Universität Heidelberg
Wintersemester 18/19

Group Parallel and Distributed Systems (PVS)
Artur Andrzejak, Diego Costa

Problem Set 6 for lecture Distributed Systems I (IVS1)

Due: 04.12.2018, 14:00 Uhr

Exercise 1 **(5 Points)**

Follow the Quick Start tutorial of gRPC presented in the official documentation[1]. This quick start works on a hello world example, provided in 10 different languages. Get yourself familiar with gRPC framework in Python and another language of your choice. For this exercise you will build upon the hello world example and create an RPC service to communicate between a server in Python and a client in a second language.

a) Define a new service in helloworld.proto called `RegisterPerson` which receives a `Person` object and returns a `Confirmation` object. The `Person` object should be defined as the example given in class (see outlook slides from Lecture 06), with name, id, email and a list of phone numbers. The `Confirmation` object should contain a string to hold the return message to the client.

b) Modify the Python server to respond to the newly created service.

- **b.1)** If no phone number is passed in `Person`, the server should return a `Confirmation` object with the message: „No phone number registered for person [name]“
- **b.2)** If a list of one or more phone numbers are passed in a `Person` object, the server should return a `Confirmation` with the following message: „The person [name] had all [amount of phone numbers] phone numbers registered successfully“.

c) Modify a client in a programming language of your choice (except Python) to submit to the `registerPerson` service.

d) Submit as your solution 1) your .proto file, 2) your server code and 3) your client code. You do not need to submit the generated code.

Hint: If you select Java as your second choice, note that there is a second project called grpc-java. In this case, you will have to modify the helloworld.proto and copy from the general grpc to the grpc-java project, and make sure both client-server use the same version.

Exercise 2 **(1 Point)**

A clock is reading 10:27:54.0 (hr:min:sec) when it is discovered to be 4 seconds fast. Explain why it is undesirable to set it back to the right time at that point and show (numerically) how it should be adjusted so as to be correct after 8 seconds have elapsed.

Exercise 3 **(3 Points)**

A client tries to synchronize with a time server by sending several messages. It measures the round-trip time of the message passing and saves both the local time stamp and the remote time stamp sent by the server in its response messages.

[1] https://grpc.io/docs/quickstart/

Which of the following three measurements should be used by the client to synchronize its own clock with the server's? To which (resulting) time should the client set its clock on the arrival of the message? Give also an error interval (as small as possible) around the actual server time which contains the inferred time.

Round-trip (ms)	**Message from the Server (hh:mm:ss)**
22	10:54:23.674
25	10:54:25.450
20	10:54:28.342

Do your answers change, if you know that both sending and receiving a message to/from the server takes at least 8 ms? How can you actually estimate such a lower bound?

Exercise 4 (2 Points)

Watch the video Linux Network Time Protocol[2] and answer the following questions:

a) What are the advantages of NTP compared to the Linux netdate command?

b) Explain the concepts of Stepping and Slewing in the NTP context.

c) What is Insane Time in NTP synchronization? How to prevent that to happen in a real system?

d) Explain how to configure the NTP protocol/daemon in a Linux system. How one can monitor the time sychronization with a NTP server?

Exercise 5 (1, Bonus Points)

Discuss the factors to be taken into account when deciding to which NTP server a client should synchronize its clock.

Exercise 6 (2, Bonus Points)

Similar to the addition of a *leap day* February, 29th to some of the calendar years, leap seconds can be added (or subtracted) to keep the Coordinated Universal Time (UTC) in sync with the mean solar time. 27 leap seconds have been added to the UTC since 1972 to account for Earth's irregular rotational speed. This difference in time is not a technical flaw, but a natural phenomenon. Still, computer science has to deal with it and offer a technical solution.

Explain the concept of leap smear[3] in your own words. Why is it beneficial to have the leap second happen right in the middle of the smearing operation?

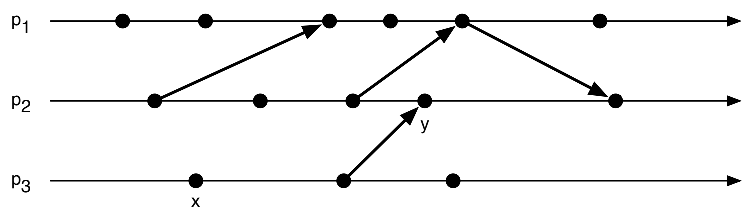
What advantage does it have to smear the leap second from a software engineering point of view? What does the protocol NTP prescribe in the case of leap seconds in order for downstream applications to be able to deal with them?

[2]https://www.youtube.com/watch?v=EkQPkQb2D3g

[3]https://developers.google.com/time/smear

Exercise 7 **(3 Points)**

Consider the events in processes p_1, p_2 and p_3 in the figure below. Denote every event with a variable name (except for the already named events x and y) and determine their Lamport timestamps. Also, list all events that are concurrent (incomparable) with x (and y, respectively).



Exercise 8 **(2 Points)**

By considering a chain of zero or more messages connecting events e and e' and using induction, show that $e \rightarrow e' \Rightarrow L(e) < L(e')$.