Procedural communication

RPC / Remote Procedure Call

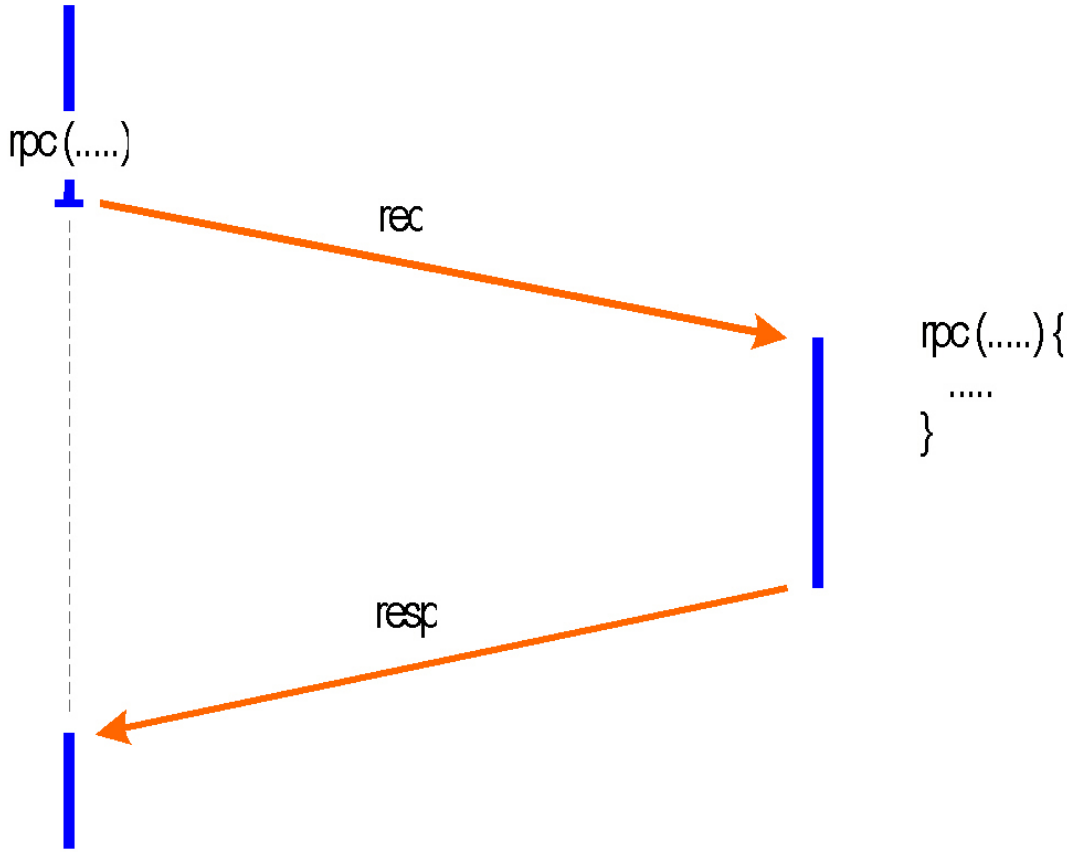
Procedural communication methods

- RPC mechanism
- RPC call semantic
- RPC call support
- transaction (Amoeba)
- rendez-vous

RPC systems

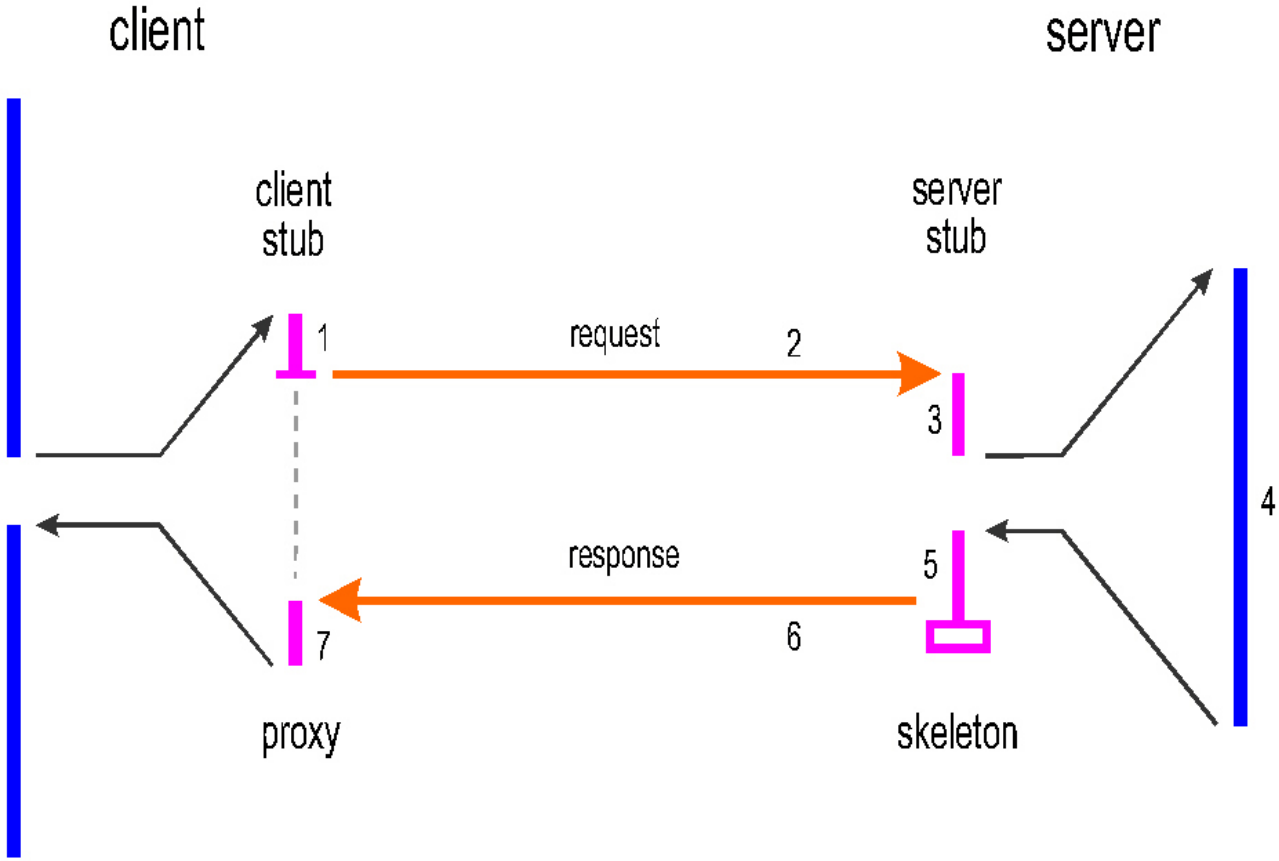
- ONC/OSF RPC
- CORBA
- Java RMI
- . . .
- XML-RPC
- gRPC

