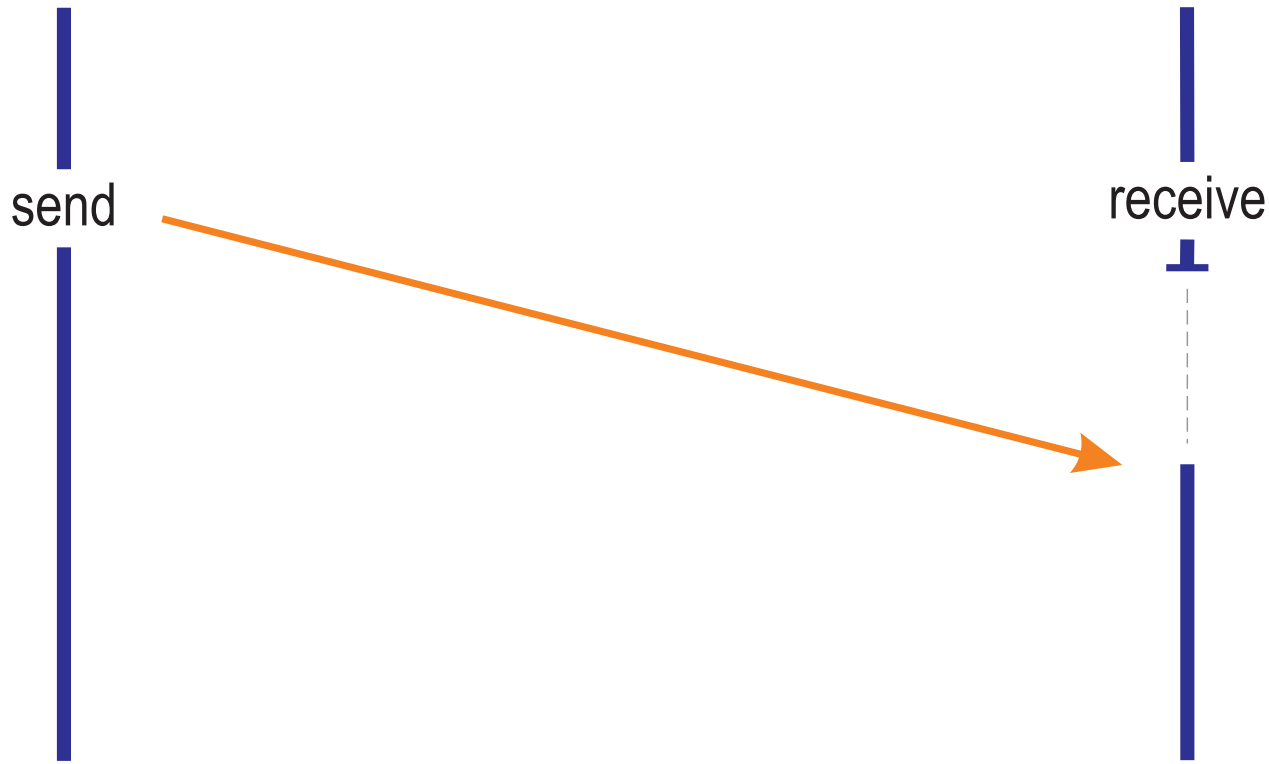
Distribuované systémy a výpočty (01)

Jan Janeček

KP FEL ČVUT

30/09/2007

- asynchronní komunikace
  - přímá
  - nepřímá
- synchronní komunikace
- příjem na více kanálech


