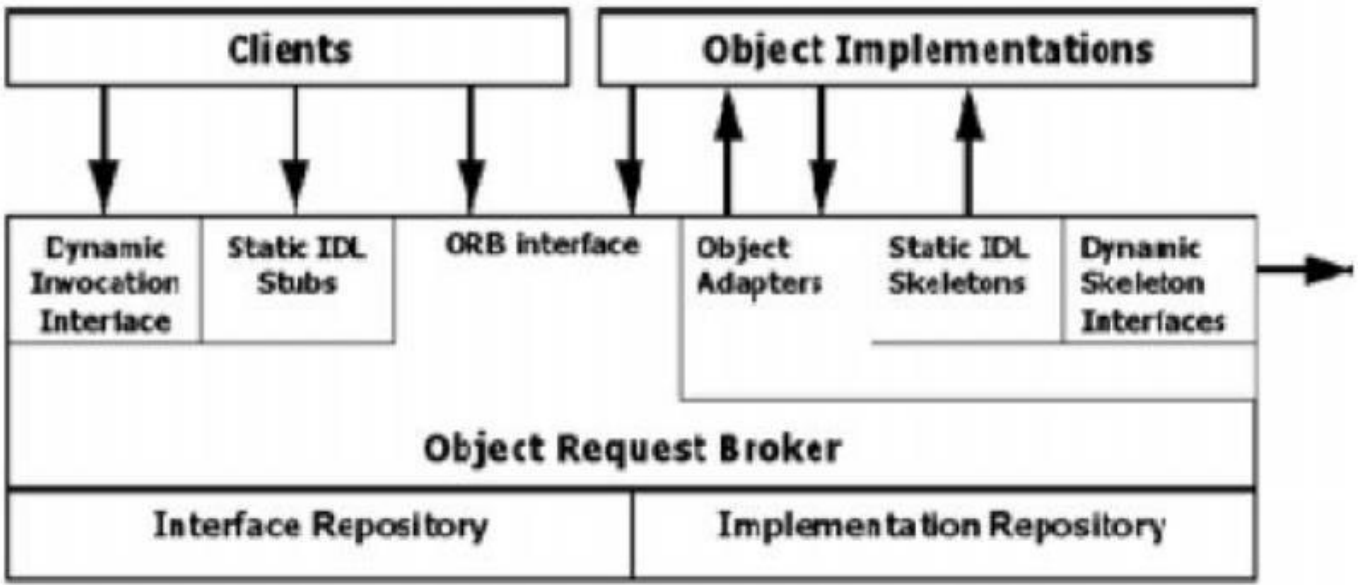
# SISTEM TERDISTRIBUSI
# REMOTE METHOD INVOCATION (RMI)

Mochammad Zen Samsono Hadi, ST. MSc. Ph.D

- CORBA (Common Object Request Broker Architecture) memungkinkan kita menggunakan aplikasi **tanpa adanya batasan platform**, teknologi jaringan, bahasa pemrograman, maupun letak objek pemberi service yang dituju.

# Komponen Utama CORBA

- CORBA disusun oleh **komponen-komponen utama**:
  1. ORB (Object Request Broker)
  2. IDL (Interface Definition Language)
  3. DII (Dynamic Invocation Interface)
  4. IR (Interface Repositories)
  5. OA (Object Adapter)



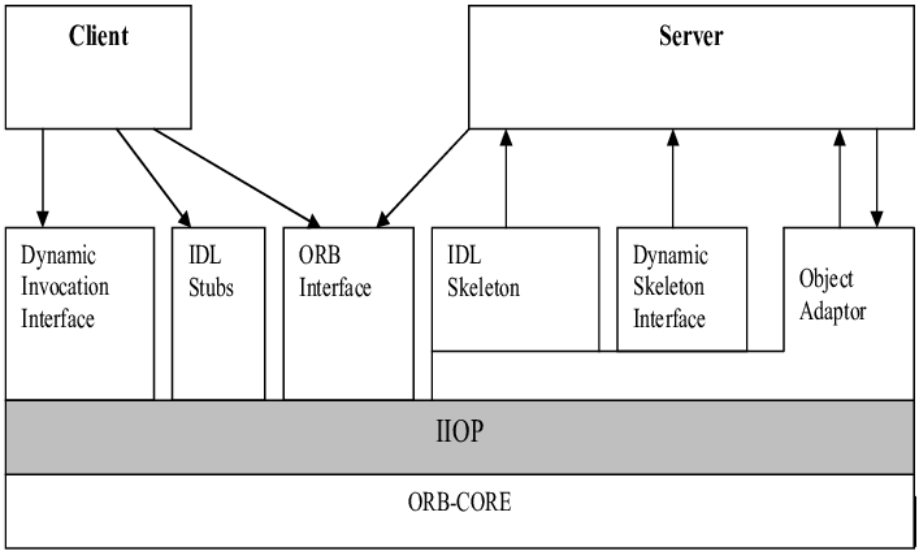
# Komponen CORBA

**Komponen CORBA pada sisi *Client*:**

1. *Client Application*
2. *Client IDL Stubs*
3. *Dynamic Invocation Interface*
4. *Interface Repository*
5. *Client Side ORB Interface*
6. *ORB Core*

