TDDD25
Distributed Systems

Exam Training

Christoph Kessler

IDA
Linköping University
Sweden

2024

LINKÖPING UNIVERSITY

Exam Training

- The example questions on the following slides are taken from the regular exam in March 2023.
  - In general, no ready-made complete answers, but hints.
  - Some solution proposals are given orally or on whiteboard.
  - The exams for 2024 will, by and large, be similar in style.
  - General exam hints on the last slide.

Exam March 2023, Question 1

1. (8 p.) **Distributed Systems Concepts**

   (a) Define *access transparency* and *replication transparency* in distributed systems. (2p)

   (b) Name and shortly describe at least 2 different kinds of *heterogeneity* that can be encountered in distributed systems. (1p)

   (c) What is the main technical principle to achieve portability of application software in spite of heterogeneity in distributed systems? (1p)

   (d) What is a *mobile agent* (in the context of a distributed computer system)? (1p)

   (e) How does a *publish-subscribe* communication system work? Which main components does it consist of, and what do they do? Be thorough! (3p)

- See the slides of the first 2 lectures for the answers.

Exam March 2023, Question 2

2. (8 p.) **Client-Server Communication**

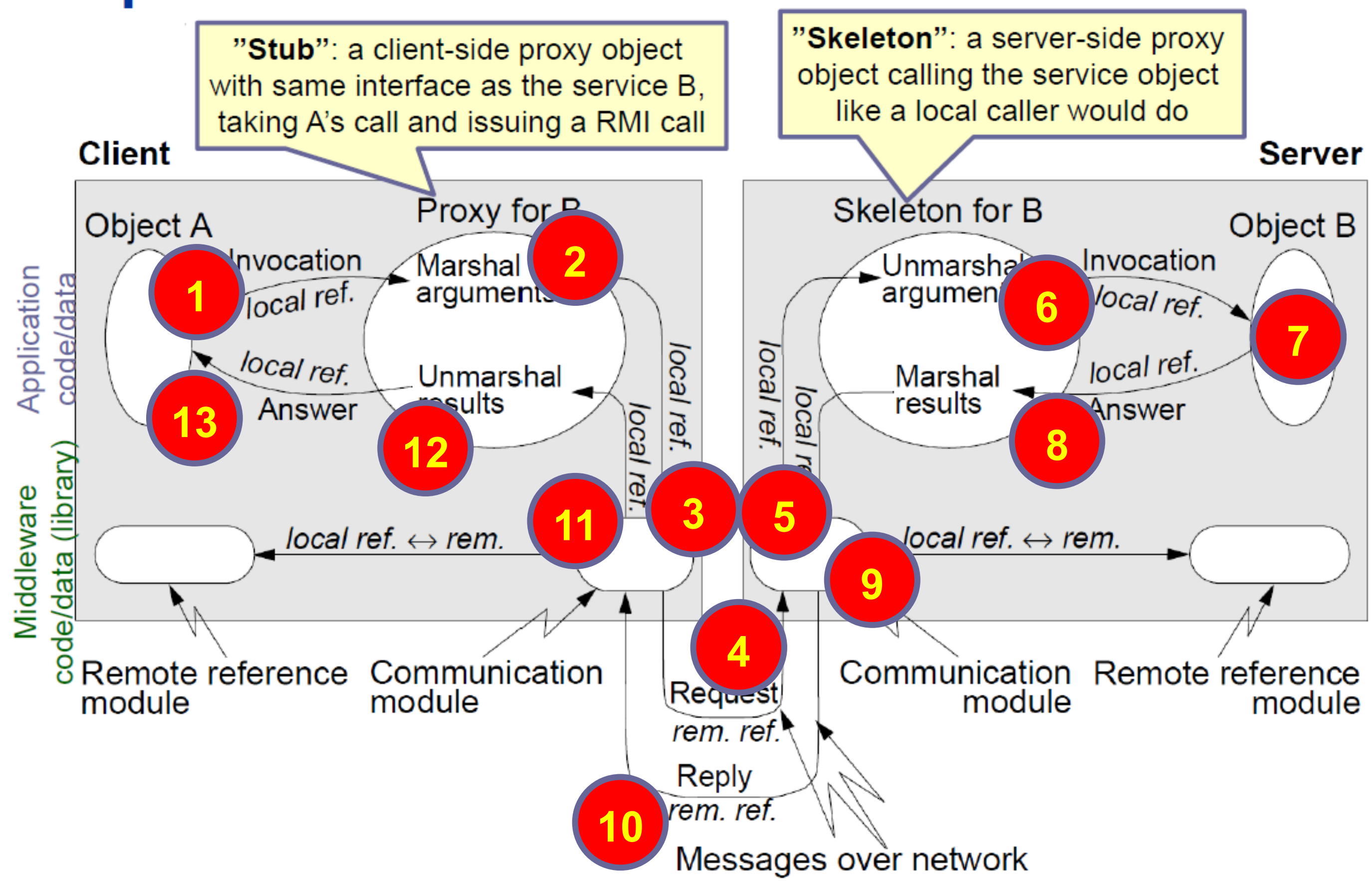
   (a) Describe the main flow of communication in remote method invocation between a client function and a service function. Which software parts are involved, what do they do, and when? Be thorough. (3p) →

