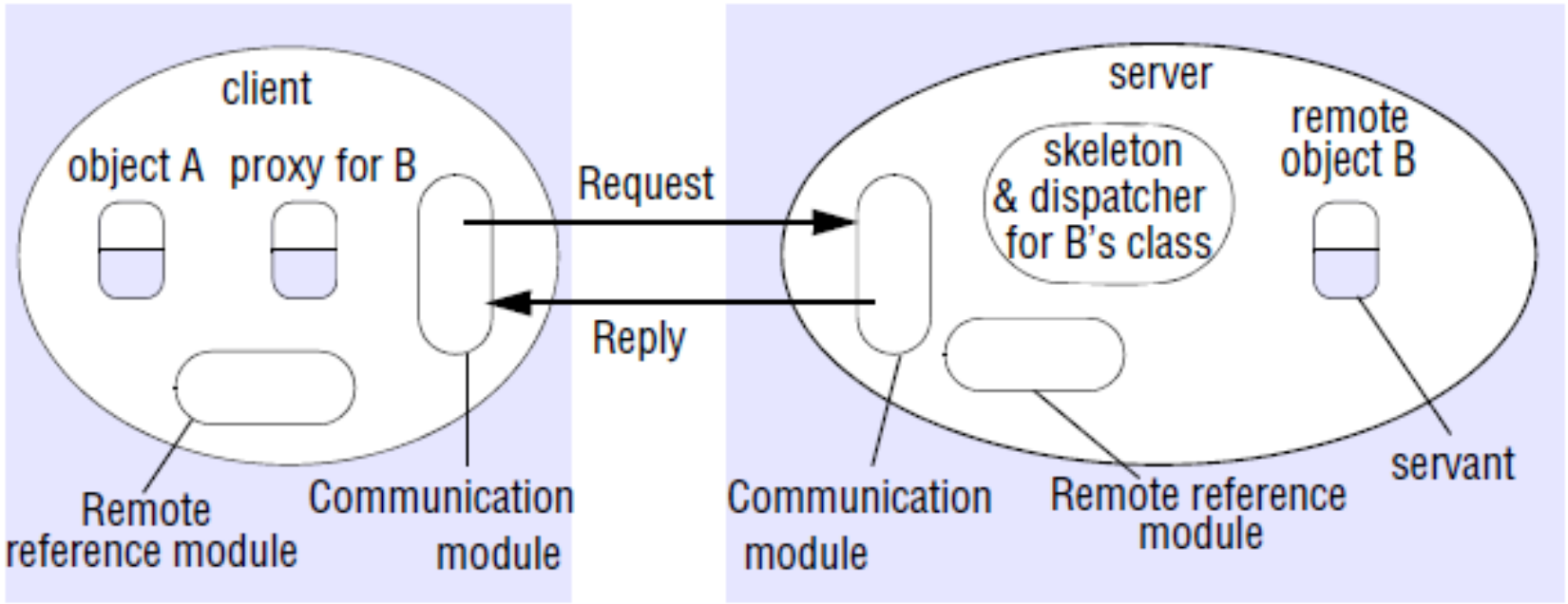


Действия в системи с разпределени обекти

Може да се създават инстанции на обекти чрез RMI. Ако L съдържа метод за създаване на отдалечен обект, то извиквания от C и K могат да доведат на инстанции на M и N.



Реализация на RMI



Communication module

Реализира request-reply протокола.

Формат на съобщение (модулите използват първите три полета)