send
receive

přímá asynchronní komunikace

- odeslání zprávy

  send *expr* to *receiver*

- příjem zprávy

  receive *var* from *sender*

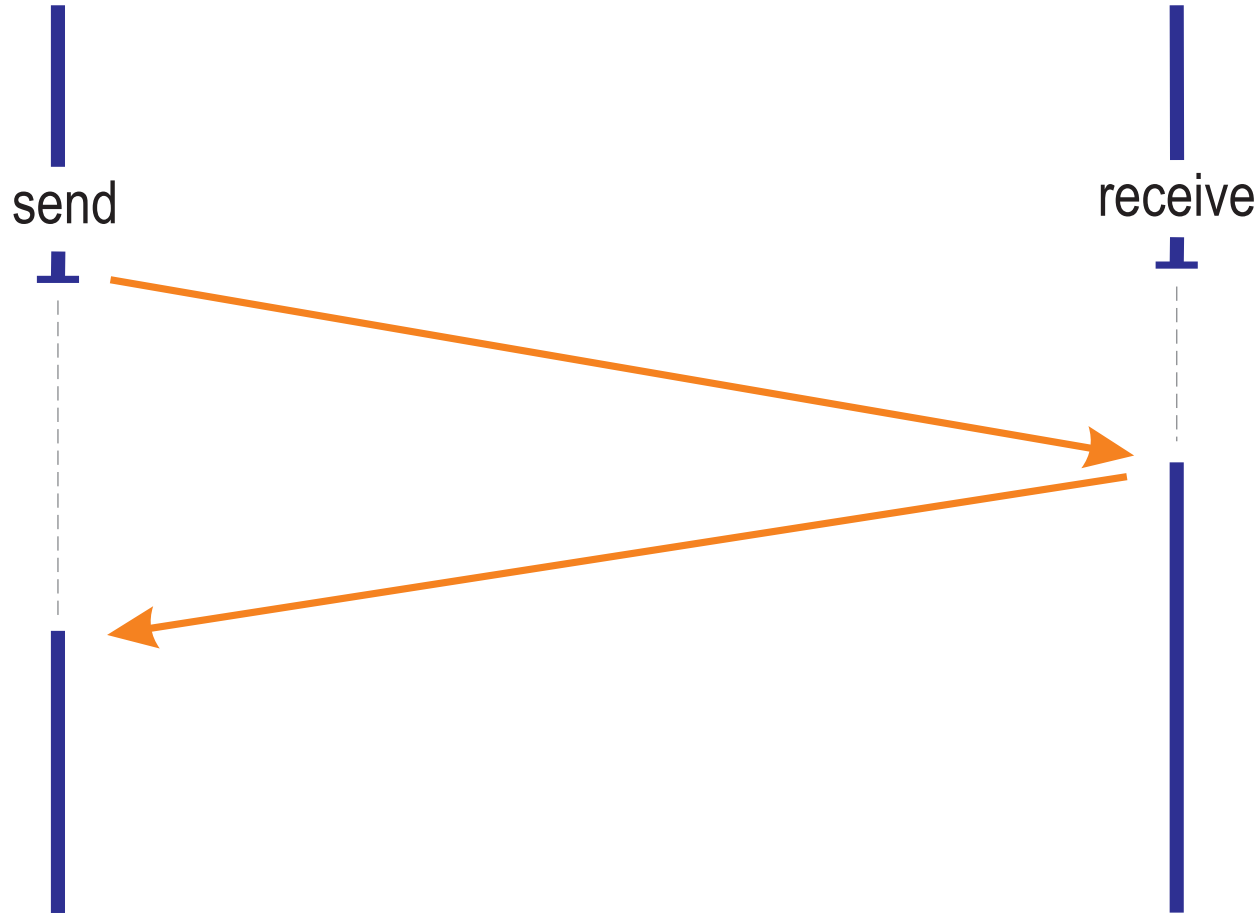
nepřímá asynchronní komunikace

- odeslání zprávy

  send *expr* to *channel*

- příjem zprávy

  receive *var* from *channel*


send
receive

přímá synchronní komunikace

- odeslání zprávy

  send E to R

- příjem zprávy

  receive V from S

přímá synchronní komunikace

- odeslání zprávy

  R ! E

- příjem zprávy

  S ? V

jednoduchý select

select

receive msg_1 from $ch_1 \Rightarrow stat_1$

or receive msg_2 from $ch_2 \Rightarrow stat_2$

or receive msg_3 from $ch_3 \Rightarrow stat_3$

end

strážený (guarded) select

select

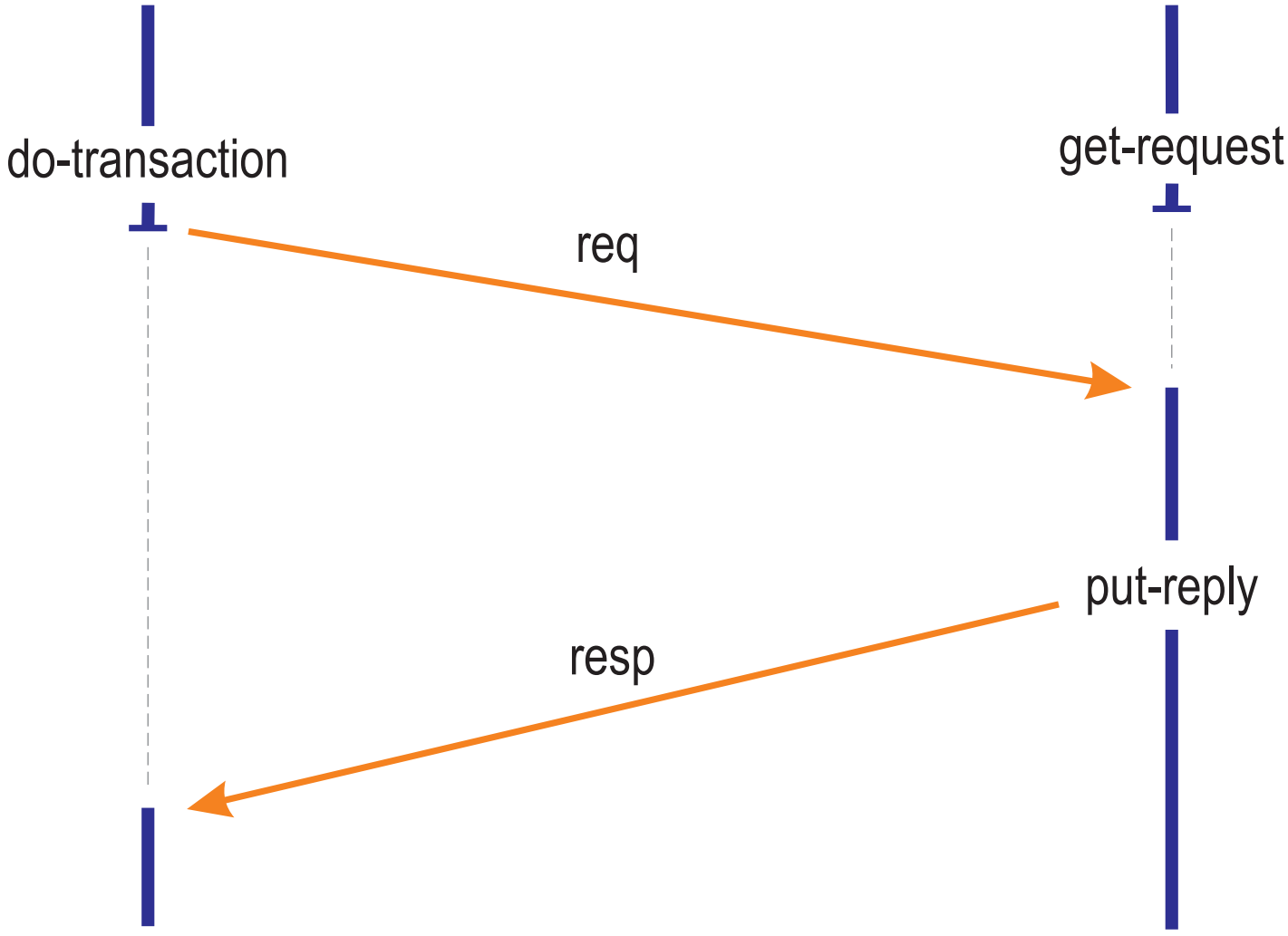
when $cond_1$ receive msg_1 from $ch_1 \Rightarrow stat_1$

