RPC – Remote Procedure Call

Stand-alone program

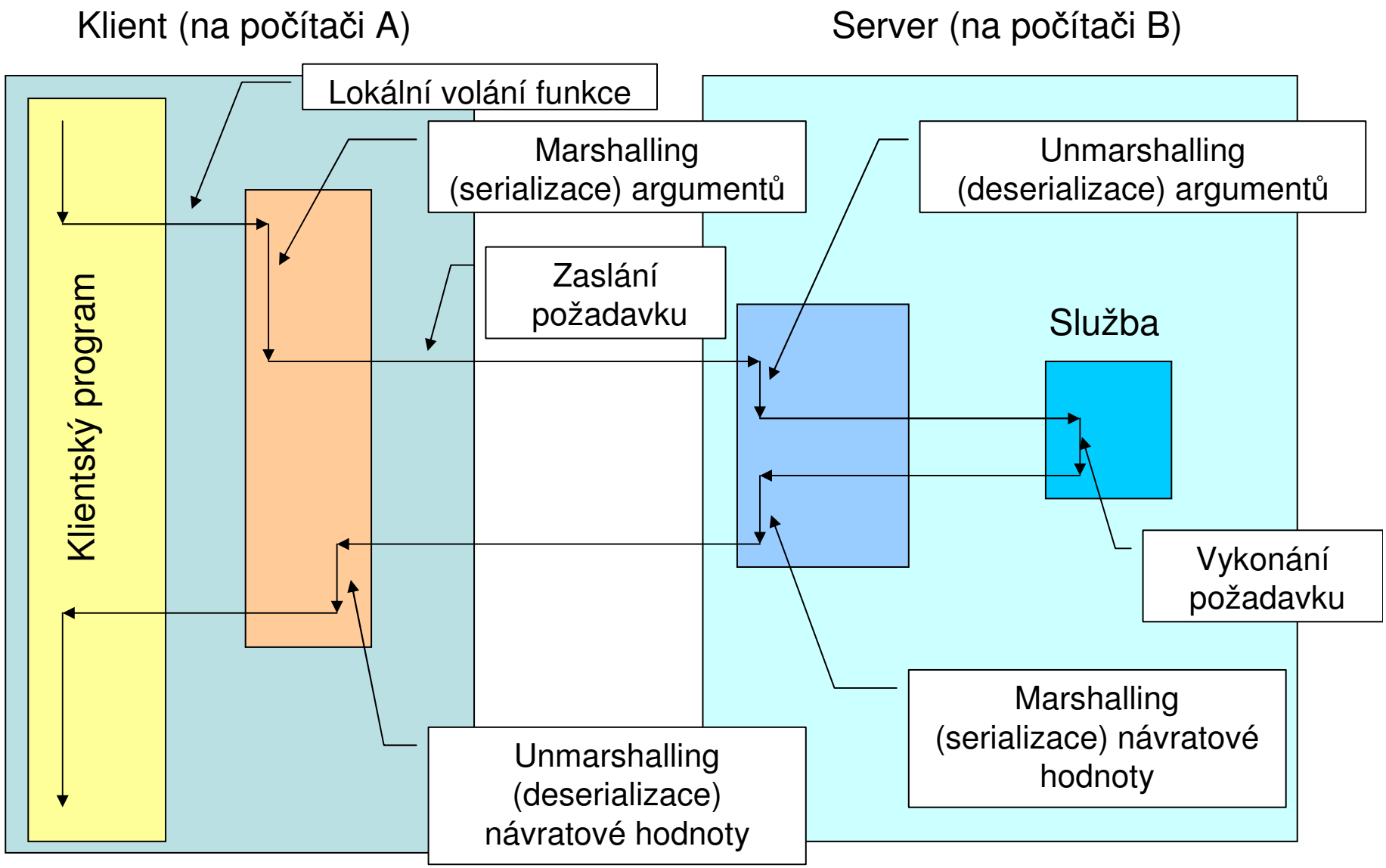
```
#include <stdio.h>
long bin_date(void);
char *str_date(long bintime);

main(int argc, char **argv) {
    long lresult; /* return from bin_date */
    char *sresult; /* return from str_date */
    if (argc != 1) {
        fprintf(stderr, "usage: %s\n", argv[0]);
        exit(1);
    }
    /* call the procedure bin_date */
    lresult = bin_date();
    printf("time is %ld\n", lresult);
    /* convert the result to a date string */
    sresult = str_date(lresult);
    printf("date is %s", sresult);
    exit(0);
}
```

RPC – Remote Procedure Call

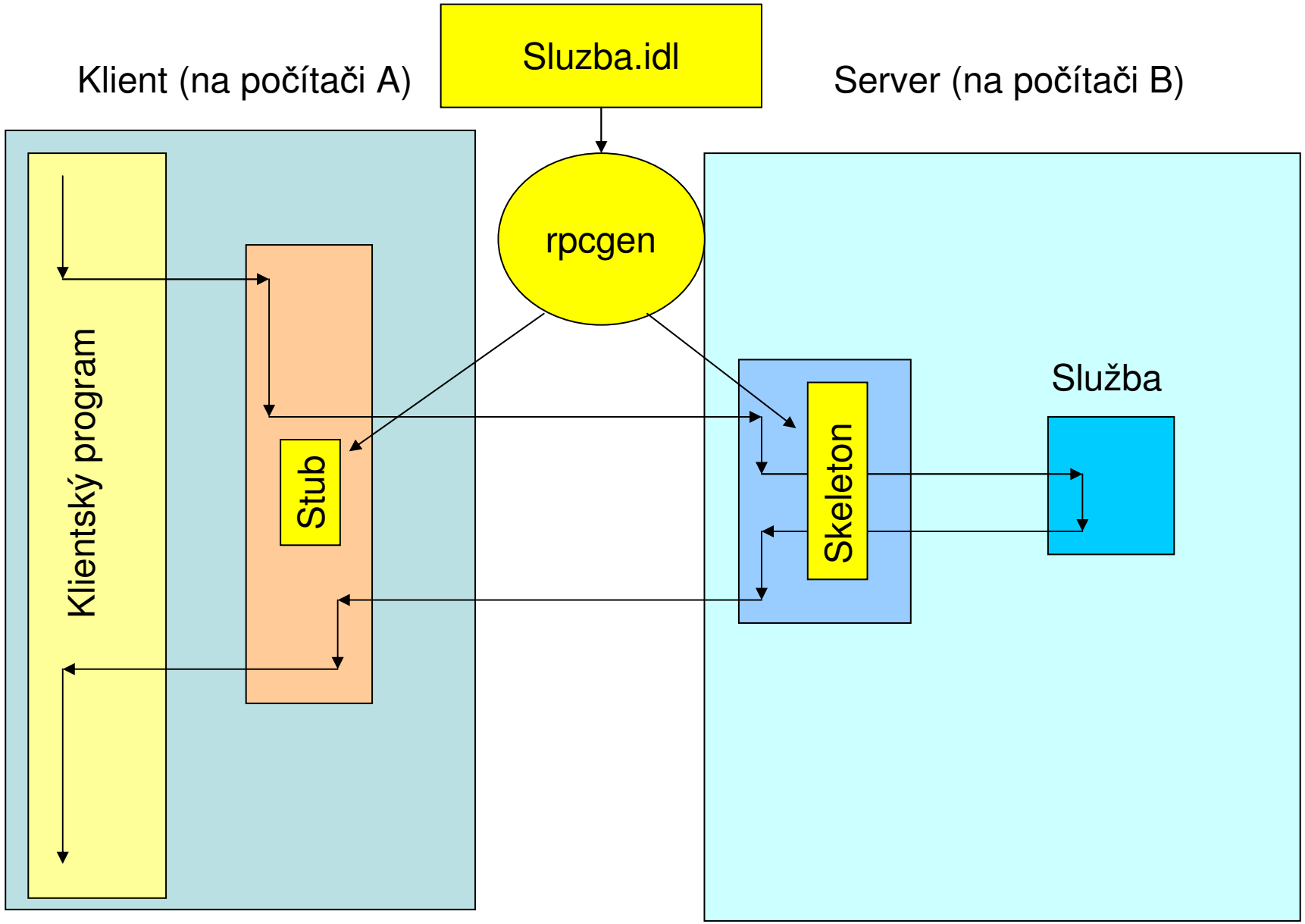
Stand-alone program: functions

```
long bin_date(void) {
    long timeval;
    long time(); /* Unix time function */
    timeval = time((long *)0);
    return timeval;
}

char *str_date(long bintime) {
    char *ptr;
    char *ctime(); /* Unix library function
*/
    ptr = ctime(&bintime);
    return ptr;
}
```