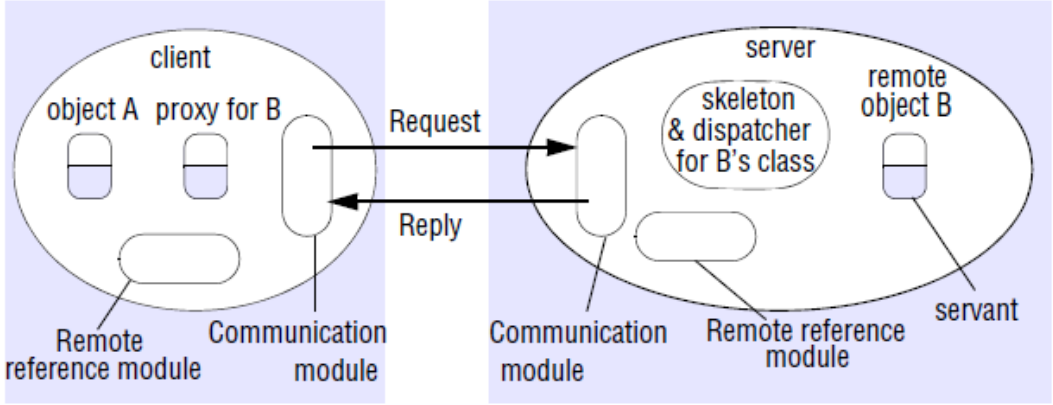
messageType	*int (0=Request, 1= Reply)*
requestId	*int*
remoteReference	*RemoteRef*
operationId	*int or Operation*
arguments	*// array of bytes*

Remote reference module

Отговаря за транслирането между локалните и отдалечени референции на обекти и за създаване на отдалечени референции на обекти.

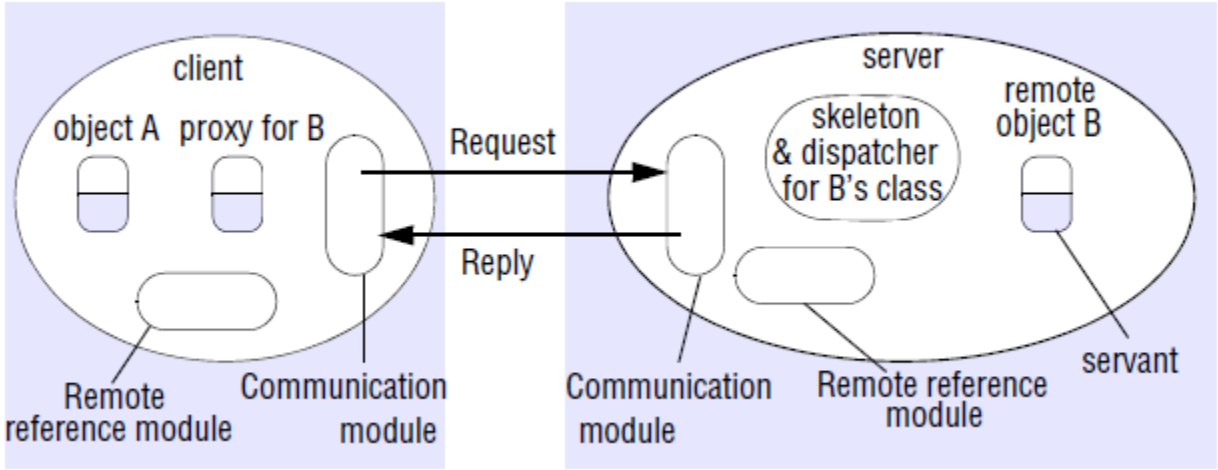
На базата на *remote object table* – съдържа съответствията между локалните референции на обекти в този процес и отдалечените (те са системно видими):

- Запис за всички отдалечени обекти за процеса (отдалечения обект B ще бъде записан в таблицата на сървъра)
- Запис за всяко локално прокси (прокси за B ще бъде записано в таблицата на клиента)



