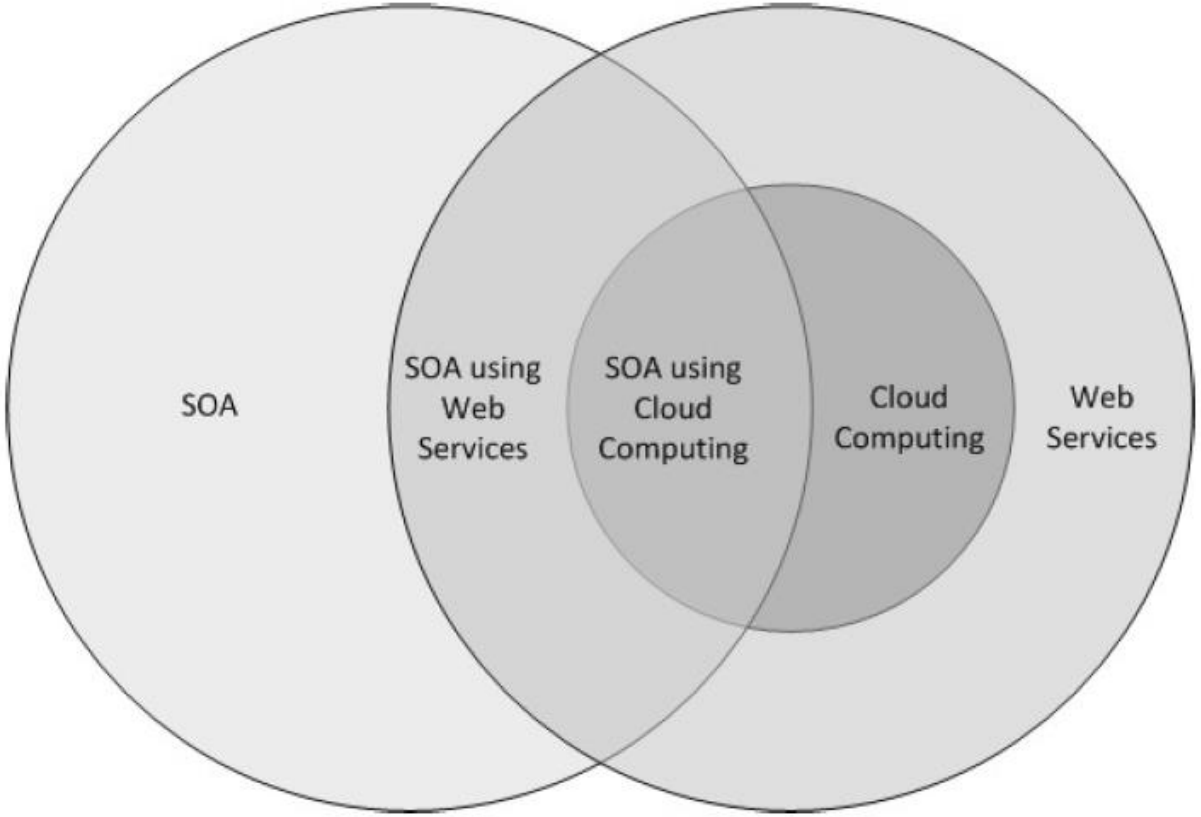
Web-Service and Cloud Computing

Web-Service and Cloud Computing



Source: Venn diagram of relationships between Web Services, SOA, and Cloud Computing (source Barry and Dick, 2013)

Web-Service and Cloud Computing

- • Cloud Computing and SOA are independent approaches.
- • Cloud Computing is a broad term for any Web service, which offers the entire “traditional IT stack”, such as software, hardware, and applications.
- • SOA, instead, focuses mainly on software services
- • Web Services encapsulates Cloud Computing in this diagram because Cloud Computing uses Web Services for connections.

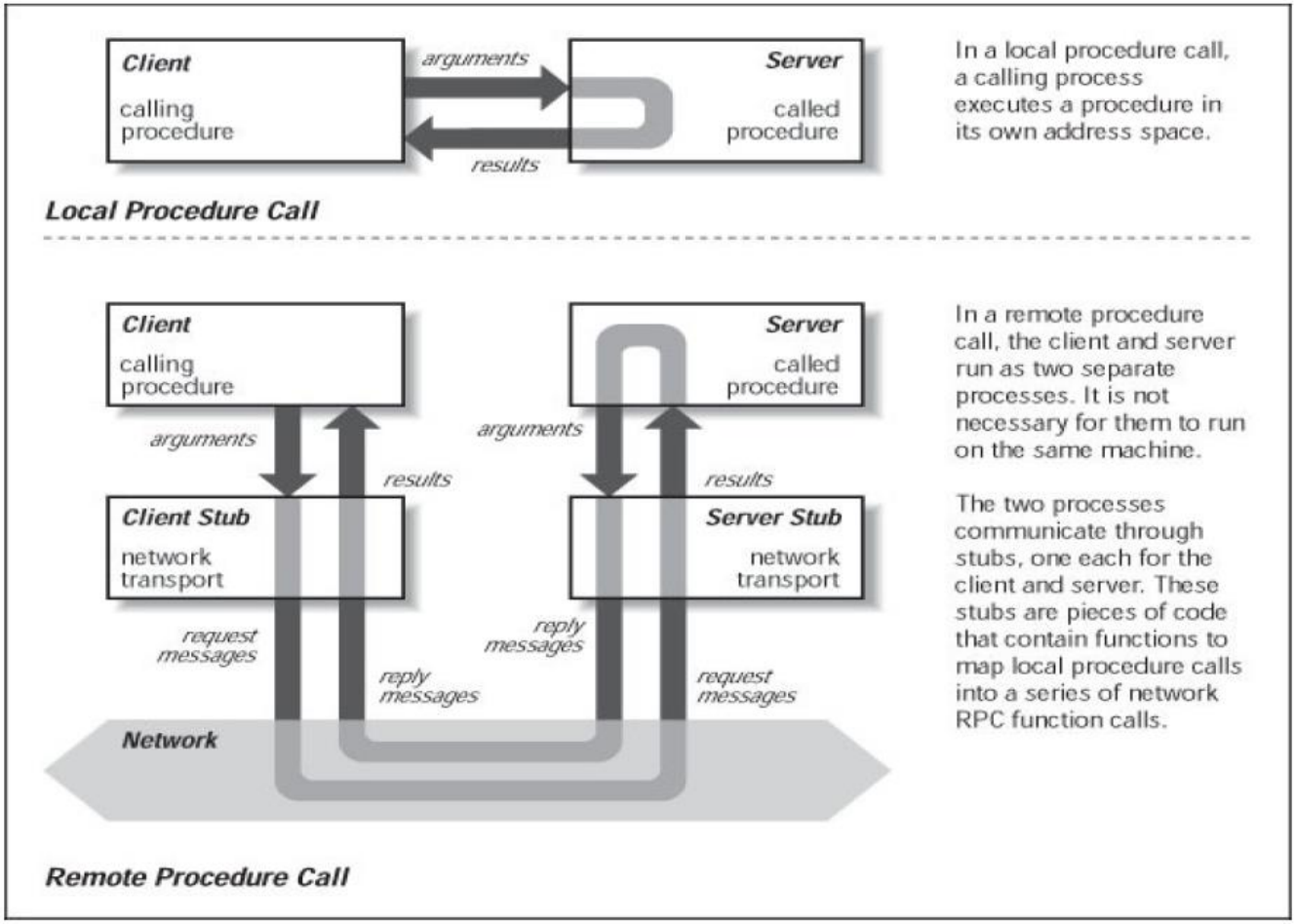
Web-Service and Cloud Computing

- It is possible, however, to use Web Services in situations other than Cloud Computing. Such use of Web Services may be part of a service-oriented architecture.
- It is possible to have a service-oriented architecture and not use Web Services for connections.

Remote Procedure Call (RPC)

- • An extension of conventional procedure call (used for transfer of control and data within a single process)
- • allows a client application to call procedures in a different address space in the same or remote machine
- • ideal for the client-server modeled applications
- • primary goal is to make distributed programming easy, which is achieved by making the semantics of RPC as close as possible to conventional local procedure call

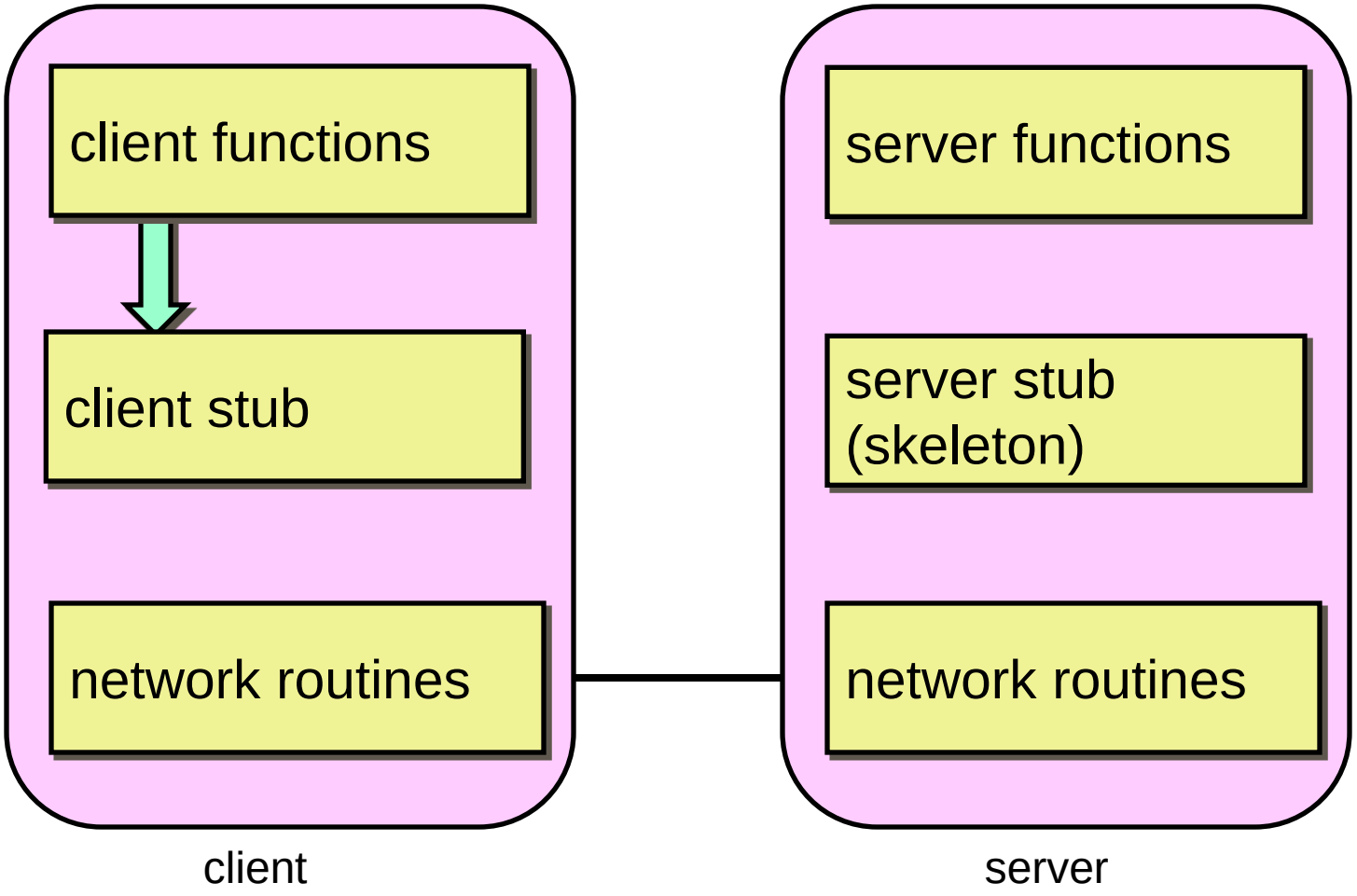
Local vs. Remote Procedure Calls



Source: “Power Programming with RPC” by John Bloomer, O’Reilly & Associates, 1992