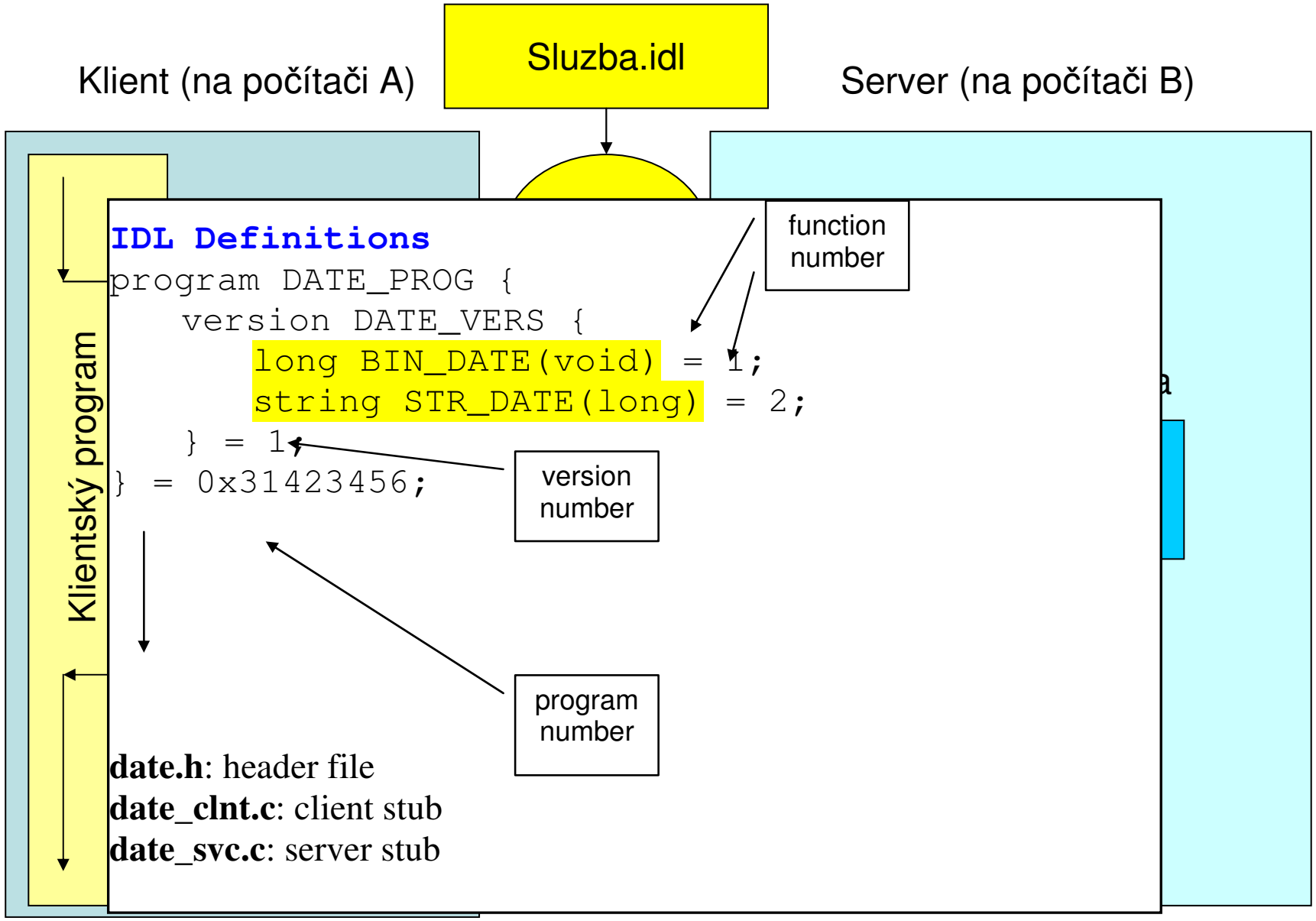
RPC – Remote Procedure Call



RPC – Remote Procedure Call



RPC – Remote Procedure Call



RPC – Remote Procedure Call

Vygenerovaný skeleton:

```
#include "date.h"
long *
bin_date_1_svc(void *argp,
               struct svc_req *rqstp)
{
    static long result;
    /* insert server code here */
    return &result;
}

char **
str_date_1_svc(long *argp,
               struct svc_req *rqstp)
{
     static char *result;
     /* insert server code here */
     return &result;
}
```