Procedural communication



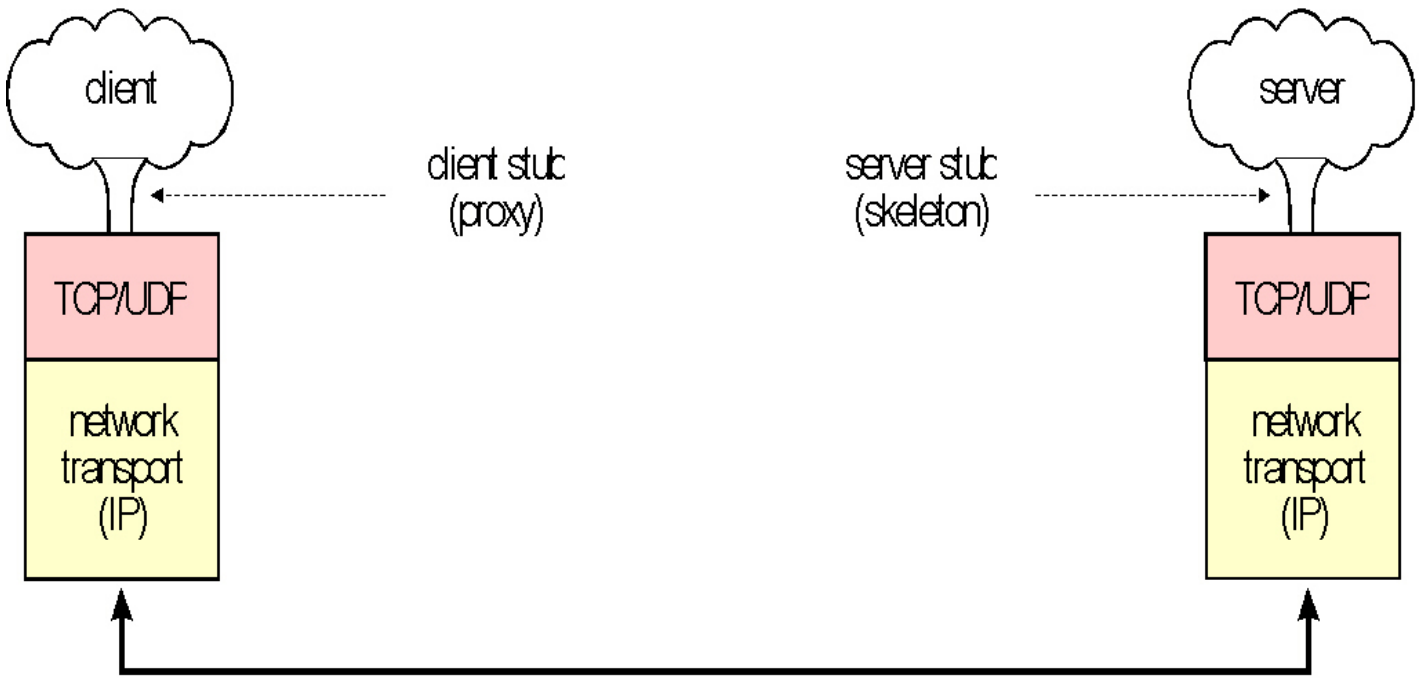
Procedural communication

Stubs of operating systems/communication



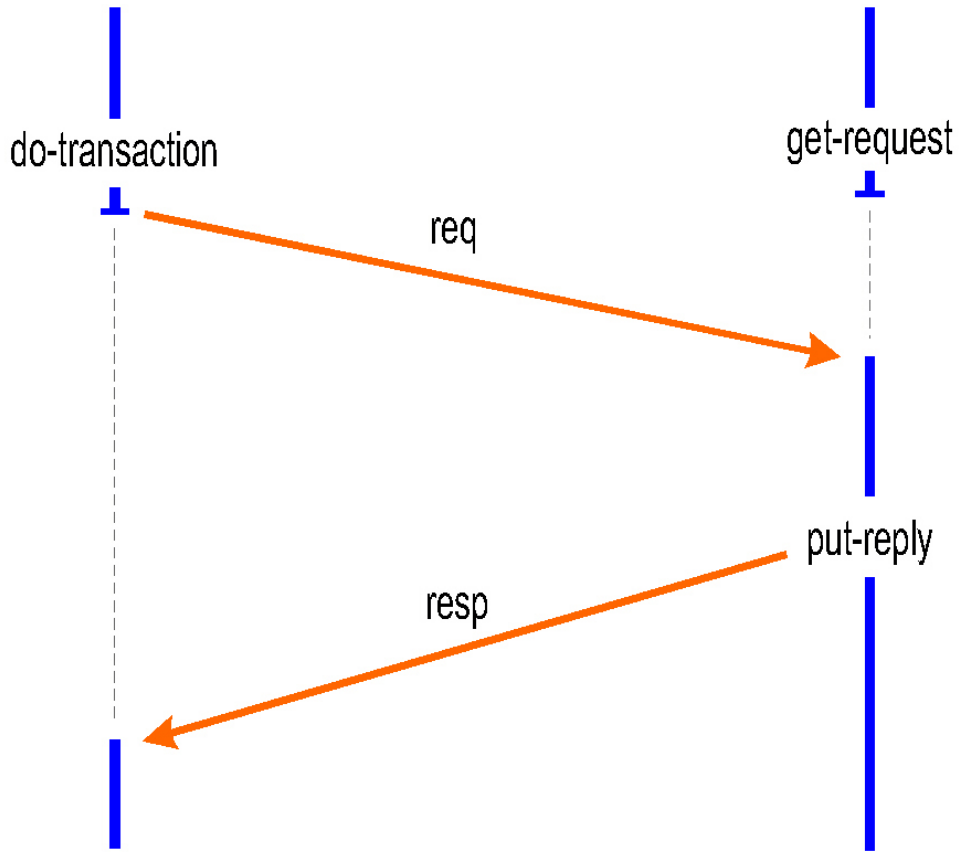
Procedural communication

Applications support by operating systems and communication protocols

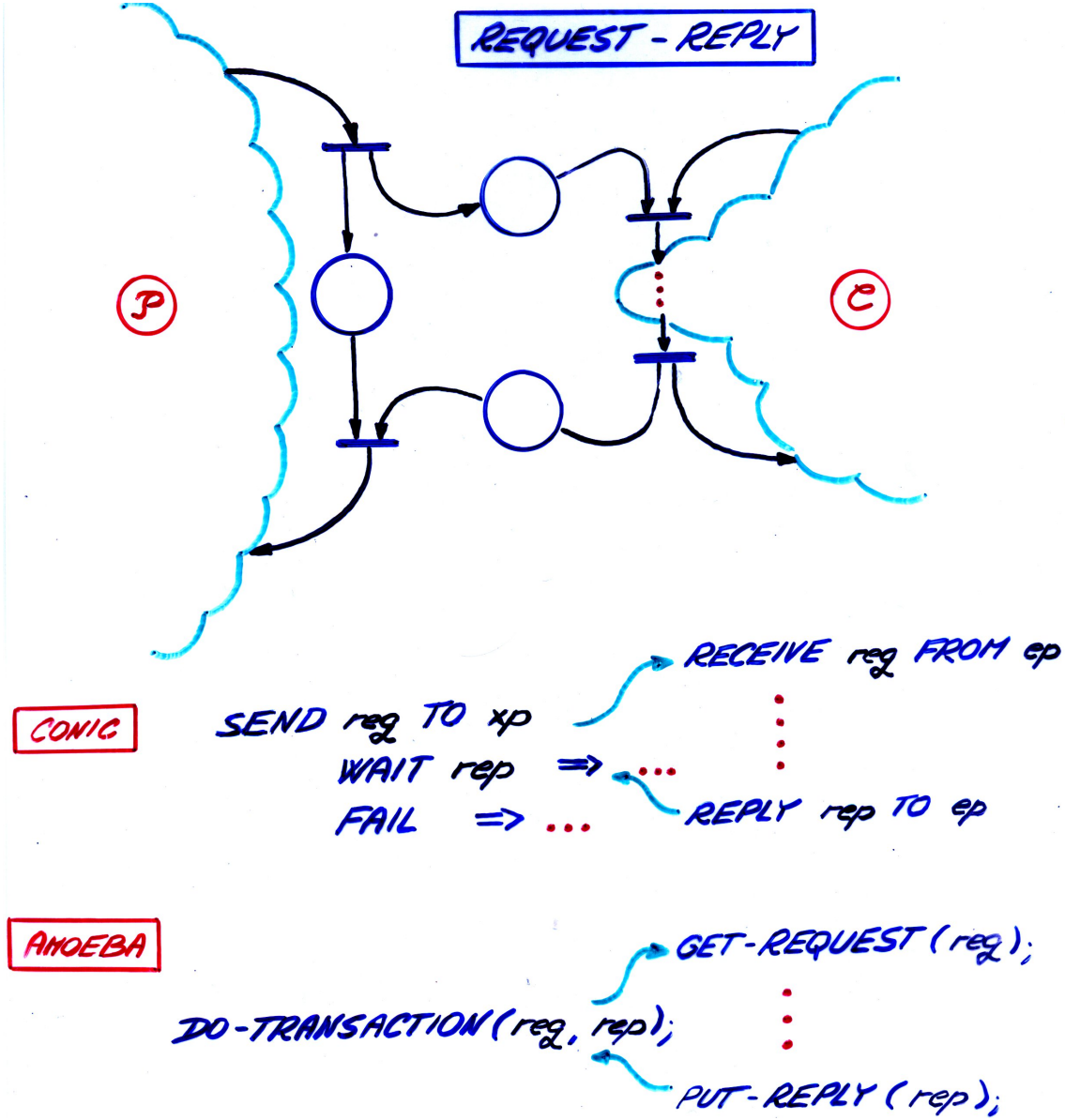


Transactions

Conic, Amoeba

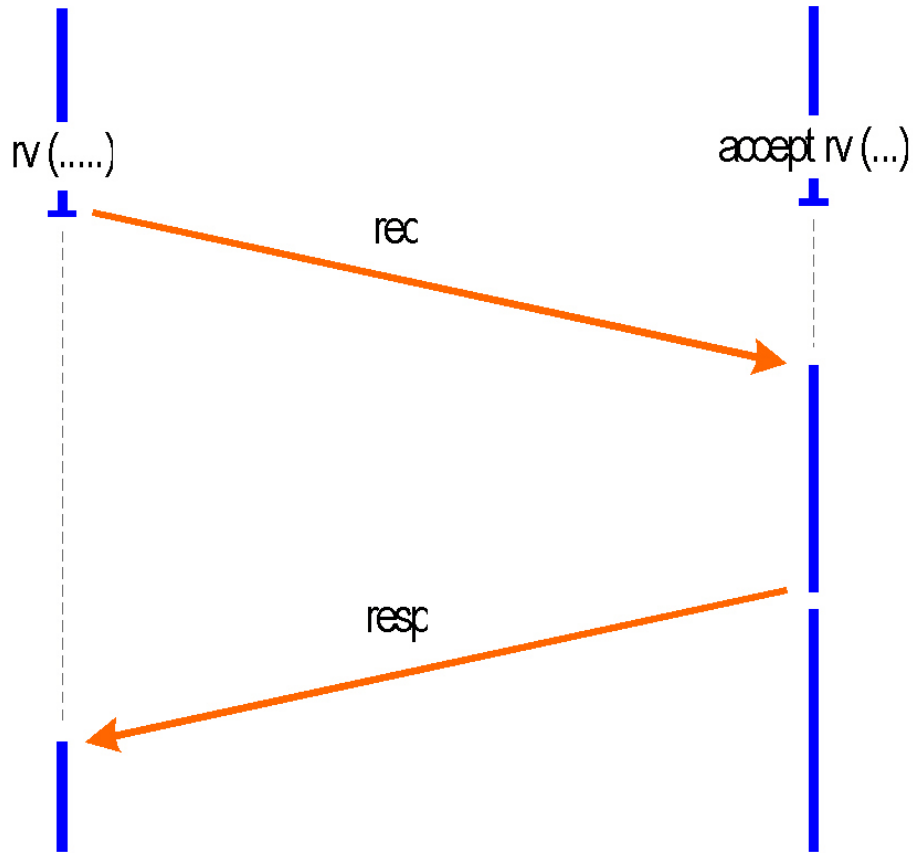


Transactions

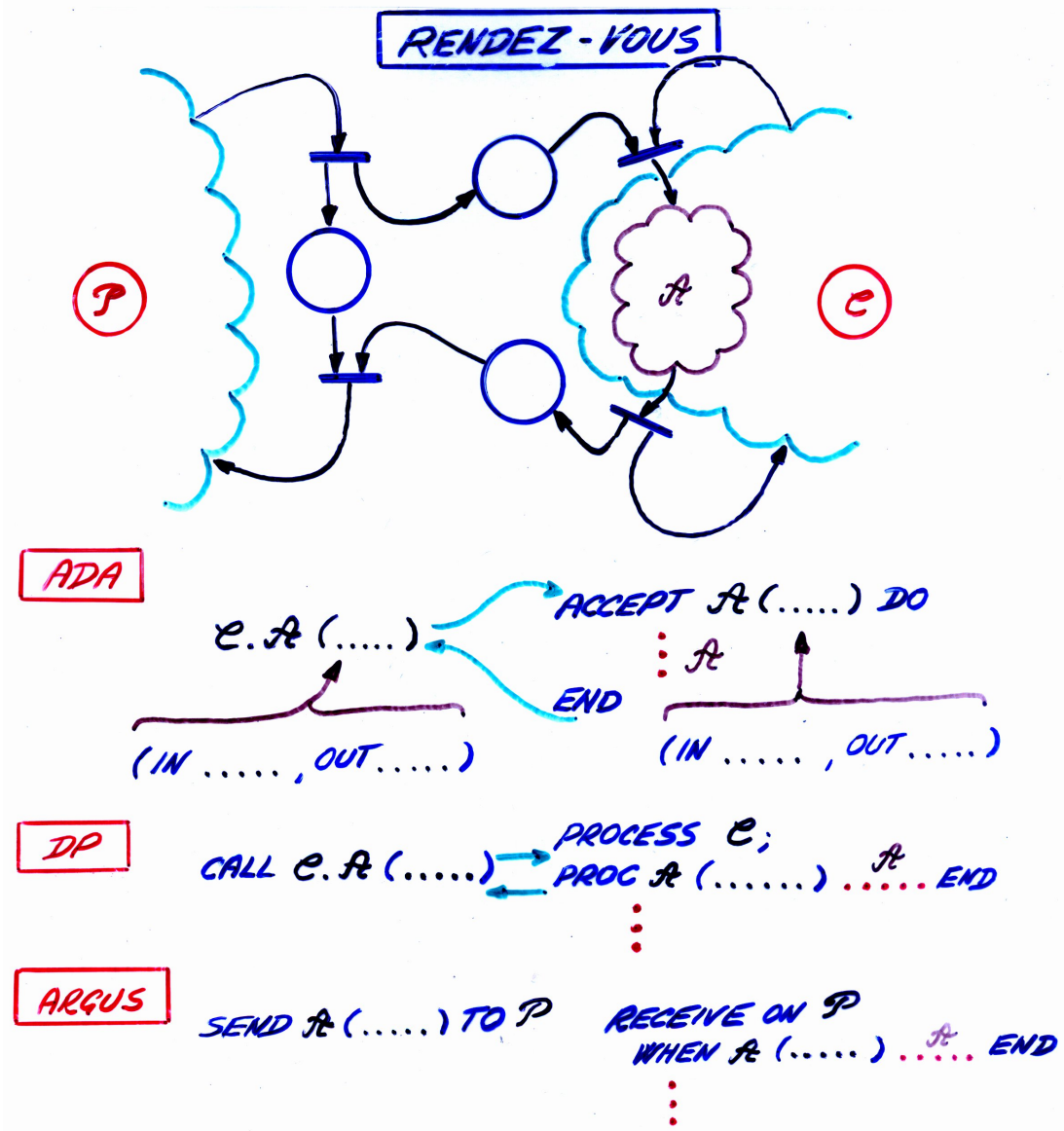


Rendez-vous

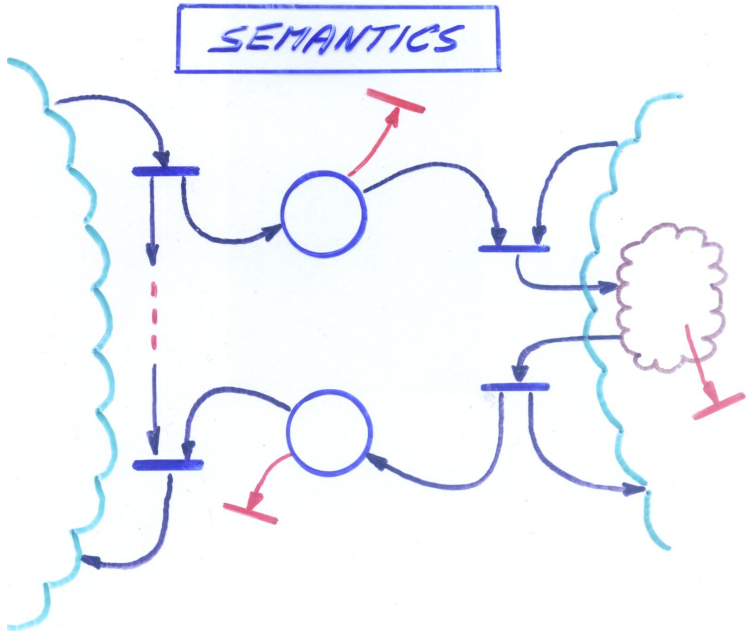
Ada



Rendez-vous

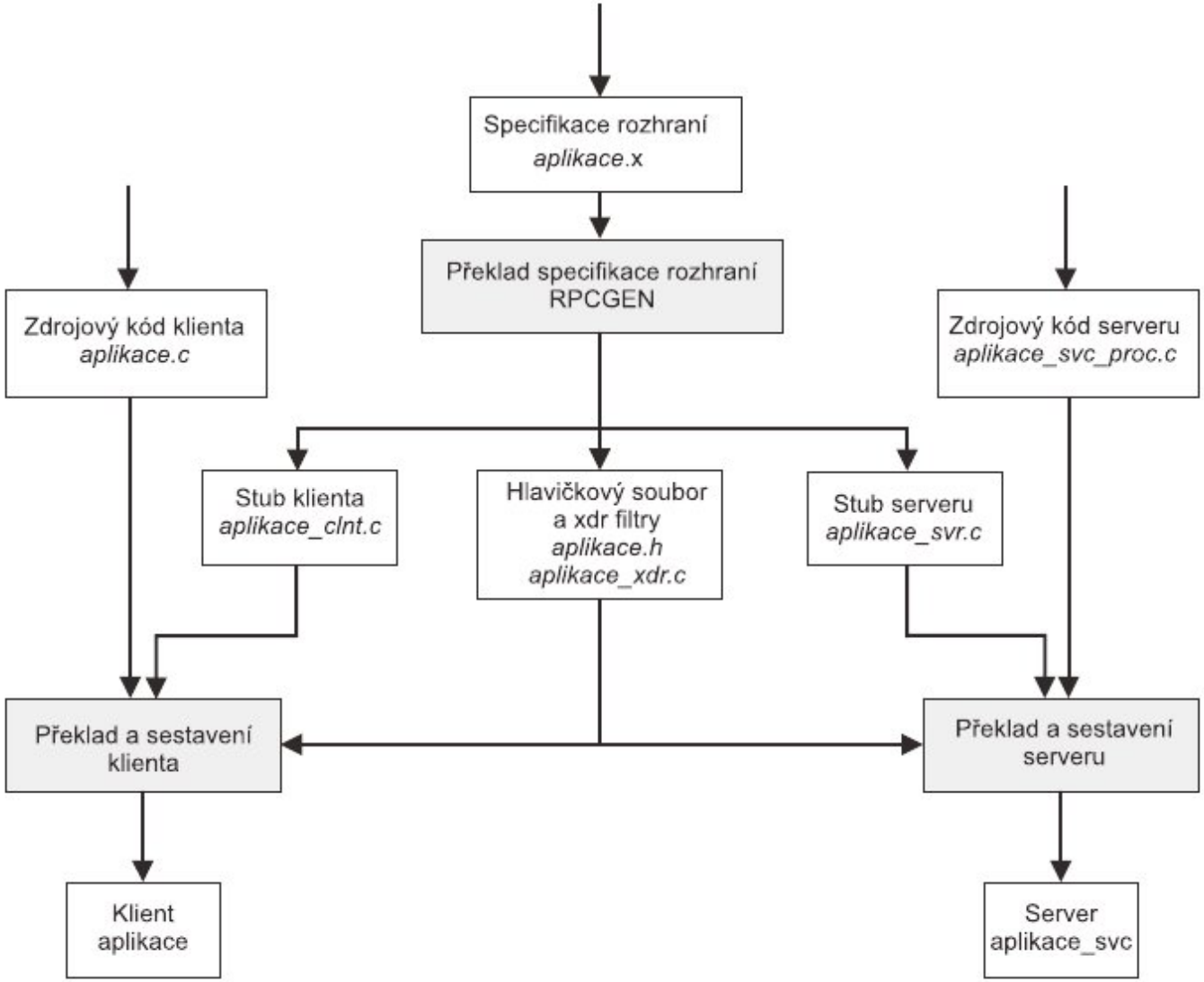

RENDEZ-VOUS
P
A
C
ADA
C.A (.....)
ACCEPT A (.....) DO
A
END
(IN ..... , OUT .....)
(IN ..... , OUT .....)
DP
CALL C.A (.....)
PROCESS C;
PROC A (.....) ..... END
ARGUS
SEND A (.....) TO P
RECEIVE ON P
WHEN A (.....) ..... END

Possible errors and protections against them



Remote Procedure Call - ONC RPC

Open Network Computing



Remote Procedure Call - ONC RPC

- integer (32 bit)
- unsigned integer (32 bit)
- enumeration - enumeration type (32 bit)
- boolean (32 bit, ekvivalent to enum type FALSE=0, TRUE=1)