Similar: Question 4 from Sample exam 1

- Remote Method Invocation: trace the way of a request and of the reply from the client to a remote server and back. Illustrate with a figure.

(3p)

Implementation of RMI



Exam March 2023, Question 2

2. (8 p.) **Client-Server Communication**

   (a) Describe the main flow of communication in remote method invocation between a client function and a service function. Which software parts are involved, what do they do, and when? Be thorough. (3p)

   (b) What is the purpose of the *Interface Definition Language* in CORBA? How is it processed? (2p)

   (c) In a CORBA-like system for remote-method invocation, what does the programmer need to change in the client (caller) function if the service component happens to reside on the *same* computer as the client? Justify your answer. (1p)

   (d) How can *exactly-once* semantics be achieved in the case of lost messages (assuming that the server never crashes)? (2p) →

Similar: Question 2 in Sample exam 1

Define the following three possible semantics for remote procedure calls:

a. At-least-once semantics

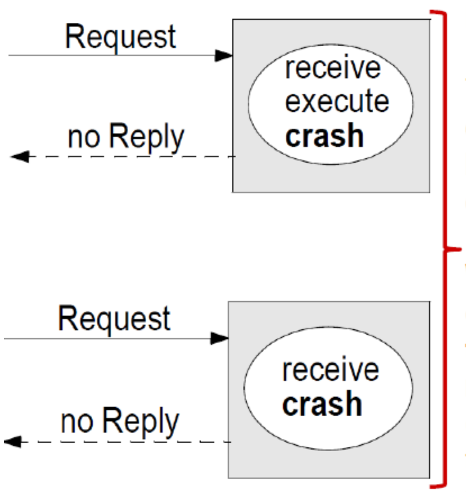
b. At-most-once semantics

c. Exactly-once semantics.

Is it possible to achieve exactly-once semantics in the case of lost messages? But in the case of server crashes? Explain.

(3p)

Summary - RMI Semantics and Failures



Big problem!

The client cannot distinguish between these cases.

What to do if the client noticed that the server might be down? (it did not answer to repeated requests)?

- If the problem of errors is ignored, **maybe-semantics** is achieved for RMI:
  - The client, in general, does not know if the remote method has been executed once, several times, or not at all.
- If server crashes can be excluded, **exactly-once semantics** is possible to achieve by resending requests, filtering out duplicates, and using history.
- If server crashes **with loss of memory** are considered, only **at-least-once** and **at-most-once semantics** are achievable in the best case.

LINKÖPING UNIVERSITY

Exam Training

- The following example questions are taken from the **sample Exam 1** (by P. Eles) on the course web page.

  - **Note**: Points as set in the pre-2023 sample exams might not be indicative for my exams.

  - General hints on the last slide.

Sample exam 1, Question 1

- Synchronous and asynchronous distributed systems.
  What are their main features,
  and what are the consequences of these features?

(3p)

Synchronous Distributed Systems

- **Main features:**
  - Lower and upper bounds on execution time of processes can be set.
  - Transmitted messages are received within a known bounded time.
  - Drift rates between local clocks have a known bound.