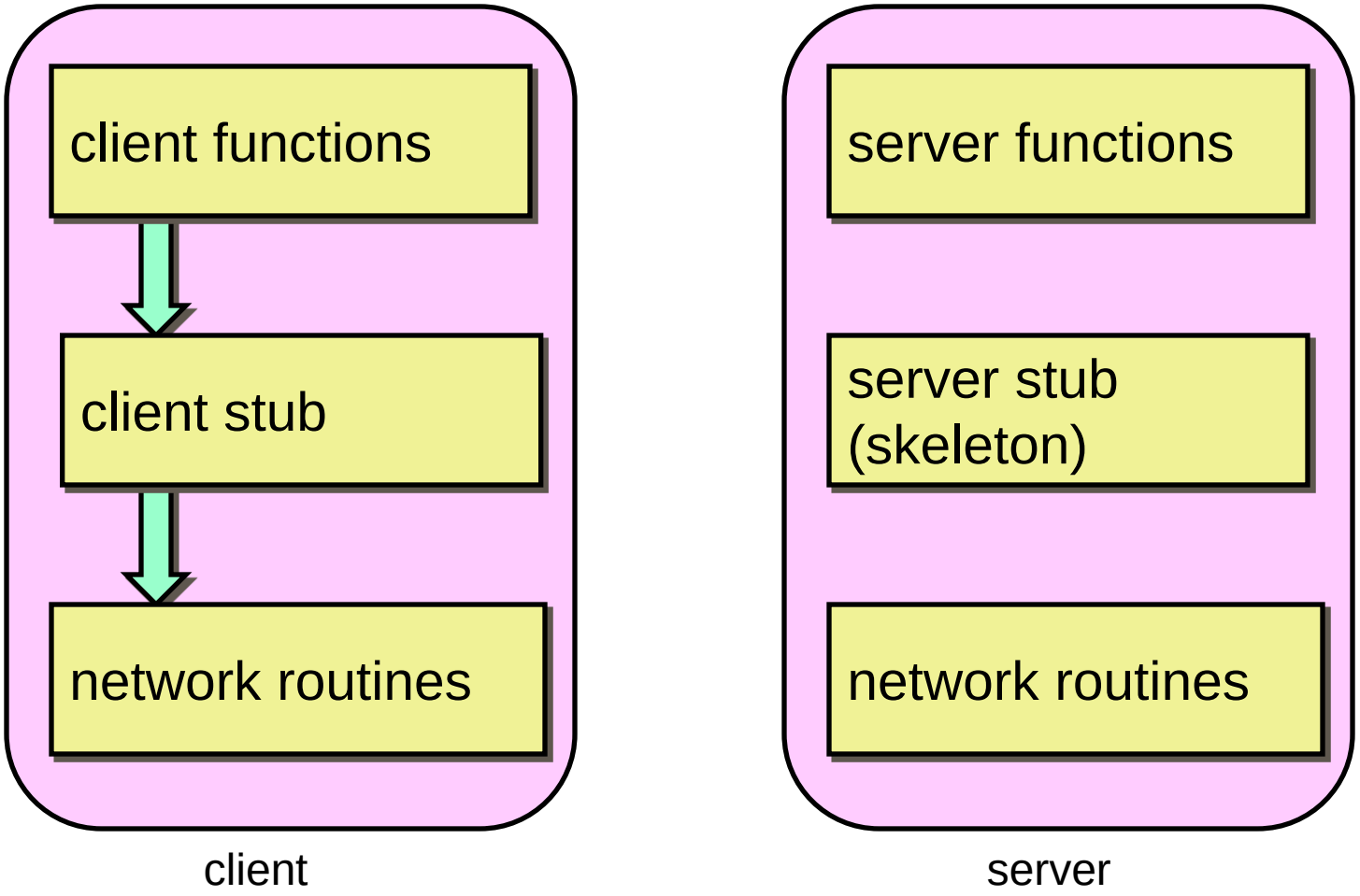
Steps of a Remote Procedure Call

1. Client calls stub



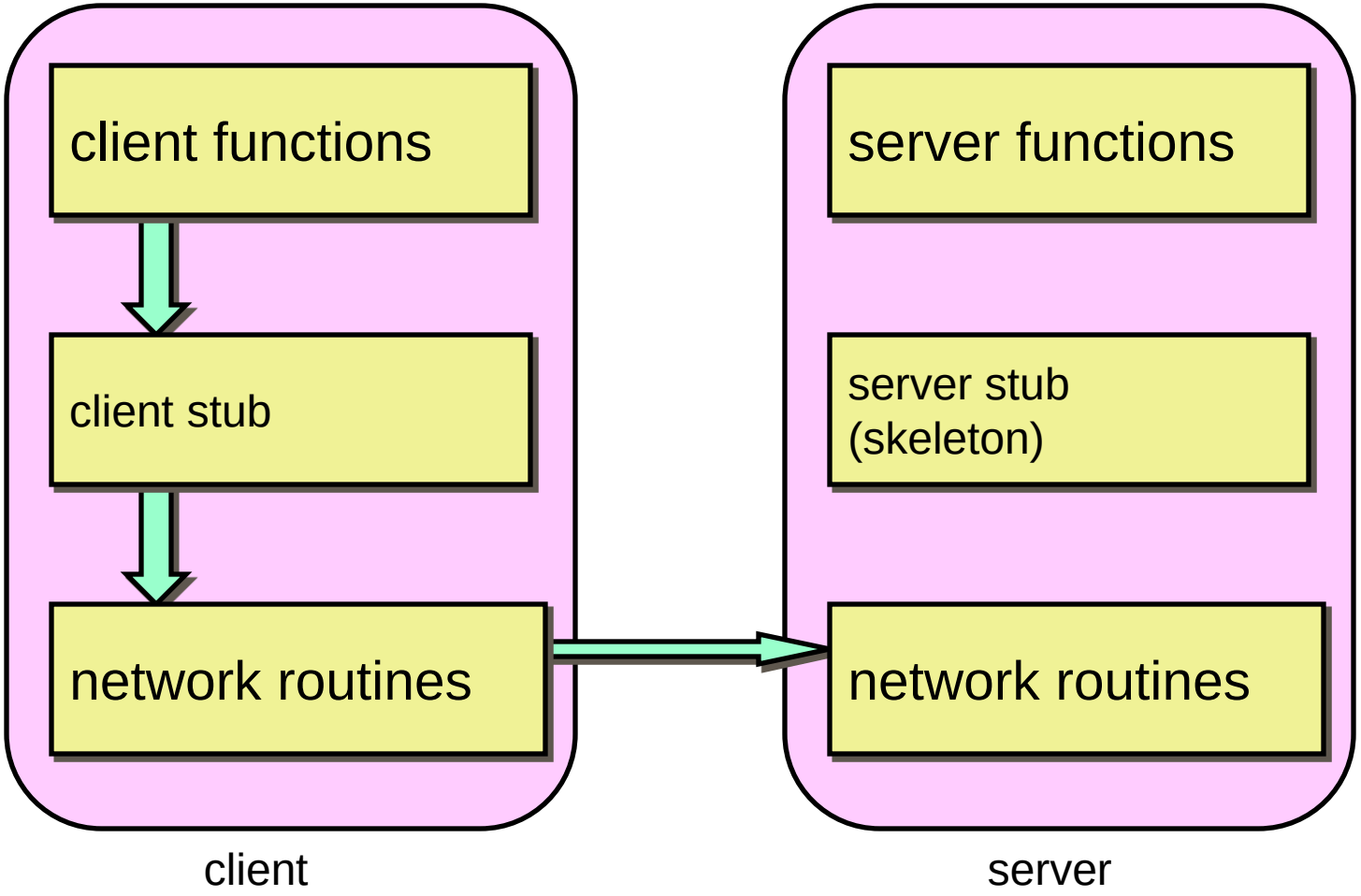
Stub functions

2. Stub marshals params to net message



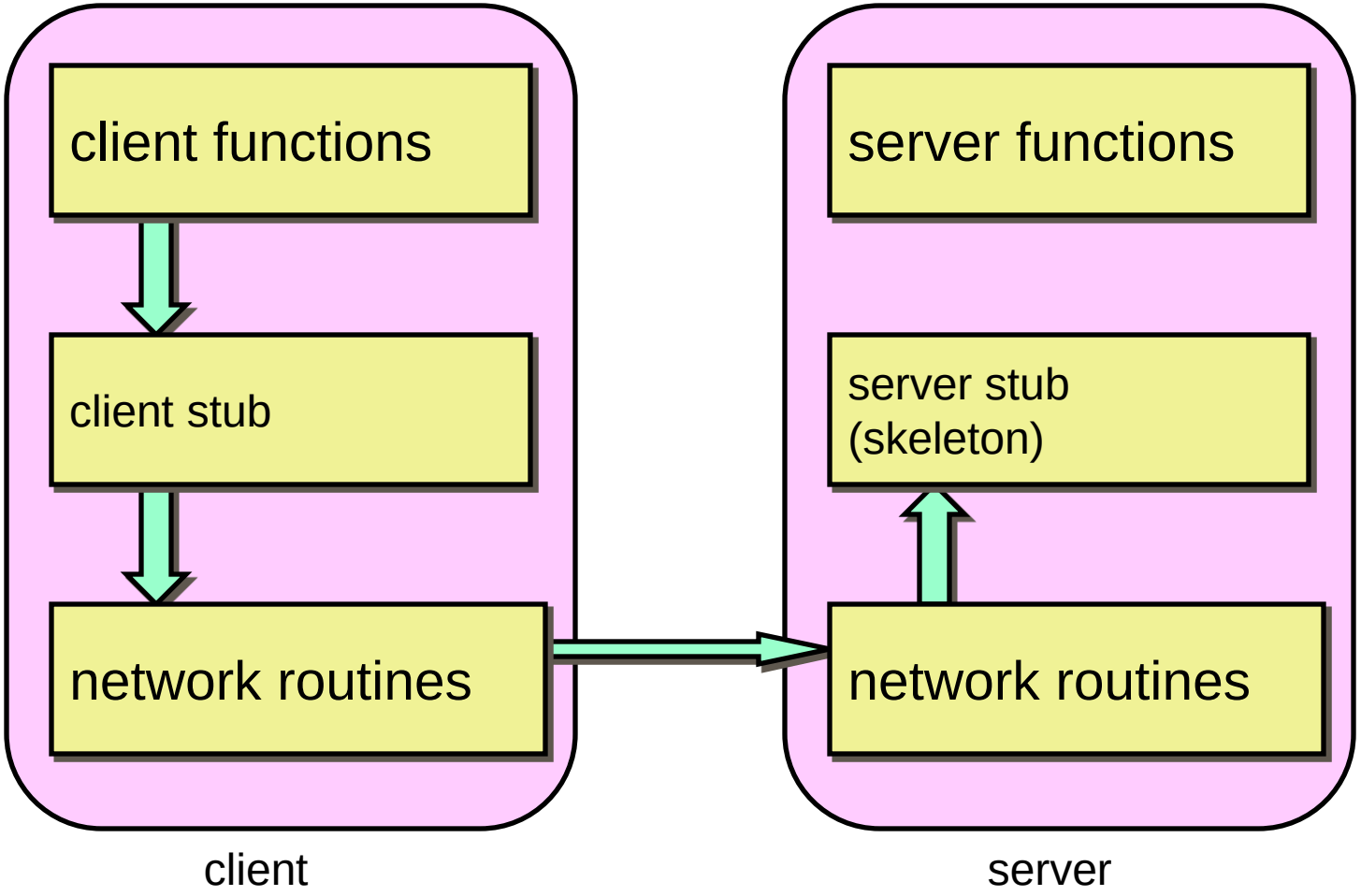
Stub functions

3. Network message sent to server



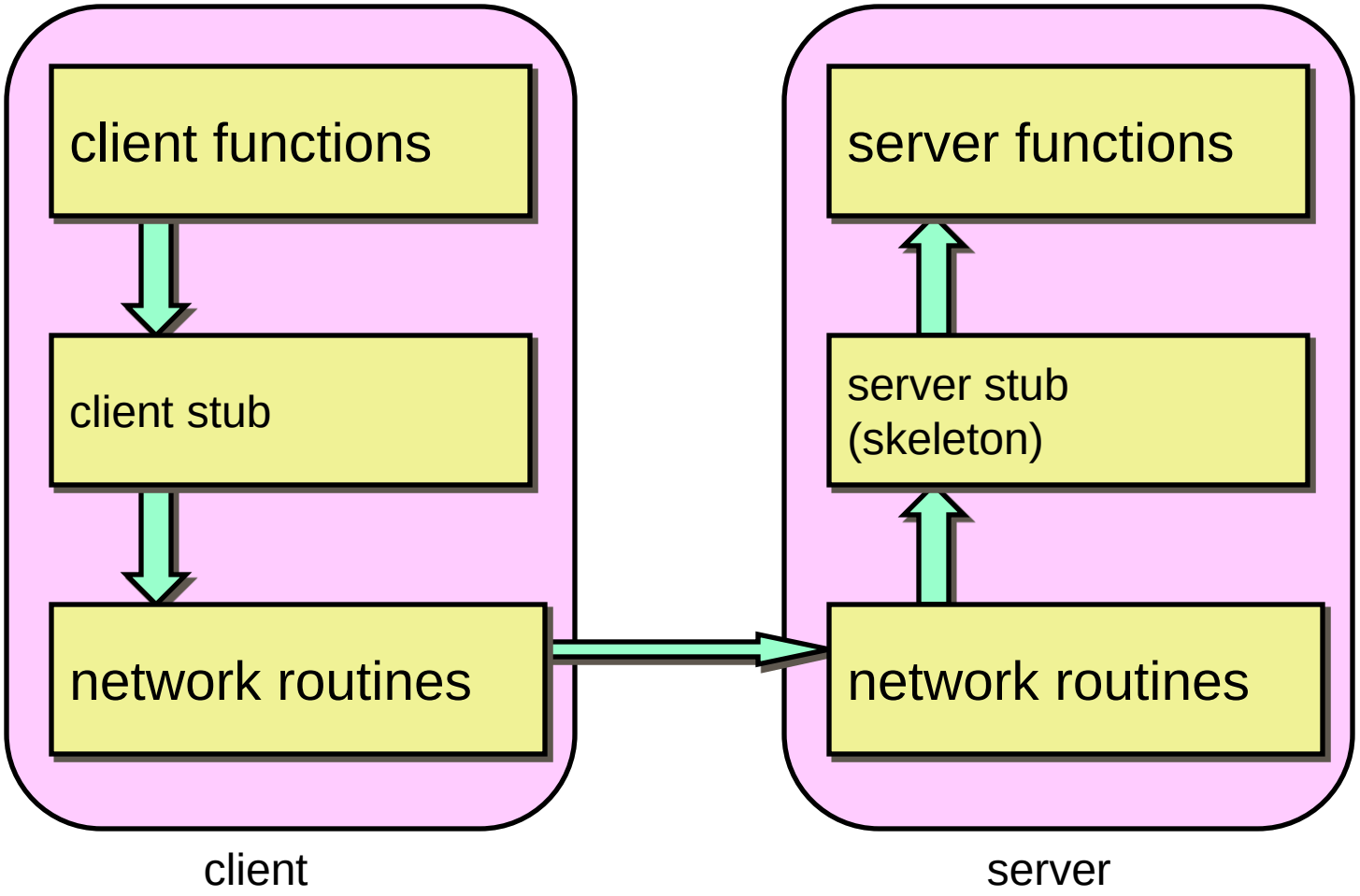
Stub functions

4. Receive message: send to stub



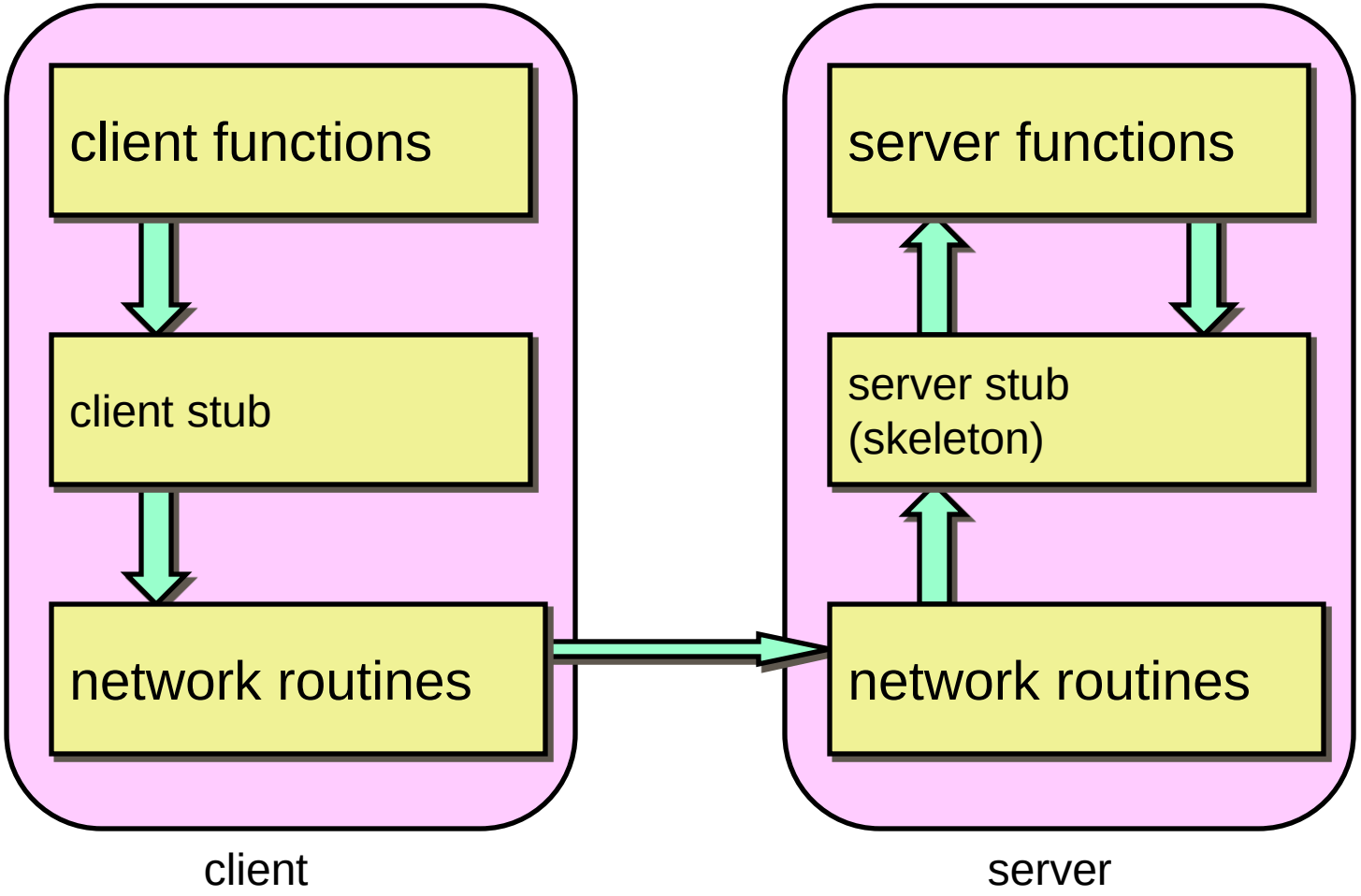
Stub functions

5. Unmarshal parameters, call server function



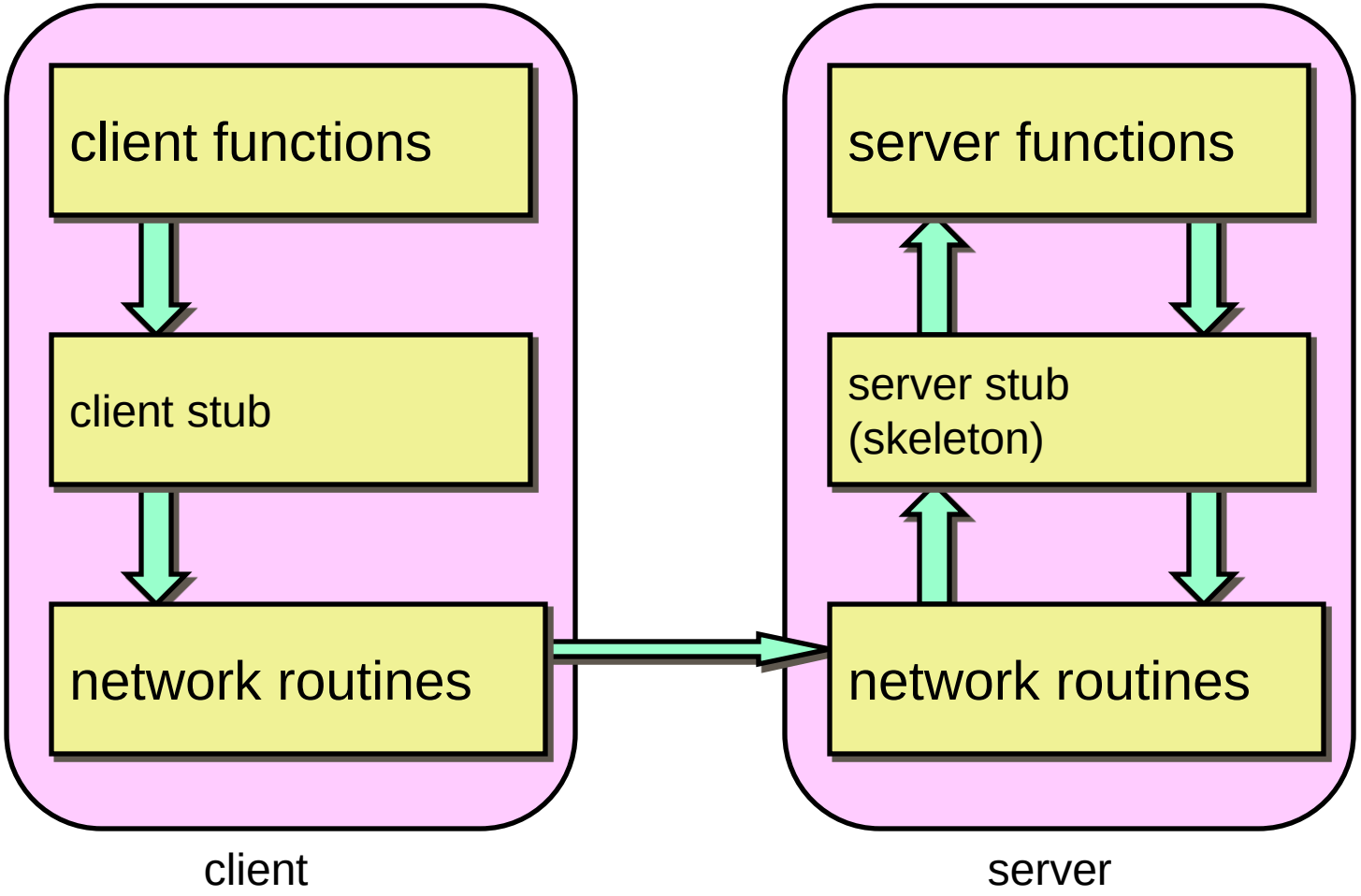
Stub functions

6. Return from server function



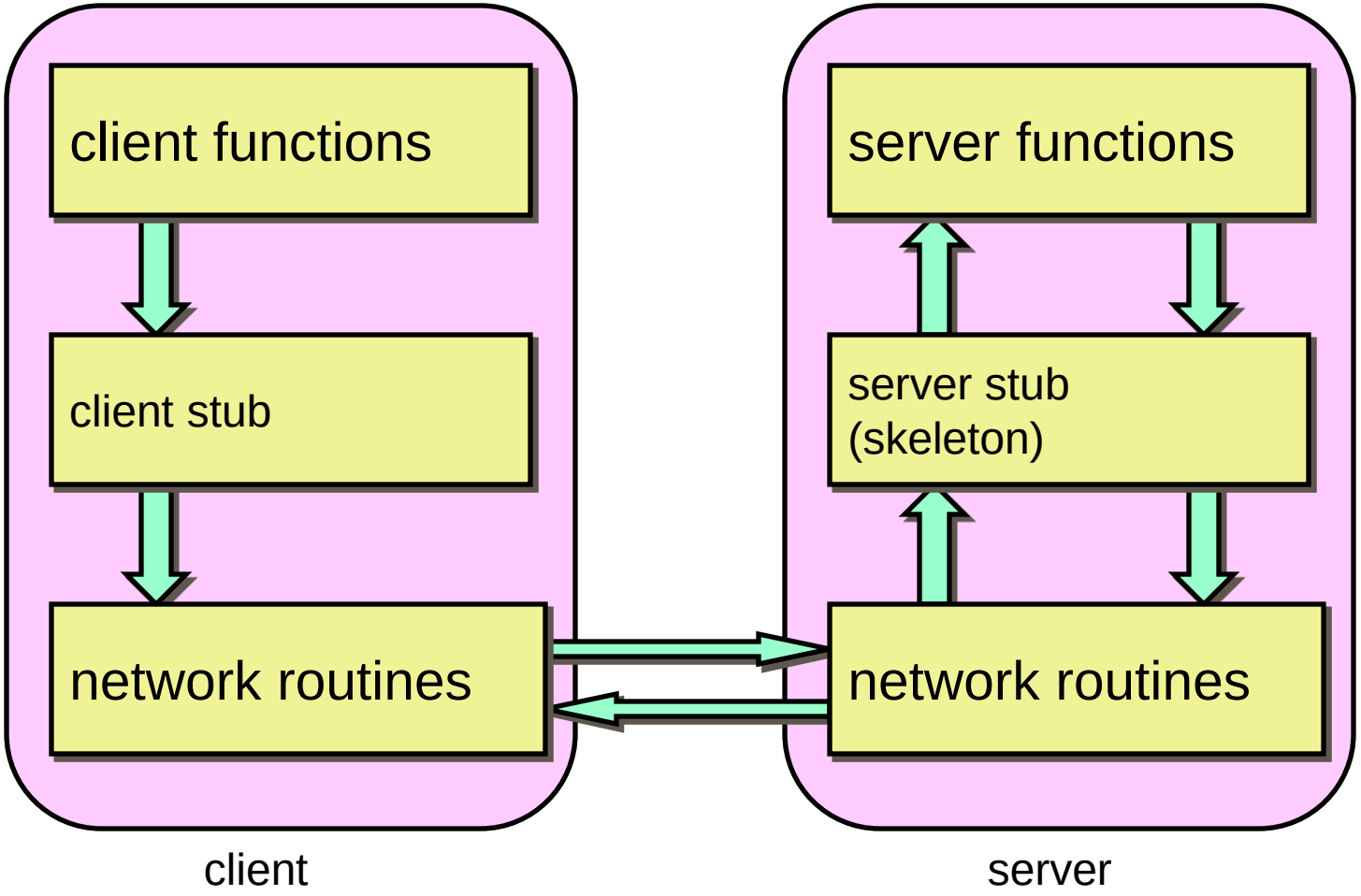
Stub functions

7. Marshal return value and send message



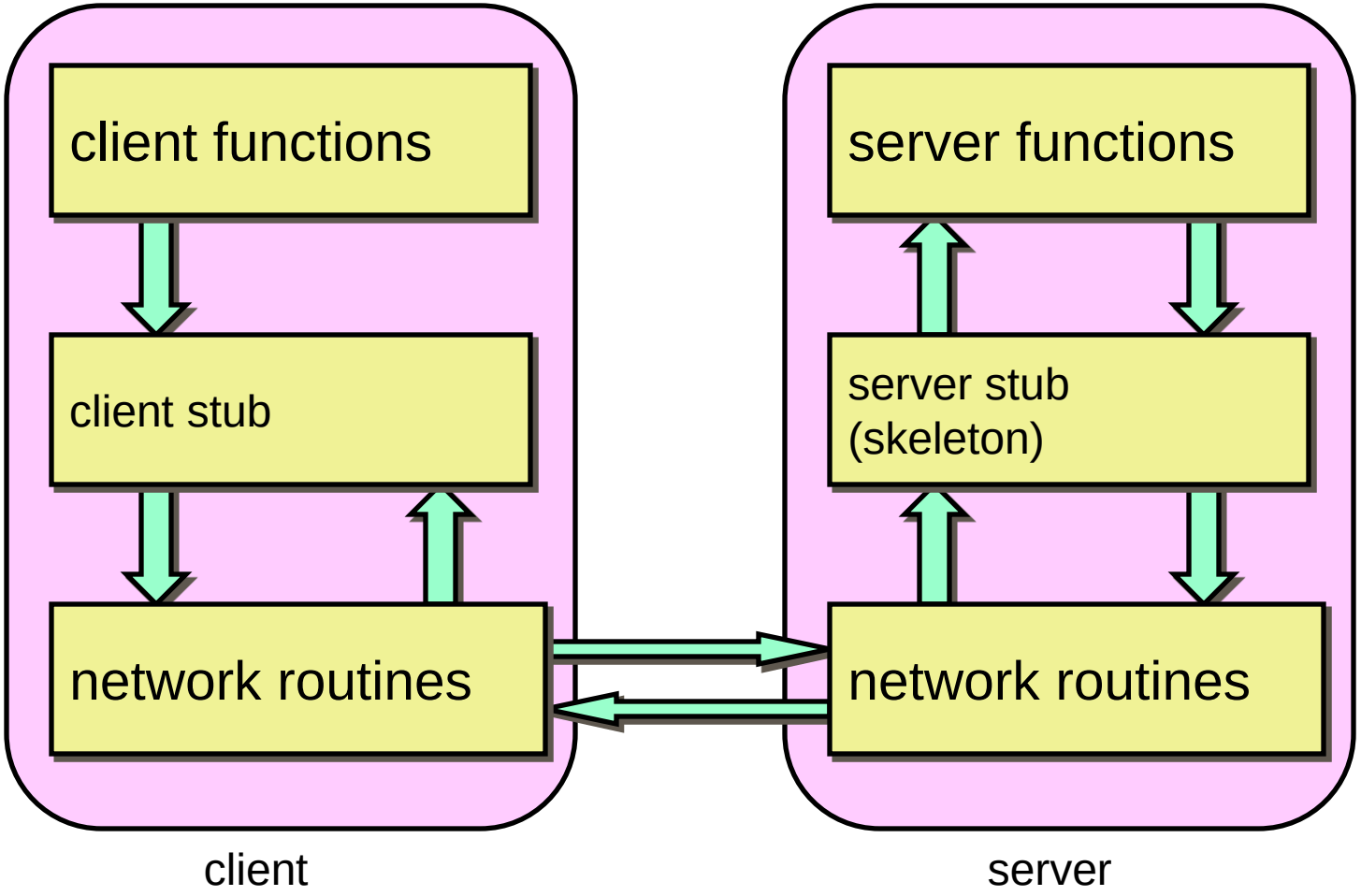
Stub functions

8. Transfer message over network



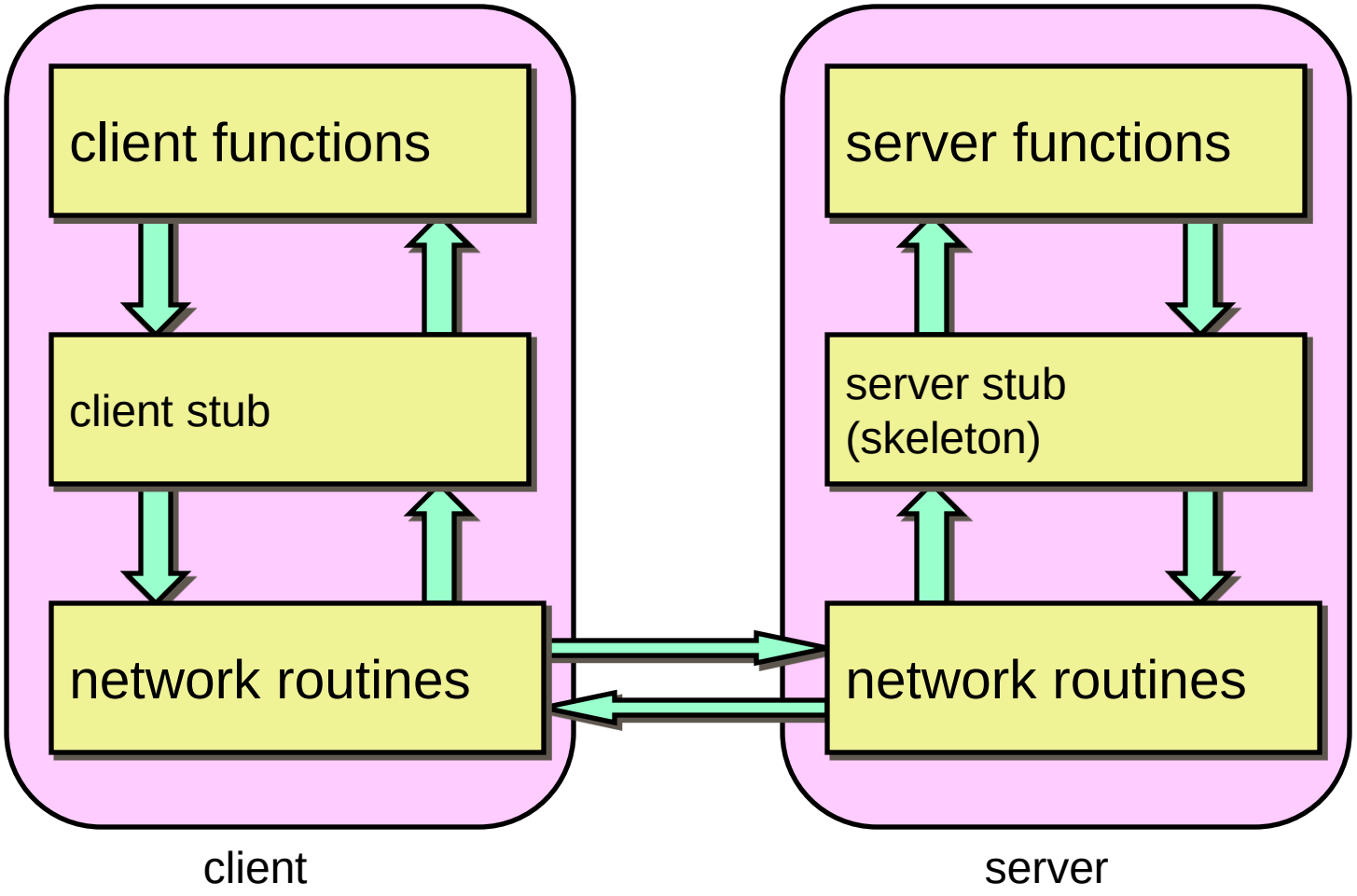
Stub functions

9. Receive message: direct to stub

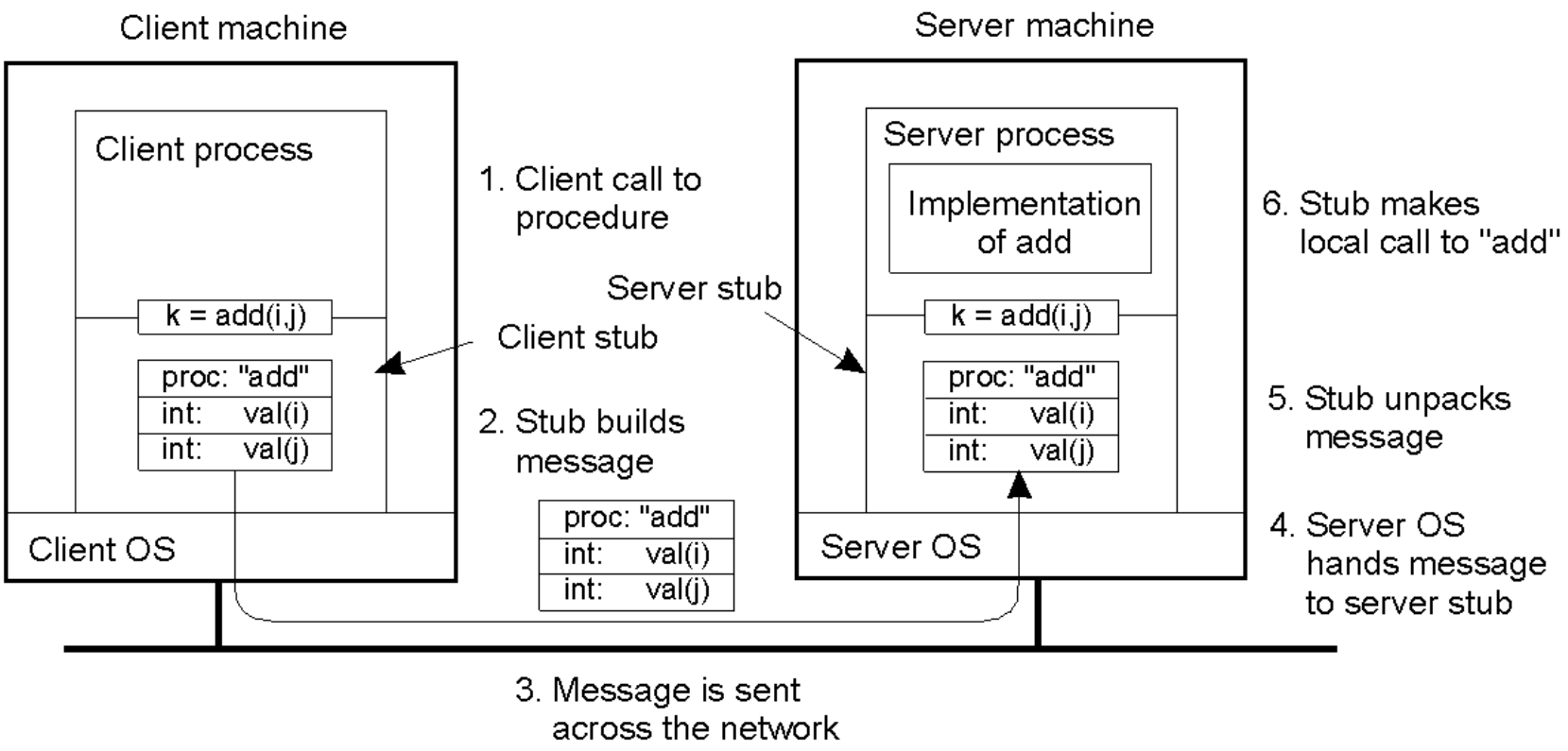


Stub functions

10. Unmarshal return, return to client code

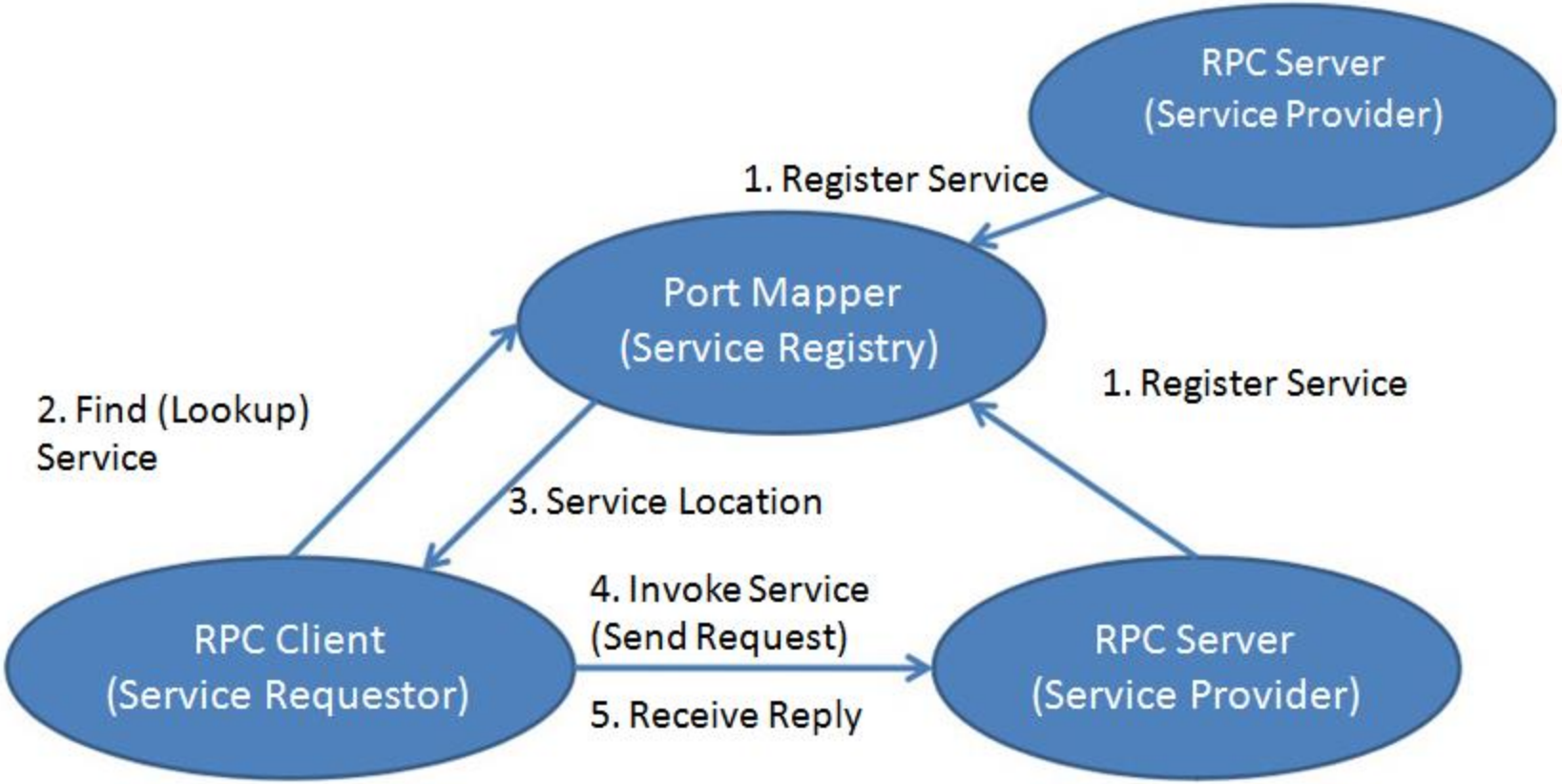