**Komponen CORBA yang terletak di sisi Server**

1. *Server Side ORB Interface*
2. *Static IDL Skeleton*
3. *Dynamic Skeleton Interface*




Internet Inter-ORB Protocol (IIOP)

# RMI (Remote Method Invocation)

- Sistem komputasi terdistribusi yang bekerja di banyak tempat mengharuskan beberapa komputer untuk bisa berkomunikasi satu sama lain.
- Untuk komunikasi, Bahasa Java mendukung pemakaian socket yang sifatnya fleksibel dan mencukupi untuk keperluan komunikasi umum. Tapi di sisi lain, untuk membuat socket, client dan server harus terhubung melalui protokol pada application level untuk meng-encode dan men-decode data-data yang akan dikirimkan.

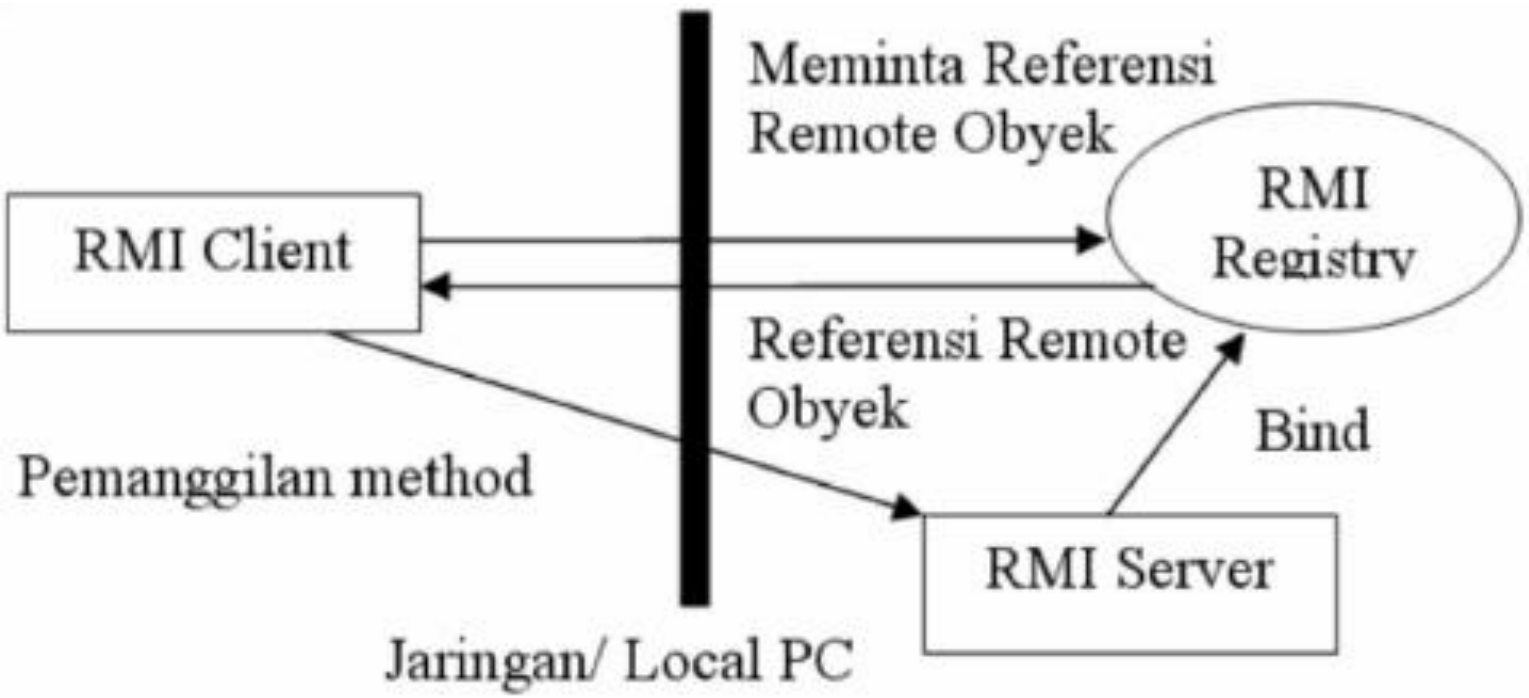
# RMI (Remote Method Invocation)

- RMI adalah salah satu bagian dari J2SE yang digunakan untuk membangun aplikasi terdistribusi menggunakan Bahasa Java.
- RMI adalah kumpulan kelas dalam Java yang digunakan untuk menangani pemanggilan (invocation) method secara jarak jauh (remote) dalam suatu jaringan internet. Dimana idenya adalah memisahkan obyek-obyek secara terdistribusi dalam mesin-mesin yang berbeda.
- RMI menggunakan prinsip pemrograman berorientasi obyek dimana obyek satu dapat saling berkomunikasi dengan obyek lainnya dan untuk membangunnya dibutuhkan interface.
- RMI terdiri dari RMI Client dan RMI Server.

