DS_2020-2021_GRR_A2.1b

1)

This is the scenario we find:
Code files: a.c and b.c.
9:30:00 = Execution of make to compile. Files obtained: a.o, b.o and a.out.
9:30:30 = Opening code files.
9:30:45 = Saving new code files and executing make. Execution of a.out.

After updating the file code, the error persists. Why does this happen? When the user was saving the file, the computer checked the time and it detected a deviation of 1 minute respect the real time. Due to this, the computer puts the clock 1 minute before and now the time of the computer is 9:29:45. Consequently, the file that has been saved is 15 seconds older than the new file, so when the client executes the latest file, the one executed is the old one because it is the one that has the latest date.

How can we avoid this? Making a correct compensation of the deviations. Once a deviation in the local physical time has been detected, it is necessary to adjust it. To adjust it, in case you add or subtract the detected deviation, errors as the ones seen on the Scenario could happen because the property of the time could be violated. As seen, some applications as UNIX *make* are very sensitive to this time property.

2)

So, to avoid this, the are two ways to adjust the deviation depending on the case:

- Negative deviation: The deviation amount can be added to the local physical time to compensate the deviation without any other operation, it will only produce a discontinuity in time.
- Positive deviation: In case the deviation amount is subtracted, it would cause a temporary inconsistency. To avoid this problem, the software clock is slowed down during a period of adjustment trying to reduce to the maximum the deviation. In other words, time runs slower during that period.

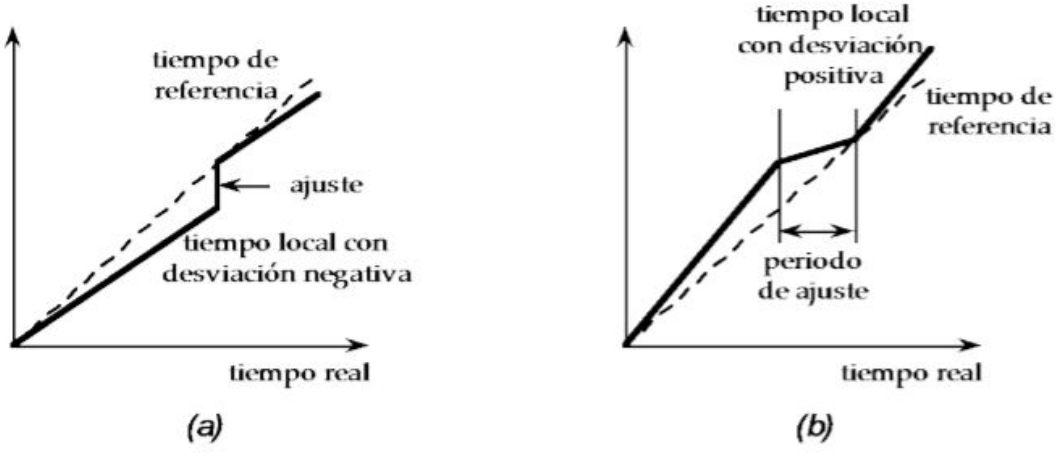

Figure 2.2. Compensation of the deviation (a) negative, (b) positive.

3)
It is asked to calculate what is the period in which the clock has to be synchronized again, so we are going to calculate it with this values:

Theta = 500ms
Precision = 50ms
Delta = 10^-5

Having this, Theta - Precision = 500 - 50 = 450ms
So, if our goal is to synchronize such that Theta is always smaller or equal to 500 :
(Theta - Precision) / Delta = 450 / 10^-5 = 45*10^2 seconds

So, if we transform seconds into hours, we get: 12.5 hours.
In conclusion, after 12.5 hours the clock has to be synchronized again.