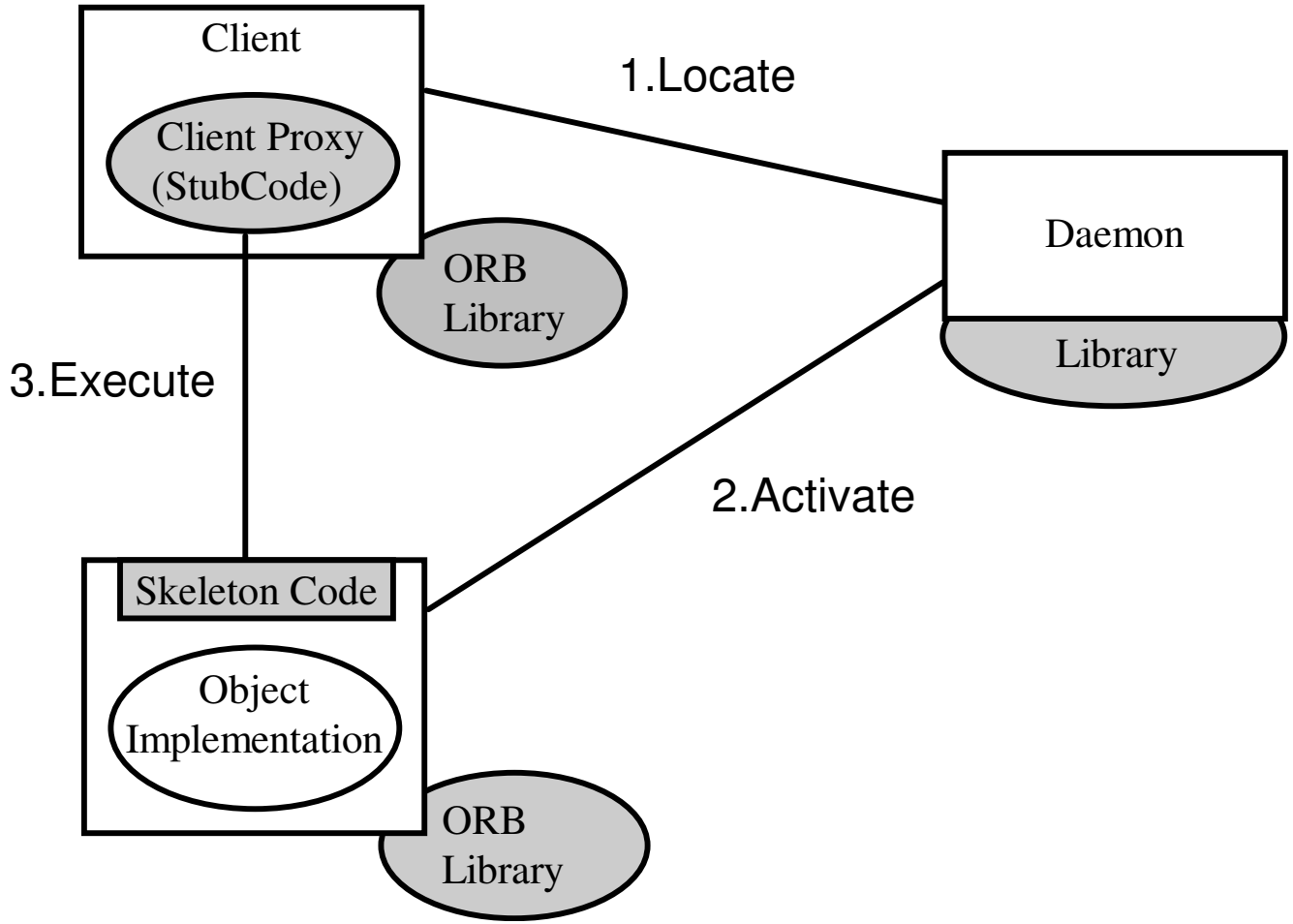
RPC – Remote Procedure Call

Modifikace klienta

```
#include <rpc/rpc.h>
#include “date.h”
CLIENT *cl; /* rpc handle */
cl = clnt_create(server,
                 DATE_PROG,
                 DATE_VERS,
                 “netpath”);

long *lresult; /* return from bin_date_1 */
char **sresult; /* return from str_date_1 */

if ((lresult=bin_date_1(NULL, cl))==NULL) {
  clnt_perror(cl, server);
  exit(1);
}
printf("time on %s is %ld\n",
       server,
       *lresult);
if ((sresult=str_date_1(lresult, cl)) == NULL) {
  clnt_perror(cl, server);
  exit(1);
}
printf("date is %s", *sresult);
```

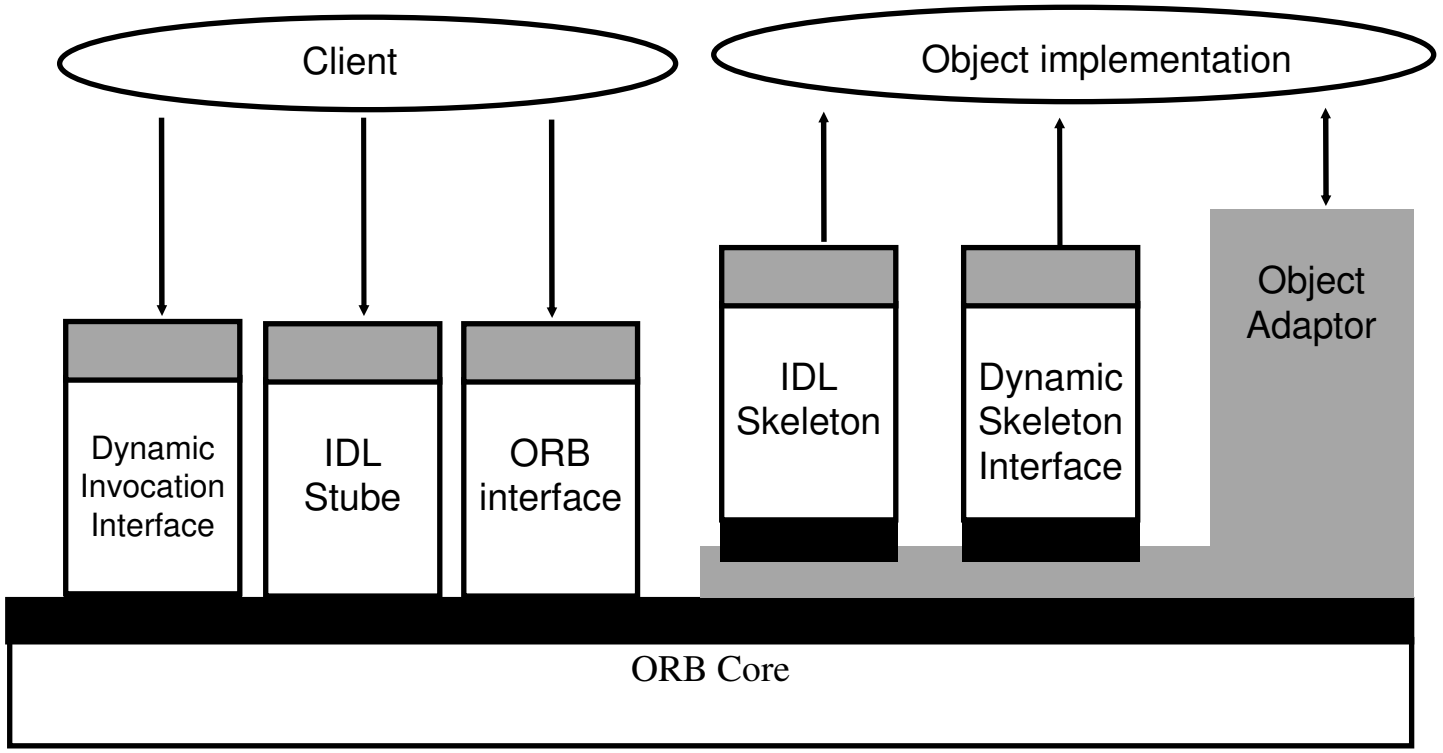
CORBA – Common Object Request Broaker Architecture