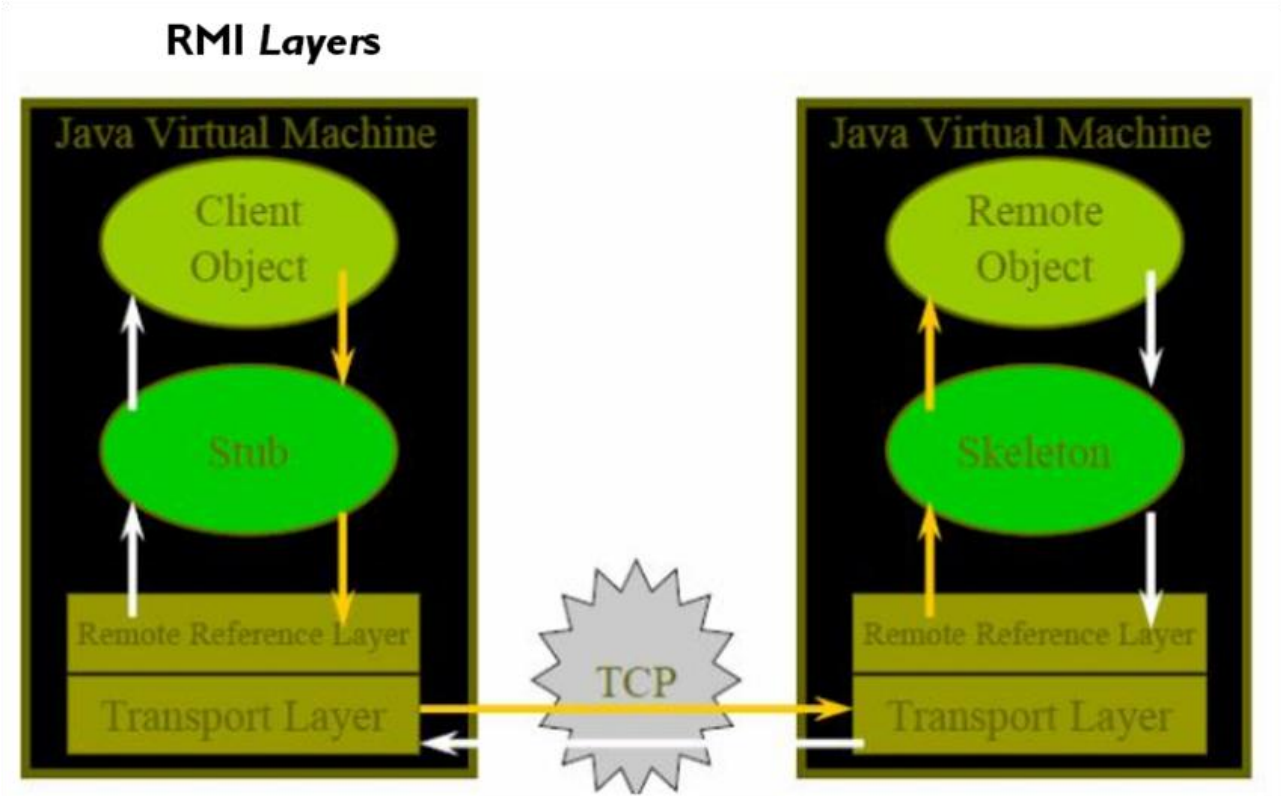
# Arsitektur RMI

- RMI Server akan mendaftarkan remote obyeknya ke RMI Registry melalui bind dengan nama unik.
- RMI Client yang akan melakukan suatu pemanggilan method dari remote obyek, harus meminta referensi obyek ke RMI Registry berdasarkan nama kelas obyek tersebut.
- Dalam RMI harus ada pendefinisian interface (behaviour) dan implementasi interface (berupa kelas).
- RMI hanya dimiliki oleh Bahasa Java saja.

# Arsitektur RMI



# RMI Layer




- Stub dan skeleton merupakan interface antara aplikasi dan RMI System. Stub bertindak sebagai client side proxy, sedangkan skeleton sebagai server side proxy.

