Distributed Systems

Client-Server Communication: Exercises

Werner Nutt

Request-Reply Protocols

Communication Types

- Asynchronous: sender continues after submission

- Synchronous: sender is blocked until
  - message is stored at receiver's host
  - message is received
  - reply is received

Request/Reply Protocols

Basically all client/server communication follows the pattern of a request/reply protocol:

- the client *sends* a request message
- the server *executes* the requested operation
- the server *responds* with a reply message

1. Give an example of a request/reply protocol in the real world.

2. Discuss the pros and cons of synchronous and asynchronous communication for a request/reply protocol. Under which conditions is synchronous communication preferrable? When asynchronous communication?

Request-reply Message Structure

Imagine a request-reply protocol where a request is an invocation of a method of a remote object.

Suppose requests have the following format:

messageType: 0	*bool (0=Request, 1= Reply)*
requestID	*int*
objectReference	*RemoteObjectRef*
methodId	*int or Method*
arguments	*array of bytes*

Request-reply Message Structure (cntd)

3. Under which circumstances is a requestID needed? (Hint 1: remember that messages are sent over networks;
   Hint 2: remember Question 2)

4. What elements should a reply message have?

Request-reply at the Transport Level

5. Which transport protocol would be more suitable to implement a request reply protocol, UDP or TCP?

6. Does the answer depend on additional circumstances?

7. Over which transport-level protocol is the request-reply protocol implemented that you mentioned as answer to Question 1?

Datagram-based RRP

Suppose a Request-Reply Protocol is implemented using UDP.

8. What can go wrong in a message exchange?
   Describe 3 possible problems.
   (Think of message loss and its consequences.)

9. For each problem above, describe a (simple) refinement of the basic request-reply protocol that overcomes it.

Invocation Semantics

For request/reply protocols, one distinguishes between three semantics that are distinguished by the guarantees given for the execution of the *server's operation*:

- Maybe Semantics: the server may execute the request once, several times, or not at all
- At-least-once Semantics: the server executes the request at least once, but may execute it more often, until the client receives an answer
- At-most-once Semantics: for each request, the server executes the operation at most once; the invoking application receives the result or an exception

Invocation Semantics (cntd)

11. Explain: for which of the three semantics is it necessary that the client transmits requests more than once?

12. Explain: for which of the three semantics is it necessary that the server remembers the requests it has answered?

13. For each of the three semantics, explain what the server should do when it receives a request.
    Hint: Find out, whether the server should remember something, and if so, what.

Idempotent Operations

14. Suppose a server receives the same client request more than once. How should it react the second time?

An operation of the server is called “idempotent” if it leads to the same result, independent of how many times it is executed.

15. When running a request/reply protocol, does it make a difference whether server operations are idempotent or not? Does this depend on the invocation semantics? Explain your answer.

Request-Reply with Acknowledgements

In the Request-Reply-Acknowledgement (RRA) protocol the client acknowledges the server’s reply messages, and the acknowledgement message contains the ID in the reply message being acknowledged.

16. For which of the three invocation semantics does RRA allow for an optimisation? Is the optimisation on the client or on the server side?

    Describe how the implementation of client and/or server changes for this semantics when moving from RR to RRA.

Request-Reply with Piggy-Backed Acknowledgements

17. Design a variant of the RRA protocol in which the acknowledgement is piggy-backed on, that is, transmitted in the same message as, the next request where appropriate, and otherwise sent as a separate message. Which changes are necessary on the server and on the client side?