The Object ManagementGroup (OMG)
http://www.omg.org/



CORBA – Common Object Request Broaker Architecture

The Object ManagementGroup (OMG)
http://www.omg.org/



CORBA – Common Object Request Broaker Architecture

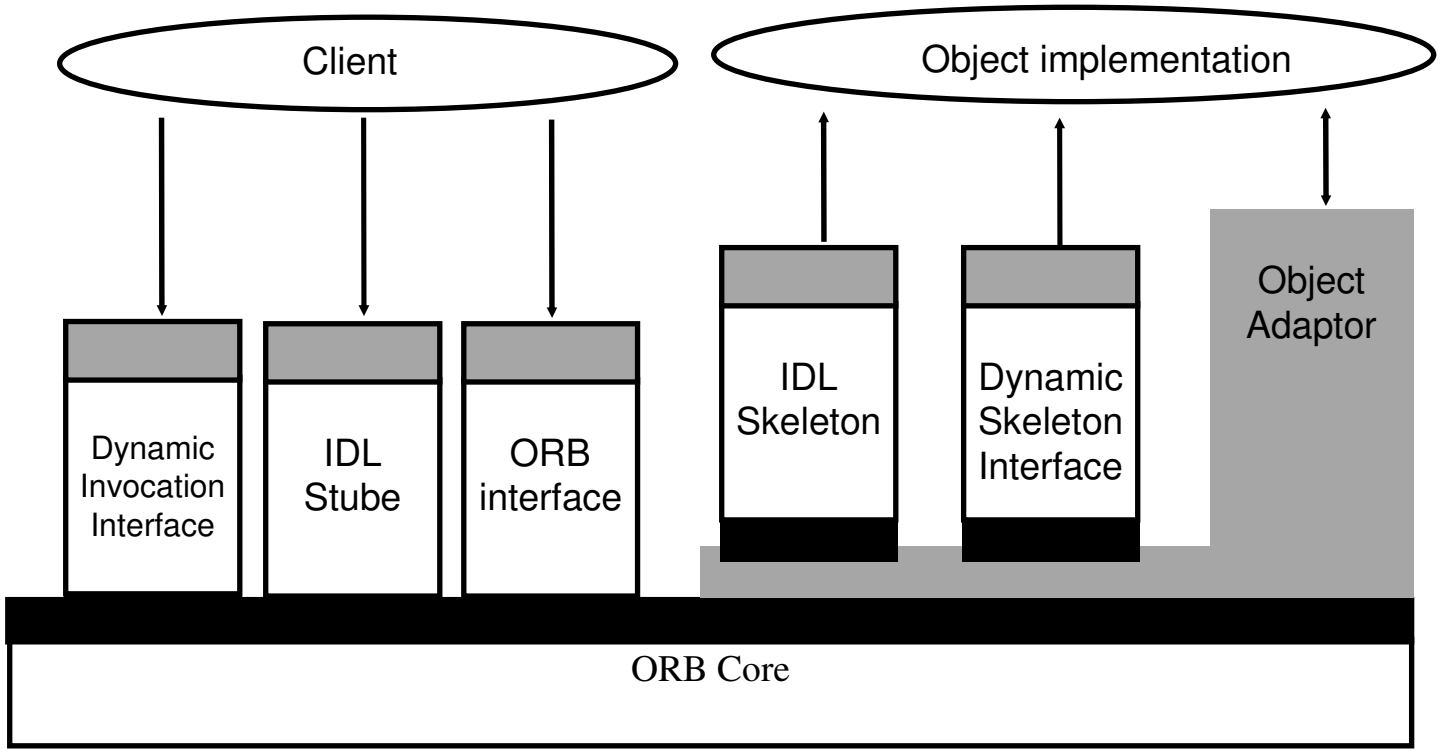
The Object ManagementGroup (OMG)
http://www.omg.org/

Příklad CORBA IDL:

```
#include "MISDefinitions.idl"

module MISData {

        struct TstData {
                unsigned short opId;
                unsigned short roleId;
        };


        interface TstDataIF {
                TstDef::TstResult GetTstData (
                        in unsigned short opId,
                        out MISData::TstData data
                        );

}
```

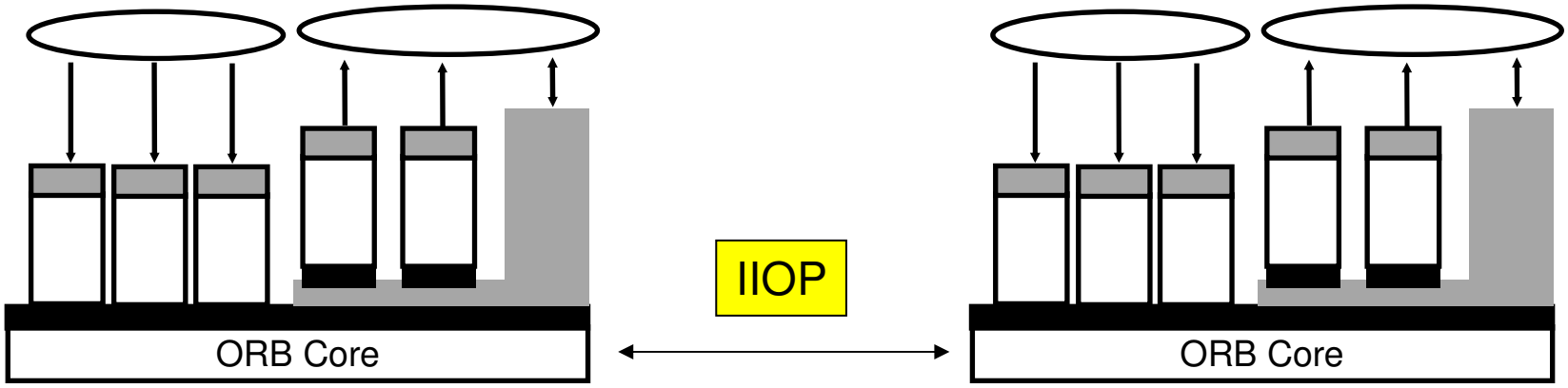
CORBA – Common Object Request Broaker Architecture