# Stub dan Skeleton

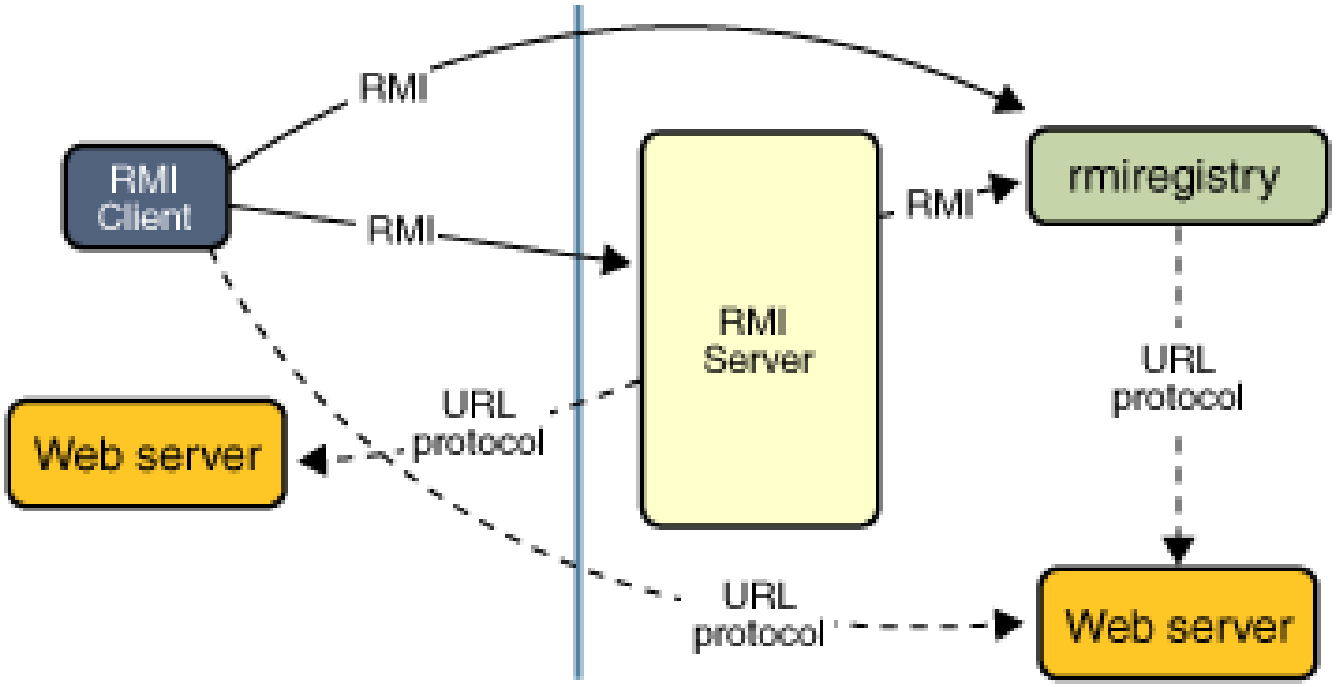
**Fungsi Stub:**

- Meminta lokasi remote server obyek pada remote reference layer
- Memberitahu remote reference layer bahwa semua data parameter telah terkirim sehingga pemanggilan method sesungguhnya dapat dilakukan oleh server.
- Memberitahu remote reference layer bahwa pemanggilan telah lengkap

