Guidelines for Protocol Development Project

This document describes the milestones and evaluation criteria for the protocol development project and gives some advice and guidelines to help you forward.

In the first meeting, you and your team should go through the assignment description and the evaluation criteria. If you have questions regarding the requirements, please do not hesitate to contact the course personnel. It is strongly recommended that you agree on a regular meeting time (at least once a week) with the team, and that in every meeting you ensure that the responsibilities, tasks and next steps are clear for everyone.

There should be a git repository for the project implementation, with read access to the course personnel. It is recommended to use ***version.aalto.fi*** for this, but GitHub is also ok if you prefer so.

Project teams are announced on Friday, February 2nd, but if you have a team ready, you can start the work right away. The deadline for submitting the team and topic form is at the end of Thursday, February 1st.

Recommended phases of work

Step 1: Discuss the use cases with the team. The use cases define how users would interact with the app. You can use the following template to describe use cases. You should be able to clearly define entities and their roles in the system after this step.

Use case name:
Actors involved: (e.g. client A, client B, server and etc.)
Preconditions:
Steps:
The client xxxx
xxx
xxx
Error:
e.g. If xxx does not xxx, xxx will be xxx

Post-conditions:

Step 2: Go through the use cases and summarize the functional requirements. Note that use cases do not typically describe non-functional requirements and constraints. You can identify non-functional requirements by analysing the operational environment (e.g. network environment, potentially large number of users, etc.), the expectations from user experience perspective (e.g. low latency, high resolution), security (e.g. data encryption, join approval), and any other relevant concerns.

Step 3: Architecture design. You could consider, for example, whether you apply a layered architecture for your system and protocol design, and whether you want to follow a distributed or centralized architecture in your system. You should consider the service provided by the protocol (or

protocols) you will use in your system, and assumptions about the environment your protocols are being executed.

Checkpoint 1 (Friday, 16.2.2024): Submit a summary of functional/non-functional requirements, a draft of the architecture design, and a brief description about the next steps. Submit a presentation (PowerPoint is sufficient) by 16.2.2024 and present it in a peer review session on **Tuesday, 20.2.2024**. Feedback will be given to each group.

Step 4: At this step, you start to define messages to be exchanged between entities. One way is to draw a message sequence diagram for each use case and define the message format used in the messages. It is also recommended to draw finite state machines for each entity. For example, if a protocol is about sending a file from A to B, then you can draw one state machine for the sender and another one for the receiver. Protocols must be prepared to deal appropriately with every feasible action and with every possible sequence of actions under all possible conditions.

When working on your design, analyse how the related protocols work in other similar systems (as referenced in topic descriptions), and try to think if you can make a better design in some respects in your solution. You can choose to first make your own designs, and then compare with previous ones. Alternatively, you can first study the existing solutions, identify their limitations, and try to design a better one. Note that it is not acceptable if you simply copy the design of any existing protocol/software.

Step 5: Implement and test the protocols. This can be done without actual user interface or developing a very simple user interface (e.g. command line). You should check if the state machines have been implemented correctly, and test the performance, efficiency and scalability (e.g. latency and traffic size when the number of clients increase).

Step 6: Implement the application logic using the designed protocols. The user interface can be simple, the focus on this course is on communication and protocols.

You may want to iterate between steps 5 and 6: you may notice during application implementation that your protocols have shortcomings and faults, or you may just want to improve something in the protocols as you learn more.

Checkpoint 2 (Wednesday, 6.3.2024): The main functions of the protocols and the application should be ready for testing. Ensure that the current version is pushed to git repository and notify course personnel about the progress.

Step 7: Test the application and demonstrate it to the course personnel. If you want to test your implementation in more challenging network environment, you can use e.g. Mininet to create a virtual network topology in your local machine with simulated delay and packet loss. Mininet is available only for Linux.

Step 8: Write the final report.

Summary of deliverables

- **Present a demo** to course personnel and ensure that **code is up to date in the git repository** by **Wednesday, March 20**
- Submit **group report** to MyCourses by **Friday, March 22**. See report template below.

- Submit **individual report** (1 – 2 pages) to MyCourses by **Friday, March 22** on what you have learnt from the project. You should discuss four aspects:
    1) What were the most important decisions you and your group made about the communication solutions in your application?
    2) What were the most difficult and challenging parts during the design and implementation phase?
    3) How did you find information and help about the communication solutions and implementation? For example, did you use RFCs or scientific publications, Google or perhaps ChatGPT?
    4) What was your contribution in the team? How would you characterise the team's functionality overall?

Group report template (10 – 20 pages)

Cover page

- name of the project
- Name and student number of each group member

Chapter 1: System architecture (1-2 pages)

- Draw a figure that illustrates the high-level architecture of the system. Add a brief description of the figure.
- List the functional requirements. If the architecture includes several layers of protocol, you need to explain the functionality of each layer and the interfaces between adjacent layers.
- List non-functional requirements.

Chapter 2: Design (3 – 8 pages)

- State machines
- Messages and their formats
- Message sequence charts
- Underlying communication framework: what other protocols your solutions is built on and why? For example, are you using TCP or UDP (or something else), or are you using some existing protocols for security? Provide references.
- Discussion and comparison with other communication solutions known to solve similar problems. Provide references.