RPC Steps



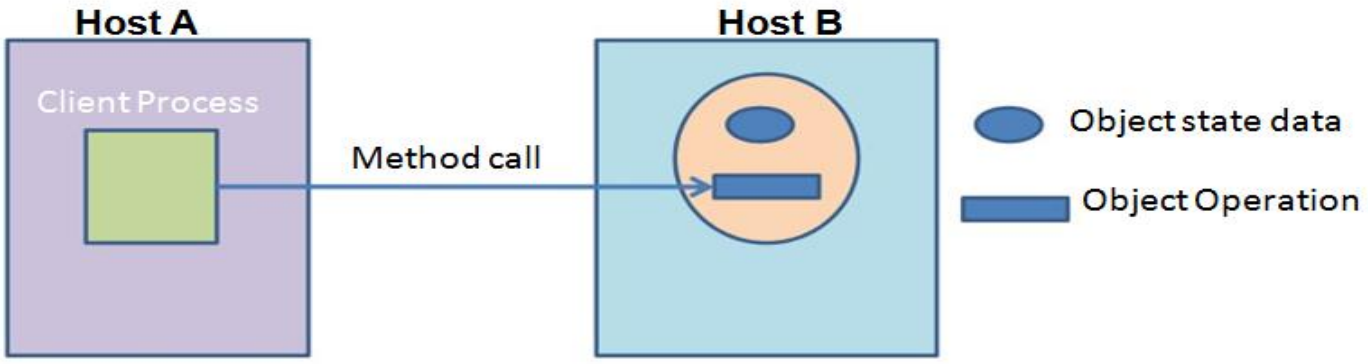
Steps involved in doing remote computation through RPC

RPC Steps


RPC Server
(Service Provider)
1. Register Service
Port Mapper
(Service Registry)
1. Register Service
2. Find (Lookup)
Service
3. Service Location
4. Invoke Service
(Send Request)
RPC Client
(Service Requestor)
5. Receive Reply
RPC Server
(Service Provider)

Distributed Objects

- A process running in host A makes a method call to a distributed object residing on host B, passing with the call data for the parameters, if any.
- A process which makes use of a distributed object is said to be a client process of that object, and the methods of the object are called remote methods.
- Distributed Object Mechanism
  - Java Remote Method Invocation (RMI),
  - Common Object Request Broker Architecture (CORBA) systems,
  - Distributed Component Object Model (DCOM)

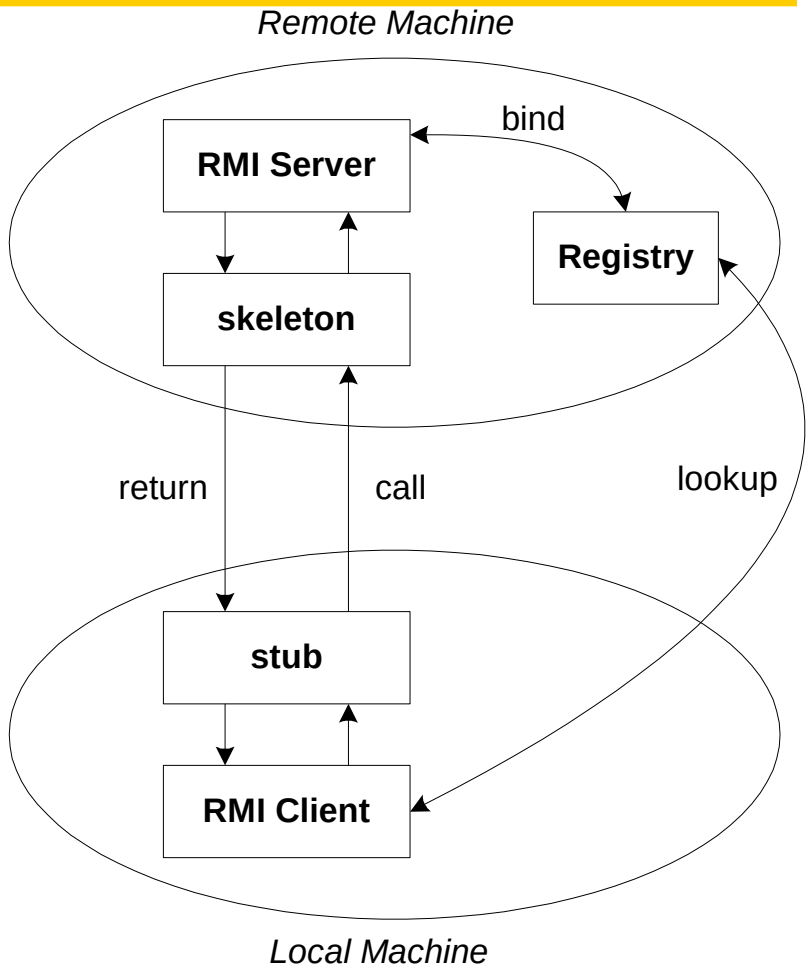


CORBA v/s RMI

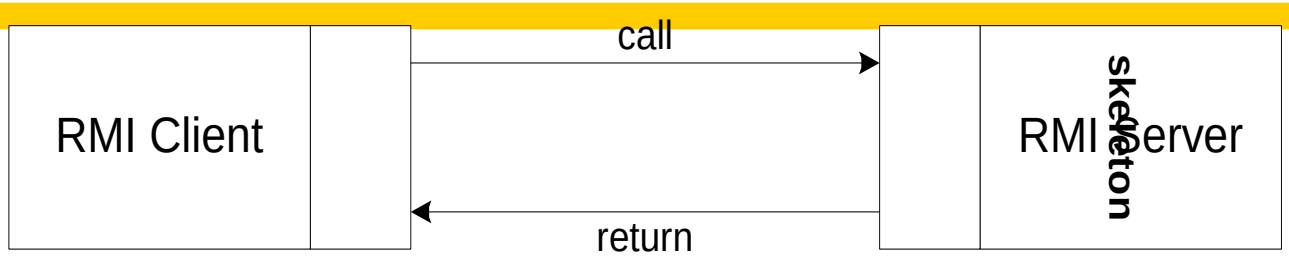