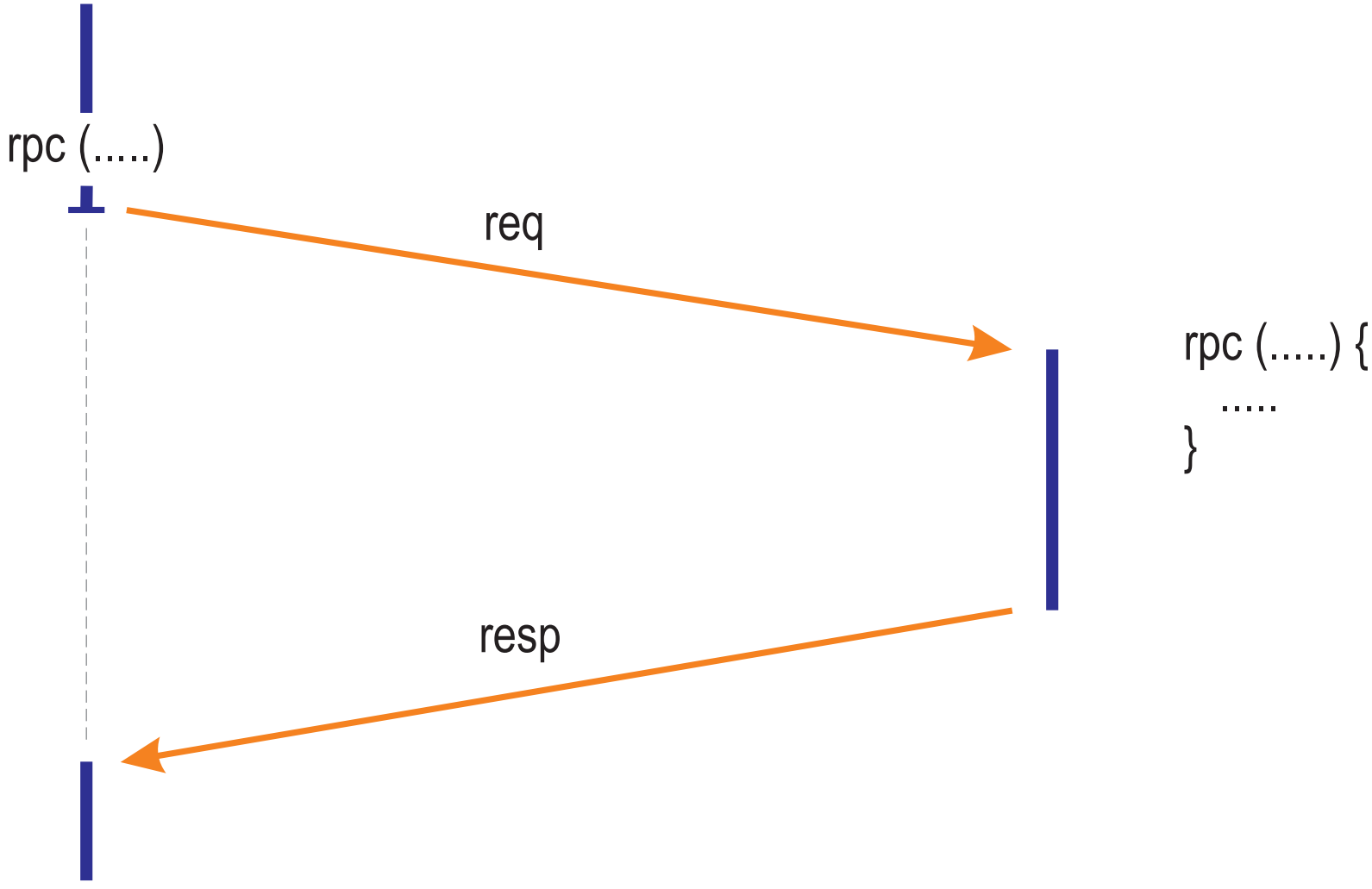
or when $cond_2$ receive msg_2 from $ch_2 \Rightarrow stat_2$

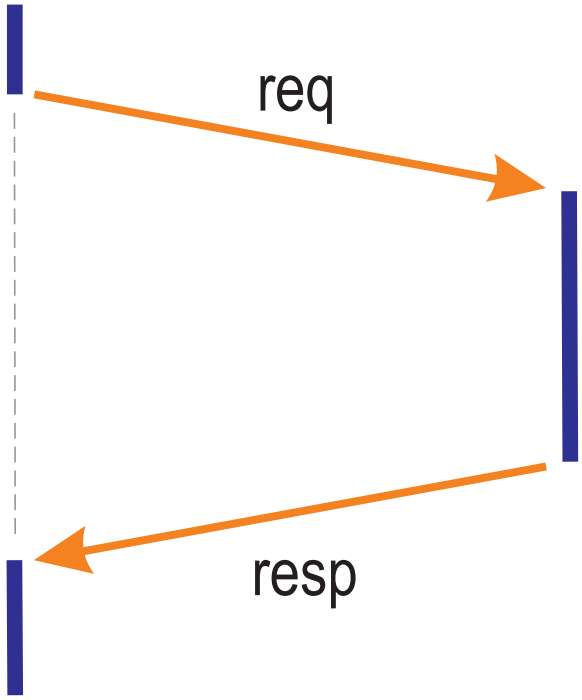
or when $cond_3$ receive msg_3 from $ch_3 \Rightarrow stat_3$

