Communication Networks 2
Exercise 7 -
Distributed Programming

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Multimedia Communications Lab
TU Darmstadt

Problem 1 Motivation

What is the motivation for building distributed systems? Name four desired properties.

Solution:

a) Scalability

b) Openness

c) Heterogeneity

d) Fault tolerance

Problem 2 Transparencies

Which eight transparencies influence the design of a distributed system?

Solution:

a) Access Transparency

b) Location Transparency

c) Mobility/Migration Transparency

d) Replication Transparency

e) Concurrency Transparency

f) Failure Transparency

g) Performance Transparency

h) Scaling Transparency

Problem 3 Location Transparency

Please explain the location transparency and give an application example.

Solution:

a) A location transparent name contains no information about the named object's physical location

b) Non location transparent names are an administrative nightmare → the migration of services becomes nearly impossible

c) Examples:

- Domain names: e.g. kom.tu-darmstadt.de
- Your Email address

Problem 4 Remote Procedure Calls

Remote Procedure Call systems are used to invoke procedures on remote hosts. To do so, some steps are necessary. Name and describe these steps. Also describe on which node these steps happen?

Solution:

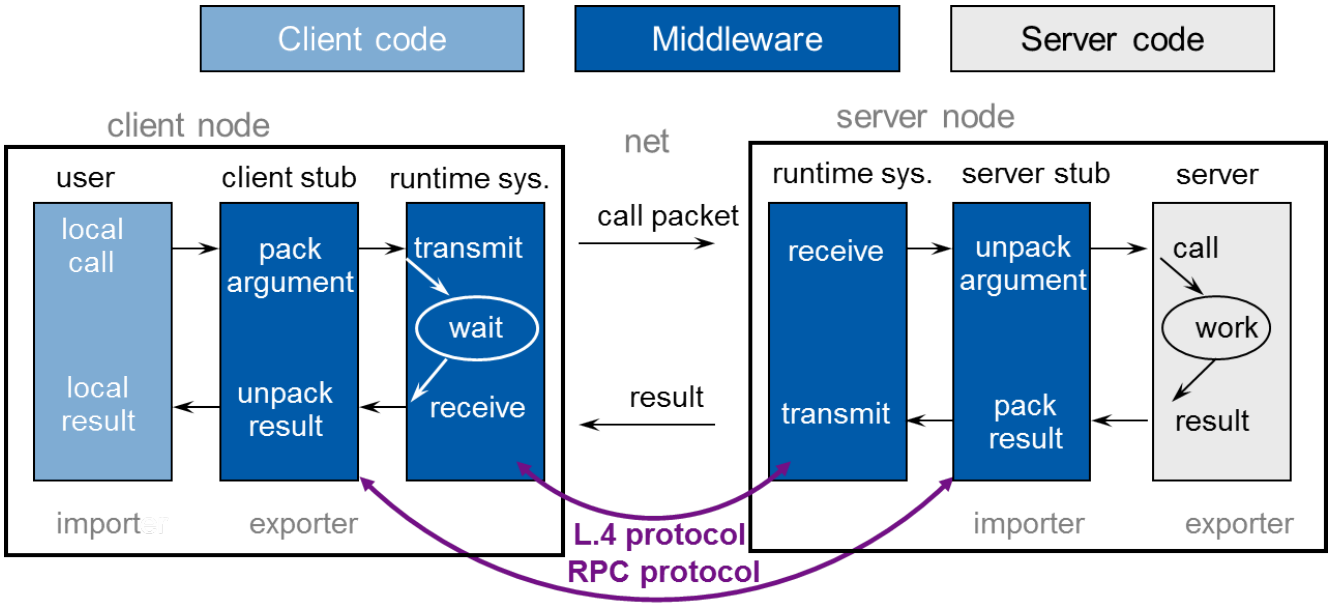

Figure 1: RPC Architecture

- On the Client Node: a server stub is called
  a) Pack arguments
  b) Transmit data to the server node

c) Receive the result

d) Unpack the result

e) The Server Stub returns the result to the caller

- The Server Node waits for data packets from the Client Node
  a) Receive the data from the client node
  b) Unpack the arguments
  c) Call the procedure implementation
  d) Pack the result to a response packet
  e) Transmit the response packet back to the client node

Problem 5 Failure Semantics

RPC usually uses UDP as transport layer protocol. As UDP is not reliable, some RPC packets might get lost. This loss causes the RPC call to fail. But there are four strategies ("failure semantics") to handle those failures. Please name and explain these semantics.