- hyper integer (64 bit)
- floating-point (32 bit)
- double precision floating-point (64 bit)
- quadruple precision floating-point (128 bit)
- fixed length opaque data - sequence of octets,fixed length
- variable length opaque data - sequence of octets, variable length
- string - řetězec znaků proměnné délky ukončený znakem (NULL)
- fixed length array - fixed length array
- variable length array - variable length array
- structure - strukture/record with fixed length
- discriminated union - variant record
- void - empty type (without value)

Technologies CORBA, Java RMI

CORBA

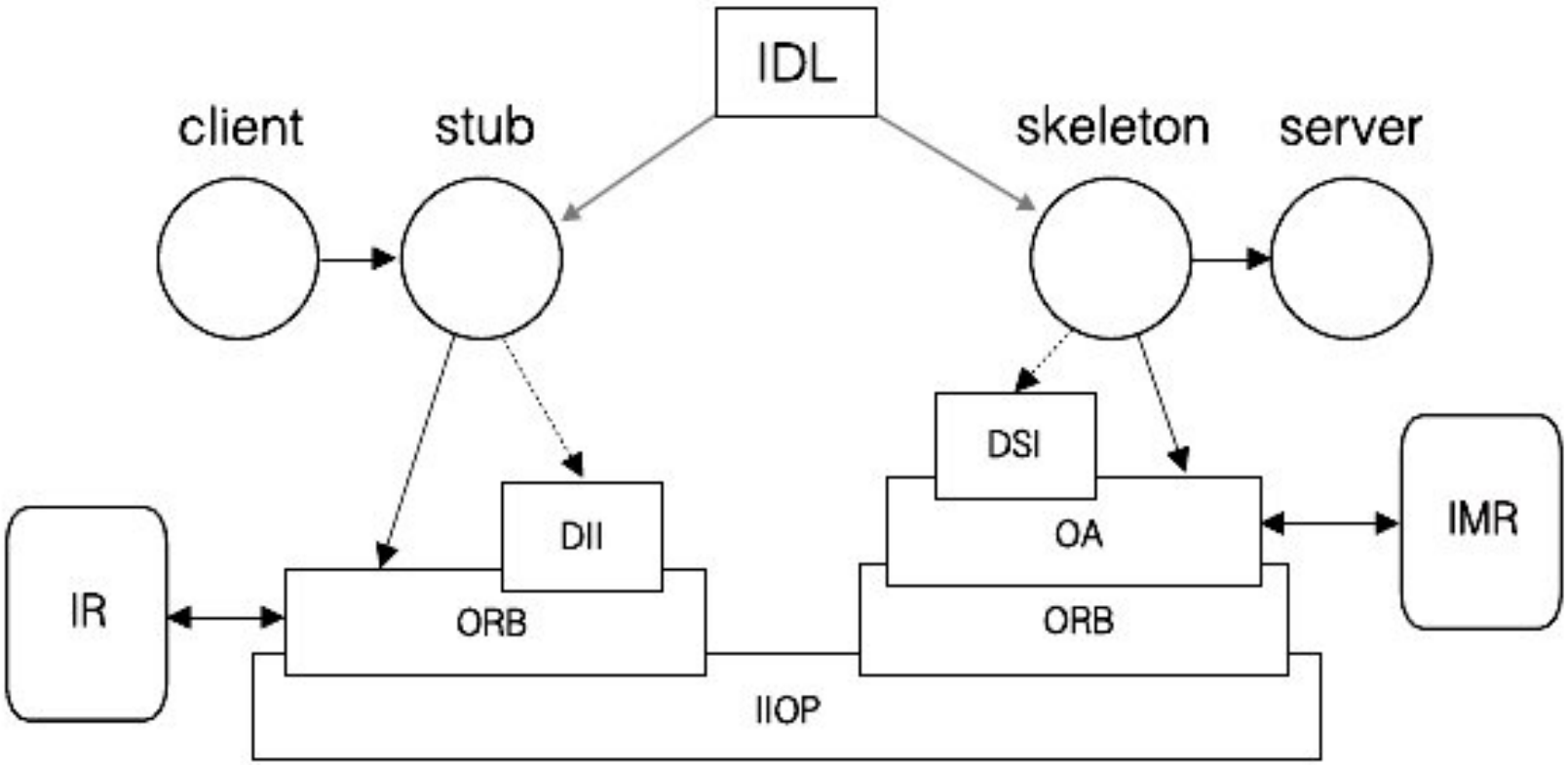
- object oriented aproach
- efficient communication
- dynamic modifications
- multilanguage

Java RMI

- object oriented aproach
- efficient communication
- Java RMI - Remote Method Invocation

CORBA

Common Object Request Broker Architecture



CORBA

IIOP

Internet Inter Object Protocol

OA

vazba komponenty na ORB, objektový adaptér
Basic Object Adapter - BOA
Portable Object Adapter - POA

