Distributed Systems

COMP90015 2018 SM1

Project 2 - High Availability and Eventual Consistency

Lectures by Aaron Harwood

Project 2

The project makes use of some parts from Project 1; namely the client. If you have not satisfactorily completed those parts from Project 1 then you'll need to work with your tutor and lecturer to catch up.

You are required to:

- Design and implement your own server protocol to provide availability and consistency among the servers, in the presence of possible server failure and network partitioning.
- Make sure that your new servers work with existing clients.
- Document your design and explain how it works to address the requirements.

The existing server protocol can be completely removed and replaced by your new protocol.

Aims

We want to implement a server protocol that:

- allows servers to join the network at any time
    - even after some clients have joined :-)
- allows clients to join (register/login) and leave (logout) the network at any time
- maintains that a given username can only be registered once over the server network
- guarantees that an activity message sent by a client reaches all clients that are connected to the network at the time that the message was sent
    - i.e. if **T** is the time that the message was sent, and **X** is the set of clients connected to the network at time **T**, then the message should be received by all clients in the set **X**
- guarantees that all activity messages sent by a client are delivered in the same order at each receiving client
    - however the order that messages are sent by different clients does not need to be enforced at receiving clients
- ensures that clients are evenly distributed over the servers as much as possible

Revised failure model

Let's assume:

- servers can crash at any time, whereby the server process abruptly terminates without a chance to inform other servers before it does so (i.e. ctrl-c or kill signal)
- clients can crash at any time, whereby the client process abruptly terminates without sending a logout to the server before it does so (i.e. ctrl-c or kill signal)
- network connections between servers and between servers and clients can be broken at any time
    - this may cause TCP connections to abruptly break (exceptions will be thrown) and/or message loss
    - UDP packets may be lost (which is the assumption for UDP in any case)
- broken network connections will eventually be fixed, such failures are transient

Availability and Consistency

The failure model and CAP theorem suggests that we cannot achieve both high availability and high consistency at the same time.

- Focus is on availability: clients should always be able send messages if they are logged in and should receive messages as soon as possible
- Consistency should be eventually reached: the aims of the system should be achieved eventually where possible, i.e. messages sent should eventually get to the clients as required, clients should eventually be load balanced over the servers, the delivery order of messages should be preserved, etc.

High Level Paradigms

- You may want to think about the use of a high level paradigm.
- The project can be completed from first principles, using basic socket communication if you wish. This is probably the preferred approach.
- If you wish, you may make use of third party high level communication libraries, but they must be embedded into the server process. There must not be any connections from servers to third party systems of any kind (e.g. connections from the server to a database or to a messaging system). Servers must make connections only to other servers and clients make connections only to servers.

Technical aspects

- Your client and server should start from the command line just as they did in the first project. In particular a server can only be told of one existing server at startup. However it may well learn of others via your protocol and make more connections if you wish.
- Use Java 1.8
- Use of third party libraries are ok, but they must be embedded into the server process. There must not be any connections from servers to third party systems of any kind.

Report

Write 1500 words to detail your design and implementation and explain how it achieves the aims of the project. You should describe the messages and protocol used by your servers and may give examples of how it works in the presence of failure, and how does it work to achieve high availability and eventual consistency. If you have used a third party library then you should explain the details of how that library aids in achieving the project goals, including message formats and protocols where applicable.

Submission

- You should submit your report plus your modified system similarly to Project 1; instructions will be given on LMS.
- You will have a similar chance in Project 2 to assess your team members using the LMS test submission.
- The due date is Saturday Week 12, 26th May, 11:59pm.