- CORBA (Common Object Request Broker Architecture) has long been king
  - CORBA supports object transmission between virtually any languages
  - Objects have to be described in IDL (Interface Definition Language), which looks a lot like C++ data definitions
  - CORBA is complex
- Microsoft supported CORBA, then COM, now .NET
- RMI is purely Java-specific
  - Java to Java communications only
  - As a result, RMI is much simpler than CORBA

The General RMI Architecture

- The server must first bind its name to the registry
- The client lookup the server name in the registry to establish remote references.
- The Stub serializing the parameters to skeleton, the skeleton invoking the remote method and serializing the result back to the stub.



The Stub and Skeleton



- A client invokes a remote method, the call is first forwarded to stub.
- The stub is responsible for sending the remote call over to the server-side skeleton
- The stub opening a socket to the remote server, marshaling the object parameters and forwarding the data stream to the skeleton.
- A skeleton contains a method that receives the remote calls, unmarshals the parameters, and invokes the actual remote object implementation.

Steps for Developing an RMI System

1. Define the remote interface
2. Develop the remote object by implementing the remote interface.
3. Develop the client program.
4. Compile the Java source files.
5. Generate the client stubs and server skeletons.
6. Start the RMI registry.
7. Start the remote server objects.
8. Run the client

Web Service

- A web service is a self-contained, language neutral, platform independent, and loosely coupled application. It can be described, published, located, and invoked over the Internet
- "Web services" is an effort to build a distributed computing platform for the Web.
- Difference between Web-site and Web-Service
  - Web-site is made for human consumption
  - Web-service is for machine (application) consumption
- Web Service Types
  - SOAP based(Simple Object Access Protocol)