IR

Interface Repository

IMR

Implementation Repository

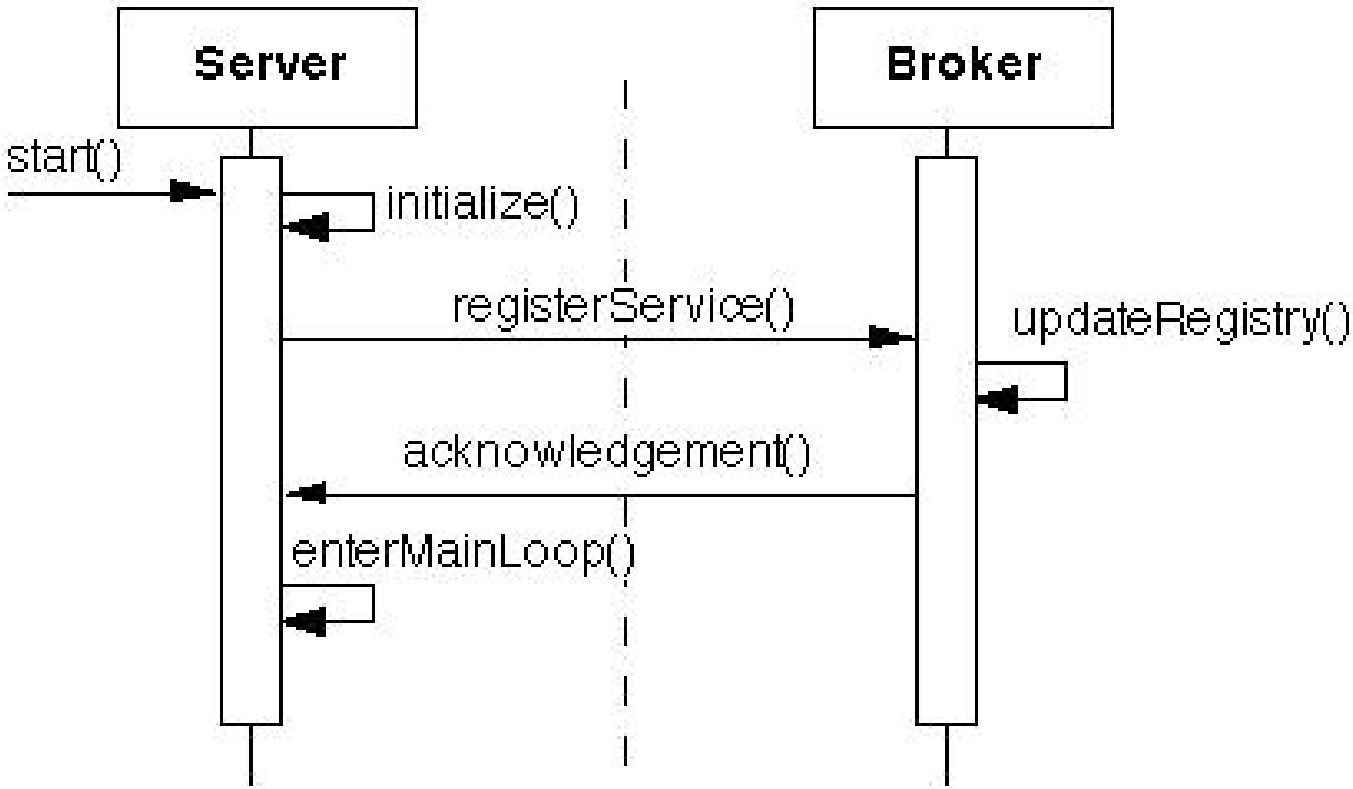
DSI

Dynamic Skeleton Interface

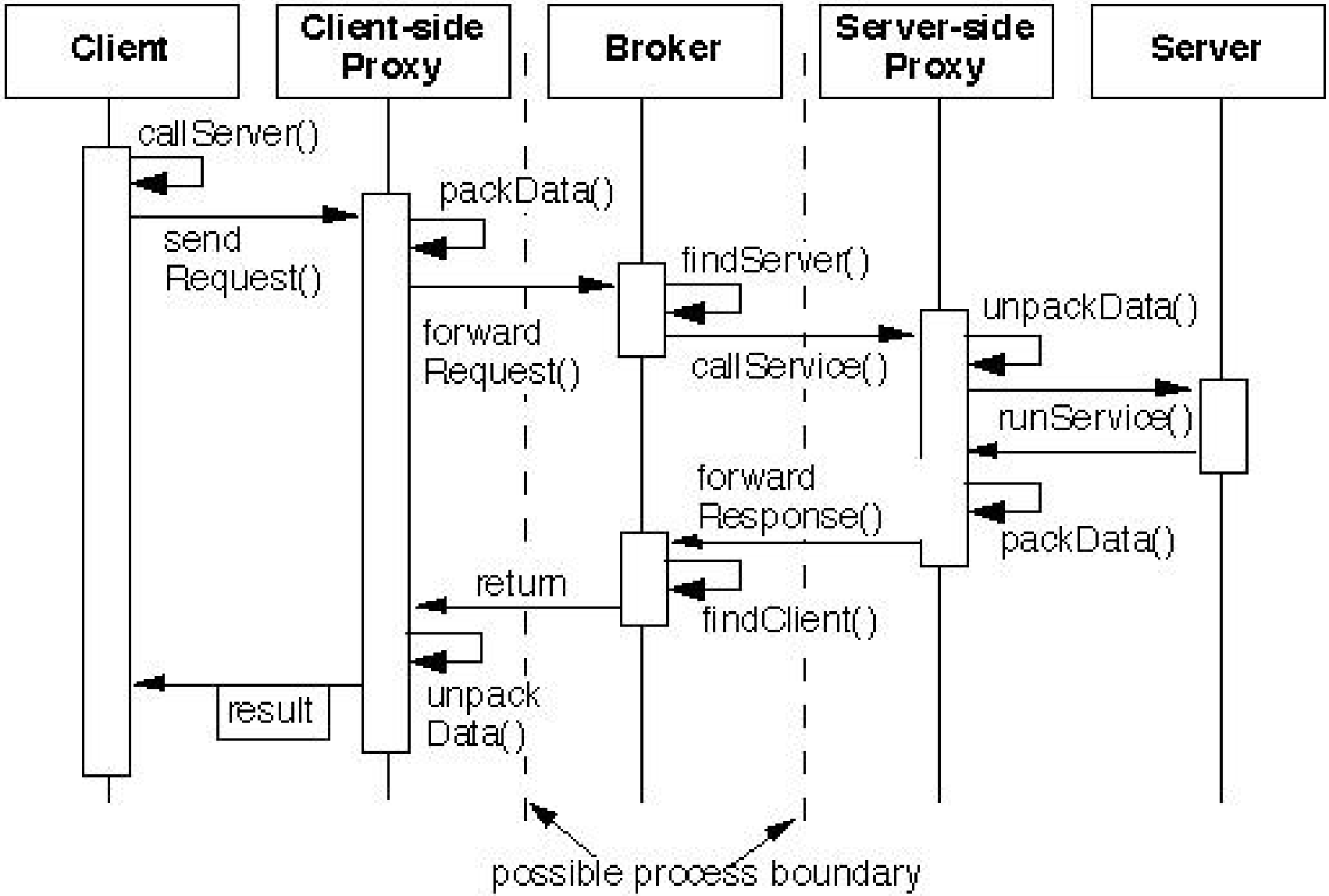
DIR

Dynamic Implementation Routine

CORBA


Server
Broker
start()
initialize()
registerService()
updateRegistry()
acknowledgement()
enterMainLoop()

CORBA


Client
Client-side Proxy
Broker
Server-side Proxy
Server
callServer()
packData()
send Request()
findServer()
unpackData()
forward Request()
callService()
runService()
forward Response()
packData()
return
findClient()
result
unpack Data()
possible process boundary

CORBA

Interface

```
interface BankAccount {
  boolean deposit(in unsigned long amount);
  boolean withdraw(in unsigned long amount);
  void balance(out long amount);
};
```

C

```
typedef CORBA_Object BankAccount;
extern CORBA_boolean BankAccount_deposit(BankAccount o,
        CORBA_unsigned_long amount, CORBA_Environment *ev);
extern CORBA_boolean BankAccount_withdraw(BankAccount o,
        CORBA_unsigned_long amount, CORBA_Environment *ev);
extern void BankAccount_balance(BankAccount o,
        CORBA_long *amount,CORBA_Environment *ev);
```

CORBA

Java Interface

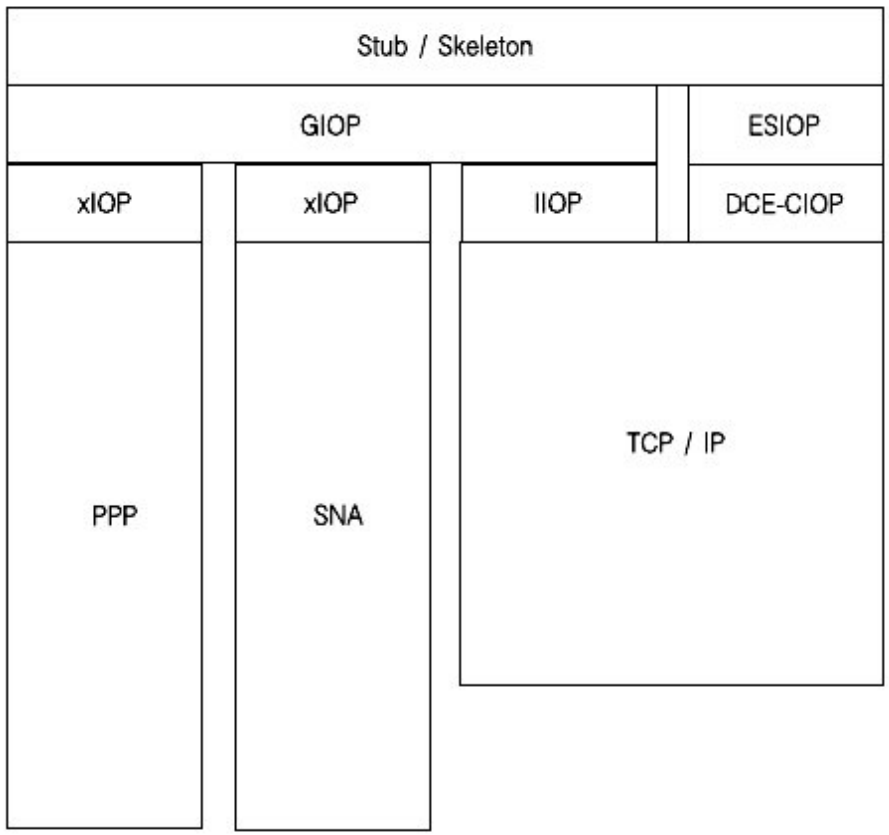
```
public Example {        public interface BankAccount {
  boolean deposit(int amount);
  boolean withdraw(int amount);
  void balance(IntHolder amount);
};
```

<type>Holder

```
final public class IntHolder {
  public Int value;
  public IntHolder() value:=0;
  public IntHolder(Int initial) value:=initial;
};
```

CORBA

Communication support



CORBA

IOP

Inter Object Protocol

GIOP

Generic Inter Object Protocol
Environment Specific Inter Object Protocol

PPP

Point-to-Point Protocol (ISO, HDLC)

SNA

System Network Architecture (IBM)