- **Important consequences:**
  1. In a synchronous distributed system, there is a notion of **global physical time** (with a known relative precision depending on the drift rate).
  2. Only synchronous distributed systems are **predictable** in terms of **timing**.
     - Only such systems can be used for **hard real-time** applications.
  3. In a synchronous distributed system, it is possible and safe to **use timeouts in order to detect failures** of a process or communication link.

But ...

- **It is difficult and costly to implement synchronous distributed systems.**

Sample exam 1, Question 1

- Synchronous and asynchronous distributed systems.
  What are their main features,
  and what are the consequences of these features?

(3p)

Asynchronous Distributed Systems

Many distributed systems (including those on the Internet) are **asynchronous**:

- No bound on process execution time
  (nothing can be assumed about speed, load, reliability of computers).
- No bound on message transmission delays
  (nothing can be assumed about speed, load, reliability of interconnections)
- No bounds on drift rates between local clocks.

Important consequences:

1. In an asynchronous distributed system, there is no global physical time. Reasoning can be only in terms of **logical time**.
2. Asynchronous distributed systems are unpredictable in terms of timing.
3. No timeouts can be used.

Asynchronous systems are widely and successfully used in practice

Sample Exam 1, Question 3

- Static and dynamic invocation in CORBA:

  How do they work? Compare.

  (3p)

Static and Dynamic Invocation

CORBA allows both static and dynamic invocation of ob

- The choice is made depending on how much informa concerning the server object, is available at compile t

Static Invocation

- Static invocation is based on compile time knowledge server's interface specification. This specification is fo IDL and is **compiled** into a proxy (client stub) code ir programming language in which the client object is e
- For the client, an object invocation is like a local invo proxy method. The invocation is then automatically fo object implementation through the ORB, the object a skeleton.
- Static invocation is **efficient** at run time, because of t low overhead.

Static and Dynamic Invocation

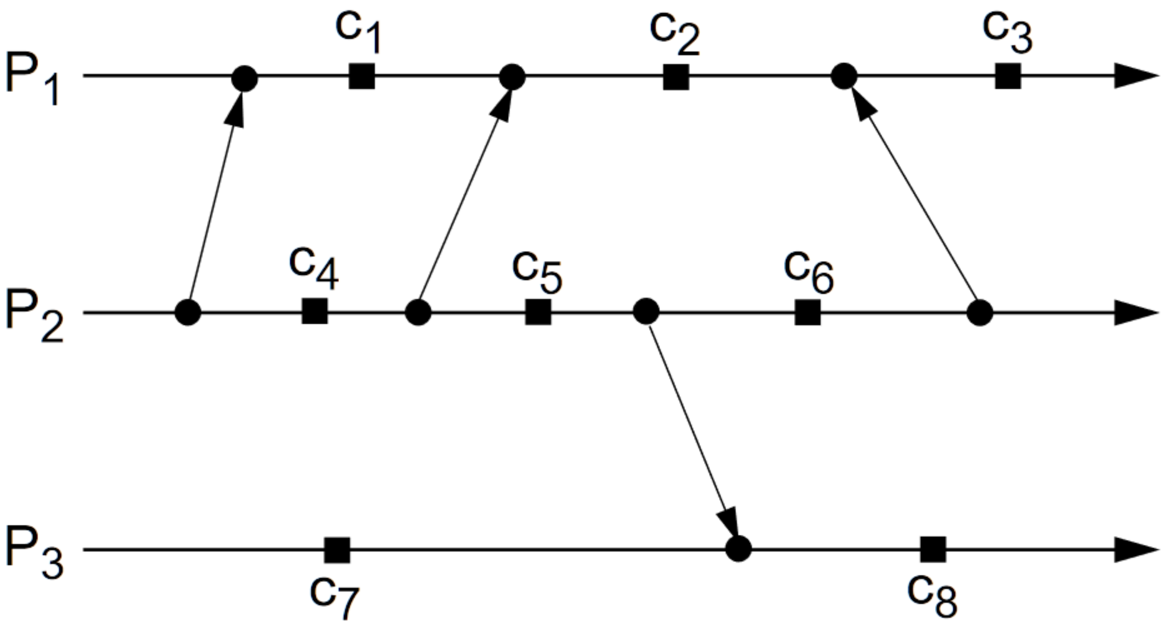
Dynamic Invocation

- Dynamic invocation allows a client to invoke requests on an object without having compile-time knowledge of the object's interface.
- The object and its interface (methods, parameters, types) are detected at run-time.
- The **dynamic invocation interface** (DII) allows to inspect the interface repository and **dynamically construct invocations** corresponding to the server's interface specification.
  - It is a **generic** stub, like an **interpreter** in contrast to to the compiled fixed-function stub for static calls
- The execution **overhead** of a dynamic invocation is **huge**.
- Once the request has been constructed and arguments placed, its invocation has the same effect as a static invocation.
  - From the server's point of view, static and dynamic invocation are identical; the server does not know how it has been invoked.
  - The server invocation is always issued through its skeleton, generated at compile time from the IDL specification.