  - REST based (Representational State Transfer)

SOAP vs REST Web Services

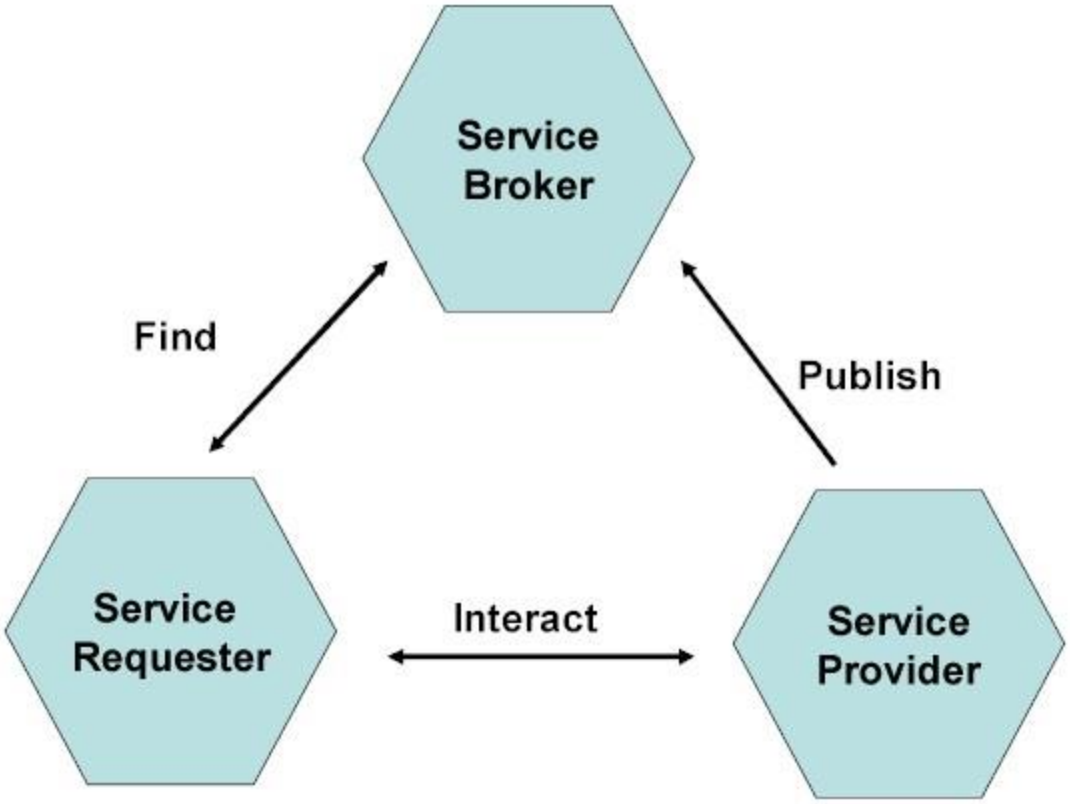
No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.
5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

Web-Service

Web Services were intended to solve three problems:

- Interoperability
  - Earlier distributed systems suffered from interoperability issues because each vendor implemented its own on-wire format for distributed object messaging
    - For example, RMI uses JRMP(Java Remote Method Protocol) and bound to JAVA language
- Firewall Traversal
- • Collaboration across companies was an issue because distributed systems such as CORBA and DCOM used non-standard ports
- • Web services use HTTP as a transport protocol and most of firewall allow access through port 80, leading to easier and dynamic collaboration.
- Complexity
- • Earlier technologies (RMI,CORBA) involve a whole learning curve
- • Web-service is a developer friendly service system.