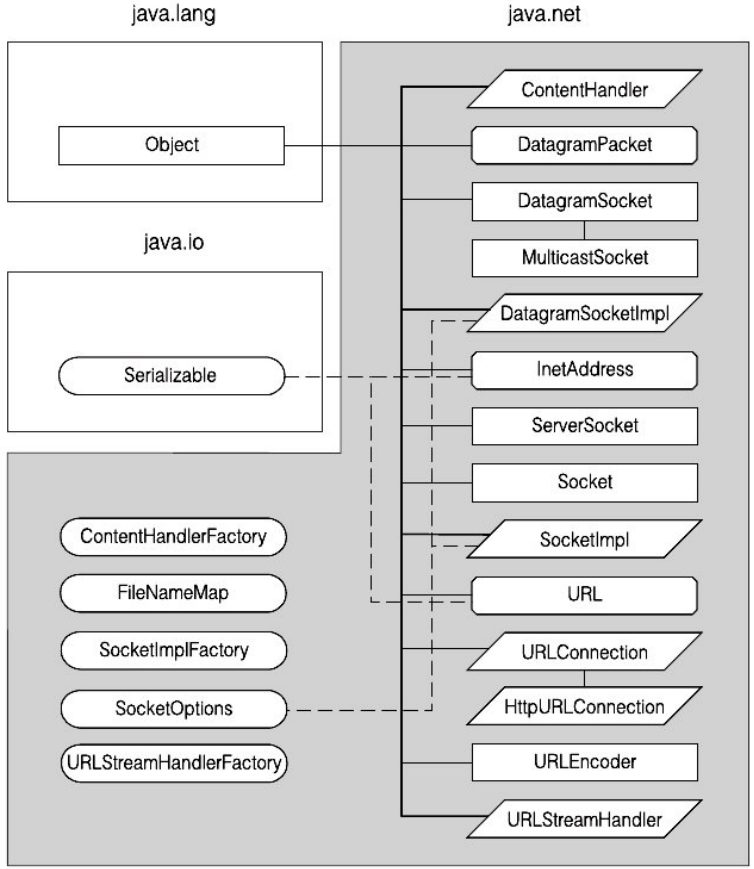
IIOP

Internet Inter Object Protocol

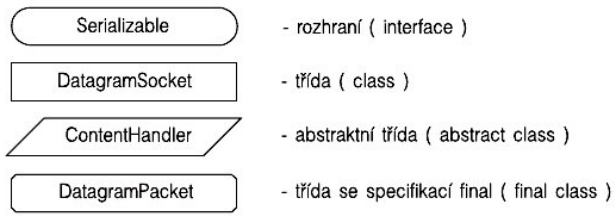
DCECIOP

DCE - Common Interoperability Protocol

Java - communication



Legenda :



Java - communication

UDP

- internet UDP
  - unicast i multicast

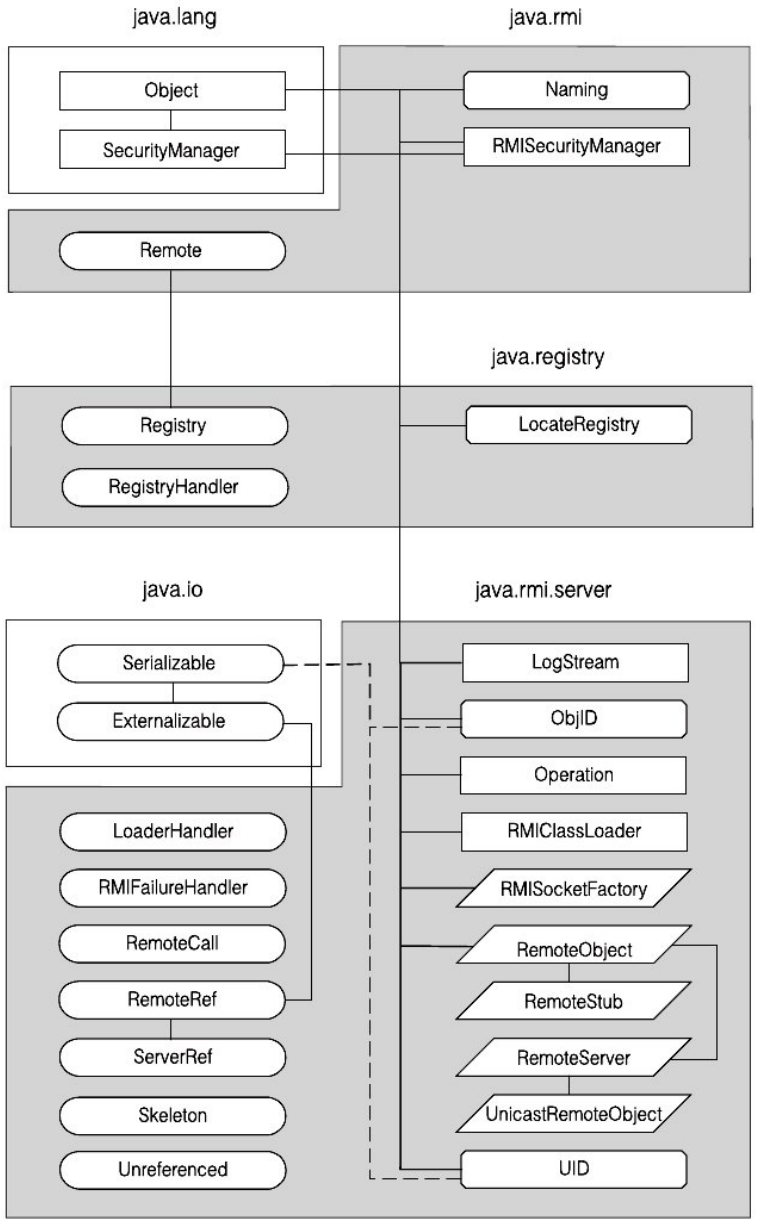
TCP

- internet TCP
  - unicast only

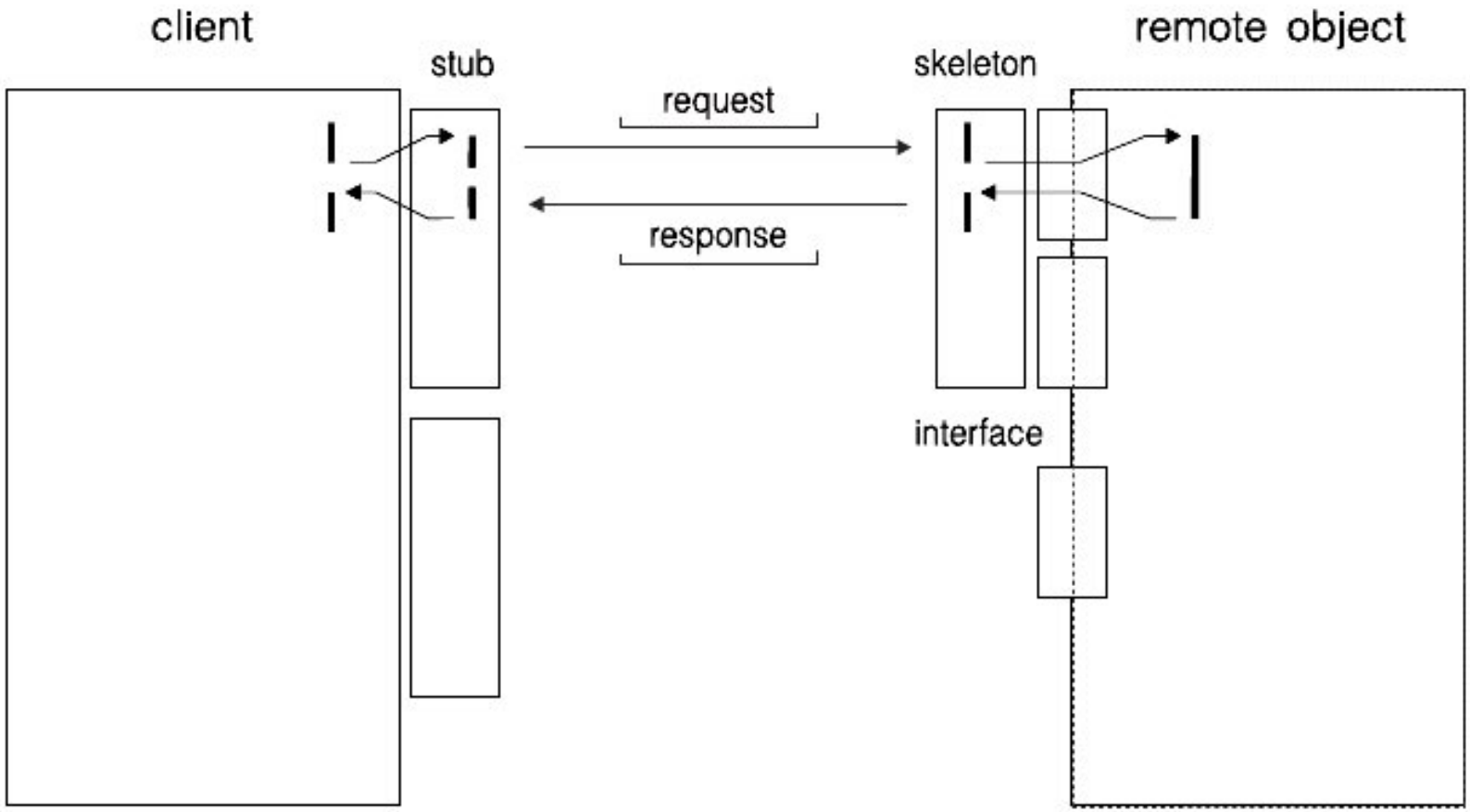
URL

- Universal Resource Location

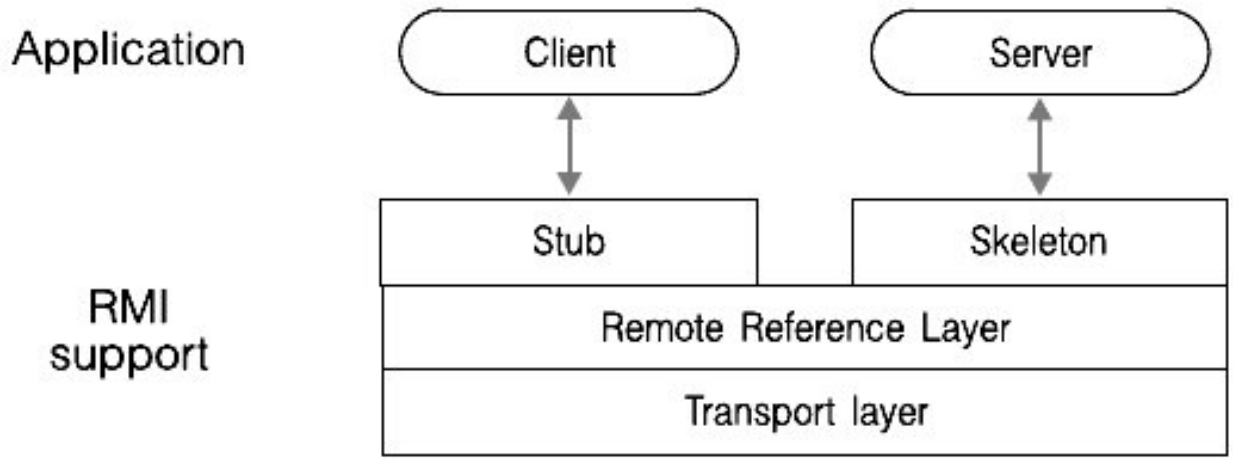
Java - RMI


java.lang
java.rmi
Object
SecurityManager
Naming
RMISecurityManager
Remote
java.registry
Registry
RegistryHandler
LocateRegistry
java.io
java.rmi.server
Serializable
Externalizable
LoaderHandler
RMIFailureHandler
RemoteCall
RemoteRef
ServerRef
Skeleton
Unreferenced
LogStream
ObjID
Operation
RMIClassLoader
RMISocketFactory
RemoteObject
RemoteStub
RemoteServer
UnicastRemoteObject
UID

