Distributed Systems **Autumn 2025**

Course Project

Project Description

The project should be carried out in groups of five people. Groups are free to organise and split the project work among members as they please as long as each member significantly contributes to the project.

Grading

The project contributes to 25% of the final grade and will be graded in the following manner:

- 65% Code functionality (relevance of the implementation and challenges solved, ease of use, results and analysis);
- 10% Project report (contents and writing);
- 25% Project presentation (contents and presentation skill).

Project Deliverables

The following items must be submitted to iCorsi by only one of the group members:

- A zip file containing the project's source code. See below for available projects.
- Report in PDF format. The report should describe the theme, usage instruction, implementation, and the challenges faced during development (max. 2 pages). It should be in single column format with font size not smaller than 10pt, and should contain the name of all the group members.

Project Presentation

A 10 minutes presentation must be made after the project submission (see schedule on iCorsi). The presentation must describe the project goal and must contain a short live demo of the Dapp. You can use prepared slides and the whiteboard during your presentation. All group members must present in-person for the presentation, we suggest at most 2 presenters per group.

Project list

Below you can find suggested projects. Project 1 is the default project with pre-defined deliverables aimed to students with limited programming experience and/or students interested in blockchains applications, decentralized finance. Project 2 and 3 are research-oriented and require a degree of programming expertise and independence. Groups undertaking these projects will engage with distributed system research and have the possibility of contributing to research projects and scientific articles. Due to the challenging nature projects 2 and 3, we will consider a good understanding of the topic and a display of significant development and evaluation attempts sufficient to get full grades in the project, even in the absence of results. The course also offers the possibility of coming up with your own project as long as the theme is relevant to distributed systems and to the material covered in the course.

1. Internet Computer decentralized application (Dapp)

This project consists in the development of a decentralized application (Dapp) on the Internet Computer ecosystem. The theme of the Dapp can be chosen freely, but must not lead to a trivial implementation; consult the TA for theme suitability.

The source code should contain a full-fledged Internet Computer Dapp. The app must contain a backend canister providing data manipulation functionalities and we strongly encourage including a frontend canister (points might be deducted for backend-only apps without enough functionality to justify the absence of a frontend).

The source code must contain a `README.md` file with

- A brief description of the Dapp including its status, e.g., whether the app is fully working or some feature are missing.
- (optional) A link to the ICP Ninja or GitHub project.
- Clear installation instructions for *all* pre-requisites (including e.g. NodeJS, IC SDK, etc.). Please include where the application was tested (Linux/MacOS) and platform-specific steps.

After installing the pre-requisites, the Dapp should build and deploy to the IC playground with the following command:

```
dfx deploy --playground
```

If the application is incomplete please include additional build instructions in the `README.md` (grade points might be deducted).

2. Datacenter algorithm simulator

Distributed coordination protocols (e.g. consensus, state machine replication) are at the core of countless services such as banking, distributed databases, configuration services, consistent frameworks for artificial intelligence training, etc. This project's goal is to build a flexible simulation framework that allows the implementation and evaluation of distributed replication protocols with real datacenter settings. The purpose of the simulator is to allow quick prototyping and evaluation of existing and novel replication algorithms before experimenting on real systems, which is often costly and time-consuming.

A good starting point is the *Dissecting the Performance of Strongly-Consistent Replication Protocols* article in which authors describe the performance evaluation of different Paxos (a popular partially synchronous consensus protocol) variants using the *Paxi* framework. This project's framework should strive to have the following features:

Simple message passing interfaces Protocols should be implementable with message passing interfaces rather than e.g., quorum systems in Paxi. A user should be able to implement a protocol specifying the number of nodes and the protocol behaviour of each node using `send` and `receive` primitives.

Parameters As in Paxi (see Table 2), the simulator should offer a number of parameters to model datacenter endhost and network behaviour. We suggest to start with working version with very high level settings, e.g., generic single-latency values for endhost processing and network communication delay, and add more details as the project progresses, e.g., per-switch latency, queuing model, congestion.

Accurate datacenter interactions The simulator should focus on datacenter protocols and not wide-area network as in Paxi. The simulator parameters, e.g. switch packet processing latency, should reflect the performance of real datacenter hardware.

Synchrony protocols Unlike Paxi, the simulator should allow modeling synchronous protocols which rely on a fixed interaction latency between processes. For instance, the simulator could offer a `sync_send` method which takes a probability of synchrony being violated as a parameter.

Some protocols that we would like to simulate on the framework for reference: Primary backup replication, Paxos (see lecture slides + large number of online resources), Looped one way imposition (LOWI) - algorithm 3.

3. Benchmark state-of-the-art RDMA consensus protocol

uKharon (paper / code) is a state-of-the-art group membership protocol in C++ leveraging Remote Direct Memory Access (RDMA) technology to achieve μs-level consensus, view-change and replication. The project consists in understanding, dissecting and evaluating uKharon's consensus protocol on real hardware using CloudLab – an open research cluster. The objectives of this project are open-ended and need to be iteratively refined with the course instructors. The project might lead to the development of a custom-made protocol for comparison with uKharon. This project will allow you to gain low-level system experience on challenging ultra-efficient modern coordination protocols, as well as giving you an insight of research methodology.