Web Services Architecture



What are Web Services

Characteristics

- A Web Service is accessible over the Web.
- Web Services communicate using platform-independent and language-neutral Web protocols.
- A Web Service provides an interface that can be called from another program.
- A Web Service is registered and can be located through a Web Service Registry.
- Web Services support loosely coupled connections between systems.

Advantages of web services

- Loose Coupling
- Standard-Based (interoperability)
- Flexibility
- Reusability
- Scalability
- Reduced Complexity
- Programmatically Accessible
- Application-to-Application Communication

Web Service Stack

Web Services Technologies

- XML Messaging
  - **Simple Object Access Protocol (SOAP)** - is an XML Messaging Protocol that allows software running on disparate operating systems and different environments to make Remote Procedure Calls (RPC)

- Web Services Description
  - **Web Service Description Language (WSDL)** – is a language that defines the interface of a Web service, required for interaction between a requester and a service provider

- Web Services Registry
  - **Universal Description, Discovery and Integration (UDDI)** serves as a "business and service" registry essential for the widespread use of Web services

e**X**tensible **M**arkup **L**anguage

- **All the technologies in Web Services are XML based**
  - Messaging
  - Description
  - Registry

  } Are all in XML

- **Why?**
  - XML is pure text with no binary data
  - Applications read the XML
  - Applications share data using XML . Any application can talk to any other application using XML (unlike binary) irrespective of the platform
  - XML is a method for putting structured data in a text file

XML Document

```
<?XML version="1.0" encoding="UTF-8" standalone="no"?>
<!-- this is an XML comment -->
<books xmlns="somename"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="somename M:\XML\Schemas\docbook.xsd">
   <book year="2000" book-title="XML in Depth">
        <author>
        <title>Mr.</title> John Doe
        </author>
        <publisher> pub </publisher>
   </book>
</books>
```

XML instance schema *comment* root attributes element

XML Rules

- **Well formed**
  - Syntax is correct (all tags opened and closed)
- **Valid**
  - All the elements matches the definitions written in the schema
- **XML Documents (.xml) == XML Instances of the Schema (.xsd)**
- **DTD**
  - Document Type Definitions – Validates XML data against it
- **XML Schema**
  - Alternative to DTD with added functionality. It supports other data types not supported by DTD
    - Predefined Simple Types (integers,booleans,dateTime…)
    - User-defined datatypes (Complex Types)
    - Validations Restrictions to types
  - XML schema itself is an XML document !
- **XML Processing**
  - Read the XML documents XML processors (Parsers)
    - SAX –Simple API for XML(based on events)
    - DOM -*Document Object Model* (reads the xml document and loads it in memory)
    - Python implements this interfaces in a package PyXML

XML v/s HTML

HTML

```
<html>
<title>Course Roster</title>
<body>
<center>
  <h1>Course Roster</h1>
  <h2>XML Programming</h2>
  <h3>Department: EECS</h3>
  <p>
  <table border=2>
    <tr>
      <th>Teacher</th>
      <td>Paul Thompson</td>
    </tr><tr>
      <th>Student<br>List</th>
      <td>Ron Jones<br>
          Uma Abingdon<br>
          Lindsay Garmon
      </td>
    </tr>
  </table>
</center>
</body>
</html>
```

XML

```
<?xml version="1.0"?>
<course>
  <name>Java Programming</name>
  <department>EECS</department>
  <teacher>
    <name>Paul Thompson</name>
  </teacher>
  <student>
    <name>Ron Jones</name>
  </student>
  <student>
    <name>Uma Abingdon</name>
  </student>
  <student>
    <name>Lindsay Garmon</name>
  </student>
</course>
```

XML v/s HTML

XML

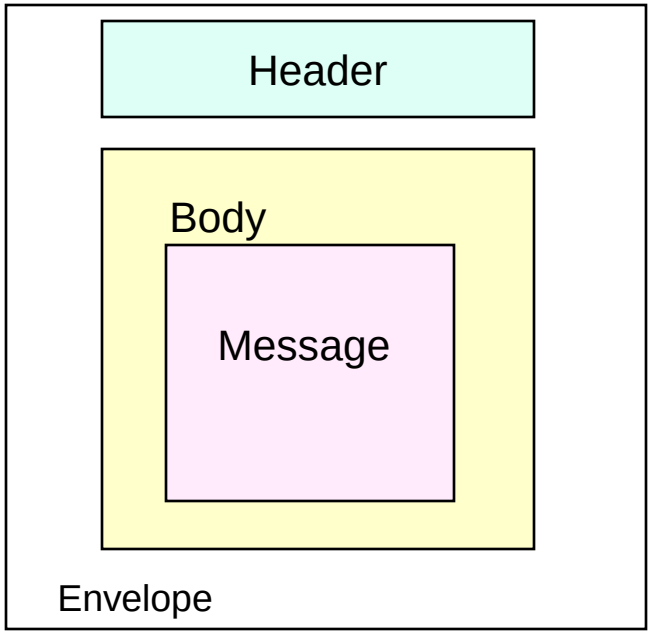
- XML is for computers
- Extensible set of tags: allows user to specify what each tag and attribute means
- Content orientated
  - Designed to transport and store data
- XML Instance Document Validation

HTML

- HTML is for humans
- Fixed set of tags: tags and attributes are pre-determined and rigid
- Presentation oriented
  - Designed to display data
  - HTML describes both structure (e.g. <p>, <h2>,) and appearance (e.g.
, <font>, <i>)
- No data validation capabilities

SOAP: Simple Object Access Protocol

- **An Internet standard specification, the goal of which is to define a platform and vendor-neutral WIRE PROTOCOL based on Internet standard protocols [HTTP & XML] to access Web Services**
- **How do we access a service???**
  - With a SOAP message: Is a XML stream which is used to transmit messages via HTTP
- **SOAP Structure**
  - Envelope: contains the entire SOAP message
  - Header
  - Body
    - Message



SOAP Example

Soap Request

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
                <m:calculateCarPayment
xmlns:m="http://www.exadel.com/services/CarPayment.xsd">
                        <loanAmount xsi:type="xsd:string">5000</loanAmount>
                        <loanTerm xsi:type="xsd:string">12</loanTerm>
                        <loanRate xsi:type="xsd:string">8</loanRate>
                </m:calculateCarPayment>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This example sends a request for a web service method called calculateCarPayment with three different arguments

You can try that with XML SPY (v 4.4)

Soap Example II

Soap 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:calculateCarPaymentResponse
xmlns:ns1="http://www.exadel.com/services/CarPayment.xsd" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                        <Result xsi:type="xsd:double">434.94</Result>
                </ns1:calculateCarPaymentResponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The response could include Fault elements to describe any error that occurred invoking the service

Invoking a web service

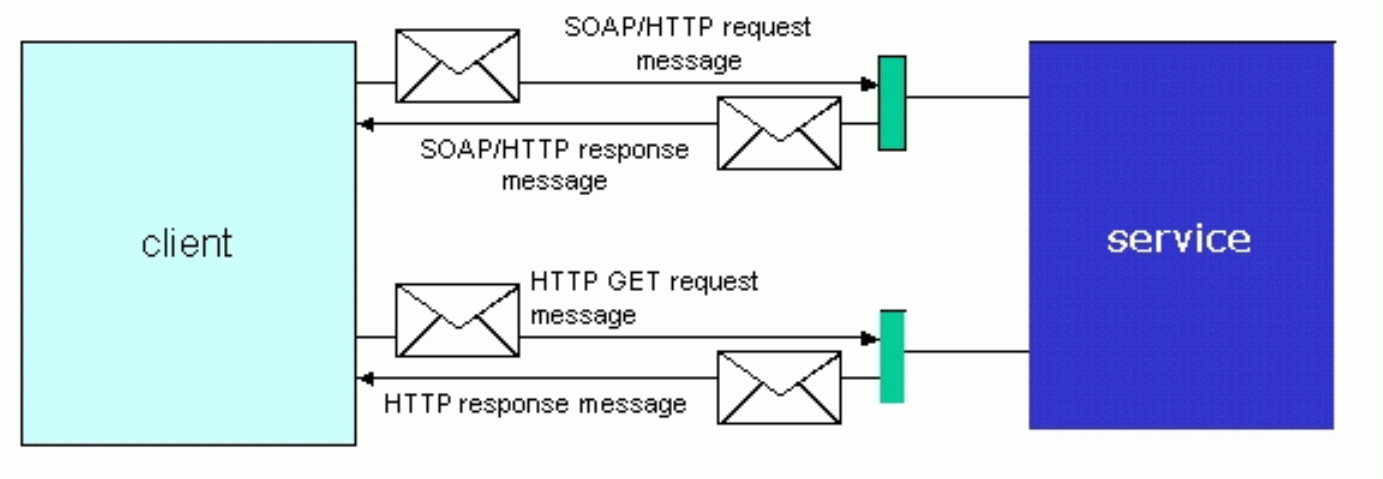


Figure 1. A client invoking a Web service

Example

- Consider a simple account-management and order processing system.
- The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.
- The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

Steps to perform this operation :

