Distributed System Final Project Instructions

The goal of your final project is to design, build and test a working real-world distributed system, or to experiment with distributed algorithms in a suitable simulation. Consider the example ideas below, but you can propose any project idea.

You should select a project such that a prototype can be completed by beginning of November and can be iteratively improved during Nov/Dec. The aim is to produce a high-quality mature system by the end of the course. Decide your scope with this in mind.

The project should be done as a team of 2-3 students

Proposal – Literature Review

You can use google scholar or IEEE to find research papers related to distributed systems. Each student should deeply understand the material presented in the papers s/he has undertaken. Most specifically, each student should:

NOTE: You must evaluate information from at least 6 articles and papers for 3 person teams

Each student should deeply understand the material presented in the papers s/he has undertaken. Most specifically, each student should:

- Know the algorithms and the techniques presented in the paper;
- Be able to answer to questions of the style «Why is each line of the code useful in the algorithms s/he will present and what could go wrong if any line was removed»;
- Evaluate the algorithm presented and devise own good/bad scenarios of execution and understand how the algorithms cope with these scenarios
- Study/devise a big number of examples to deeply understand how the algorithms work;
- Understand the high-level idea of the analysis of the algorithms included in the paper;

You should submit a project proposal of 2-3 pages providing more details on the project you are going to implement. your proposal should explicitly state the following:

- A description of the problem
- The motivation for the problem (e.g., why is the problem interesting, why is it challenging, who will benefit from a solution to the problem, etc.)
- A brief discussion of previous work related to this problem.
- Your initial ideas on how to attack the problem. This includes a (rough) methodology and plan for your project including high level architecture, implementation, and deployment
- Be sure to structure your plan into a set of incremental, implementable milestones and include a milestone schedule
- The resources needed to carry out your project, software, hardware, cloud, etc
- Citation and link to resources used or will be used.

ONE submission per team.

Project Timeline

Component	Due Date
Partner Selection	9/26/2024 @11:59pm
Proposal	10/08/2024 @11:59pm
Presentations	12/05/2024 @11:59pm
Final Report	12/06/2024 @11:59pm
Code Submission	12/05/2024 @11:59pm

Project Ideas

You can choose any of the ideas presented below or you should suggest your own project based on the research papers selected.

1. Distributed Social Network
Implementation of some kind of a distributed social network application. Quoting from Wikipedia: "A social network is a social structure made up of individuals (or organizations) called "nodes", which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, common interest, financial exchange, dislike, or relationships of beliefs, knowledge or prestige". All these are parameters specified by each node.

The general idea is the following. Each node must be able to join the social network and exit from the social network. During its presence to the network each node can specify or modify various parameters. The distributed system must be able to correlate nodes with common parameters and form "groups of relevance". Exiting at any arbitrary time from the social network may cause a lot of problems and have several implications, according to the services you will choose to implement and the algorithms you selected to implement them. These problems or implications must be clearly identified and their solutions must be described.

You can choose the functionality of your distributed social network so that it takes into consideration the constraints of the project described below.

2. Library of Collaborating Tools
Implementation of a library of collaborating tools, such as:

- Distributed Calendar (like the google calendar but probably with enhanced functionality)
- Distributed Meeting Arrangement (like doodle but with enhanced functionality)
- Distributed Meeting and Sharing (like facebook). The nodes can chat with each other, form chat-rooms or chat-groups, send messages to multiple recipients, etc. When chatting with each other, it is required the messages to be displayed in the same order in both participants. The same is required when talking to chat-rooms or chat-groups.
- Distributed Editor (like recent version of emacs). Users can create and edit documents, which can be concurrently edited by several other users.

    Choose **some** of the above items so that the result should be a comprehensive project.

3. Middleware
Implementation of middleware that provides **some** of the following functionality:

- Naming service. Obviously, each node has to define a "name" when it enters the network. This service requires implementing a mechanism that permits to some node to find the information of any other node by searching its "name" and this must be implemented in a distributed way.
- Group creation, node joining and exiting, and communication. Maintaining the groups of relevance in a distributed way, when nodes can join and exit the network arbitrarily, is challenging.
- Security service: The security service may include the following: (1) authentication of principals, (2) access control on the reception of remote method invocations, (3) security of communication between clients and servers, (4) facilities for non-repudiation, etc.
- Trading service: Allows the location of objects by their attributes, i.e., it is kind of a directory service. Its database contains a mapping from service types and their associated attributes onto remote memory references of objects. Clients make queries specifying the type of service required, together with other arguments specifying constraints on the values of attributes, and preferences for the order in which to receive matching offers.
- Transaction and concurrency control services.
- Persistent state service.
- Any other functionality you would like to suggest.

4. Publish-Subscribe System
Implementation of a publish-subscribe system. Some node wants to publicly share a piece of information (e.g., to announce an event) with any other member included in the groups of relevance it participates. Implement algorithms that efficiently diffuses the information (e.g., efficient broadcasting, or mobile agents, or gossiping protocols)

5. Distributed Hash Tables
Implementation of a P2P system, structured or unstructured.

6. Distributed Data Structures
Implementation of a library of distributed data structures for distributed memory machines.