The Object ManagementGroup (OMG)
http://www.omg.org/



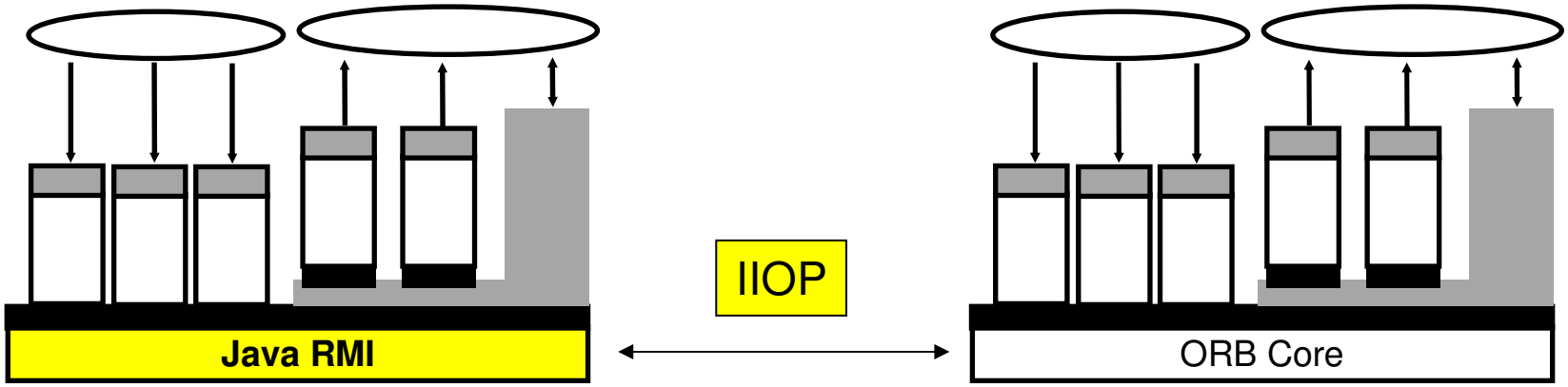
CORBA – Common Object Request Broaker Architecture

The Object ManagementGroup (OMG)
http://www.omg.org/



Internet Inter-ORB Protocol

Java RMI – Remote Method Invocation



Java RMI – Remote Method Invocation

CLIENT

```
public class Client
{
     public static void main(String[] args)
     {
          try
          {
               Registry registry = LocateRegistry
                                   .getRegistry("localhost", 1099);
               ServerExecutable exec = (ServerExecutable)
                                       registry.lookup("scitacka");
               System.out.println("Client question: 15 + 3");
               System.out.println("Server answer: " + exec.add(15, 3));
          }
          catch(NotBoundException ex)
          {
               System.err.println(ex);
          }
          catch(RemoteException ex)
          {
               System.err.println(ex);
          }
}
```

Java RMI – Remote Method Invocation

SPUŠTĚNÍ SERVERU

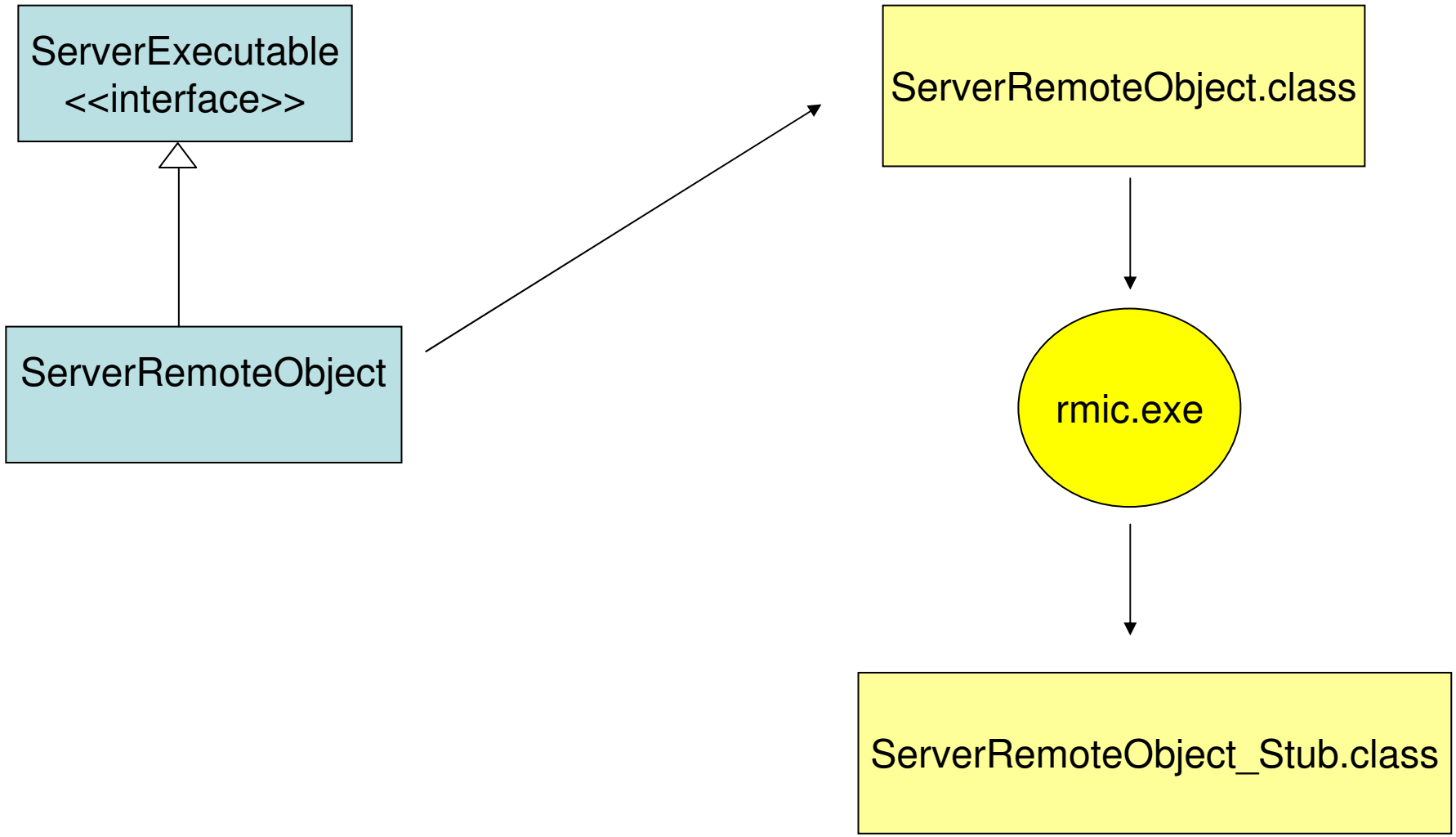
```
public class Server
{
    public static void main(String args[])
    {
        try
        {
            ServerExecutable stub =
            (ServerExecutable) UnicastRemoteObject.exportObject
                                        (new ServerRemoteObject(), 0);
             Registry registry =
                          LocateRegistry.createRegistry(1099);
            registry.rebind("rmi:///scitacka", stub);
            System.out.println("RMI Service is running.");
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
            return;
        }
    }
}
```

