Distributed Pub-Sub System with Replication

Ajinkya Prabhu - amprabhu **Athilesh Arputham - aarputha**

Introduction

Our aim in this project is to create a publisher-subscriber system which will follow the naming server and storage server architecture used in the lab 3 distributed file system project.

Design

Some definitions

1. Topic - A location/URI where messages will be posted by publishers and consumed by subscribers.
2. Publisher/Producer - will generate messages and will follow a schema per topic
3. Subscriber/Consumer - will consume messages from a topic for later use

Our Design includes two kinds of servers.

1. Naming Server - Will hold the mappings between topics and Servers. This will then be the bookkeeper and maintain a copy of all the servers assigned to a specific topic and store their IP address and port.
2. Topic Server - will hold the actual data related to the topic such as the timestamp of each message and current producers and consumers

The topic servers will first register with the naming server with the topic they intend to serve and their IP address and port.

The Topic servers will replicate their data across multiple instances so as to maintain fault tolerance in the event a topic server fails and is uncommunicative the client will automatically request the naming server for a new topic server with the same topic and try to recover.

Each topic server will also replicate which clients have read certain messages. To make sure that deletions are consistent across all replicated topic servers the topic servers will only delete messages in topics after all clients connected to a topic have read the message. A way in which this can be achieved is if all clients connected to a topic server have been marked as having seen the message the topic server can delete the message and ask other servers to delete the message themselves. If all other server's clients have also the message then they also will delete those messages.

Project Plan and Testing and evaluation

1. Implement the naming server, Server will hold in a key-value store the details mentioned above
2. Implement the Topic server will have a theoretically unlimited buffer of messages and will delete messages in the process as described above.
3. Implement replication of messages between topic servers and forwarding messages to clients on the same topic but different servers
4. Testing with unit tests and regressions tests which check for the following
   - Checkpoint
     - Topic Server Creation
     - Register on the naming server - 20
     - Communication between client and topic server - 20
   - Final
     - Topic server replication - 25
     - Topic server message production - 20
     - Topic server message deletion - 20

- Topic server message consumption - 20

Tooling (Subject to change based on Course staff approval)

Language - We would like to use Rust
Libraries

- tokio for async and threading
- serde for serialization deserialization
- tarpc for RPC communication

Makefile - will include how to build and test our project using the testing framework provided by rust

Evaluation

We do not have a concrete plan for evaluation but have a rough estimate on the points distribution above

125 - auto grading 30 - documentation