HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Distributed Systems (2019)

Task 1: Basic Distributed System Architectures

Practical information:
Due date: Before Tuesday, **19.11.2019**, 23:59
Each hour late after midnight is 5% less points
– After 10 hours, value of the task is 50%
– This task can be done in a team of max 4 persons (3 is preferred).

Submit your solution to dianlei.xu@helsinki.fi CC. xiang.su@helsinki.fi

Task description

Design and implement a simple distributed system running on 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.

Distributed communication between the nodes, i.e. each application instance exchange data with two other instances to form a shared state distributed across the client or server 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
• Ultimately, nodes must work together, forming a functional distributed application running on three (3) nodes.
Few topic examples: 1) Multi player games, e.g. card, gambling, paper, rock, etc; 2) Online shops/stores; 3) Synchronized music player; and 4) Distributed analytics for sensing devices.

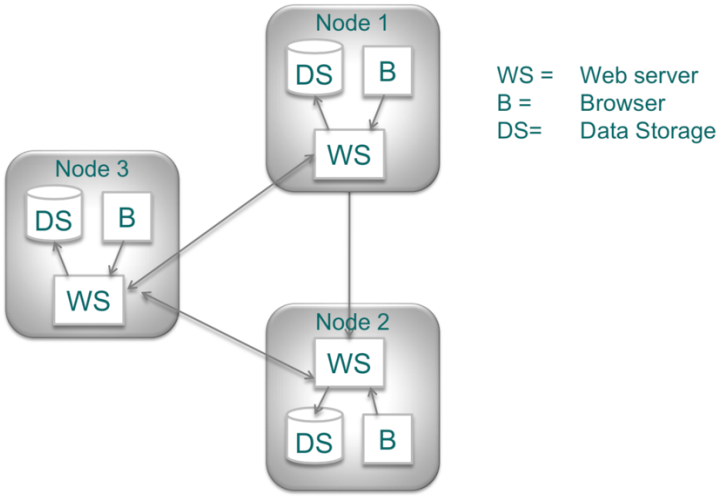

Figure 1. Running example.

General guidelines an tips:

• No enforced topics, but a list of enforced technical requirements.
• Use programming language of your preference.

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

Deliverables:

- Source code with readable, commented code (preferred in GitHub)
- Screenshots or 1 min video of your working implementation
- Report: 1) A thorough final report about the project's purpose and core functionality; 2) your system design using the 3 principles from the lectures (architecture, processes, communication), 3) problems encountered, and lessons learned. 4) Instructions for installation and execution; 5) Answer the following questions with your implementation: i). What is the average time for sending 50 messages between two nodes (random payload)? ii). Choose 3 different fixed message sizes (payloads for min, average, max), what is the average time when sending 25 in each case? iii). Choose a unique payload, e.g., average size, and then measure the inter arrival rate between messages? iv). How reliable is your architecture? What kind of applications can benefit from this architectural flavor?