Java RMI – Remote Method Invocation

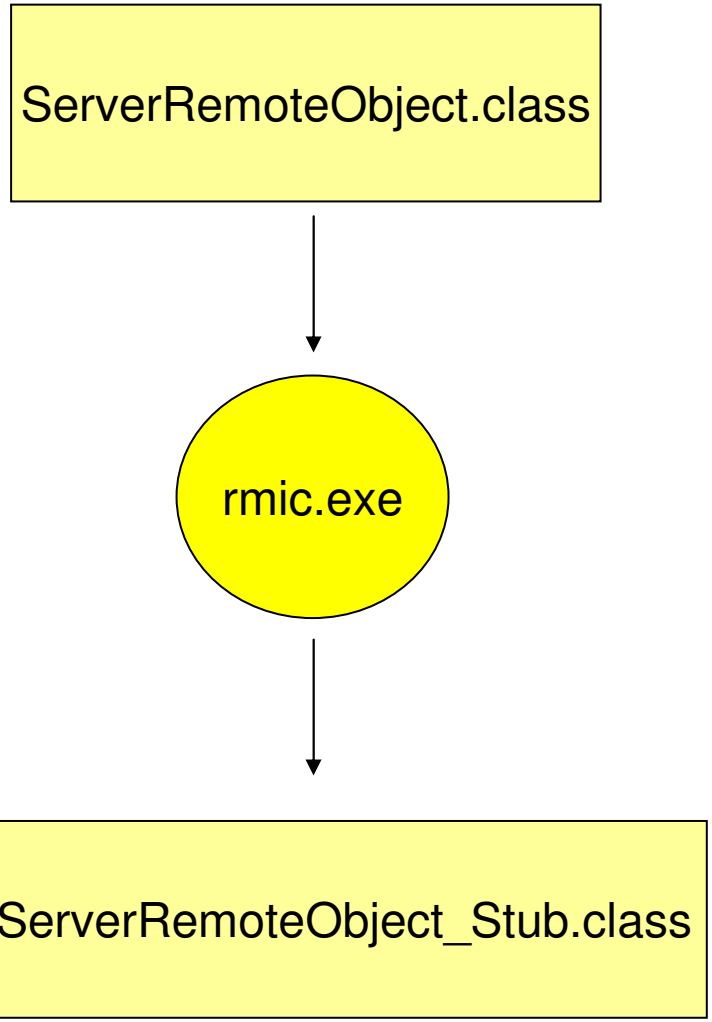
SERVER

```
public class ServerRemoteObject
                        extends UnicastRemoteObject
                        implements ServerExecutable
{

   public int add(int a, int b) throws RemoteException
   {
      return a + b;
   }

   private static final long serialVersionUID = 1L;
}
```