end

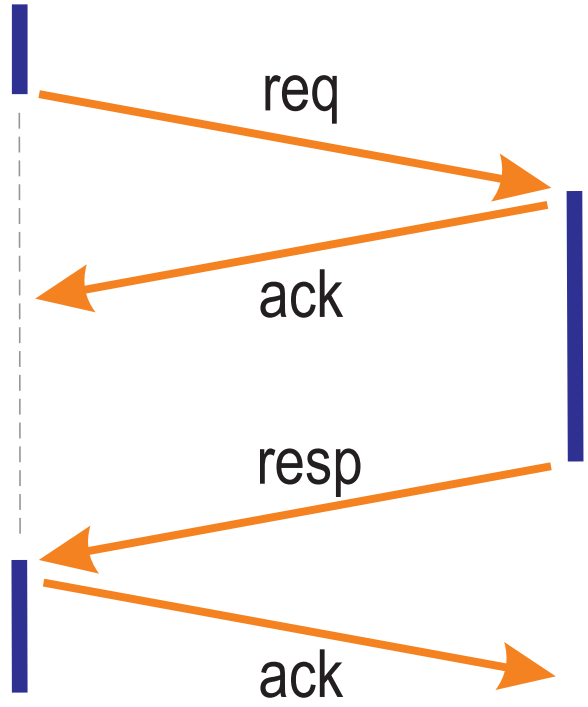
- transakce (Amoeba)
- RPC mechanismus
- sémantika RPC volání
- podpora RPC volání
- rendez-vous



