Large-Scale Distributed Systems - LSDS 2016
Project A - Chord On Demand

November 17th, 2016

1 Introduction

This is **one of the three project assignment proposals**. You need to **decide before November, 23th 2016 which project you would like** to work on and notify via email the assistant and the instructor. All projects require the same amount of work. You have to do, and we will grade, only one project.

1.1 Report and Grading

You will have to deliver a report of your work to the assistant. This report will be taken into account for the final grading (see first Lesson's slides).

Your project report has to contain:

- A text report, containing the various graphs you will construct during the sessions, the associated explanations and discussions. Remember that a graph without explanation, or an assertion that is not conveyed by a graph, do not have much interest.
- Source code, formatted in a proper way with indentations, and with appropriate comments.

Format: Reports are **individual**. Your report has to be written in **English**. The deadline is set to **December, 15th, 14:00**. *Send the report to raziel.carvajal@unine.ch (CC to raziel.carvajal@gmail.com) and etienne.riviere@unine.ch .*

Use the following mail subject: `LSDS2016 - project A - FIRSTNAME LASTNAME`. Only **PDF files** for the report and ".lua", ".gpi" (gnuplot), ".sh", etc. files for the code will be accepted.[1]

Grading: We will grade your progress, the correctness, and the clarity of both the code and the report. The content is what matters most, but the presentation has an impact on the final grading.

[1] If you want to use another language for processing the logs, feel free to do so as long as the language is an interpreted script language: python, ruby, perl, etc.

2 Project overview

Note: This project is based on a Gossip-based protocol; hence, it requires the use of the peer sampling service. You can either use your own implementation of the Peer Sampling Service from the first assignment or the implementation of the peer sampling service that you will find on ILIAS.

The Chord routing protocol was implemented for the second assignment, using the explicit construction mechanisms from the original paper. As mentioned during lecture, it is also possible to bootstrap the Chord structure using gossip-based protocols, and maintain this structure in the face of churn as a result of its self-organizing properties.

You will implement the *Chord on Demand* algorithm and evaluate it. The following tasks are to be performed.

- Reading the Chord on Demand paper;
- Implementing the T-Man gossip-based protocol that selects peers in the view based on a distance function, given as a parameter;
- Using the T-Man implementation along with the Peer Sampling Service to create the Chord's protocol views:
  - the predecessor;
  - the successor list;
  - the finger nodes that allow the efficient routing.[2]
- Evaluating the protocol in terms of routing efficiency and churn resilience. The comparison to the explicit Chord implemented in the previous assignment is a good idea, but it is not mandatory to fulfill the assignment.

Note that the experiments to conduct in the projects are less guided than the previous assignments. You should first implement the simplest version of the protocol and test it using simple cases (first week). It is then *your responsibility to contact the assistant* and meet in person to discuss on the experiments you will provide in the report. You are free—and even encouraged—to propose other measurements (e.g., based on the original paper), and/or to compare with your version of Chord from the previous assignment.[3]

Reading: You have to read the Chord on Demand paper *before starting the implementation*. The algorithm is described in the paper and in the course slides, but do not hesitate to ask for a meeting to discuss the algorithm if anything is unclear. The paper is available on ILIAS and can also be downloaded from `http://members.unine.ch/etienne.riviere/papers/chordondemand.pdf`

[2]Note that the structure shall always evolve towards the perfect structure, hence it is not necessary to notify other nodes that a newly joined node is a better neighbor–this should take place as part of the self-stabilization protocol.

[3]Note that it is allowed to compare with another students' explicit Chord code from the second assignment. This has no impact on the final grading but should be mentioned in the report.

3 The Peer Sampling Service provided implementation

The provided PSS implementation has the following API:

- `pss_init()`: initialize the PSS. When this function returns, the active thread of the Peer Sampling Service is running and maintains the view. The call lasts for some time, and returns once a few cycles of the Peer Sampling protocol have been performed;
- `pss_getPeer()`: returns a random peer from the view.

4 Suggested workplan

This section suggests a work plan for the project assignment. You should be careful to have an implementation of the simple protocol running as soon as possible and focus on the evaluation and the report. You should propose your own work plan, which should be approved by the assistant. The underlying items are suggestions.

Task 4.1: Implement a self-organizing ring where each node gets its predecessor and a set of k successors on the ring.

Task 4.2: Implement the fingers selection algorithm using T-Man.

Task 4.3: Add support for nodes failures. When a node fails, the fingers pointing to it will be updated after some time. The routing mechanism has to be aware of the fact that some links may be stale, and it is reasonable to add an aging mechanism to these links similarly to the aging mechanism used in the Peer Sampling Service. If you integrate this task in your work plan, you will need to use a churn trace. Ask the assistant to provide a churn trace.

Task 4.4: Evaluation of the protocol. Several metrics can be observed including, but not limited to, bootstrap efficiency, routing efficiency, correctness of the structure, etc.

Task 4.4: *(optional)* Proximity-aware Chord. In T-Chord, fingers are selected according to their distance to the node *on the ID space* (ring space). It is possible to add some proximity awareness in the choice of fingers to take into account network delays. Implement T-Chord-Prox, which uses communication proximity as the finger selection algorithm in order to construct low latency paths between nodes. Compare your performance results with the original T-Chord. Note: if you decide to integrate this task in your work plan, contact the assistant who will give you a code snippet that will emulate network delays in the context of the cluster (where normally all communications take the same, very short time).