UMEÅ UNIVERSITY, DEPARTMENT OF COMPUTING SCIENCE

Persistent GCom project (30 points)

Distributed Systems - 5DV147

1 PURPOSE

The goal of this project is to increase your knowledge of distributed systems by extending the functionality of the **GCom** middleware for group communication. The project must be solved by groups of two students (same groups as for GCom). The specific goals of this project are:

- *Understanding fault tolerant mechanisms*, their characteristics and applicability with emphasis on maintaining robustness and availability of a system under network partitions and unexpected events.
- *Understanding how distributed data stores work*, with focus on how to use them to achieve highly scalable and fault tolerant services.
- *Understanding how current Internet Services are able to achieve high levels of scalability, availability, and fault tolerance.*
- *Development of software engineering skills.* Practice the ability to add features to extend the functionality of a previous large implementation.
- *Development of written communication skills.*

2 DESCRIPTION

In this project you will have to extend the functionality of the GCom middleware on three aspects:

1. Allow clients to disconnect from the group (either voluntary or due to unexpected events) and, once they reconnect, receive all communication sent to the group during the disconnected period.

2. Increment the robustness of the group by maintaining operations during network partitions. Clients separated by a partition should be able to communicate, normally, with other clients on the same partition. The group should achieve full connectivity once the partition is resolved.

3. Increment the robustness of the *registry* service through replication so that correct operation (i.e., new clients can still become members of the group) is maintained even if some instances crash.

2.1 MANAGEMENT OF RE-CONNECTING OPERATIONS

In dynamic groups, members may join and leave the group at any time. The GCom middleware offers basic management of dynamic groups with the provision that processes that re-join the group are handled as any other joining process, i.e., as if they have never been group members. A direct effect of this design decision is that since all processes are "new", they have never participated in previous group communication. For several applications (think *Skype*) this group membership functionality is too simple. In those cases what is necessary is also to recognize that a joining process was a member of the group some time before.

Another desirable feature is to be able to store messages sent to the group. If messages are persisted on permanent storage, processes re-joining a group are able to retrieve messages sent to the group during the time that they were disconnected. Moreover, storing messages on permanent storage helps in maintaining a history of the group communication so that processes can query and fetch messages previously sent to the group, even when they were not group members.

Due to performance and reliability concerns you will have to use more than one storage unit (replica). Each unit must manage a certain number of clients, i.e., not all clients are connected to the same storage unit. Moreover, it is important to maintain storage units synchronized so that clients connecting to any unit can access all group messages.

The specific extensions to GCom required are:

- To *maintain an identity for processes belonging to a group*: The group management module should know whether a joining process is a new group member or if the process is re-joining from a disconnected period.

- To *retrieve messages sent to the group when a process re-joins*: All messages sent to a group while a process is disconnected should be delivered and displayed once the process connects back.

- To *maintain causal order of messages*: All messages delivered when a client reconnects to a group should be delivered in causal order.
- To *maintain replicas synchronized*: Each replica must manage a number of client connections but connecting to any replica should provide identical information.

Consider the following issues when designing these functionalities:

- *Reliability and Robustness*, e.g., at which point in time should messages be persisted to permanent storage? think of what may go wrong or fail.
- *Limitations*, e.g., What can the system not guarantee?

2.2 Management of network partitions

You have to extend GCom so that processes can maintain normal operations under network partitions, e.g., to a great extent processes should be unaware of partitions. The specific requirements are:

- Processes on either side of a partition must be able to communicate normally with processes on the same partition.
- Processes at the other side of a partition should appear as disconnected processes.
- New processes wishing to become members of a group that is divided by a network partition should be allowed to do so.
- Messages should be delivered to all processes (at the other side of the partition) once the network partition is fixed.
- Conflicts that may arise due to the group segmentation should be resolved.

2.3 Registry service

In GCom the registry service is a single point of failure. If the registry crashes, group communication can continue but no other processes can join the group. You will have to fix this problem. The specific requirements are:

- To maintain multiple instances of the registry service so that group membership functionality is still possible even if some instances crash.
- To persist the state of the registry service on permanent storage.
- To synchronize registry information across instances so that clients can connect to any instance indistinctly.

You will make use of the same storage system that you use for the management of reconnecting operations.