Remote reference module

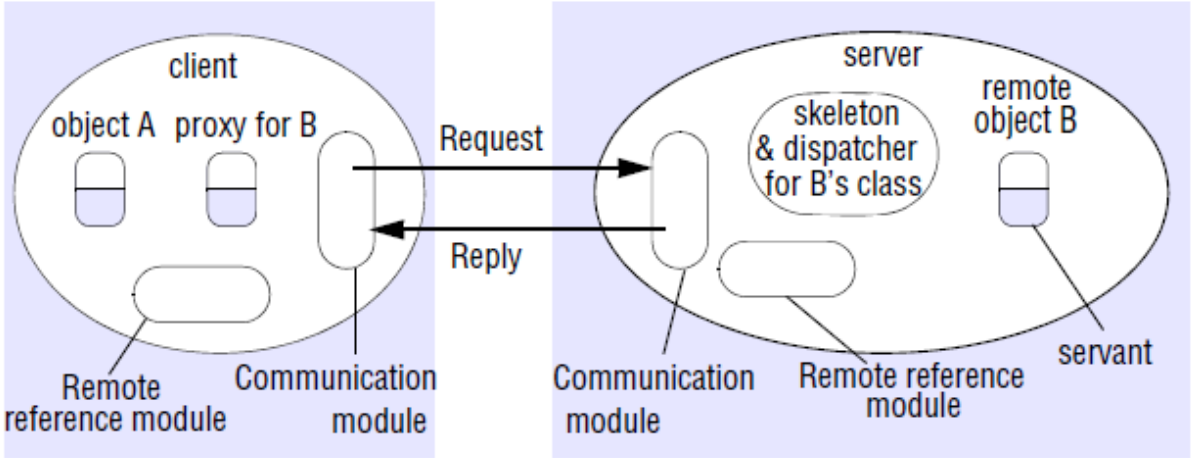
- Когато отдалечен обект са предаде за първи път като аргумент или резултат, RRM създава референция към отдалечен обект и я записва в таблицата.
- Когато референция към RO пристигне в заявка или отговор, RRM се запитва за съответстващата локална референция на обект, която може да е или към прокси или към RO. Ако ROR не е в таблицата, RMI създава ново прокси и указва на RRM да го добави в таблицата



Servant

Инстанция на клас, предоставяща тялото на отдалечения обект. Съществуват в сървърния процес. Обработва отдалечената заявка пратена от съответстващия скелетон.

Създават се когато се инстанциират отдалечени обекти и остават докато повече не са необходими (garbage collection)

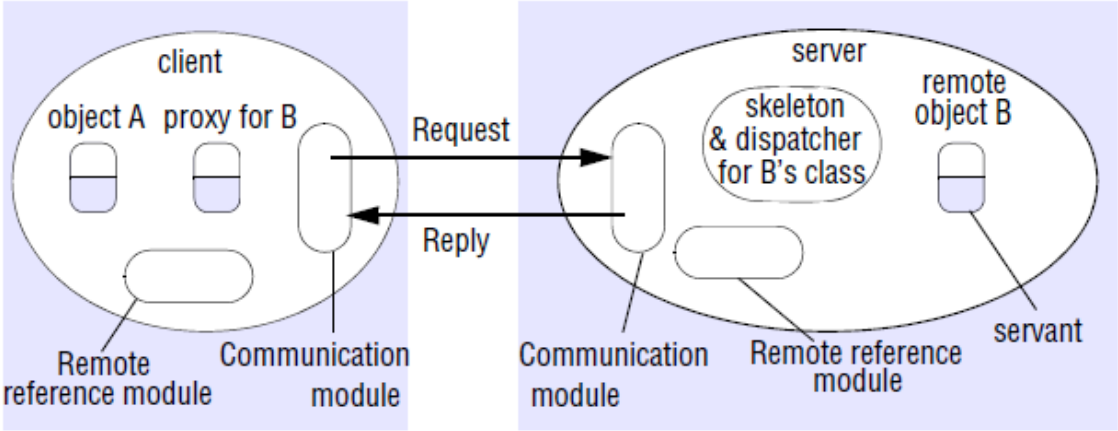


RMI софтуер

Proxy – прави извикването на отдалечен метод прозрачно за клиента (като локално извикване), препраща извикването в съобщение към RO. Скрива детайлите по маршалинг на аргументите, унмаршалинга на резултата и пращането и получаването на съобщенията за клиента.

За всеки RO за който процесът има отдалечена референция трябва по едно прокси.