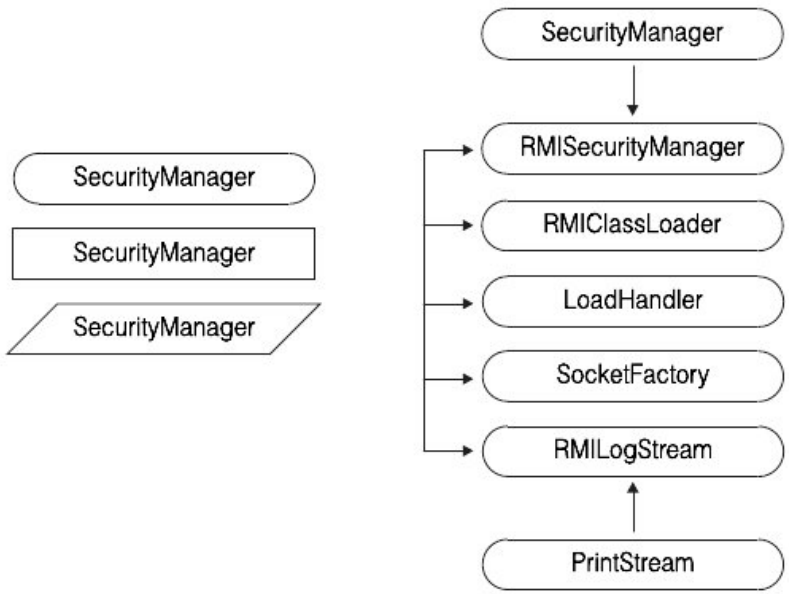
Java - RMI


client
stub
request
skeleton
remote object
response
interface

Java - RMI

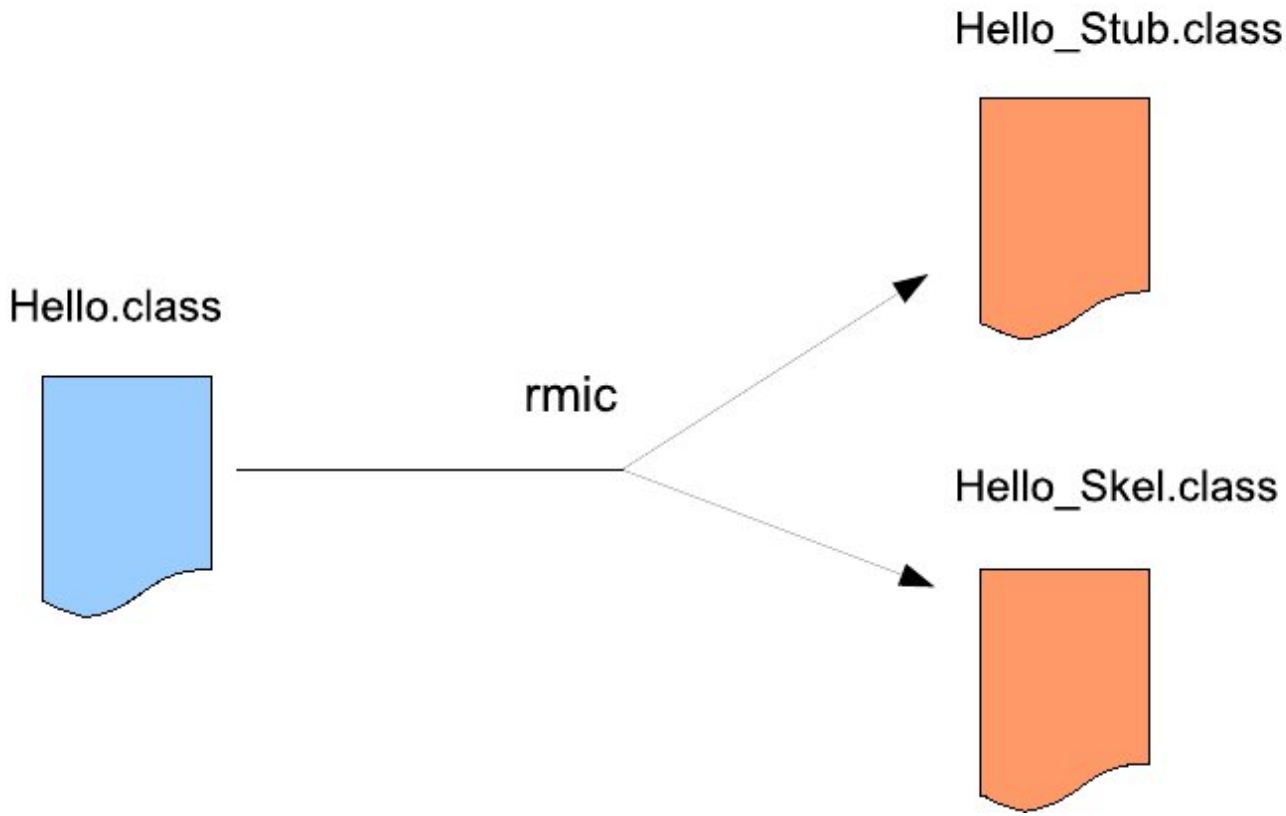

Application
Client
Server
Stub
Skeleton
RMI
support
Remote Reference Layer
Transport layer

Java - RMI


SecurityManager
SecurityManager
SecurityManager
SecurityManager
RMISecurityManager
RMIClassLoader
LoadHandler
SocketFactory
RMILogStream
PrintStream

Java RMI - stubs generation

stub generation



Java RMI - interface

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

Java RMI - server

```
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+"!";
  }
```

Java RMI - server

```
  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);
    }
  }
}
```

Java RMI - client

```
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);
  }
}
```

Java RMI - applet

```
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);
  }
```

Java RMI - HTML page for applet

```
<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>
```

XML-RPC, SOAP/RESTful technologies

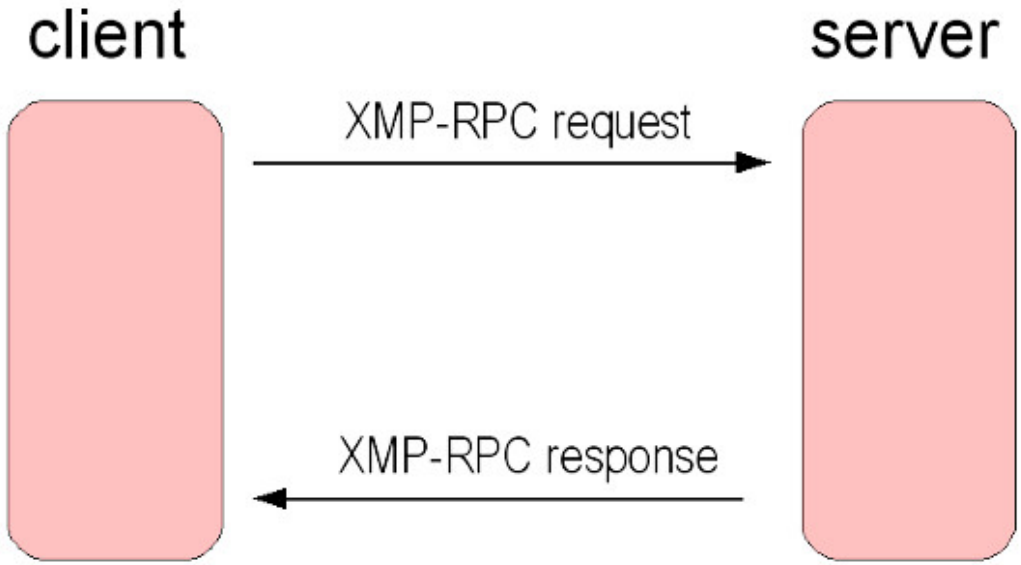
XML-RPC

- procedural communication
- for education mainly
- XML format
- simple server implementation

SOAP, RESTful

- object oriented
- SOAP - primarily RPC
- RESTful - primarily access to data sources (clouds)

XML-RPC



- simple definition

  www.xml.com

- implementation based on XML parser

XML-RPC simple data types

boolean

```
<boolean> 1 </boolean>
```

int

```
<i4> 42 </i4>          <int> 42 </int>
```

double

```
<double> -12.53 </double>
```

string

```
<string> Hello world! </string>
```

dateTime.iso8601

```
<dateTime.iso8601>
    19980717T14:08:55
</dateTime.iso8601>
```

XML-RPC structure data types

array

```
<array>
    <data>
        <value><i4>1404</i4></value>
        <value>
            <string>Something here</string>
        </value>
        <value><i4>1</i4></value>
    </data>
</array>
```

struct

```
<struct>
    <member>
        <name>foo</name>
        <value><i4>1</i4></value>
    </member>
    . . .
    </struct>
```

XML-RPC additional data types

base64

```
<base64>
    eW91IGNhbid0IHJlYWQgdGhpcyE=
</base64>
```

nil

```
<nil/>
```

XML-RPC request

```
POST /RPC2 HTTP/1.0
User-Agent: . . .
Host: . . .
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
        <param>
            <value><i4>40</i4></value>
        </param>
    </params>
</methodCall>
```

XML-RPC response

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 2005 19:55:08 GMT
Server: . . .

<?xml version="1.0"?>
<methodResponse>
    <params>
        <param>
            <value><string>South Dakota</string></value>
        </param>
    </params>
</methodResponse>
```

XML-RPC fault