Chapter 3: Implementation and Evaluation (3 – 8 pages)

- Implementation overview: software modules, system requirements, needed software libraries, instructions on building and using the software.
- Description of experimentation setup: test environment, methodology, tools
- Description of different test cases (workloads, network environments, etc.)
- Results analysis (e.g. throughput, delays, reliability related attributes)
- Known shortcomings and ideas for improvements.

Chapter 4: Teamwork (0.5 pages)

- Team processes during the project (e.g., meeting arrangements, what channels were used for communication)
- Responsibilities of each team member

Evaluation matrix

Total maximum is 50 points.

A. Interim report submitted at checkpoint 1. (5 points)

Topic (weight)	Unacceptable (0)	Marginal (1)	Acceptable (2)	Exceptional (3)
Requirement analysis (1)	Little or no grasp of the problem.	Some understanding of the problem. Some use cases are defined.	Overall sound understanding of the problem. Most use cases are clearly defined.	A list of functional and non-functional requirements is provided.
Project management (1)	Next steps are not planned.	Next steps are clearly defined.	Next steps are clearly defined. The schedule and task distribution are well planned.	

B. Implementation (19 points)

Topic (weight)	Unacceptable (0)	Marginal (1)	Acceptable (2)	Exceptional (3)
Features (2)	Software does not implement any of the required features from the topic description	Software implements only few of the required features, or the implementation is very buggy	Software implements most of the required features, or there are some defects in the implementation	Software implements all required features, there are no major problems
State management (1)	Different users cannot see consistent system state (e.g. in editing collaborative document)	Consistent system state can be seen, but it needs special actions (e.g., manual update by user), or is malfunctioning at times	System state is updated in real-time for all users, also after leaving and joining session.	In addition, system can handle cases of simultaneous modifications by different users on the system
Communication performance (1)		System supports transfer for larger objects (files, images)	Uploading or downloading a large object does not unnecessarily interfere with other users' interaction with the system	In addition, system can handle multiple simultaneous upload/download sessions of large objects
Exception management (1)		Basic error conditions are handled (connection refused, host unreachable)	If one client dies unexpectedly, the system remains operational for other users	
Session management (1)	Users cannot join the system	Multiple clients can join and leave the system at any time.	All clients have a real-time information about the other users within the same	Application supports multiple separated sessions that maybe created

			session. Users are identified by a name.	and removed dynamically (e.g., different collaborative documents or chat teams)
Security (1)			Communication is encrypted	

C. Final report (23 points)

Topic (weight)	**Unacceptable (0)**	**Marginal (1)**	**Acceptable (2)**	**Exceptional (3)**
Architecture design (1)	Little or no grasp of the problem.	Entities and their roles are clearly defined.	The service provided by each protocol is properly defined. Structures (e.g. layering, if applied) and role of interfaces between them are defined.	Architectural design choices are explained and justified. Design alternatives are discussed.
Protocol Design (2)	Not capable of achieving the objectives	The design can fulfil most of the functional and non-functional requirements.	The design can fulfil the functional and non-functional requirements.	Additionally, state machines, message sequence charts and/or class diagrams are used for explaining the design.
Evaluation (2)	The design is not evaluated.	Some tests have been done, but the test cases are not sufficient.	Performance in terms of latency, efficiency in terms of traffic size, and scalability are evaluated.	Illustrate how the design fulfils the non-functional requirements using the experimental results.
Discussion (2)	The other related work is not discussed in the documentation	Own design is compared with at least one other alternative protocol. Differences in the design are identified.	Additionally, the limitations of own design are discussed.	
Academic Writing (1)	The report is difficult to follow. Many grammar errors in the text. A lot of text is irrelevant to the topic.	The report is otherwise easy to follow, but some important details are missing.	The report is easy to follow, and the ideas are well expressed. The paper is well written, except for a few places which require clarification.	Report is well written and concise. Ideas are well expressed. Report is well organized and easy to follow. Plots and diagrams are understandable,

				and they support the text.

D. Teamwork (1 points)

Topic (weight)	Unacceptable (0)	Acceptable (1)
Teamwork (1)	Tasks are not assigned to each member. The group does not manage to complete the tasks together.	Responsibility and workload are equally distributed between group members in fair way.

E. Individual report (2 points)

Topic (weight)	Unacceptable (0)	Acceptable (1)	Exceptional (2)
Reflection (1)	Individual report does not address the required aspects	Only some of the required aspects are discussed	All the required aspects are discussed

Grade limits and points

The project points scale to course grading as follows. Points from lecture quizzes are added to project points. For passing grade, there should be no “unacceptable” rows in grading matrix, regardless of total points.

Grade	Points
5	44 – 50
4	38 – 44
3	32 – 38
2	26 – 32
1	20 – 26
0	0 – 19

Change history

- 31.1.2024: Minor modification in evaluation matrix.
- 13.3.2024: Page counts in template guidance adjusted. Points vs. grading table added.