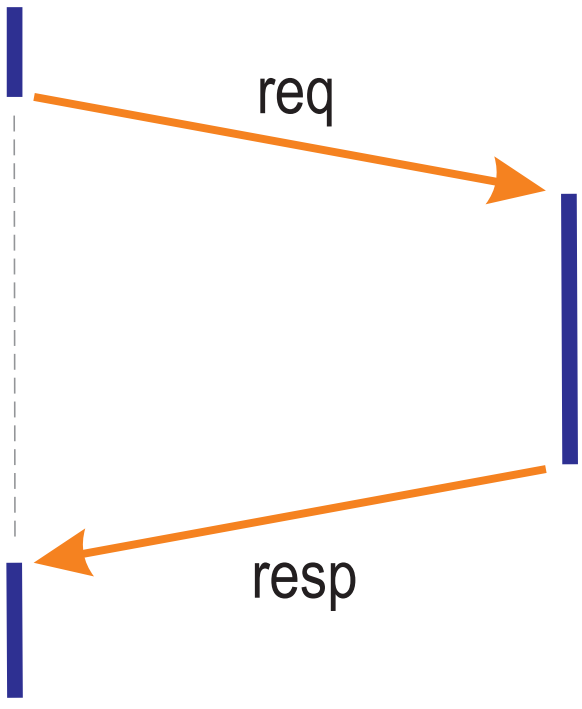




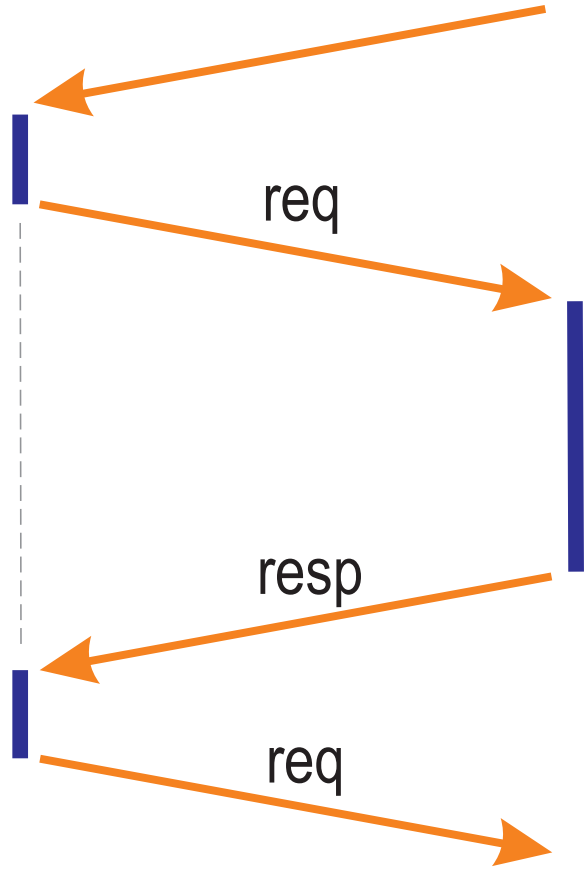
a



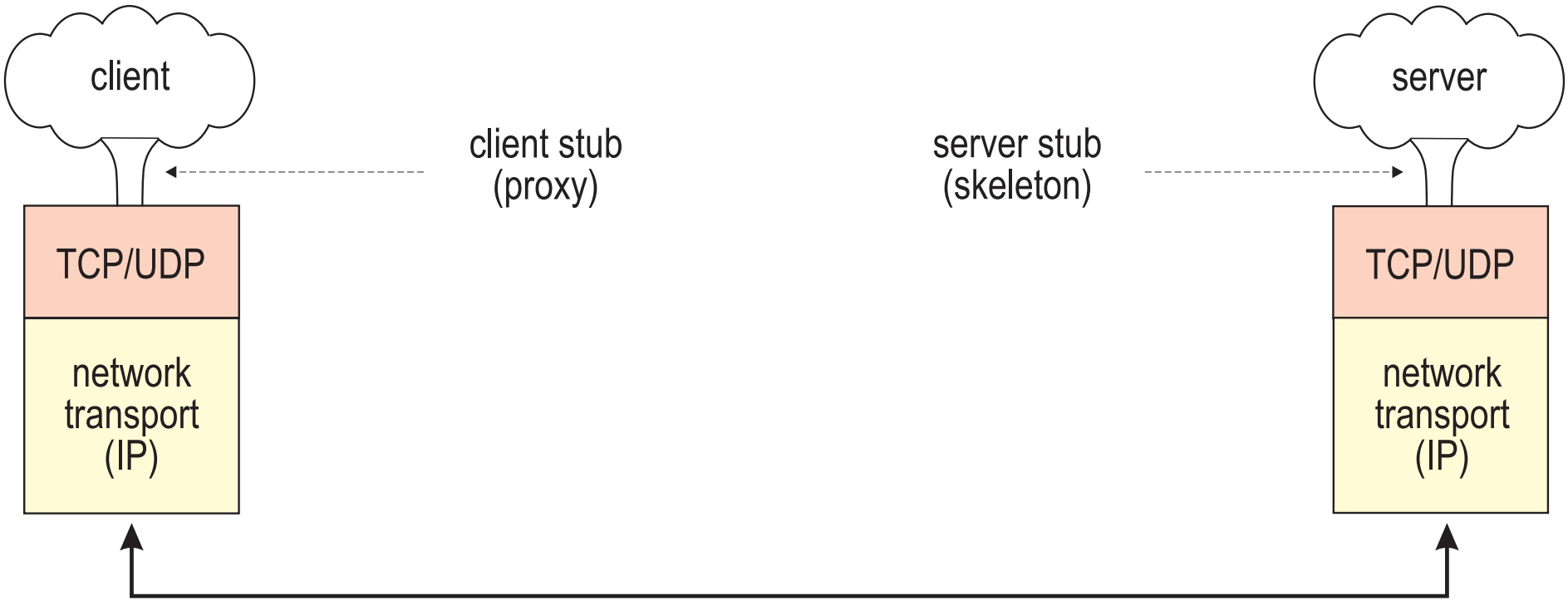
b

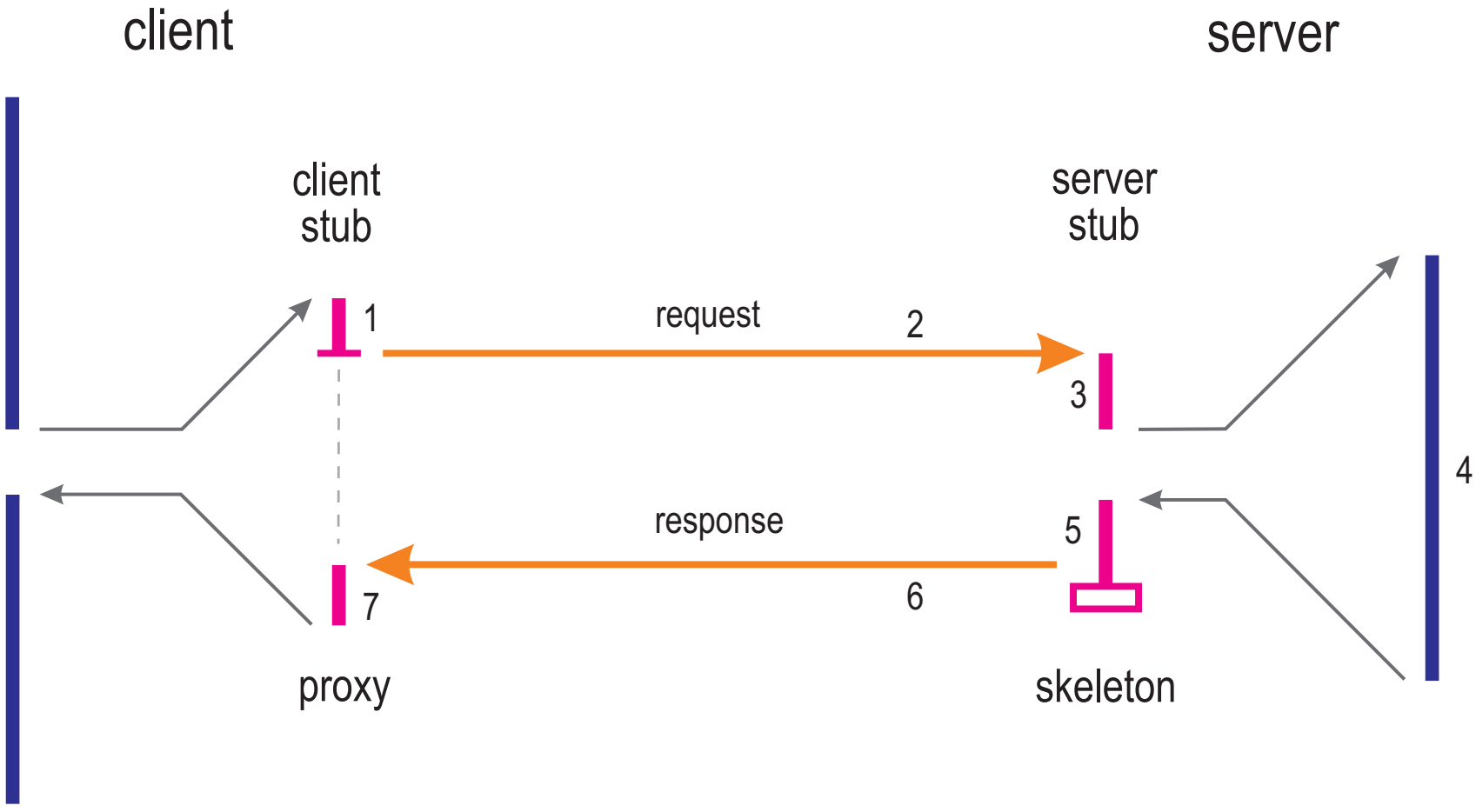