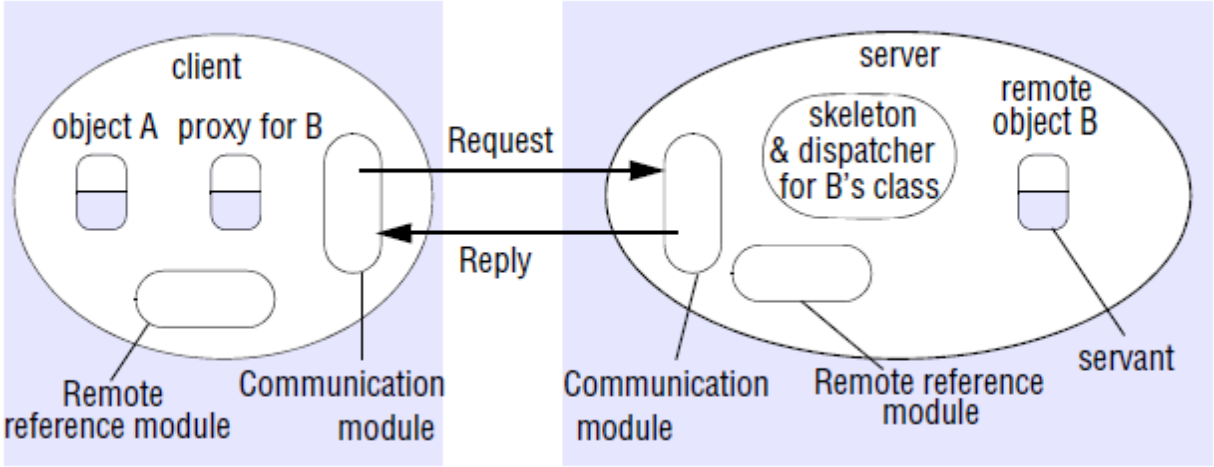
Класът на проксито реализира методите на RI на RO който представлява.



RMI софтуер

Dispatcher – за всеки клас представящ RO по един диспечер.

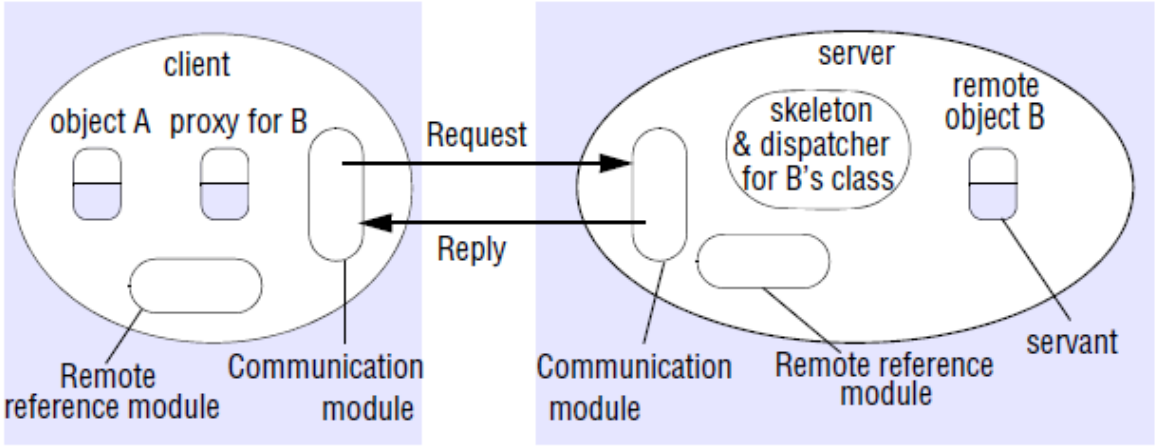
Получава съобщението-заявка от комуникациония модул. Използва *operationalID* да избере подходящия метод в скелетона.



RMI софтуер

Skeleton – класът на RO има скелетон, който реализира методите на RI. Те са реализирани различно от тези на серванта, който въплъщава RO. Метод на скелетона демаршалва аргументите в съобщението-заявка и извиква съответстващия метод в серванта. Чака завършването на извикването и маршалва резултата, заедно с изключение в съобщение-отговор към метода на изпращащото прокси.