Sample Exam 1, Question 5

(similar question in March 2023 exam)

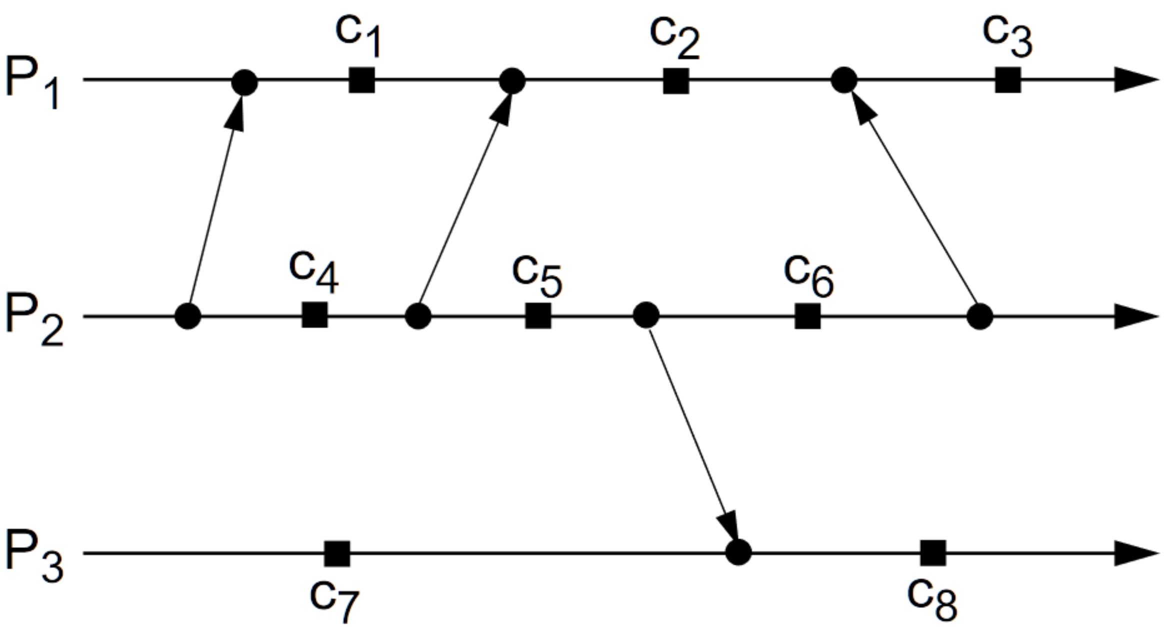
- What is a *cut* of a distributed computation?
- What means a *consistent* and a *strongly consistent* cut?
- Consider the following set of events:



Sample Exam 1, Question 5

(similar question in March 2023 exam)

- What is a *cut* of a distributed computation?
- What means a *consistent* and a *strongly consistent* cut?
- Consider the following set of events:



Determine for each of the following cuts if it is: **inconsistent**, **consistent** or **strongly consistent**:

- $\{c_2, c_6, c_8\}$, s.c.
- $\{c_1, c_4, c_7\}$, s.c.
- $\{c_1, c_5, c_7\}$, c.
- $\{c_1, c_6, c_8\}$, c.
- $\{c_1, c_6, c_7\}$, c.
- $\{c_3, c_6, c_8\}$ i.