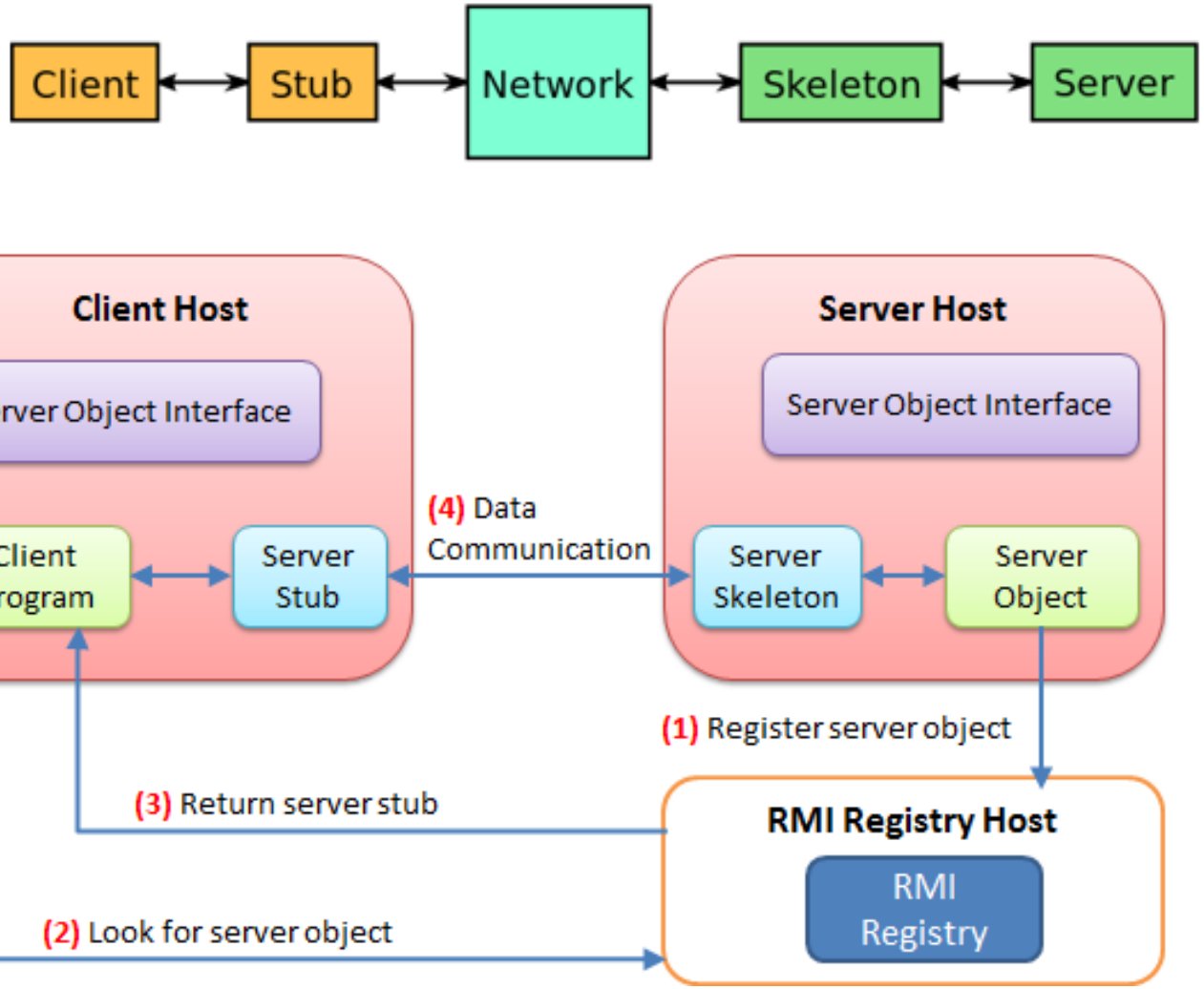
**Fungsi skeleton:**

- Mengirimkan panggilan method pada server obyek sesungguhnya.

