Peaceland

Preliminary questions

1) What technical/business constraints should the data storage component of the program architecture meet to fulfill the requirement described by the customer in paragraph «Statistics» ?

Statistics are computed based on information that comes from everywhere in the country, which implies that the database needs to be distributed and partition tolerant. Also, it needs to comprehend every information gathered so far to be meaningful and avoid biases, hence it also needs to be consistent. We thus need a CP database.

SQL cannot be a solution, as it relies on a master-slave system and is thus not fully partition tolerant. We need a NoSQL database.

Additionally, statistics often require large data aggregation, and so we should favor colum-oriented solutions.

Taking these points into account, we chose to use Apache HBase, for it is a column-oriented NoSQL CP datastorage solution.

2) What business constraint should the architecture meet to fulfill the requirement describe in the paragraph «Alert»? Which component to choose?

The business constraint described in the paragraph “Alert” is that of time. It is mandatory that an agitated individual is “handled” as soon as possible to avoid the spread of their agitation. This constraint has two implications on the design:

- Upon receiving the initial report from peacewatchers, the gateway must immediately send an alert to the peacemakers;
- To allow peacewatchers to do their job as quickly as possible, they need to be able to fetch data about the individuals surrounding them at any time. It means that we need an AP datastorage to handle the data needed for day to day operations. Here, we chose to use Scylla, a column-oriented and fast database;

3) What mistake(s) from Peaceland can explain the failed attempt?

The biggest mistake is to have hired data scientists who have probably stored their data on `.csv` files in order to easen the training of their keras models on their Jupyter notebooks. Whereas they should have recruited data engineers instead.
To be more specific, they should have hired Epita students like us earlier. That would have solved all of their problems.

Jokes aside, here follows a list of mistakes they may have done:

- using either an SQL, a centralized database, or a non-CP datastorage solution to make statistics and to store big volume of data;
- not using streams to forward their data towards their databases;
- using an interpreted language like Python or R for not only their Proof Of Concept, but also their real life solution;
- forgetting about the peacewatchers’ need of quickly fetching data, and thus not having set up an AP database with a list of each citizen’s peace score.

4) Peaceland has likely forgotten some technical information in the report sent by the drone. In the future, this information could help Peaceland make its

peacewatchers much more efficient. Which information?

Currently, peacewatchers have the responsibility of computing the peace score of their surrounding citizens. Therefore, they have to be capable of fetching the most up-to-date information at any given time. However, they may be improved in various manners by sending additional information in their reports, as well as by slightly modifying some of the information they are sending.

First, it is possible that the communication channels that are established between a given peacewatcher and our servers are disturbed. In such an event, the servers may not receive all of the reports in order, nor instantly. To prevent any of such problems to happen, the peacewatchers should add a timestamp to their reports.

Furthermore, there is a high probability that two citizens of Peaceland share the same name. It can thus be a problem for us to handle the collisions when updating peace scores. Thus, it would be useful for them not to send the name of a citizen, but rather a unique identifier.

Moreover, in the current state of things, some unfortunate edge cases may occur. For instance, consider the following scenario:

1. a citizen does a bad deed and has their peace score reduced to a value right above the alert threshold (`peacescore_at_t0 = THRESHOLD`);
2. the citizen does a good deed, and their peace score should be updated positively. However, there is some delay for this to happen (`peacescore_at_t3 = THRESHOLD + ω`);
3. at `t1`, another peacewatcher fetches the citizen's peace score (`peacescore_at_t1 = peacescore_at_t0 = THRESHOLD`), and the citizen performs a minor bad deed that is recorded. This other peacewatcher records that, computes a new peacescore (`peacescore_at_t2 = THRESHOLD - ε, 0 < ε < ω`) that is below threshold;
4. Because this peacescore is below the threshold, it induces an alert, which immediately calls the peacemakers. However the theoretical score of the citizen at `t2` should have been `theoretical_peacescore_at_t2 = THRESHOLD + ω - ε`, which is above the threshold;

This scenario is one of many examples of bugs that could be induced by having peacewatchers send computed scores instead of delta values of scores *(i.e. the values of the modifications related to a citizen's deed)*.

The peacewatchers should thus send a list of a citizen's deeds, or the sum of the peacepoint value of these said deeds in each report rather than recomputing the peacescores every time.