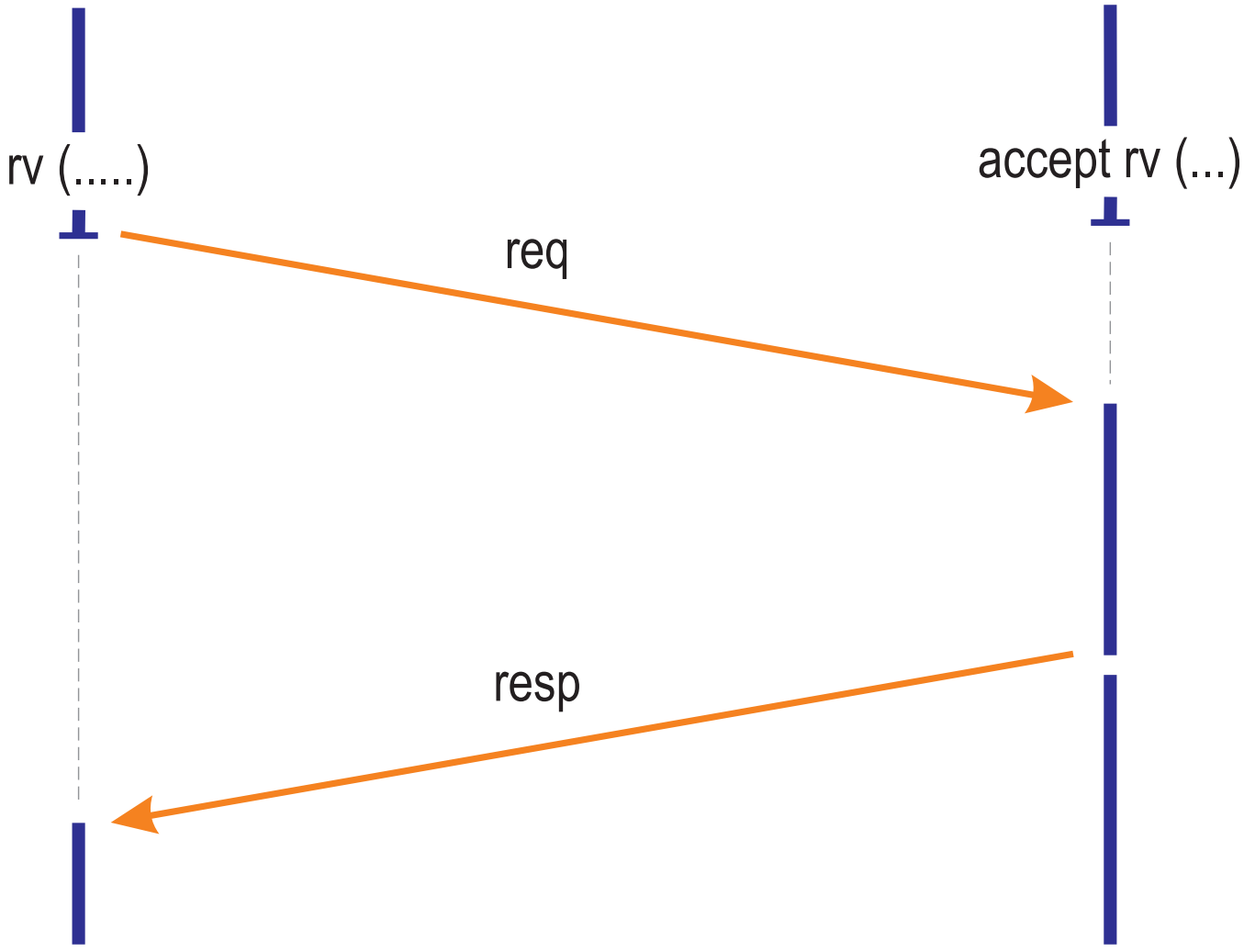
a



c


client
client stub
(proxy)
server stub
(skeleton)
server
TCP/UDP
network
transport
(IP)
TCP/UDP
network
transport
(IP)





client

call pname(aplist)