2.4 Testing and Demonstration

In order to ensure the correctness of the extensions to the GCom middleware, you must modify the chat application developed during the previous project. You must also adapt and extend the debugging application so that you can correctly test and demonstrate that your system is working properly. Similarly to the GCom project, your test application and your debugging utility will be used during your practical demonstration of this project

- *That messages are delivered according to the specified ordering*
- *How messages are propagated in the network*: Show both the path a message takes and how many times a certain process has received a certain message.
- *The content of hold-back queues and other buffers*: Present all messages waiting to be sent or delivered as well as values of vector clocks and other counters.
- *Current system performance*: As a measure of the system performance, count the number of messages (including control messages) required to perform an operation (send one message with a specific ordering and certain multicast).
- Also, for clarity reasons, *the test and debug application must feature graphical user interfaces*, as the amount of information that needs to be presented exceeds what is reasonable for a text-only user interface.

2.5 Tools

For this project you will have to use a distributed data store (e.g., HBase, mongoDB, or Cassandra). In addition to a data store, you are free to use any tools that you want. You will however have to motivate your choice of tools with respect to, and in terms of, what you expect to learn from this course and project. Similarly to the previous project (GCom), if you find a tool that solves the assignment with little or no effort from your side, you are naturally **not** allowed to use it. You are of course allowed to use documentation and research papers from existing solutions.

More specifically:

- You are free to use any data store that you like, however, the teachers will only provide help for Apache Cassandra.
- You may use any programming language and middleware of your choice (as long as the rule about non-trivial solutions is not broken), however, the teachers will not provide help for other systems than Java using Java RMI or Python.
- Make use of any IDE and/or plug-ins you feel makes the job easier.

2.5.1 APACHE CASSANDRA

During the Apache Cassandra tutorial you learned the basics of using Cassandra. For the project you are free to use the scripts and code that were provided for the Tutorial. For the project you will have to access Cassandra within your source code. The teachers have prepared sample code that will allow you to do so (you can download the code[1] as an Eclipse project).

```
1  package se.umu.cs.ds.cassandratutorial;
2
3  import com.datastax.driver.core.*;
4
5  public class Example {
6
7    public static void main(String[] args) {
8      // Connect to the Cassandra cluster
9      Cluster cluster = Cluster.builder()
10         .addContactPoint("127.0.1.1")
11         .addContactPoint("127.0.1.2")
12         .addContactPoint("127.0.1.3")
13         .build();
14
15     // Connect to the "mykeyspace" keyspace
16     Session session = cluster.connect("mykeyspace");
17
18     // Prepare a statement
19     SimpleStatement toPrepare = new SimpleStatement("SELECT * FROM users WHERE
           user_id=?");
20     toPrepare.setConsistencyLevel(ConsistencyLevel.ONE);
21
22     // Execute the statement
23     PreparedStatement prepared = session.prepare(toPrepare);
24     ResultSet resultSet = session.execute(prepared.bind(1));
25
26     // Print results
27     Row result = resultSet.one();
28     System.out.println(String.format(
29         "Users with ID 1: %s %s",
30         result.getString("firstName"),
31         result.getString("lastName")));
32
33     // Execute another statement
34     resultSet = session.execute("SELECT * FROM users");
35
36     // Print results
37     System.out.println("All users:");
38     for (Row row : resultSet) {
39       System.out.println(String.format("%d %s %s",
```

[1] http://www8.cs.umu.se/kurser/5DV147/HT13/assignments/cassandra-client-example.tar

```
40              row.getInt("user_id"),
41              row.getString("firstName"),
42              row.getString("lastName")));
43      }
44    }
45  }
```

Listing 1: Java code for accessing Cassandra

2.6 ADVICE

Some helpful hints:

- Reuse the first part of GCom implementation.
- Data/Messages should be persisted when delivered to the application but not when received by the middleware.
- To solve the registry service (naming service) problem you can replicate the leader reference for each group in all replica instances (i.e., read optimized replication).

2.7 DETAIL LIST OF PROJECT REQUIREMENTS

The mandatory features that must be implemented and their value in points toward the project final score are listed below.

Feature	Points
Management of re-connecting operations	15
Management of network partitions	7
Add fault tolerance to the registry service	4
Implementation points	26
Written report	4
Total points of project assignment	**30**

In addition, you must demonstrate your solution to us and hand in a complete report fulfilling the requirements as described below.