```
<?xml version="1.0"?>
<methodResponse>
    <fault>
        <value>
            <struct>
                <member>
                    <name>faultCode</name>
                    <value><int>4</int></value>
                </member>
                <member>
                <name>faultString</name>
                <value>
                    <string>Too many parameters.</string>
                </value>
                </member>
            </struct>
        </value>
    </fault>
</methodResponse>
```

XML-RPC

simple implementation

- even without a web server,
- standalone program
- generally based on obvious (slow) XML parser

implementations in more languages

Perl

Python

C / C++

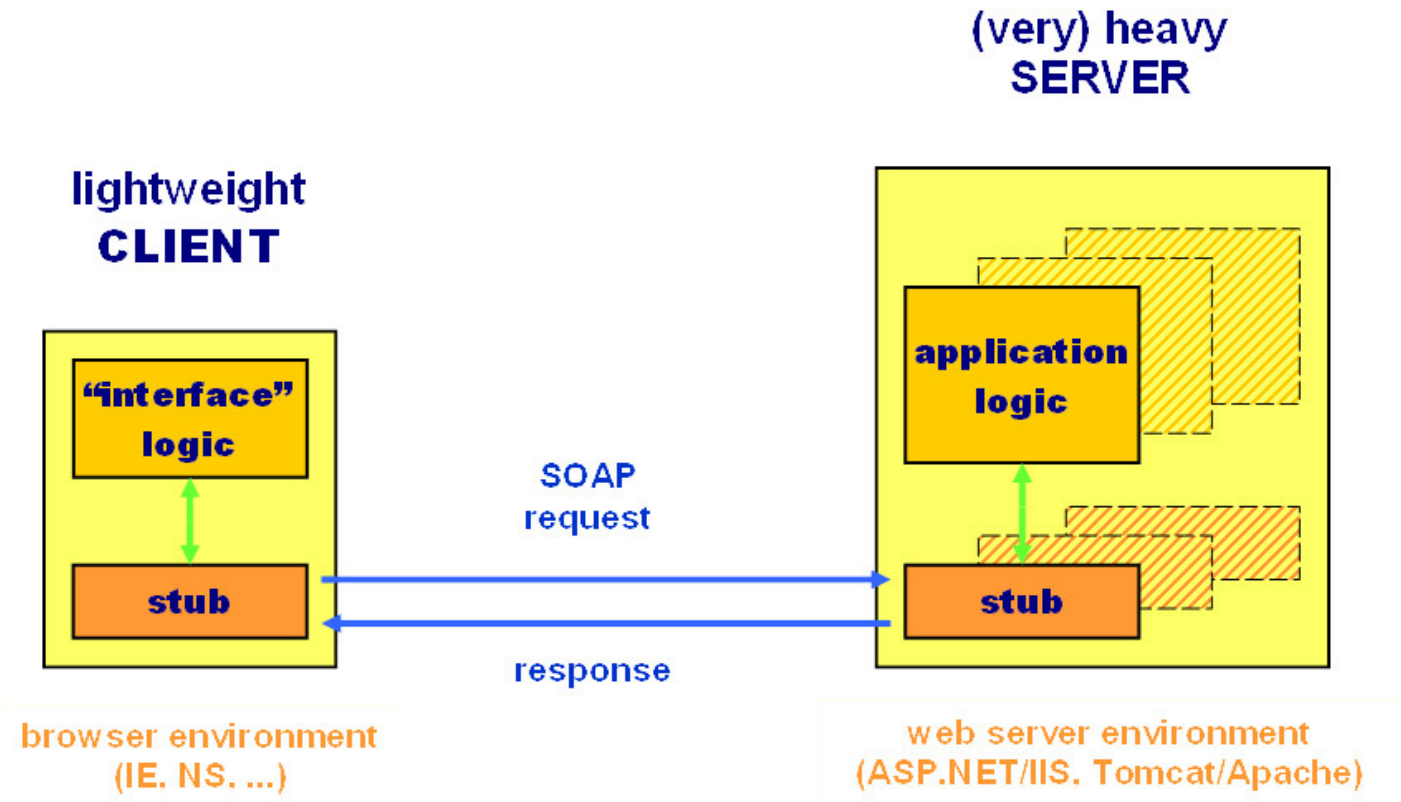
Java

Ruby

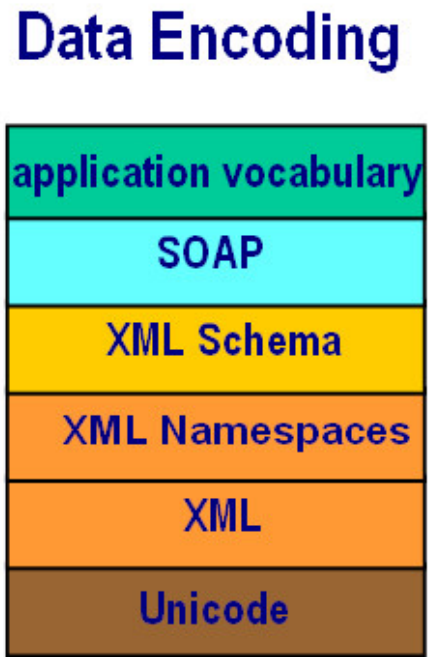
languages can be combined

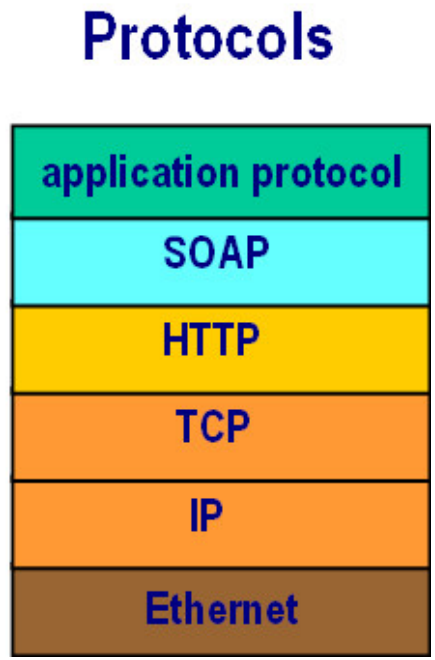
SOAP

Simple Object Access Protocol



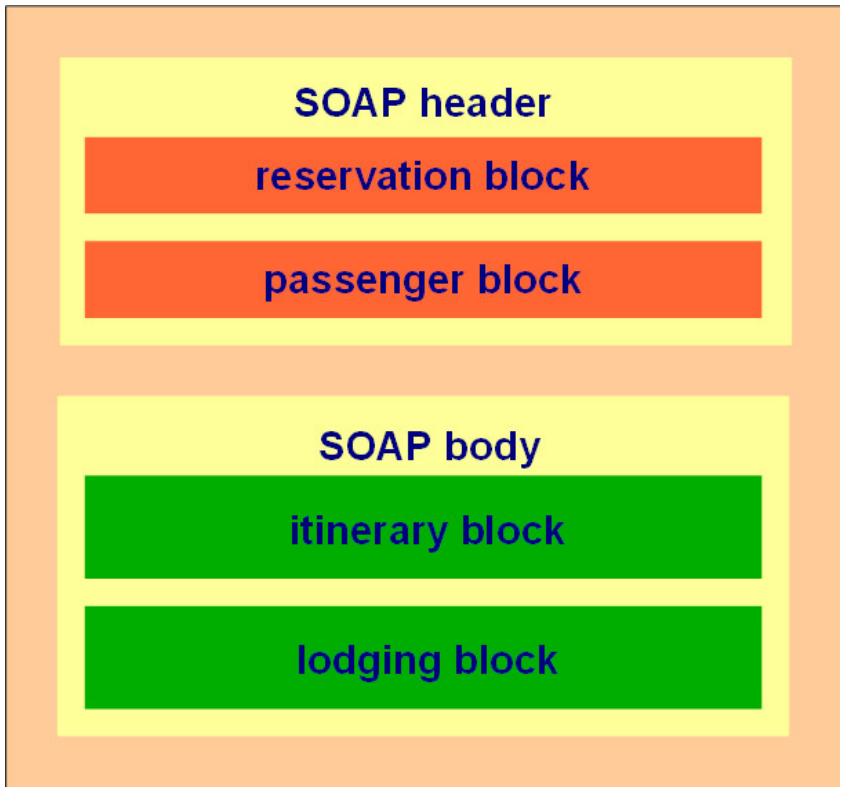
SOAP


Data Encoding
application vocabulary
SOAP
XML Schema
XML Namespaces
XML
Unicode


Protocols
application protocol
SOAP
HTTP
TCP
IP
Ethernet

SOAP

Simple Object Access Protocol



SOAP - HTTP GET request

GET /travelcompany.example.org/reservations?code=FT35ZBQ HTTP/1.1
Host: travelcompany.example.org
Accept: text/html;q=0.5, application/soap+xml

SOAP - HTTP response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope»
<env:Header>
. . . . .
</env:Header>
<env:Body>
. . . . .
</env:Body>
</env:Envelope>
```

SOAP - HTTP POST request

```
POST /Reservations HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope»
<env:Header>
. . . . .
</env:Header>
<env:Body>
. . . . .
</env:Body>
</env:Envelope>
```

SOAP - SMTP request

From: a.oyvind@mycompany.example.com
To: reservations@travelcompany.example.org
Subject: Travel to LA
Date: Thu, 29 Nov 2001 13:20:00 EST
Message-Id: <EE492E16A090090276D2084@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope»
<env:Header>
. . . . .
</env:Header>
<env:Body>
. . . . .
</env:Body>
</env:Envelope>

SOAP - SMTP response

From: reservations@travelcompany.example.org
To: a.oyvind@mycompany.example.com
Subject: Which NY airport?
Date: Thu, 29 Nov 2001 13:35:11 EST
Message-Id: <200109251753.NAA10655@travelcompany.example.org>
In-reply-to: <EE492E16A090090276D2084@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope»
<env:Header>
. . . . .
</env:Header>
<env:Body>
. . . . .
</env:Body>
</env:Envelope>

SOAP - RPC

RPC request

```
<?xml version="1.0"encoding="UTF-8"standalone="no"?>
  <SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema»
    <SOAP-ENV:Body>
      <ns1:doubleAnInteger xmlns:ns1="urn:MySoapServices»
        <param1 xsi:type="xsd:int»123</param1>
      </ns1:doubleAnInteger>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

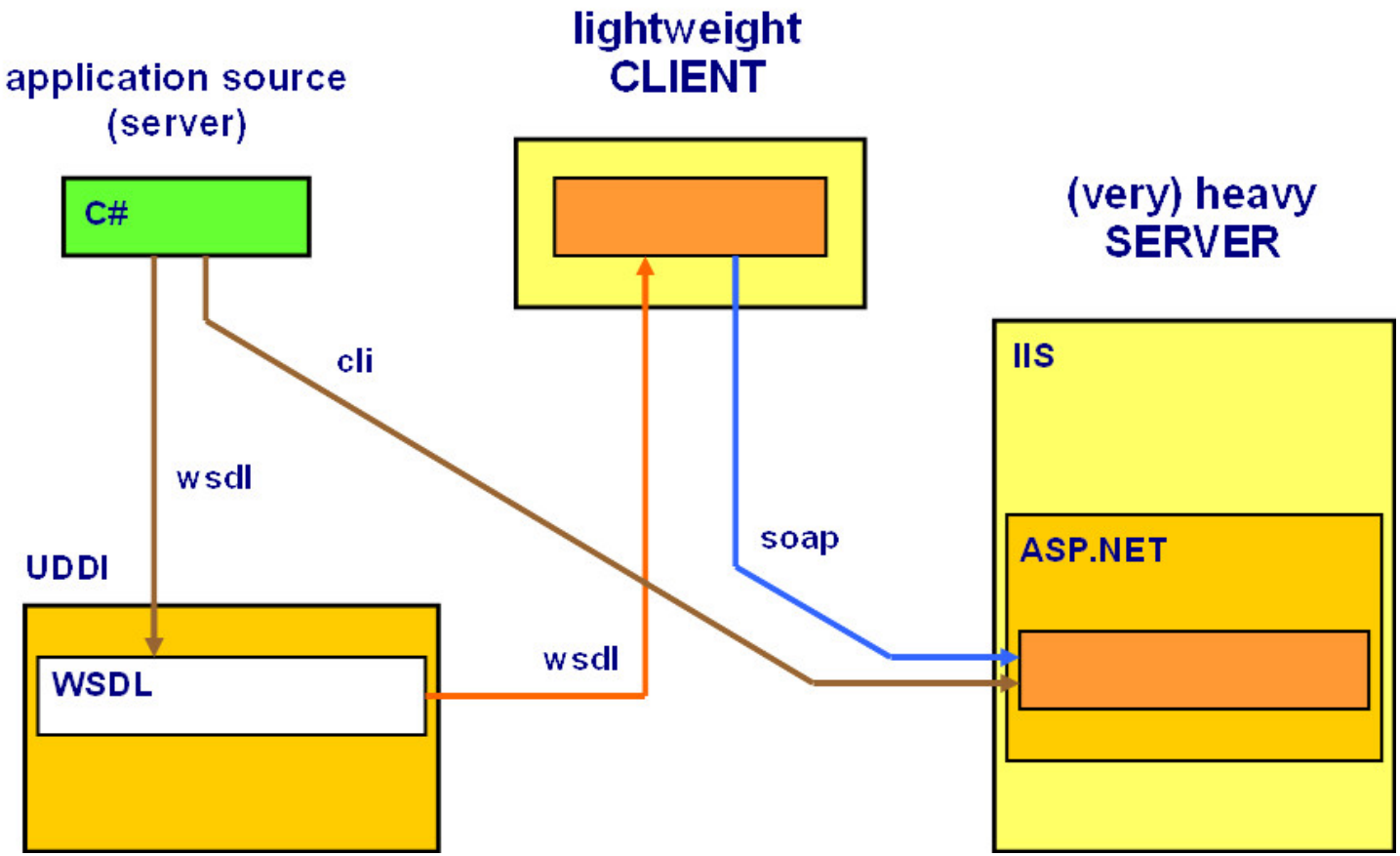
SOAP - RPC

RPC response