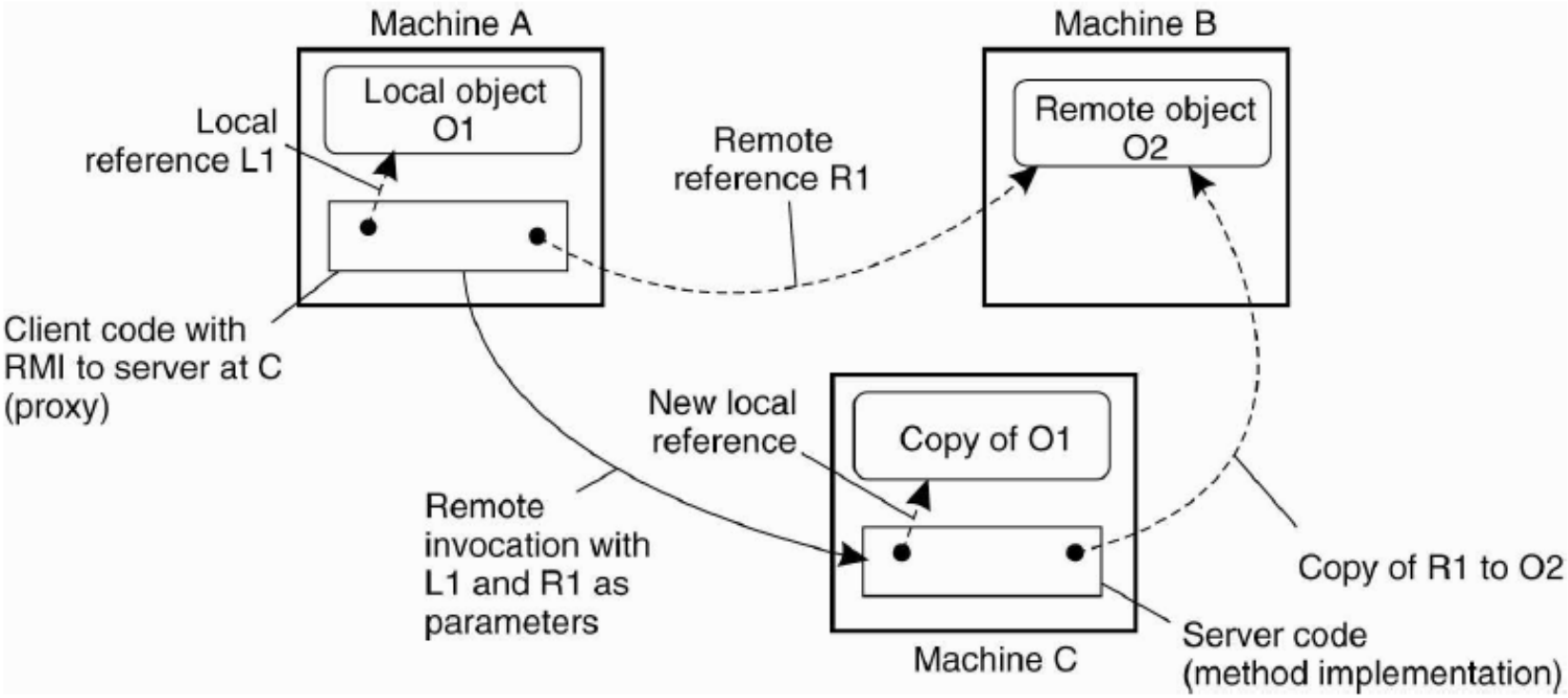
Java RMI компилатора генерира проксито, диспечера и скелетон класовете от класа на RO.



Предаване на параметри

- **Object reference** – По-лесно в сравнение с RPC
  - Сървърът просто се присъединява (*bind*) към указания обект и извиква метод;
  - Когато повече не е необходим указания обект, сървърът се разкача (*unbind*);
- **Object-by-value** – Клиентът може да изпрати цял обект като стойност на параметър
  - Обектът трябва да се маршалва
    - Маршалва състоянието му
    - Маршалва се методите му или дава референция къде може да се намери тяхната имплементация
  - Сървърът демарашалва обекта (има вече копие на оригиналния обект)
  - Този тип предаване може да доведе до неприятни резултати

Предаване на параметри



RMI - използване

```
import java.rmi.*;
import java.util.Vector;

public interface Shape extends Remote {
  int getVersion() throws RemoteException;
  GraphicalObject getAllState() throws RemoteException;
}

public interface ShapeList extends Remote {
  Shape newShape(GraphicalObject g) throws RemoteException;
  Vector allShapes() throws RemoteException;
  int getVersion() throws RemoteException;
}
```