Distributed Systems (2020)

Programming Task

Design and implement a simple distributed application running on at least three (3) nodes, implementing some applications, such as a multi-player game, a collaboration tool, distributed user interface, distributed data processing for sensing devices, etc. Each application instance exchanges data with at least two other instances to form a shared state distributed across the nodes.

The participating nodes must: 1) be able to exchange messages; 2) be able to express their state and/or readiness for sessions towards other nodes; 3) All nodes must log all important events of their activity either locally or to a central location.

There are no enforced topics, but a list of technical requirements

1. Use programming language of your preference.
2. Focus of this course work is not on the user interface or application logic -- focus is on implementing a distributed system.
3. The notions of session and state are dependent on the application implemented; thus, they depend on design and are not identical across implementations.
4. Do not overcomplicate things – a good initial design will help a lot.
5. Virtualization technologies can be used for emulating several individual machines on a single computer (e.g., VMWare, Virtual Box)
6. This is a lot easier than working with three distinct physical machines for development and demonstration purposes.
7. Nodes do not have to be identical, e.g., one can act as a monitor/admin and another one as a sensor actuator.

Few topic examples:

1. A multiplayer game
2. An online store.
3. Synchronized music player.
4. Distributed analysis of some publicly available datasets.

The deliverables include a report and a video (max 5-minute duration) demonstrating your system.

Report contents:

1. The project's goal(s) and core functionality. *Identifying the applications / services that can build on your project.*
2. The design principles (architecture, process, communication) techniques.
3. The key enablers and the lessons learned during the development of the project.
4. How do you show that your system can scale to support the increased number of nodes?
5. How do you quantify the performance of the system and what did you do (can do) to improve the performance of the system (for instance reduce the latency or improve the throughput)? (Extra points for improvement).
6. What functionalities does your system provide? For instance, naming and node discovery, consistency and synchronization, fault tolerance and recovery, etc? For instance, fault tolerance and consensus when a node goes down.

Source code with readable, commented code (preferred in GitHub or version.helsinki.fi)

Suggested Schedule:

1. 13.11.2020 - The project topic and team members finalized, the expected goal and functionality, finalizing the programming language, the repository, and basic skeleton/outline of the code, identification of the strategy to evaluate and demonstrate the system.
2. 20.11.2020 - A basic running system that can work on 3 nodes.
3. 27.11.2020 - Preliminary evaluation, and implementation of the scaling and functionalities.
4. **08.12.2020 - Final deliverables**

Grading (30 points)

1. A running system that implements the basic goals and functionality (10 points).
2. Demonstration of system scaling (5 points). For instance, begin with 3 nodes, and then show the system can support 4 nodes, 5 nodes, and more than 5 nodes.
3. Evaluation of the performance in term of relevant metrics such as throughput or latency etc. Identifying what can be done to improve the performance (5 points).
4. Functionalities provided by the system (10 points). At least three of the following: a) naming and node discovery, b) synchronization and consistency, c) fault tolerance, d) consensus.