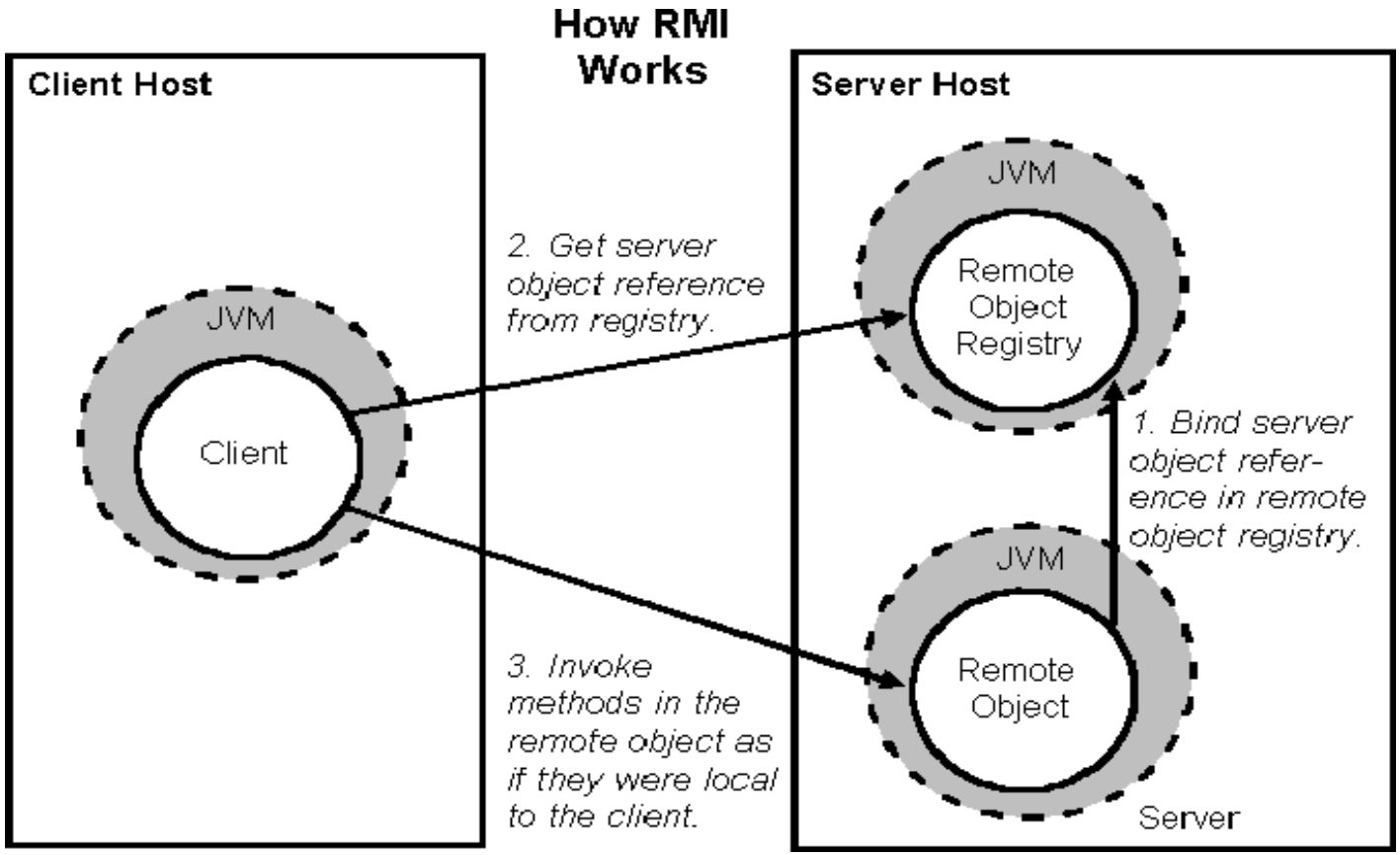
# Gambaran Aplikasi RMI



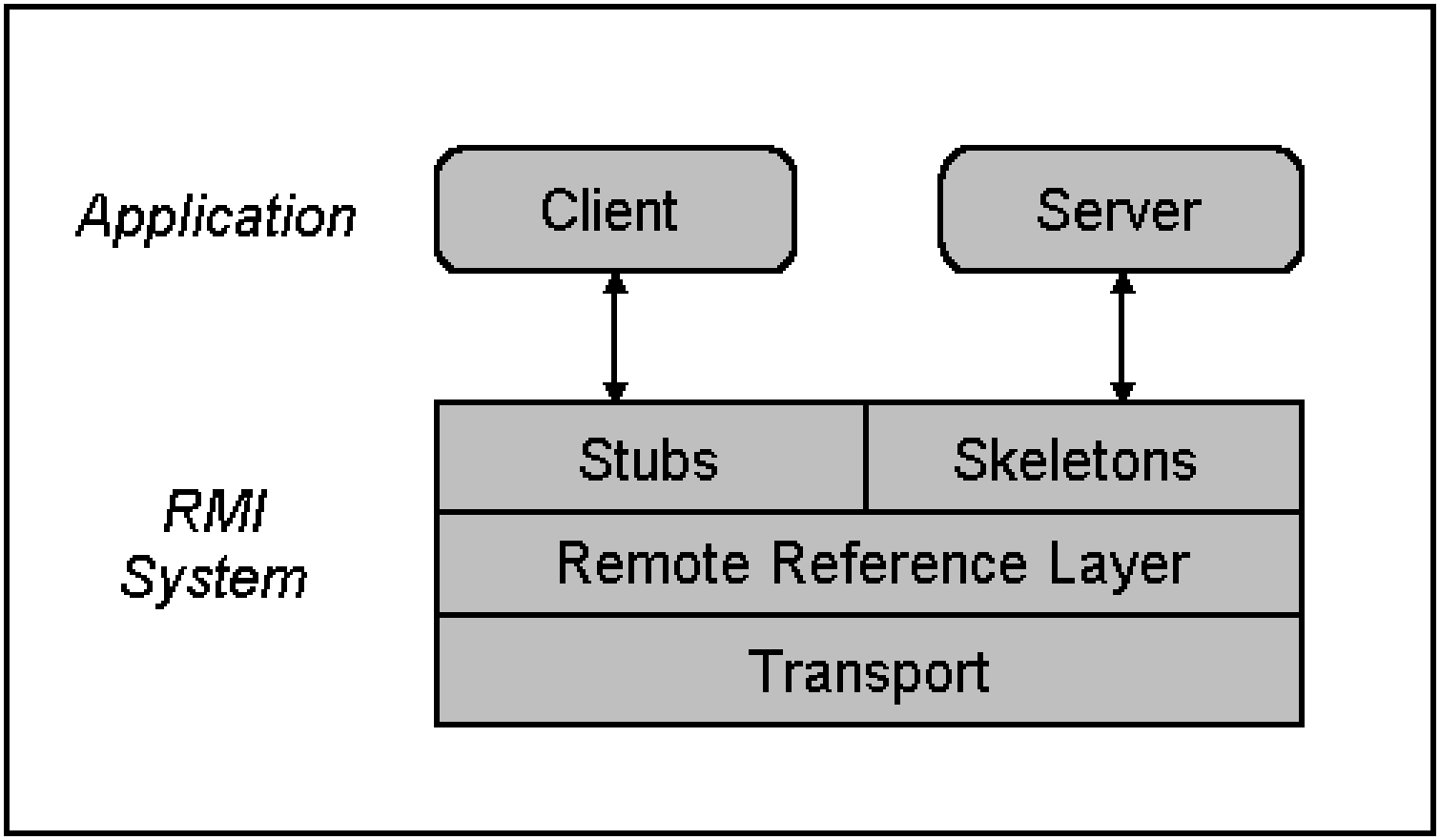
# Model Client Server RMI



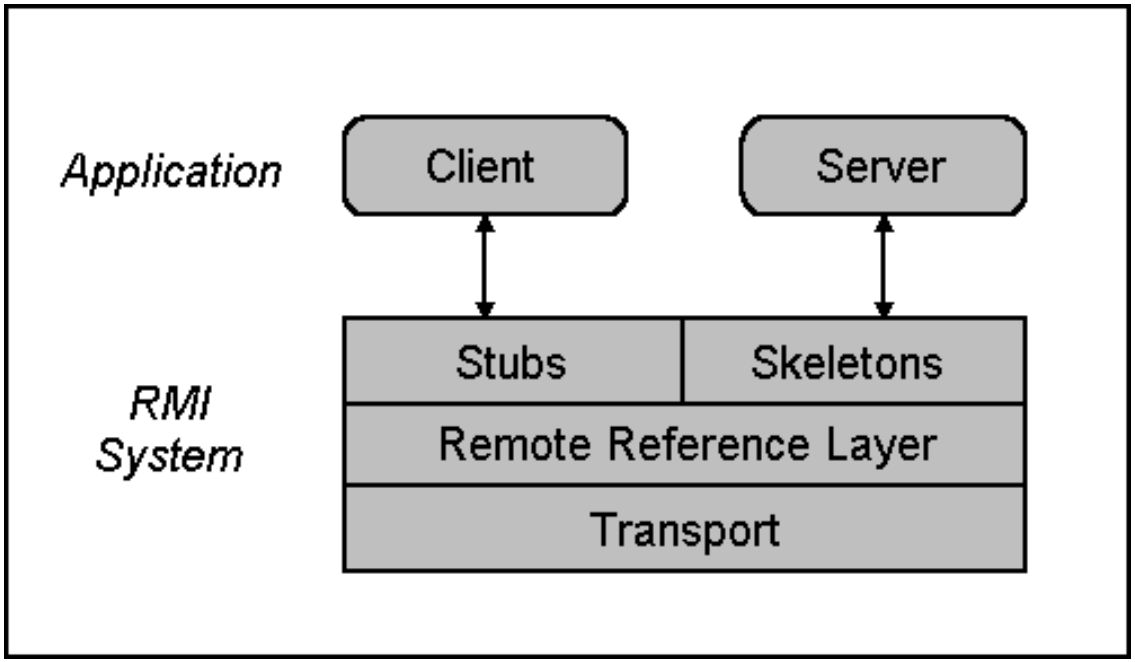
# Proses RMI


How RMI Works
Client Host
JVM
Client
2. Get server object reference from registry.
3. Invoke methods in the remote object as if they were local to the client.
Server Host
JVM
Remote Object Registry
1. Bind server object reference in remote object registry.
JVM
Remote Object
Server

# Sistem RMI


Application
Client
Server
RMI System
Stubs
Skeletons
Remote Reference Layer
Transport

# Sistem RMI


Application
Client
Server
Stubs
Skeletons
RMI System