```
<?xml version="1.0"encoding="UTF-8"?>
  <SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema»
    <SOAP-ENV:Body>
      <ns1:doubleAnIntegerResponse
      xmlns:ns1="urn:MySoapServices"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/»
        <return xsi:type="xsd:int»246</return>
      </ns1:doubleAnIntegerResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

SOAP

Simple Object Access Protocol



WSDL description generation

server interface C -> WSDL

helloWorld.cpp

```
[WebService]

public string helloWorld()
{
   return (“Hello world!“);
}
```

WSDL generator

```
wsdl helloWorld.cpp
```

WSDL description generation

server interface Java -> WSDL

helloWorld.java

```
String[] helloWorld()
{
  return (“Hello world!“);
}
```

WSDL generator

```
java2wsdl helloWorld.java
```

WSDL

Web Service Description Language

```
<?xml version="1.0" encoding="utf-8">
<definitions>
  <types> describes SOAP parameters' types
      . . . . .
  </types>
  <message> describes structure of SOAP messages
      . . . . .
  </message>
  <portType> defines method invocations
      . . . . .
  </portType> (bind requests and responses)
  <binding>
      . . . . .
  </binding>
  <service>
      . . . . .
  </service>
</definitions>
```

WSDL

definitions

```
<definitions>
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="uri:diy"
   targetNameSpace="uri:diy"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
>
      . . . . .
</definitions>
```

WSDL

messages

```
<messages>

<message name="helloWorldRequest>
<part name="name" type="soap:anyType" />
</message>

<message name="helloWorldResponse>
<part name="helloWorldResult" type="soap:anyType" />
</message>

</messages>
```

WSDL

portType

```
<portType name="helloWorldPort>
  <operation name="helloWorld">
    <input message="tns:helloWorldRequest" />
    <output message="tns:helloWorldResponse" />
  </operation>

  . . . . .

</portType>
```

WSDL

binding

```
<binding name="helloWorldSoap" type="helloWorldPort">
  <soap:binding style="rpc"
    transport="http://schemas.soap.org/soap/http" />
  <operation name="helloWorld">
    <soap:operation soapAction="radio" style="rpc" />
    <input>
      <soap:body use="encoded" namespace="uri:helloWorlds"
        encodingStyle="http:// schemas.soap.org/soap/encoding" />
    </input>
    <output>
      <soap:body use="encoded" namespace="uri:helloWorlds"
        encodingStyle="http:// schemas.soap.org/soap/encoding " />
    </output>
  </operation>

  . . . . .

</binding>
```

WSDL

service

```
<service name="helloWorldPort>
<document>
  textual service description
</document>
<port name="helloWorldPort" name="helloWorldSoap" />
<soap:address location="http://127.0.0.1:5555/" /> </port>
</service>
```

SOAP - vývoj aplikace

Server

1. application logic in C++, C#, or Java
2. (compilation) and generation of WSDL description
3. deployment into a web server (IIS, Axis)
4. WSDL registration in UDDI service

Client

1. readingof WSDL description from UDDI
2. generation of SOAP client proxy
3. encoding of the client application in C++, C# or Java
4. running the client in corresponding environment

SOAP - embedded server

Charon II

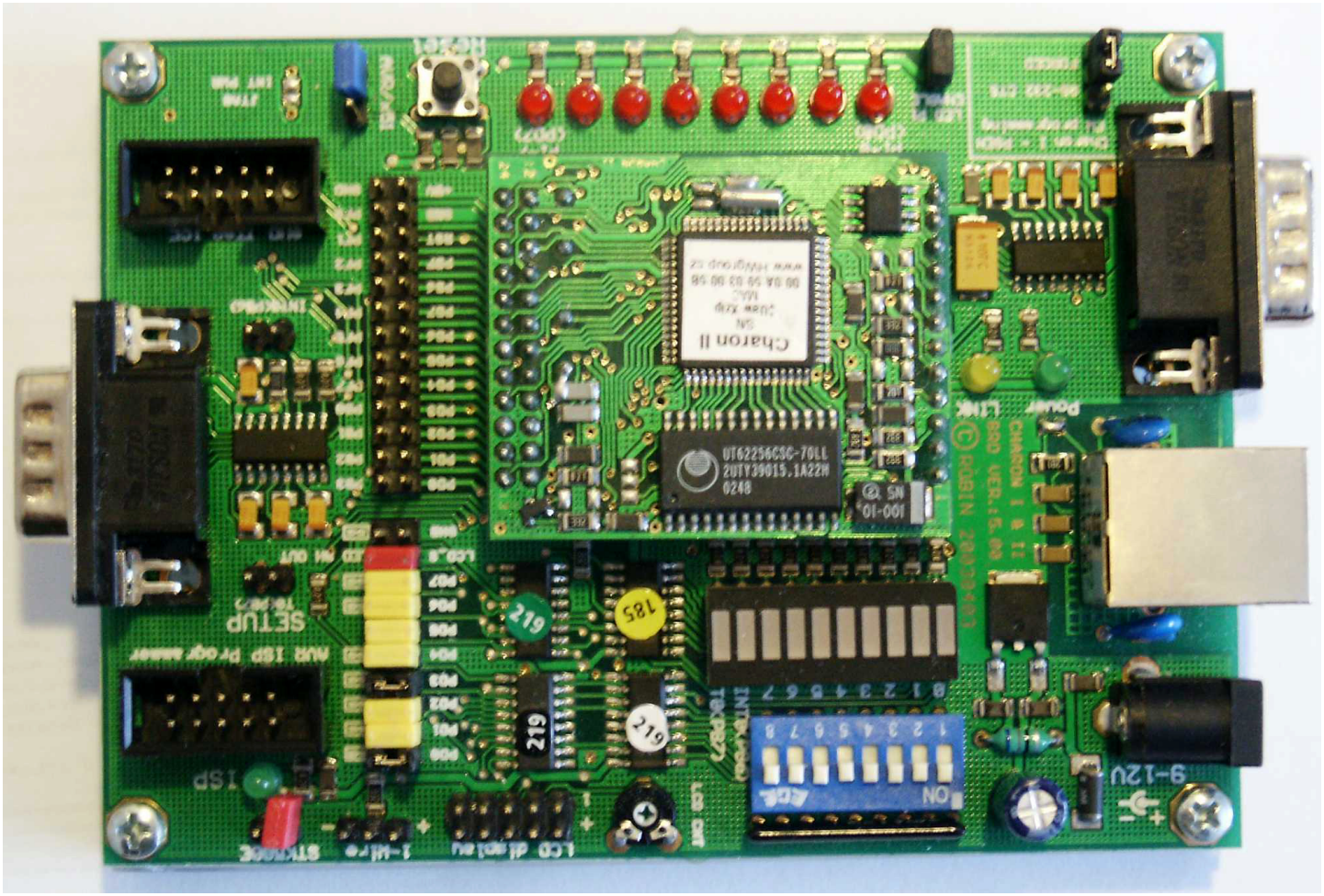
HW Server, Praha CZ
Egnite Software GmbH, Herne BRD

Atmel ATMega128
128 kB Flash EPROM
32 kB RAM
Realtek RTL8019 Ethernet Controller

Nut/OS - Multithreading real-time kernel
Nut/Net - TCP/IP implementation
simple HTTP “server”

SOAP - embedded server

Charon II



RESTful

REpresentational State Transfer

SOAP

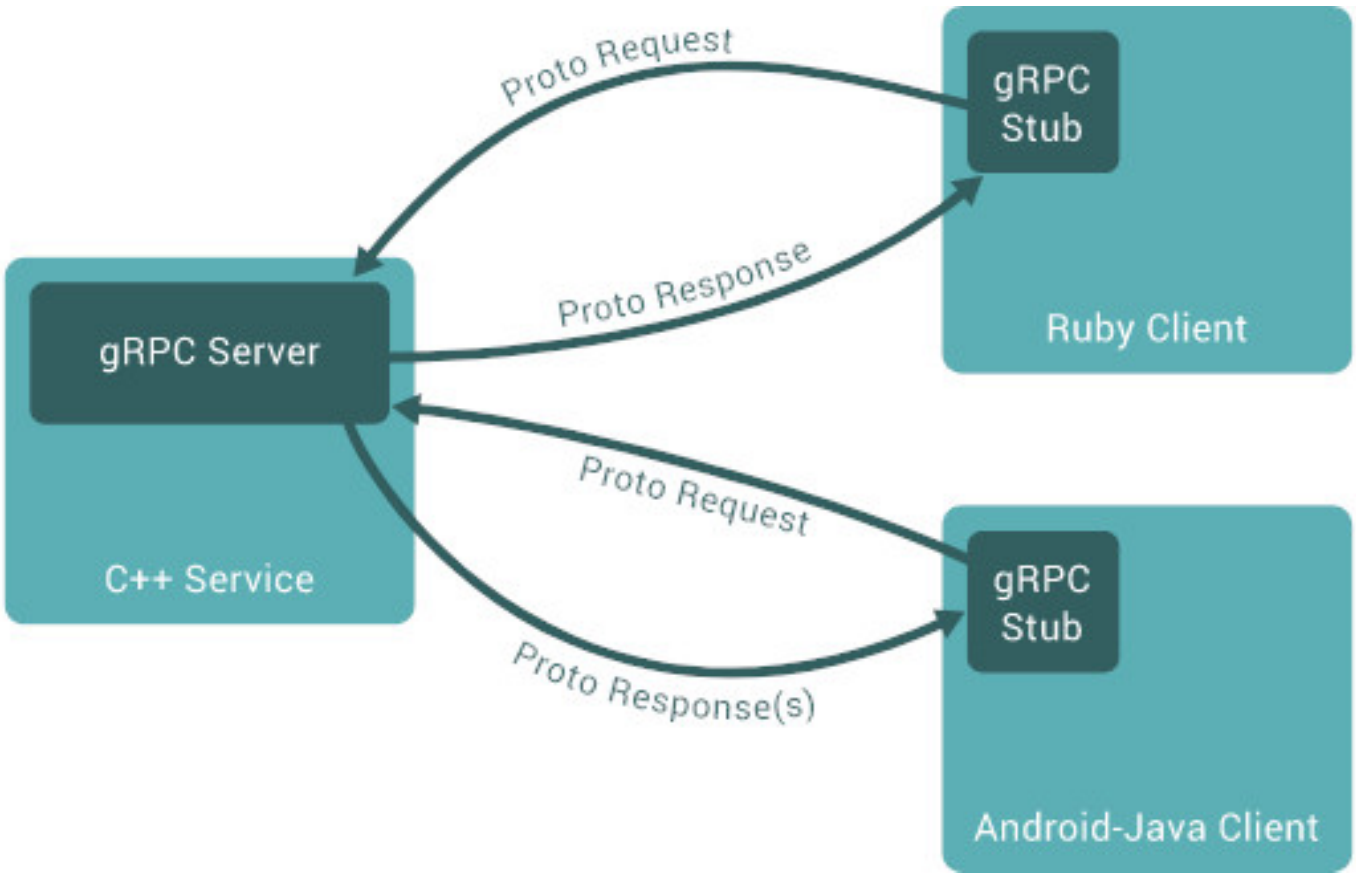
- primarily a RPC application
- XML requests
  - GET
  - PUT

RESTful

- primarily access to data sources (databases, clouds)
- CRUD (Create, Read, Update, Delete) operations
- XML requests
  - POST
  - GET
  - PUT
  - DELETE

gRPC

application structures



Asynchronous Procedural communication

asynchronous methods

- async. Java RMI
- gRPC
- node.js