Solution:

a) *Maybe-Semantics*: No repeated requests
   - Simple, fast but often not sufficient
   - Idea: The user will try it again in case of a failure

b) *At-Least-Once-Semantics*: Infinite Retry
   - Repeated Requests but the server is stateless
   - Sufficient for read operations

c) *At-Most-Once-Semantics*: Tolerate omissions
   - Repeated requests & replies, duplicates recognized → RPC executed only once
   - No result in case of a server crash. Server may have (not) executed RPC.

d) *Exactly-Once-Sematics*: Tolerates crashes
   - For normal commercial RPC systems, this remains a dream
   - Transactional systems, fail-safe solutions needed

Problem 6 RMI Middleware

Please name and describe the three main components of the RMI middleware. Please also draw an image that describes how these components communicate with each other.

Solution:

a) *ServerStub* Interface to call the remote method

- Pack the arguments and transfer them to the server host
- Unpack the result and return it to the caller

b) *Server Skeleton* performs the wiring on the server side

- Unpack remote method call parameters
- Invoke the method on the server object
- Pack the result of the invocation
- Transfer the result to the client host

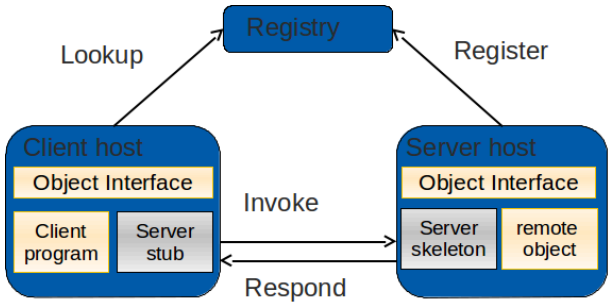

Figure 2: RMI Architecture

a) Registry Provides a name service for server objects

- The Server Host register server objects
- The Client Host lookup server objects by their name
- ⇒The client is supposed to know the name of the server object

Problem 7 Local/Distributed Objects

Please name and describe the differences between local and distributed objects. Hint: Which operations exist in the object live cycle of local and remote objects? How do those operations differ for local and remote objects?

Solution:

- Distributed objects have more meta data (object location, unique id, type meta data, . . .)
- Remote method invocations have a much higher latency (three to four orders of magnitude)
- Network communication necessary:
  - Creation: Objects are either created or migrated to another host
  - Cleanup: To decrease reference counter on dst. objects
  - Referencing: To increase reference counters on dst. objects

Problem 8 Preparing a soap client

In the HTTP exercise you wrote a parser that parsed the mensa menu and published it on the web. For this task, we created a soap based webservice that gives you the mensa card for the current day and that is able to let you communicate about the food with the rest of the course.

Solution:

a) The namespace is: de.tu−darmstadt.e−technik.kom.kn2.mensa−soap−webservice

b) The implemented procedures are:

- getTodaysMenus,
- getComments,
- downvote,
- addComment and
- upvote

c) The procedure upvote takes the parameter fid which is an integer.

d) fid stands for Food ID, its a unique id for a meal.

Problem 9 Implementing a soap client

It's now time to implement a client for the webservice. Our client will be implemented using Spyne and Python. However, all other SOAP Libraries could be used, to solve this exercise. To get an idea how such an soap webservice works and how to build a small client in python, we recommend to take a look at the helloworld description[1]. Now its up to you, your amount and time and your motivation to implement a nice working client. Choose a programming language of your choice (python is really easy but others are also very welcome) to create a web service client.
If you created a cool terminal-, smartphone-, web- or desktop application that you want to share with the others feel free to present it at the exercise discussion, we will let the service run at least to the end of the semester.

Solution:
Check the attached code set.

[1] http://spyne.io/docs/2.10/manual/02_helloworld.html