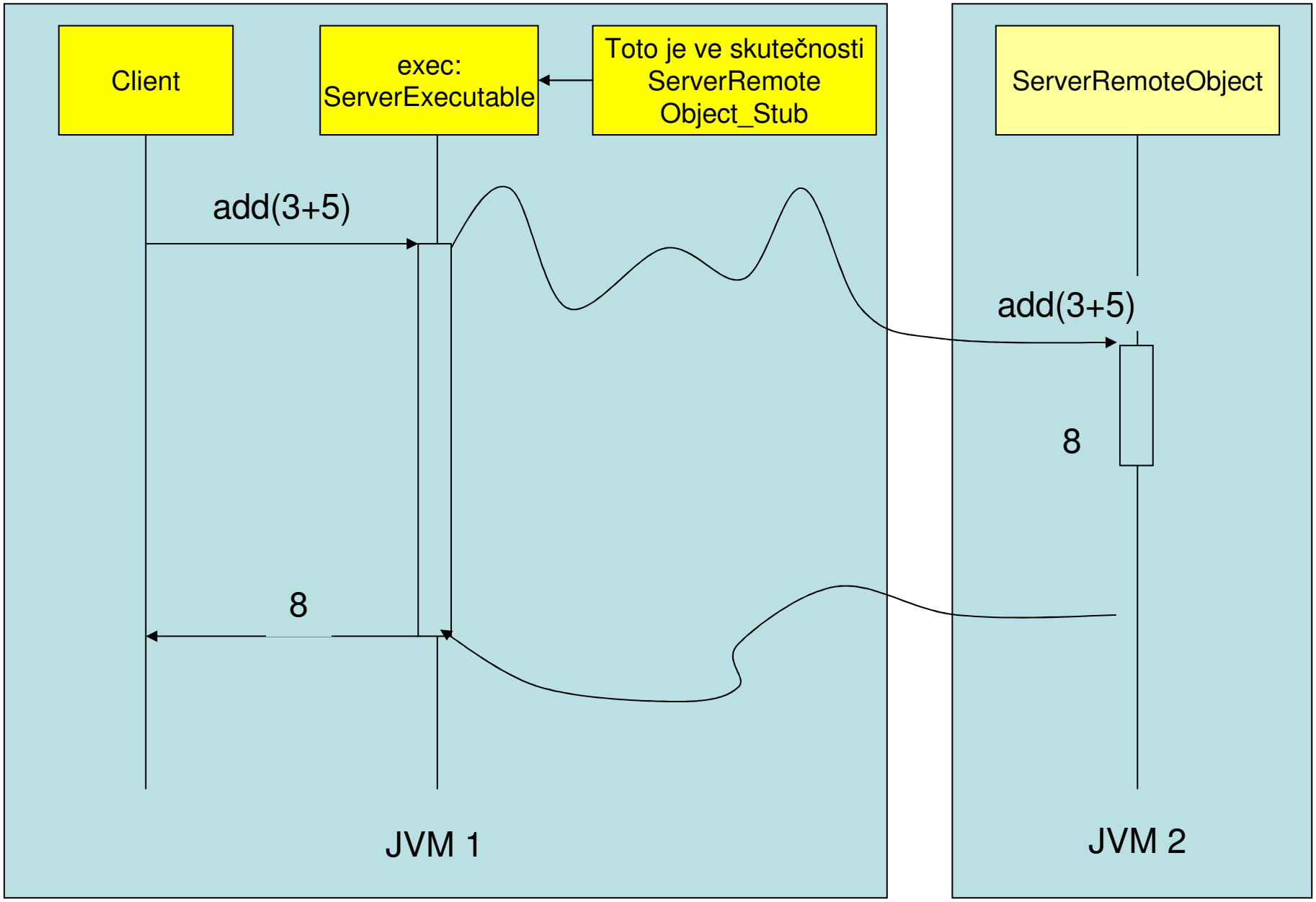
Java RMI – Remote Method Invocation



Java RMI – Remote Method Invocation



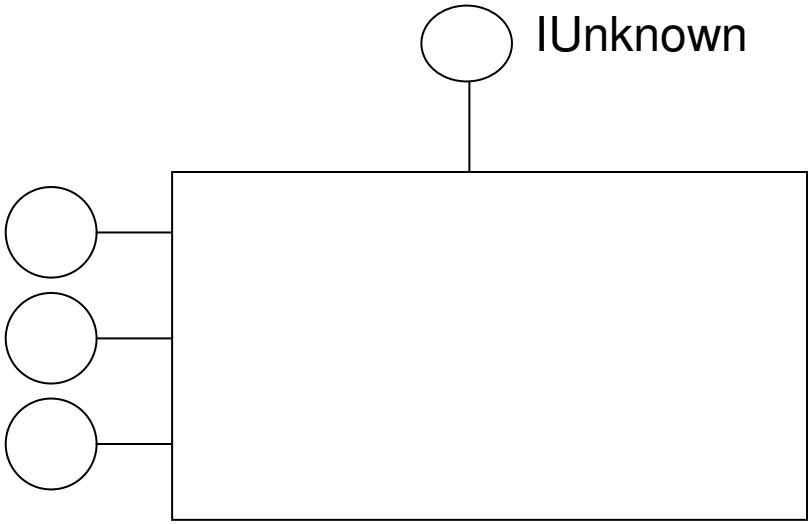
Java RMI – Remote Method Invocation



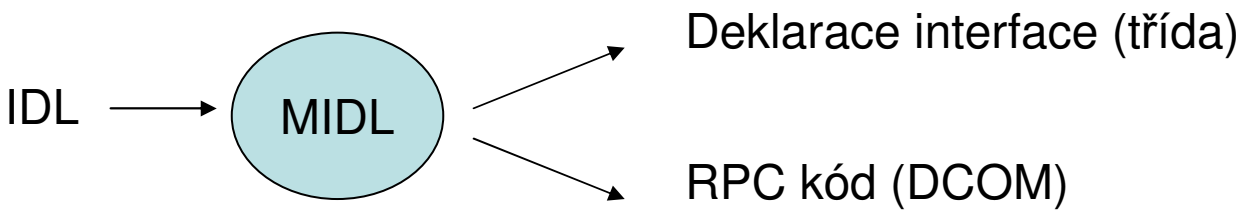
COM

Component Object Model

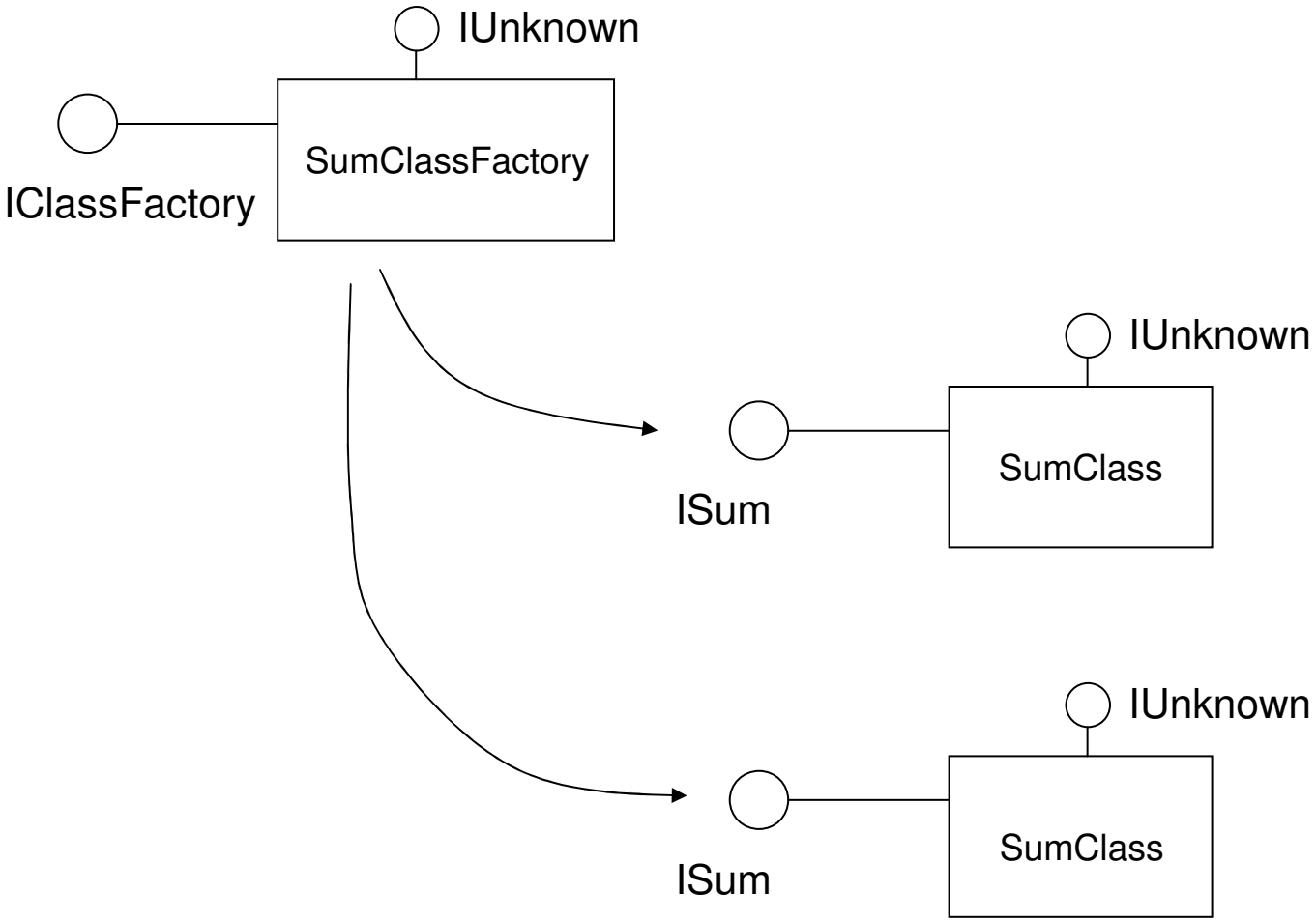
Interface Identifier – např. {A46C12C0-4E88-11CE-A6F1-00AA0037DEFB}
GUID – Global Unique Identifier
GUIDGEN – nástroj generující GUID



COM



COM

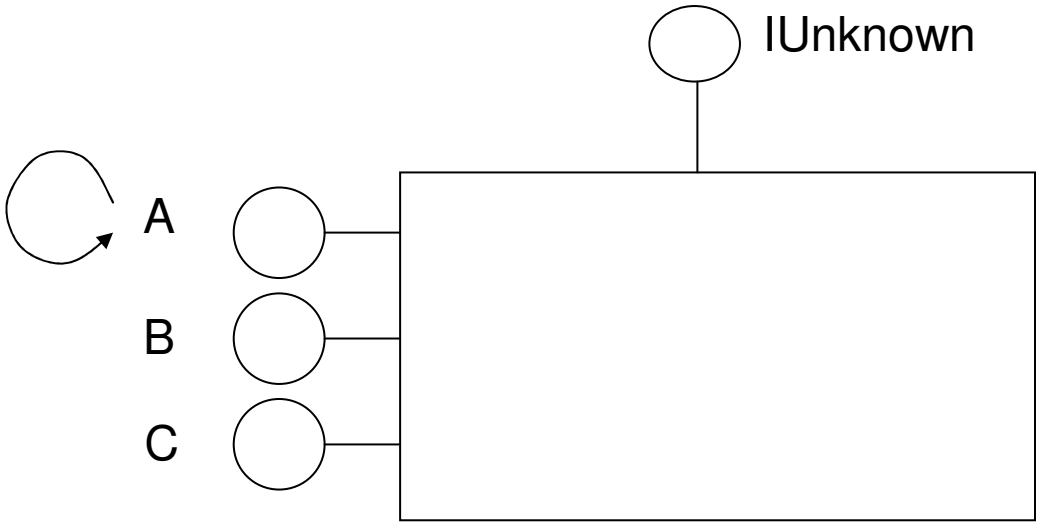

IUnknown
IClassFactory
SumClassFactory
IUnknown
ISum
SumClass
IUnknown
ISum
SumClass

