Prof. Dr. Annette Bieniusa
Philipp Lersch, M. Sc.

RPTU Kaiserslautern-Landau
Fachbereich Informatik
AG Softwaretechnik

Exercise 6: Programming Distributed Systems (Summer 2025)

Deadline: 24.6.2025 AOE

Submission

- You need a team and a Gitlab repository for this exercise sheet.
- In your Git repository, create a branch for this exercise sheet, for example with

```
git checkout -b ex6
```

- Create a folder named “ex6” in your repository and add your solutions to this folder.
- Create a merge request in Gitlab and assign Philipp Lersch as assignee. If you do not want to get feedback on your solution, you can merge it by yourself.

1 CRDT Questions

In the lecture you started with researching CRDTs. Answer these questions to test your fundamental knowledge on CRDTs.

a) What does CRDT stand for? Do alternative long forms exist?

> Regular: conflict-free replicated data type
> - convergent replicated data type (CvRDT, state-based)
> - commutative replicated data type (CmRDT, operation-based)

b) Which two major categories exist for CRDTs? How do they differ?

> - State-based CRDTs: Transfer (partial for Delta CRDTs) states, update sizes grow
> - Operation-based CRDTs: Transfer operations, often requires more guarantees from the transport layer

c) What are typical drawbacks of CRDTs?

> - Growing meta-data (e.g. tombstones)
> - No total order, strong eventual consistency (TLDR; If two or more nodes receive the same updates they will reach the same state)
> - Internal complexity and semantic details

d) What are typical benefits of CRDTs?

> - High availability (local first, offline capable)
> - Low local latency (even more complex local processing is often much faster than 1+ round trips over the network)
> - No coordination required

e) Give a short description of the CAP Theorem.

The CAP Theorem states that it is impossible to have consistency, availability and partition tolerance at the same time. It is only possible to choose two or a compromise.

f) Evaluate CRDTs in the context of the CAP Theorem.

CRDTs are meant to be highly available and partition-tolerant. Considering the CAP Theorem strong consistency is not possible.

g) What properties should CRDT operations exhibit?

- idempotent ($A * A = A$): Updates can be reapplied.
- commutative ($A * B = B * A$): The order of update integration does not matter.

Associativity and intent preservation depend on the implementation and desired semantics.

h) How could a state-based counter CRDT be implemented? It will only be incremented.

A state-based counter CRDT may be implemented by using a vector clock as metadata. Each replica increases its own value assoziated with its own PID on local increment operations. The current locally known and observed value of the counter is the sum of all vector elements.

2 CRDT Implementations

In this section you will implement some basic CRDTs. To simplify the task at hand, we will create network agnostic implementations. These have three classes of operations:

1. The initialization: Here an initial state is created and the value for the application layer is returned.
2. The operations: Operations take some input and the current state to generate an update. They do not directly modify the local state, instead they only return the update for an arbitrary transport layer.
3. The update integration: The updates from any CRDT-operation are integrated into the local and remote replica states. Each integration returns the new value for the application layer.

Hint: If you want to play around with CRDTs you can check out a simulator that includes two set implementations on `https://softech-git.informatik.uni-kl.de/research/crdt-showcase`. Run the master-branch version.

2.1 Two Phase Set

Research and create a state-based two phase delete-wins set CRDT. It should implement the operations for adding and deleting entries.
Hint: A state-based two phase set uses internally two sets, which can be transmitted on updates. One set tracks the added elements. The other set tracks the deleted elements. The state of the set as presented to the upper layer is the set of added elements without the elements of the deletion set.
Hint: You can check the "Delete wins double set" preset in the above mentioned simulator.

2.2 Expanded 2P Set

How could your solution for 2.1 be changed such that deleted content can be re-added? Would the created CRDT have different semantics?

2.3 Add Wins Set

Modify the previous two phase set CRDT so that it is operation-based and exhibits add-wins semantics. It should also implement the operations for adding and deleting entries. It should be possible to add and delete the same content as often as needed.
Hint: You can check the "Operation-based set" preset in the above mentioned simulator.

2.3.1 Turtles all the way down

What guarantees are needed by the transport layer to ensure that your implementation works correctly?

2.4 Multi-Value Register

Implement a multi-value register. It should provide a set operation.
Hint: What should be the return-value of the init and integrate update operations?
Hint: Vector clocks can help to detect concurrent operations in an only partially-ordered distributed system.