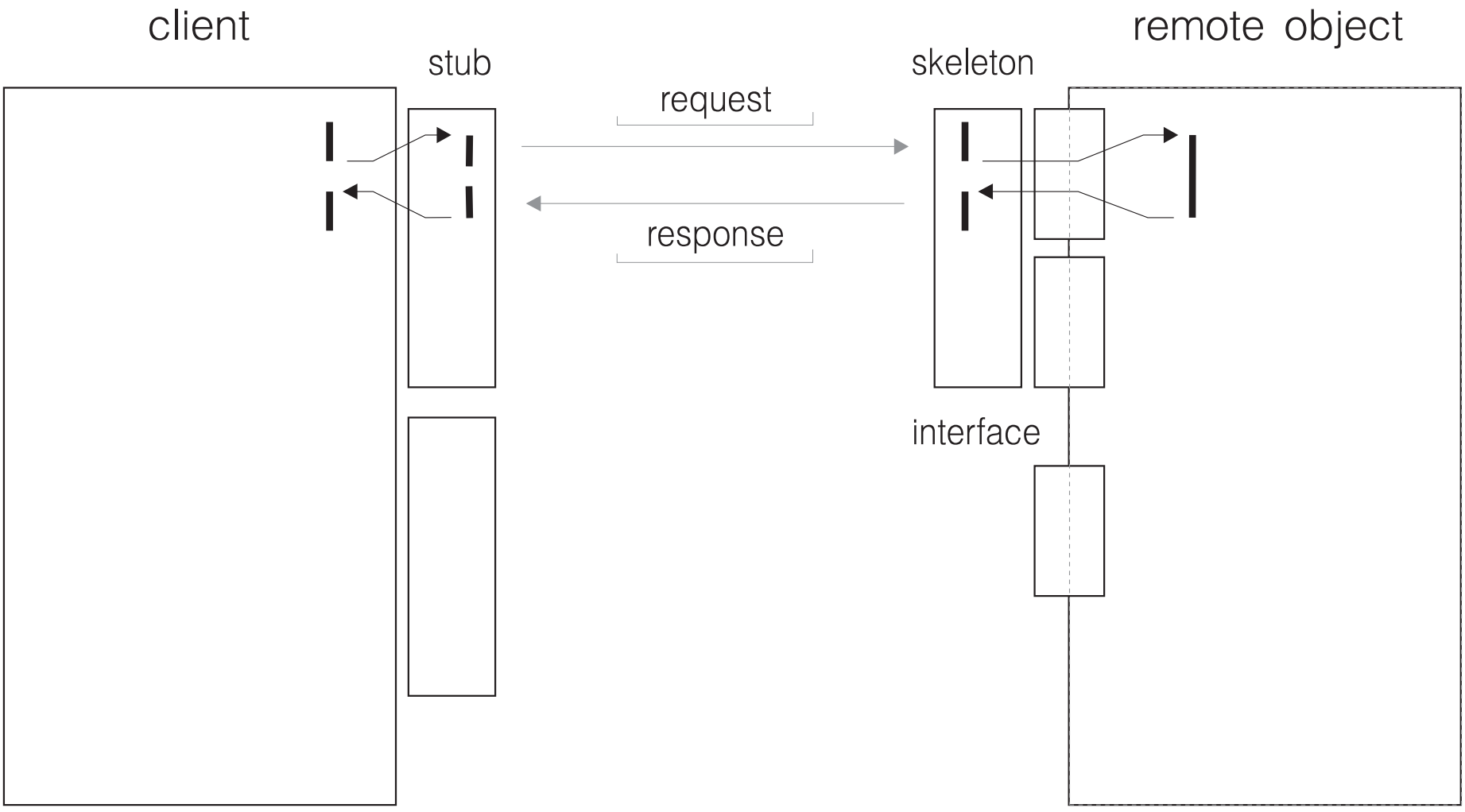
server

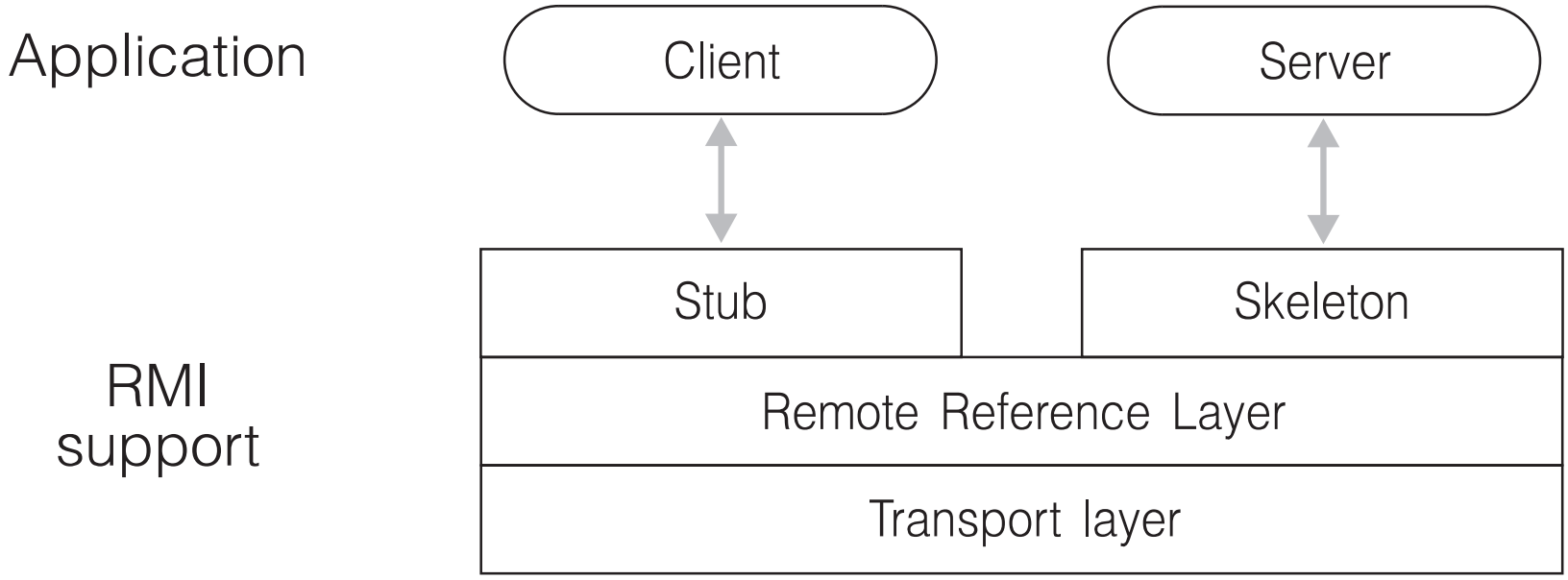
accept pname (fplist)