2.8 BONUS POINTS

Everything on the mandatory part + some creative idea that extends the functionality of the basic system. In order to obtain the bonus points, all features from the mandatory part **must** work correctly and you must score at least 50 points on the theoretical part of the evaluation (from the written assignments). We can add up to a maximum of 10 extra points to your score.

If you decide to add additional features to be considered for bonus points, make sure that

you and your partner have the same ambitions and goals for this project. You **must** also come by our office and present what you intend to do so that we can decide whether it is feasible giving the time frame as well as agree on how many points you will obtain. Bring your idea in writing. This is to avoid misunderstandings that end up costing your group important time that would be better spent testing and maintaining your obligatory code.

3 Deliverables

As an aid in planning your work, we divided the project into two deliverables. Read the specification carefully before starting working on deliverable 1.

3.1 Deliverable 1 - Analysis and Design (Optional)

This deliverable is optional but we encourage you to do it since we can offer you feedback about your overall plan and design. If you decide to work on this deliverable, you must include at least the following sections:

- *Analyze the problem.* What should be done, and what problems need to be solved? Formulate a requirements specification for Persistent GCom, based on the problems you identify. What *must* be implemented, and what *may* be implemented, time permitting? See [RFC 2119][2] for suggestions on how to write such sections.
- *Identify suitable tools.* Besides the distributed data store, are there any third party components that could be used? Learn the benefits and drawbacks of possible choices by implementing simple test applications before choosing a tool that you will devote energy into using. The sooner you learn the limitations of a technology you use, the better.
- *Plan your project* Write a project plan, complete with milestones (dates when you will have completed parts of the project) and division of work. You will be required to refer back to this in your final report, and discuss how well your project followed the project plan.
- *Design a solution.* Write a design (as detailed as possible) for your solution. Try to walk through the conceptual parts of the design to convince yourself that it will be feasible when you implement it. Having a sound, modular design is important. Your design should be detailed enough to not need any major additions during implementation. Minor modifications are allowed of course - even the most carefully designed system may require changes due to e.g., unforeseen properties of the tools used.

[2] http://www.ietf.org/rfc/rfc2119.txt?number=2119

3.2 Deliverable 2 - Implementation and full report

Please note that you can *base* the full report on the first analysis and design document, which means that a good first report will make the full report easier to complete!

It is a **strong** requirement that your system can easily be started from a normal Unix terminal. If you have a complicated command line, please include the appropriate scripts for each component. See this basic scripting tutorial[3] for a quick guide on scripting.

4 Hand-ins

Deadline for handing in solutions are found in the course schedule[4]. Keep in mind the policy for late submissions.

The project consists of two or three hand-ins:

1. Written hand-in of deliverable 1 (optional).
2. Written hand-in of deliverable 2 (complete report) and implementation.
3. Demo of your system.

4.1 Demo

During your demo, you will need to convince the teachers that your implementation works. Bring a test protocol, i.e., a series of tests that **clearly** demonstrates that your Persistent GCom fulfills the requirements, and a test tool which can be used to apply it. The protocol should include tests for a combination of clients disconnecting from the group chat service in a way that you can show that messages sent to the disconnected client are correctly received once the client reconnects to the group chat. Part of this test should show that messages are still delivered using causal ordering. You must consider that you will have at least three storage units (replicas), at least two clients connected to each storage unit, and that a network partition would divide the storage units with two storage units on one site and one unit on the other side. Remember to bring a copy of the test protocol on paper. Your test protocol must clearly state your names as well as user names.

4.2 Report

Similarly to the previous project (GComm), you will need to submit a report. In the report include any assumptions you made during the analysis phase. Also discuss problems encountered and alternative solutions considered in the analysis. The report should also discuss to what extent the requirement list is fulfilled, as well as to which extent you could adhere to the project plan. Once again, remember that GCom is a general middleware,

[3] http://www.linuxconfig.org/Bash_scripting_Tutorial
[4] schedule.html

which just so happens to be used by a chat application for the project, this means that it is important to document how one would use it to develop other applications requiring group management and group communication.

Try to maintain a focus on readability while writing the report. Make the description of your system as clear as possible, and consider your report's level of detail. When writing the report keep in mind the feedback that you received from your previous report.

4.3 Final note

Remember, you need to get at least 50 points on the aggregate of all projects to pass the course. Any bonus points from the assignment can only be used to get a higher grade once you pass. Furthermore, any bonus points obtained only count during the three exams offered for the course during the next 12 months, not for later offerings of the course.