COM

```
Class CFactory: public IClassFactory
{
   public:
        // IUnknown
        … AddRef();
        … Release();
        … QueryInterface(REFIIF, void**);

        // IClassFactory
        …  CreateInstance(IUnknown *, REFIID, void **);
        …  LockServer(BOOL);
        …  CFactory(): m_cRef(1) {g_cLocks++};
        …  ~CFactory() {g_cLocks--};

   private:
        ULONG m_cRef;
}
```

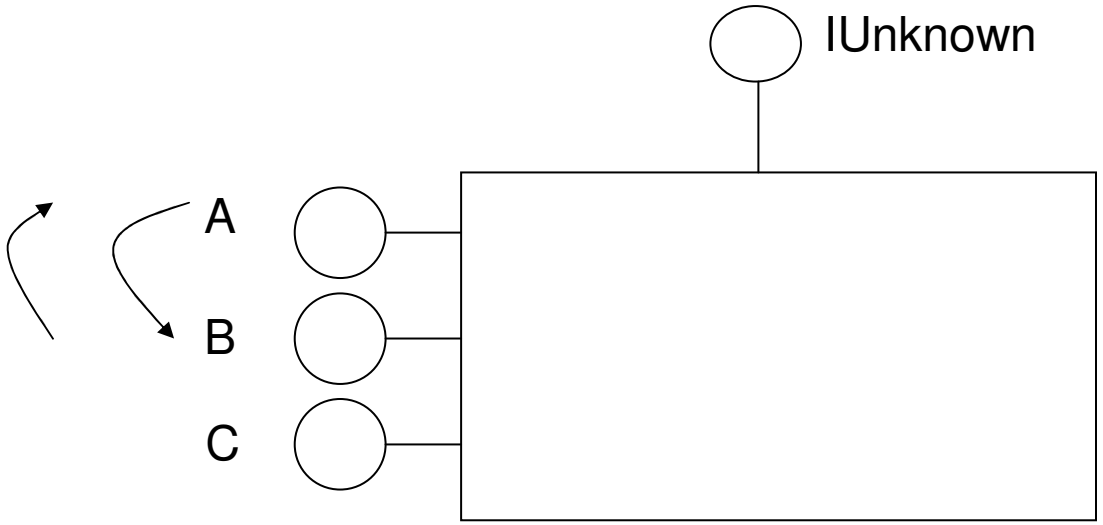
COM

QueryInterface - reflexivnost



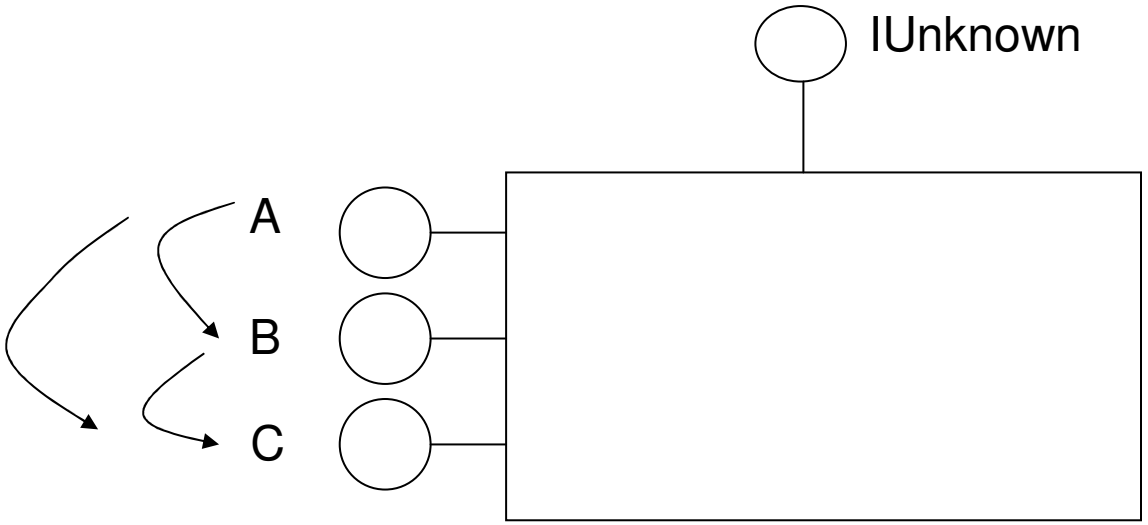
COM

QueryInterface - symetrie



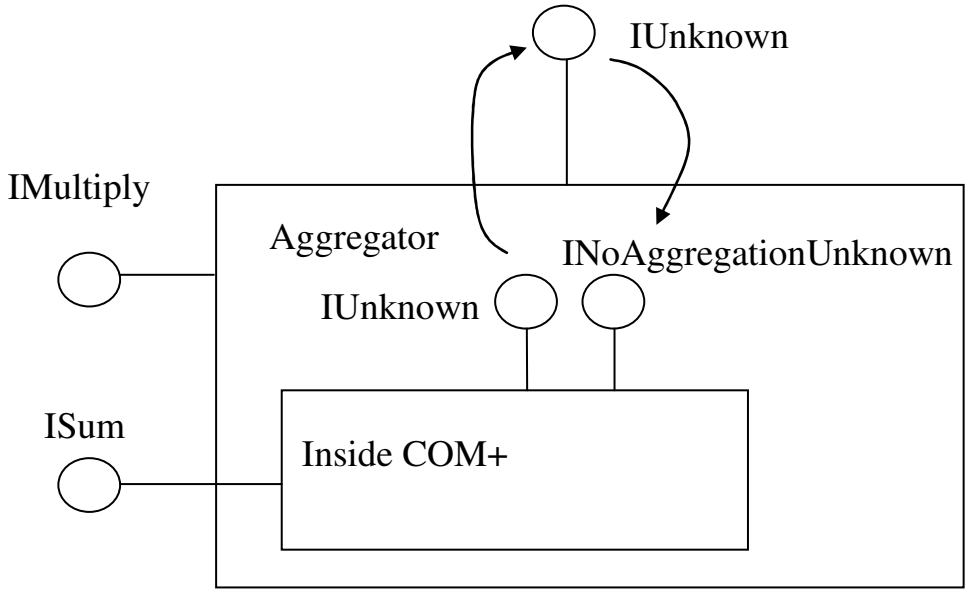
COM

QueryInterface - transitivita



COM – dědění na binární úrovni

COM+ Agregace komponent:



COM – dědění na binární úrovni

COM+ delegace komponent (containment):