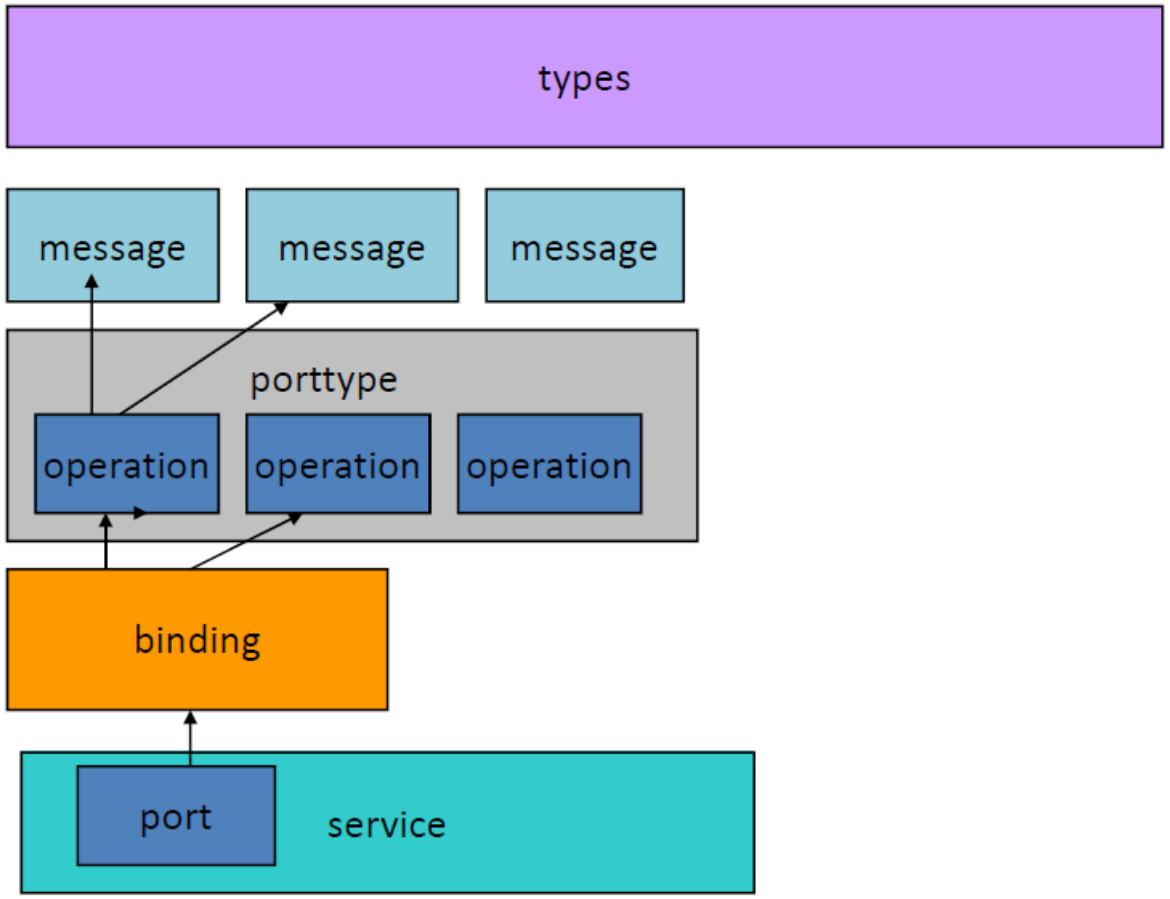
- The client program bundles the account registration information into a SOAP message.
- This SOAP message is sent to the web service as the body of an HTTP POST request.
- The web service unpacks the SOAP request and converts it into a command that the application can understand.
- The application processes the information as required and responds with a new unique account number for that customer.
- Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
- The client program unpacks the SOAP message to obtain the results of the account registration process.

WSDL Web Service Definition Language

- WSDL is an XML-based language used to define Web Services and describe how to access them.
- It is the external interface for a client (IDL)

- WSDL includes information about
  - Data types it uses
  - Parameters it requires and returns
  - Groupings of functionality
  - The protocol to be used to access the service
  - The location or address of the service

WSDL Structure



WSDL Structure

- <definition> - Root element
- <types> - Provides data type definitions
- <message> - Represents the abstract definition of the data being transmitted
- <portType> - Defines a set of abstract operations
- <binding> - Specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType
- <port> - Specifies an address for a binding
- <service> - Used to aggregate a set of related ports
- <serviceType> - Mechanism to aggregate portTypes

```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>
```

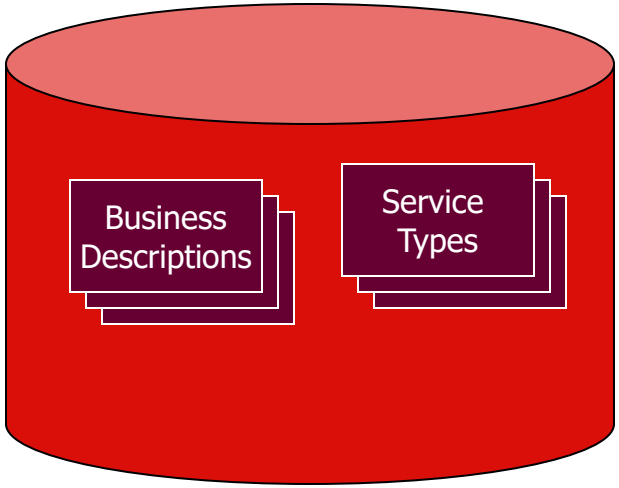
```
   <binding name = "Hello_Binding" type = "tns:Hello_PortType">
      <soap:binding style = "rpc"
         transport = "http://schemas.xmlsoap.org/soap/http"/>
      <operation name = "sayHello">
         <soap:operation soapAction = "sayHello"/>
         <input>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </input>

         <output>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </output>
      </operation>
   </binding>

   <service name = "Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding = "tns:Hello_Binding" name = "Hello_Port">
         <soap:address
            location = "http://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

- Example Analysis
- **Definitions** − HelloService
- **Type** − Using built-in data types and they are defined in XMLSchema.
- **Message** −
  - sayHelloRequest − firstName parameter
  - sayHelloresponse − greeting return value
- **Port Type** − sayHello operation that consists of a request and a response service.
- **Binding** − Direction to use the SOAP HTTP transport protocol.
- **Service** − Service available at http://www.examples.com/SayHello/
- **Port** − Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.

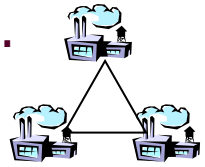
UDDI Universal Description Discovery and Integration

1. 
SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

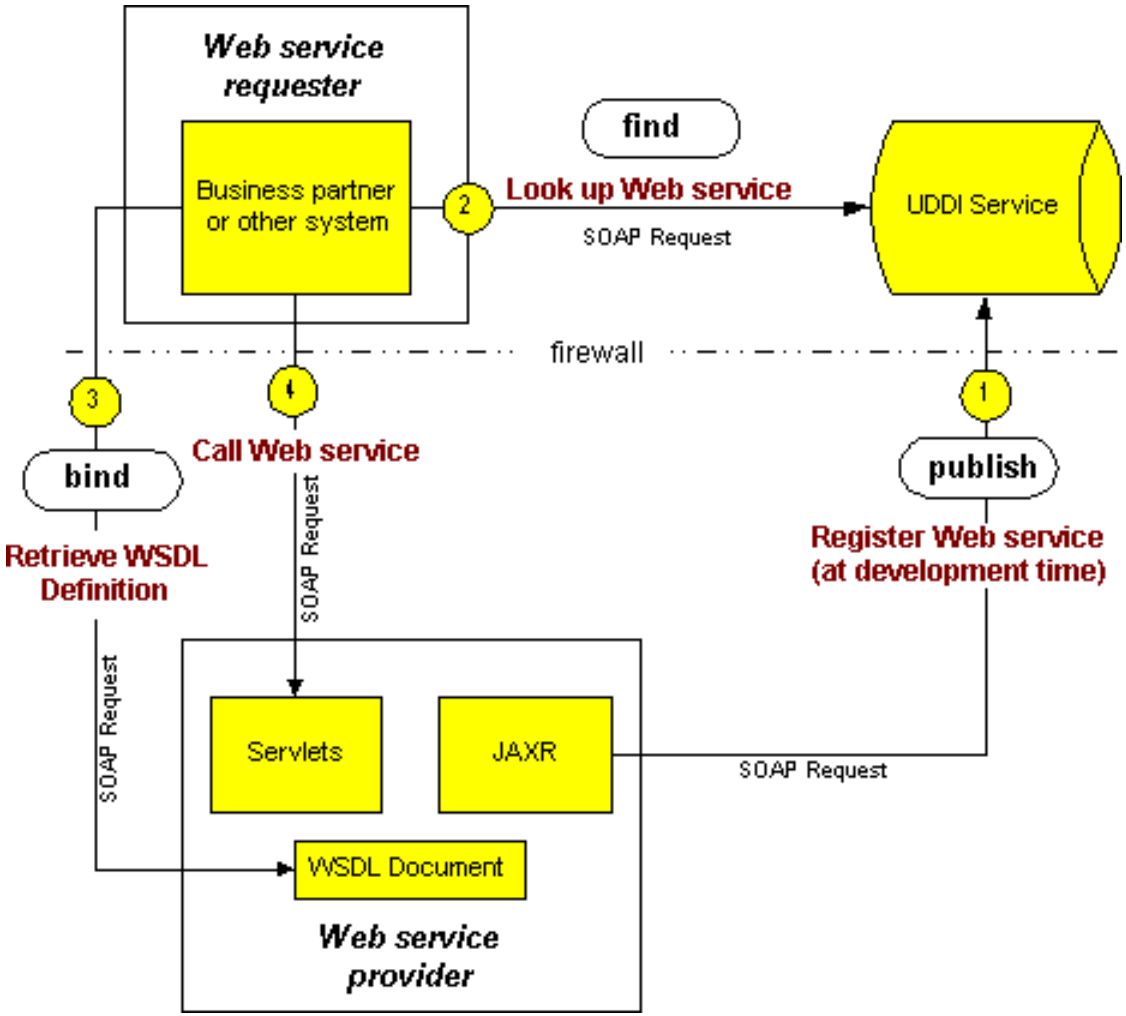
2. 
• Businesses populate the registry with descriptions of the services they support



3. 
Marketplaces, search engines, and business apps query the registry to discover services at other companies

4. 
Business uses this data to facilitate easier integration with each other over the Web

How it works all together



UDDI Registries

- https://uddi.ibm.com/testregistry/registry.html
- http://demo.alphaworks.ibm.com/browser/
- http://uddi.microsoft.com
- http://uddi.hp.com

Comparison of Web-Service with other Technologies

	JAVA RMI	CORBA	Web-Service
Programming Language	JAVA	Independent	Independent
Interface Definition	Java Interface	CORBA IDL	WSDL
Data Structure	Java Objects	IDL-Specified objects	XML Data
Transport Protocol	RMI (JRMP)	GIOP / IIOP	HTTP, SMTP
Packaging	Java Object Serialization	ORB	SOAP

Approaches for creating web-service

- **Code First Approach**
  - Writing source code for web service and then WSDL File
- **Contract First Approach**
  - Creating Web-service based on given WSDL File