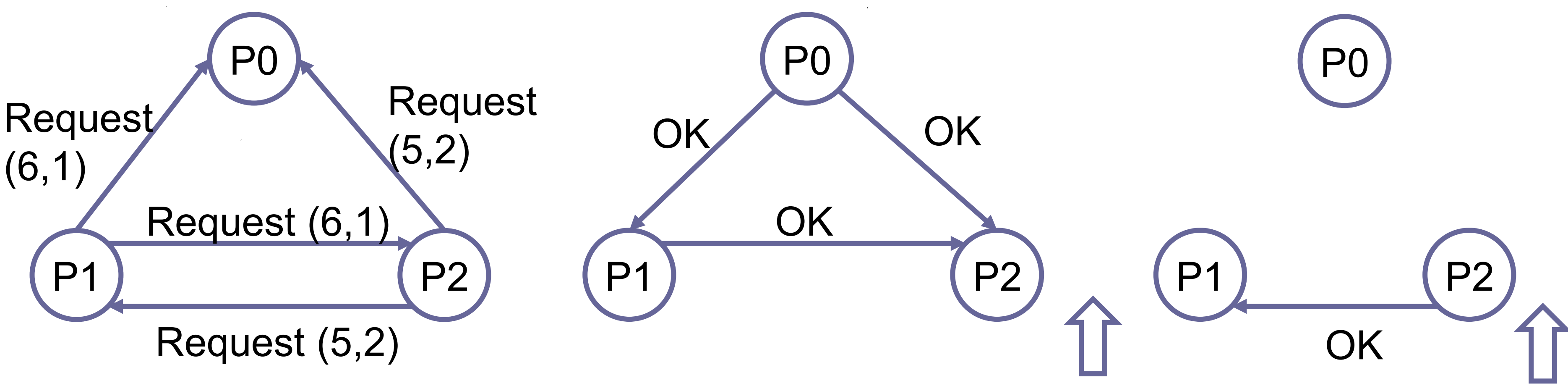
- A cut is *consistent* if every message that was *received before* a cut event was *sent before* the cut event at the sender process.

Strongly consistent cut: a consistent, transitless cut

Sample Exam 1, Question 6

- Consider mutual exclusion with the Ricart-Agrawala algorithm (the *first* algorithm, not using a token).
  Imagine three processes: P0, P1, and P2.
  P1 and P2 are requesting the same resource, and the timestamp of the requests is (6, 1) and (5, 2) respectively.
  Illustrate the sequence of messages exchanged (use figures).
  Who gets the resource first?

  (3p)



P2 gets access first (due to lower logical time of request), then P1

Sample Exam 1, Question 8

- Consider a bully election with 6 processes, P1, ..., P6.
  P6, the current coordinator, fails and P3 starts the election.
  Illustrate the sequence of messages exchanged (use figures).

  (3p)

(on whiteboard)

Sample exam 1, Question 9

- Explain the following types of redundancy:
  - Time redundancy
  - Hardware redundancy
  - Software redundancy
  - Information redundancy

  (3 p)

(See slide set on fault tolerance)

Sample exam 1, Question 10

- What is the basic idea with voting protocols for updating replicated data?
- How do they work?
- Consider a set of 11 replica managers. Define two voting protocols. One for a situation when the number of writes is relatively large compared to that of reads, and the other for the reverse situation.
  Give examples of read and write quorums (use figures).

  (3p)

Sample exam 1, Question 12

- Cristian’s algorithm for clock synchronization.
  Describe how it works.
  How does it estimate the time at the receiver?
  What is the accuracy of this estimation?

  (4p)

(see slide set on real-time distributed systems, time)

Sample exam 1, Question 13

- You know the maximum drift rate of the clocks on two processors and the maximal allowed skew between them. How do you determine the maximum interval between two successive synchronizations between the clocks?

(2p)

(see slide set on real-time distributed systems, time)

General Hints for the Exam

- **Read** each question carefully and *completely* before you begin to answer it.
- After having written down your answer, **re-read** the question to make sure you have answered it completely (with all sub-questions)
- **Explain** your pseudocode and calculations.
- **Justify** your reasoning.
- If you need to make additional **assumptions**, write them down.
- **Time plan** (for my 40p exams): About 6 min per point, if you attempt to answer all questions (recommended – 21p are needed to pass)
- **How much to write?** No general rule, but as a hint:
  - Questions for 0.5p can usually be answered within 1 line of text
  - Questions for 1p usually require a few lines of text, sometimes a simple commented drawing
  - Complete answers to questions for 3-4p usually require a calculation, drawing, and/or ~1 page of text.