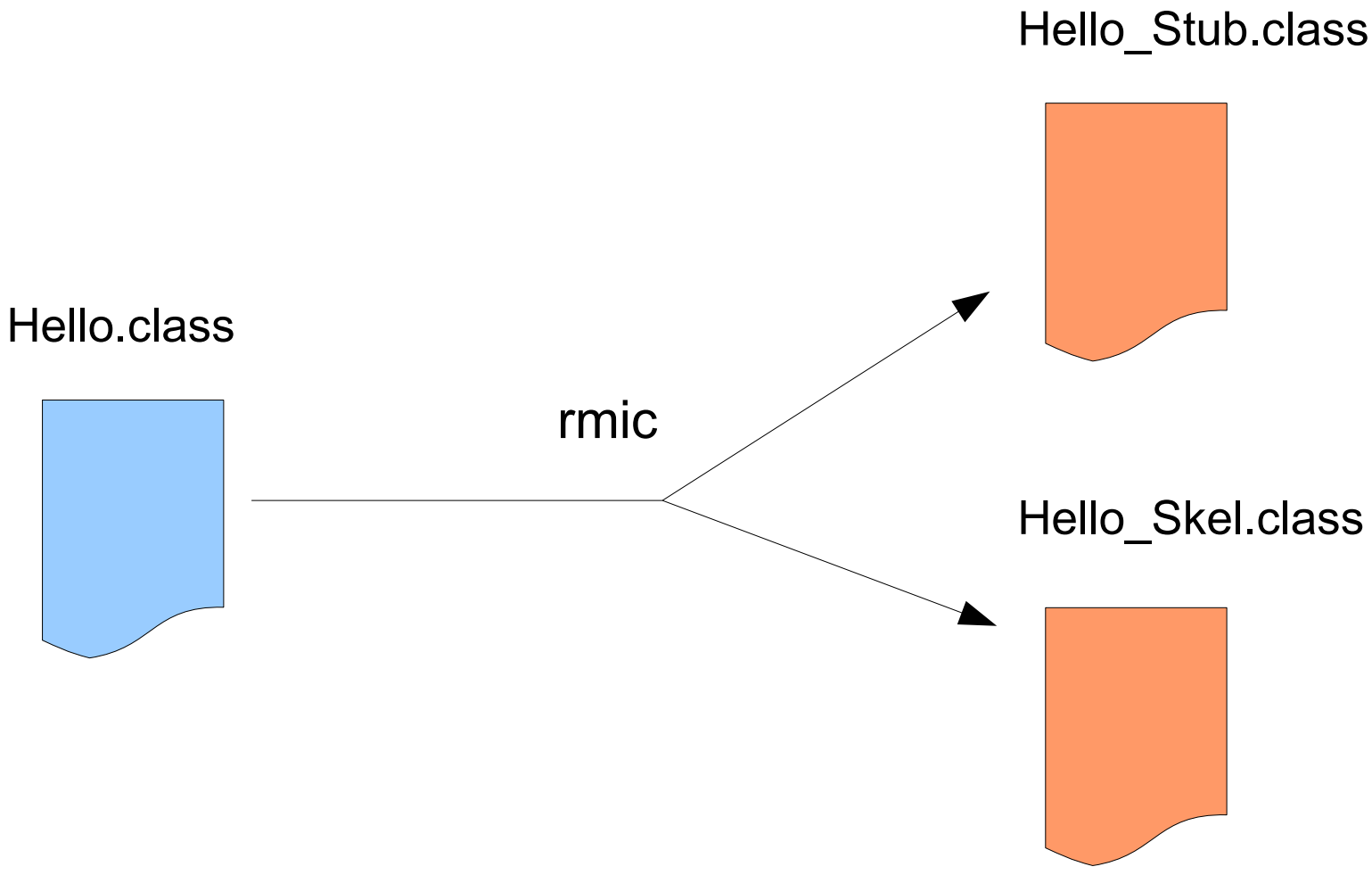
begin

...

end


client
stub
request
response
skeleton
remote object
interface


Application
Client
Server
Stub
Skeleton
RMI
support
Remote Reference Layer
Transport layer



```
package hello;
public interface Hello extends java.rmi.Remote {
  String sayHello(String s) throws java.rmi.RemoteException;
}
```

```
package hello;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
public class HelloImpl extends UnicastRemoteObject implements
Hello {
  private String greeting = "Hello";
  public HelloImpl(String s) throws RemoteException {
    super();
    greeting = s;
  }
  public String sayHello(String s) throws RemoteException {
    return greeting+" "+s+"!";
  }
```

```
  public static void main(String args[]) {
    try {
      System.setSecurityManager(new RMISecurityManager());
    }
    catch(Exception e) {
      System.out.println(e); return;
    }
    try {
      HelloImpl obj = new HelloImpl("Salut");
      Naming.rebind("//java/HelloServer",obj);
      System.out.println("HelloServer bound in registry");
    }
    catch(Exception e) {
      System.out.println(e);
    }
  }
}
```

```
package hello;
import java.rmi.*;
public class HelloClient {
  static String message = "";
  public static void main(String[] args) {
    try {
      Hello obj = (Hello)Naming.lookup("//java/HelloServer");
      message = obj.sayHello("client");
    }
    catch(Exception e) {
      System.out.println(e);
    }
    System.out.println(message);
  }
}
```

```
package hello;
import java.awt.*;
import java.rmi.*;
public class HelloApplet extends java.applet.Applet {
  String message = "";
  public void init() {
    try {
      Hello obj =
(Hello)Naming.lookup("//"+getCodeBase().getHost()
                    +"/HelloServer");
      message = obj.sayHello("client");
    }
    catch(Exception e) {
      System.out.println(e);
    }
  }
  public void paint(Graphics g) {
    g.drawString(message,25,50);
  }
```

```
<HTML>
<title>Hello World</title>
<center><h1>Hello World</h1></center>
The message from the HelloServer is:
<p>
<applet codebase="../.."
  code="examples.hello.HelloApplet"
  width=500 height=120>
</applet>
</HTML>
```