Remote Reference Layer
Transport

# Proses RMI Dasar

1. The server program that has control of the remote object registers an interface with a naming service.
2. The interface contains the signatures for those methods of the object that the server wishes to make publicly available.
3. Stub.
4. Skeleton.
5. The client program invokes a method of the remote object.
6. An equivalent method is being called in the stub.
7. Marshalling (penggabungan)
8. UnMarshalling.
9. Finally, the skeleton calls the implementation of the method on the server.

# Proses RMI

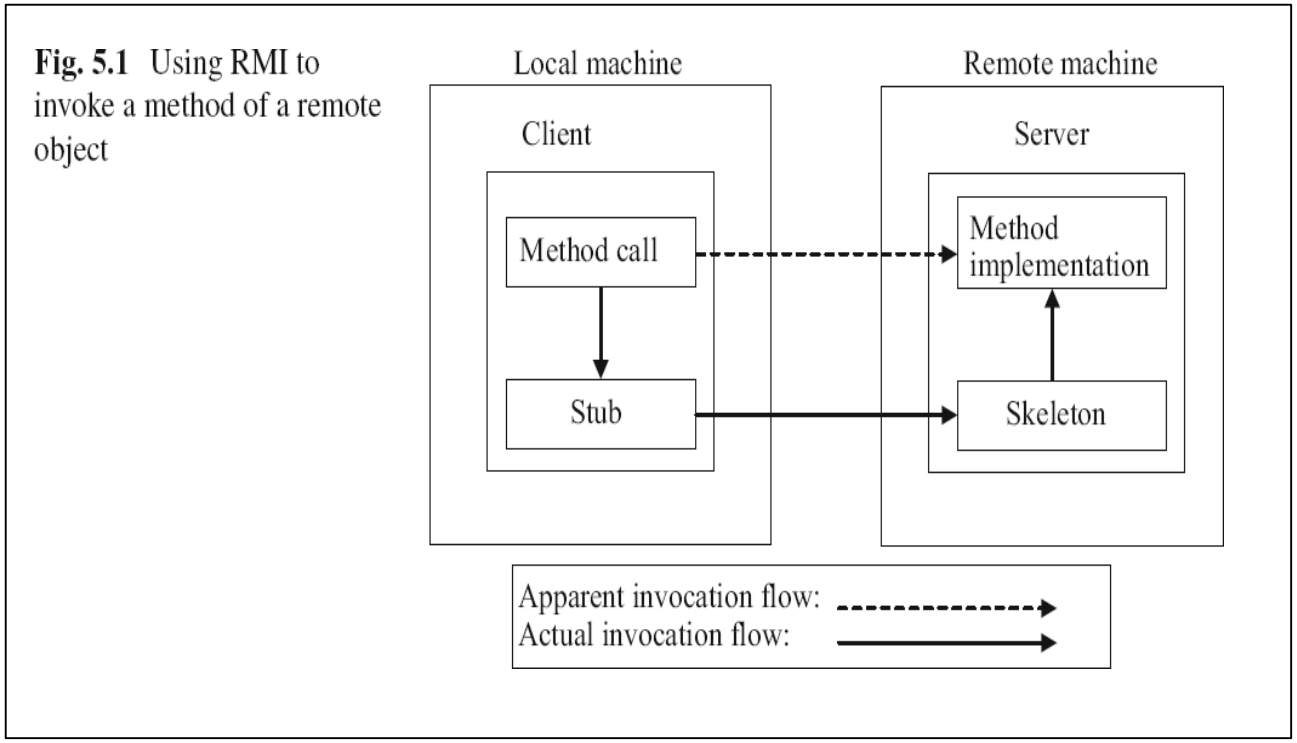
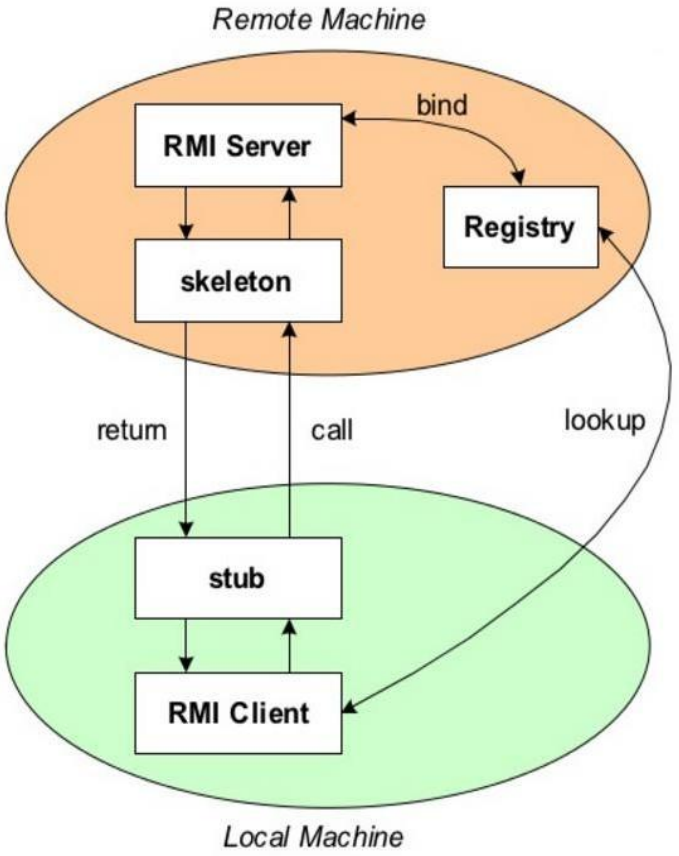


