SDIS 2019/2020 - 2nd Semester
Project 2 -- Distributed Backup Service for the Internet

---

1. Introduction

In this project you will develop a peer-to-peer distributed backup service for the Internet. The idea is to use the free disk space of the computers on the Internet for backing up files in one's own computer. As in the first project, the service must support the backup, restore and deletion of files. Also, the participants in the service must retain total control over their own storage, and therefore they may delete copies of files that they have previously stored.

2. Specification

The design of the service is up to you. E.g., you can choose to replicate full files rather than their chunks. Also, you can use some centralized server to manage the replicas, or you can use a totally distributed design, e.g using Chord to locate a file's replicas or chunks.

The ceiling of your project's grade depends on your design choices:

1. A basic solution using a single centralized server to manage the replicas, as described above, and using TCP has a ceiling of 14 (out of 20), as long as it uses thread-based concurrency. If you use no concurrency, then the ceiling will be of 12.
2. The use of JSSE for secure communication raises the ceiling by **up** to 2 points (out of 20):

   `SSLSockets`
   : The use of the basic interface provided by `SSLSockets/SSLServerSockets` will raise the ceiling of your project by 1 point. Note that using this interface may limit the level of concurrency achievable.

   `SSLEngine`
   : The use of this more advanced interface will raise your ceiling by 2 points. Note that this interface is also more flexible allowing you to use it together with other APIs to achieve higher concurrency.

3. Addressing each of the following issues, will also raise your ceiling by 2 points (per issue):

   Scalability
   : This can be at the design level, e.g. using Chord, or at the implementation level, e.g. using thread-pools and asynchronous I/O, i.e. Java NIO. (If you use Chord and Java NIO with thread-pools, your ceiling will raise by 4 points)

   Fault-tolerance
   : The goal is to avoid single-points of failure. E.g. if you choose a centralized server you can replicate it and use, e.g., Paxos or just plain primary-backup. If you choose a decentralized design, you can implement Chord's fault-tolerant features.

3. Constraints

This project must be developed in groups of **3 or 4** students. The implementation language is Java and you can use only Java SE. **Exceptionally,** we may allow you to use other libraries. In this case, **you must request permission via this project's Moodle forum** and wait for our decision. Using code without permission, may penalize your final grade.

4. What and how to submit?

You must submit all the source code files via FEUP's Git Service. Your project shall be named **`sdis1920-t<n>g2<p>`** , where `<n>` is the number of your section (turma) and `<p>` is one digit with the number of your group, e.g. `sdis1920-t0g22`. Your group id will be assigned to you by the staff after you fill this form. In addition to the source code files, you should submit a plain ASCII file named **`README`** with instructions for compiling and running your application also via the Git project.

Furthermore, you will have to submit a report according to these instructionsand fill this project's self-evaluation form. Please note that because there were no lab sessions, it is important that you detail what you have done for the project. You are expected to collaborate in our effort to minimize the effects of the current circumstances.

4 Demo

You will have to demo your project in your last lab class.