7. Consistency and Replication
Implementation of a quorum system where quorums are appropriately formed to guarantee consistency.

A Revision Control System. It is used when a team of people participating to the same project want to concurrently update the same files. Changes may be identified by a number or letter code, termed the "revision". Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged. Revisions can be made either to independent files or to the whole project.

8. Distributed Memory Allocator or Garbage Collection
A distributed memory allocator is a distributed algorithm that efficiently allocates memory requested by different nodes.

The garbage collector attempts to reclaim memory occupied by objects that are no longer in use by the program. Distributed Garbage Collection, is a particular case of Garbage Collection, where references to some object can be held by remote clients. Mechanisms, e.g. leases, exist so that an object can be identified either as useful or as garbage.

Implementation

You can choose to implement any of the projects described above and you are encouraged to propose and implement any other service you want (which is related to the topics of this course, i.e. it incorporates any class of distributed algorithms discussed during the lectures).

The implementation of your project must consider issues such as fault tolerance, performance, scalability, availability, consistency, concurrency and other challenges discussed in class.

Your choice should be so that **at least** three of the following list of algorithms should be implemented in the project:

1. Broadcast and/or multicast through spanning tree construction
2. Mobile Agents
3. Gossiping Protocols
4. Mutual Exclusion
5. Leader Election
6. Distributed Snapshots
7. Timestamps
8. Resource allocation algorithms
9. Concurrency control
10. Resource discovery
11. Agreement protocols
12. Replication and consistency protocols
13. Distributed shared memory protocols

Final Reports

Your team's report should be minimum of 10-15 pages (excluding images and diagrams) using the IEEE double column template, and include at least these sections:

1. Team member names, student IDs and email addresses on cover page
2. Project Goals and motivation
3. Description of Distributed System challenges addressed
4. Detailed discussion of previous work related to this problem (Literature review)
5. Project Design
   a. Describe key design goals in designing your distributed systems i.e how your design allows the system to handle the core distributed systems challenges: Heterogeneity, Openness, Security, Failure Handling, Concurrency, Quality of Service, Scalability, and Transparency.
   b. Describe the key components and algorithms
   c. Describe your architecture and how the components will interact. You must include several UML diagrams to illustrate your design (at least 3 diagrams from below list to illustrate proper overview of the system.
      i. Use Case style diagram to show the overview of your system functionality and different functions in your system.
      ii. Use the Deployment and Component diagrams to show the software and hardware components of your system.
      iii. Communication diagram to illustrate the connections between the components,

    iv. Sequence diagram explains the sequence of actions, events, and processes between the components, objects, and actors in your system.

    v. Activity and State diagrams to explain more detail about a process or object in your system.

6. Evaluation
   a. Evaluate the selected approach and analyze why the selected approach is good? Provide an intuitive description of the algorithms, their correctness and their complexity
   b. How are the following issues addressed in your project
      i. Fault Tolerant
      ii. Performance
      iii. Scalability
      iv. Consistency
      v. Concurrency
      vi. Security
7. Implementation Details
   a. Describe the choice of Architectural Style
   b. You must include several UML diagrams to illustrate your implementation such Class Diagrams, Object diagrams, Interaction diagrams, etc (At least 3 diagrams)
   c. Describe Software components and services implemented and how they interact (i.e RPC/RMI, Web Services, Socket APIs, etc)
8. Demonstration of example system run
   a. Define and explain a successful scenario and include snapshots
   b. Define and explain 3 failure scenarios and include snapshots for each and how your system tackle such failure scenarios
9. Analysis of expected performance (growth rate of major algorithms, etc).
   a. Describe the overall result of your project and reflect on your design decisions.
   b. Define measurements metrics to analyze and understand how it performs or works (ideally in comparison to another system).
   c. Simulate workloads and measure the performance of your system against the metrics you developed above (Example in terms of throughput and latency)
10. Testing results
    a. Develop and explain set of test cases (at least 5)
    b. Report the result of test cases and discuss how well your systems works.
11. Conclusions, lessons learned possible improvements, etc.

Final Project Presentation

A recorded presentation is required for the project. The presentation will approximately be 15 minutes for each team. Your presentation slides should be understandable to anyone who has taken our distributed systems course.

An ideal presentation will:
- Introduce your project and the problem that it solves.
- Briefly present related work/systems.
- Describe the key design choices of your system, with a discussion of how they relate to some of our distributed systems design challenges.
- Overview of components, architecture and features of the system
- Live demo of your running project. Be sure to demo and focus on core distributed systems concepts such as reliability, scalability , fault tolerance, concurrency, etc

For example:
1. Show what happens when you cause a crash.
2. How your system scales as you increase concurrent access to the system
3. How is performance affected as you increase the number of users
- Discuss the overall impact of your design and present some evaluation of how well the system works (use metrics).
- Describe what you plan on doing next
- Conclude your presentation with a summary of your project's key points.

What to submit

These deliverables are due at or before the last class of the quarter

1. Final report.
2. Recorded presentation/demo,
3. Presentation slides used to illustrate your presentation
4. Code submission
5. Readme file containing how to setup/compile your project and step by step instructions on setting up your project
6. Division of work: you are responsible for making it clear which portions each student was responsible for

ONE submission per team.