Fig. 5.1 Using RMI to invoke a method of a remote object
Local machine
Remote machine
Client
Server
Method call
Method implementation
Stub
Skeleton
Apparent invocation flow:
Actual invocation flow:

# Arsitektur RMI Umum

- Pertama, server harus mengikatkan (bind) nama-nya ke registry
- Client mencari (*lookup*) nama server di dalam registry untuk membangun referensi remote
- Stub menyusun (*serializing*) parameter-parameter ke skeleton, skeleton memanggil metode remote dan menyusun hasil balik ke stub tersebut



# Rincian Implementasi

Paket yang digunakan dalam implementasi aplikasi client-server RMI adalah

- java.rmi
- java.rmi.server dan
- java.rmi.registry

Langkah-langkah dasarnya adalah

1. Membuat interface.
2. Mendefinisikan suatu kelas yang mengimplementasikan interface ini.
3. Membuat proses server.
4. Membuat proses client.

# Kompilasi dan Eksekusi

1. Kompilasikan semua file dengan javac.
2. Buka command prompt pertama, jalankan registry RMI.
3. Buka jendela (console) baru dan jalankan server
4. Buka console ketiga dan jalankan client-nya.

# Kompilasi dan Eksekusi (Lanj.)

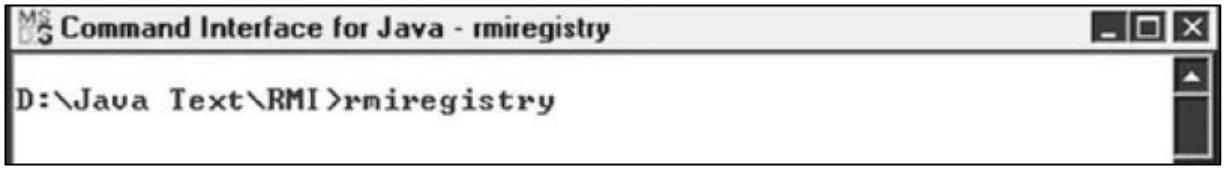
**1. Kompilasikan semua file dengan javac.**

- javac Hello.java
- javac HelloImpl.java
- javac HelloServer.java
- javac HelloClient.java

**2. Jalankan registry RMI.**

Masukkan perintah: rmiregistry

Saat ini dieksekusi, indikasi bahwa sesuatu telah terjadi hanyalah perubahan pada judul jendela command prompt.

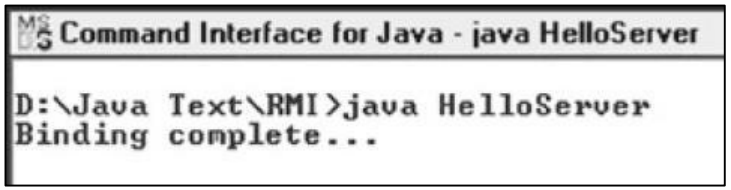


# Kompilasi dan Eksekusi (Lanj.)

**3. Buka console baru dan jalankan server.**

Dari jendela baru tersebut, panggil Java compiler: java HelloServer

Output server terlihat sebagai berikut:




**4. Buka console baru ketiga dan jalankan client.**

Lagi, panggil Java compiler: java HelloClient

Output diperlihatkan sebagai berikut: