COURSE DESCRIPTION CARD - SYLLABUS

Course name

High-Reliability Systems

Course

Field of study

Computing

Year/Semester

1/2

Area of study (specialization)

Distributed and cloud systems

Profile of study

general academic

Level of study

Second-cycle studies

Course offered in

Polish

Form of study

full-time

Requirements

elective

Number of hours

Lecture

30

Laboratory classes

30

Other (e.g. online)

Tutorials

Projects/seminars

Number of credit points

5

Lecturers

Responsible for the course/lecturer:

Michał Szychowiak, PhD

e-mail: Michal.Szychowiak@cs.put.poznan.pl

tel. 61 665 2964

Faculty of Computing and Telecommunications

ul. Piotrowo 3 60-965 Poznań

Responsible for the course/lecturer:

Rafał Skowroński, MSc

email: Rafal.Skowronski@cs.put.poznan.pl

tel. 61 665 2963

Faculty of Computing and Telecommunications

ul. Piotrowo 3 60-965 Poznań

Prerequisites

Student starting this module should have basic knowledge regarding operating systems, computing networks, distributed computing, distributed computing environments (incl. distributes shared memory), databases, and security of computer systems.

Student should understand the need to extend his/her competences. In addition, in respect to the social skills the student should show attitudes as honesty, responsibility, perseverance, curiosity, creativity, manners, and respect for other people.

Course objective

1. Provide students with basic knowledge in the field of dependable processing, mainly in the reliability

of distributed computing, distributed fault detection, masking and non-masking fault tolerance, backward and forward recovery of distributed processing, distributed agreement problems, process replication, and self-stabilization.

2. Develop ability to solve particular security problems of reliability in a fault-prone distributed environment.

Course-related learning outcomes

Knowledge

1. the student has well-established theoretical knowledge regarding algorithms and computational complexity, especially distributed algorithms with high reliability

2. the student has detailed theoretical knowledge related to selected areas of computer science, such as fault tolerance, distributed fault detectors, rollback-recovery, voting, distributed consensus, Byzantine agreement, process replication, group communication, self-stabilization, among others

3. the student has knowledge regarding trends and the most important new developments in computer science and related disciplines, concerning in particular dependability of distributed computing

4. the student has basic knowledge regarding life-cycle computing systems, with special attention on reliability

5. the student knows the basic methods, techniques and tools used to solve problems in the field of reliable distributed systems

Skills

1. the student is able to acquire, combine, interpret and evaluate information from literature, databases and other information sources (in mother tongue and English); draw conclusions and formulate opinions based on it

2. the student is able to plan and arrange self-education process

3. the student is able to employ analytical, simulation, and experiment methods to formulate and solve engineering tasks and basic research problems

4. the student is able to combine knowledge from different areas of computer science (and if necessary from other scientific disciplines) to formulate and solve engineering tasks; and use system approach that also incorporates nontechnical aspects

5. the student is able to formulate and test hypotheses regarding engineering problems and basic research problems

6. the student is able to assess usefulness and possibility of employing new developments (methods and tools) and new IT products

7. the student is able to propose enhancements (improvements) to existing technical solutions

8. the student is able to evaluate usefulness of methods and tools (also to identify their limitations) used to solve engineering tasks, i.e., building IT systems or their components

9. the student is able to solve (using, e.g., new conceptual methods) complex computer science tasks, including non-typical tasks and tasks with research components

Social competences

1. the student knows examples / case studies of data mining and analysis and understands their limitations

2. the student is able to correctly assign priorities to own tasks and tasks performed by others

3. the student is able to think and act in an entrepreneurial way

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:

a) lectures:

- based on answers to question in the written exam,

b) laboratory classes:

- evaluation of doing correctly assigned tasks (following provided lab. instructions).

Total assessment:

a) verification of assumed learning objectives related to lectures:

- evaluation of acquired knowledge on the basis of the written exam,
- discussion of correct answers in the exam .

b) verification of assumed learning objectives related to laboratory classes:

- evaluation of student's knowledge necessary to prepare, and carry out the lab tasks,
- monitoring students' activities during classes.

Additional elements cover:

- discussing more general and related aspects of the class topic,
- showing how to improve the instructions and teaching materials.

Programme content

The lecture program covers the following topics:

The basic classification of dependability issues, dependability attributes, dependability threats and means of achieving dependability. Failure models, fault-tree, reliability measurement. Methods of forward and backward recovery of distributed processing, local and global checkpoints, the role of message semantic knowledge and determinism in optimizing the recovery. Consistency conditions of communication channels. Restoring a distributed processing state by successive rollbacks, domino effect. Recovery line and the problem of output commit. Coordinated (synchronous) checkpointing: methods and sample algorithms. Independent (asynchronous) checkpointing: methods and sample algorithms. Optimistic and pessimistic message logging. Reducing the cost of message logging. Checkpoint garbage collection. Hybrid-recovery algorithms, adaptive checkpointing, and a quasi-synchronous recovery. Restoring a distributed shared memory state. Atomic operations and reliable distributed transaction commitment. Mechanisms of static and dynamic reliable voting. Basic problems of distributed agreement and formal limitations on their solvability. Binary consensus. Relations between various agreement problems. Solutions for the distributed consensus with different failure models. Solutions for agreement with Byzantine failures. The use of cryptography in the Byzantine agreement. The use of active and passive replication processes in distributed fault-tolerance. Group communication mechanisms required to implement replication and their reliable implementation. Asynchronous systems with distributed failure detectors, solving agreement problems with distributed failure detectors. The stabilization and self-stabilization of distributed systems. Classes and limits of stabilization. The use of self-stabilizing algorithms in computer networks.

The lab-classes include the following topics:

Fault tolerance with High-Availability Clusters: configuration and management of sample HAC systems and technologies – ClusterIP, Linux Virtual Server, Pacemaker, DRBD. Determination of a recovery line in a distributed processing. The use of message logging mechanisms in the recovery of processing state. Using garbage collection in asynchronous checkpointing. Unreliable communication channels in a sample problem of distributed mutual exclusion. The design of a distributed system tolerating communication failures for the selected distributed mutual exclusion algorithm. Misra's distributed algorithm for the lost token detection in the ring. Extension of the Misra's algorithm for nonFIFO channels. Application of the non-blocking 3PC protocol for unreliable processes. Self-stabilizing graph algorithms in asynchronous distributed environment.

Teaching methods

1. Lectures: multimedia presentation, presentation illustrated with examples and showcases

2. Labs: practical exercises, discussion, teamwork

Bibliography

Basic

1. Klaus Schmidt, "High Availability and Disaster Recovery: Concepts, Design, Implementation", Springer, 2006

2. Kenneth P. Birman, "Guide to Reliable Distributed Systems", Springer, 2012

3. Floyd Piedad, Michael Hawkins, "High availability: design, techniques, and processes", Prentice Hall, 2001

4. Michel Raynal, "Distributed algorithms for message-passing systems", Springer, 2013

Additional

1. Ajay D. Kshemkalyani, Mukesh Singhal, "Distributed Computing: Principles, Algorithms, and Systems", Cambridge University Press, 2008

2. Sander van Vugt, "Pro Linux High Availability Clustering", Apress, 2014

3. Daniel J. Sorin, "Fault Tolerant Computer Architecture", Synthesis Lectures on Computer Architecture No.5, 2009

4. Jerzy Brzeziński, Michał Szychowiak, "Self-Stabilization in Distributed Systems – a Short Survey", Foundations of Computing and Decision Sciences, vol. 25, no. 1, 2000

5. A.A. Helal, A.A. Heddaya, B.B. Bhargava, "Replication Techniques in Distributed Systems", Kluwer Academic Publishers, 1996

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,0
Classes requiring direct contact with the teacher	60	2,5
Student's own work (literature studies, preparation for laboratory classes/tutorials, preparation for tests/exam, project preparation) [1]	65	2,5

[1] delete or add other activities as appropriate