GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Known Bugs and workarounds - Developer

Design Project - Runamic Ghent

Group 16

Depauw Hendrik

van Herwaarden Lorenz

Aelterman Nick

Cammaert Olivier

Deweirdt Maxim

Dox Gerwin

Neuville Simon

Uyttersprot Stiaan

1 Application

- `Improvement:` In the current implementation, the eventbroker calls the method handleEvent() from its own thread. This means the classes that make use of this service from the eventbroker have to keep the handleEvent() method as short as possible as not to keep the eventbroker busy for too long. The solution that is used in our implementation is that the different classes just start a worker inside the handleEvent() method. This adds extra complexity to these classes that is not necessary. A better implementation would be to start separate threads in the eventbroker itself. The eventbroker can make use of a threadpool to limit the amount of simultaneous threads and avoid the delay of creating threads.
- `Improvement:` The EventBroker uses Strings as way to identify EventTypes. It would be better to use classes for this. The new approach has 3 important advantages. First and foremost: it is not possible to have collisions as a Class is uniquely defined, in contrast to a string. Secondly: classes can form a hierarchy, resulting a more flexible way of subscribing to related event types. Lastly: the specific event class can carry some data, eliminating the need for an Object as second argument, that is then cast to whatever the EventListener was expecting.
- When the routeEngine has to decide whether you followed the new route or the original route, it can sometimes make a mistake if the following points of the new route and old route are to close to each other. Currently 5 future points are compared with the old and new route to check on which route the user currently is. If these 5 points are close to each other, then no decision can be made about the instruction list. This can be resolved by increasing the amount of future points, but will require more time to determine on which route the user currently is.
- Problems in the route directions can arise whenever there is a u-turn. Currently we are looking to 5 future instructions to check if the user has skipped a point in the instruction list. This can happen when the GPS is not accurate enough or when the GPS location updates are to slow. With a u turn it is possible that there is an instruction before the u turn and after the u turn. Instruction after the u turn can play if the user is approaching the u turn because the location of the future instruction and the user are the same. We have solved this by not looking to future instructions after an u turn. However the route instructions get stuck in the u turn if the user never passes the location of the u turn. A new approach of instruction handling is necessary to solve this problem.

2 Server

- Tags sometimes stop working when the server reboots. This is because, instead of encoding all node indices, tags just save the first node and then every offset of the edge: [17263, 1, 1, 3, 1, 2] means: go to node 17263, take the first connection in the list, then the first, then the third, and so on. This yields high compression ratios, but if slight changes happen to the graph (or if the order of the connected nodes changes (gasp!)), then the tag stops working.
- The server is still single threaded due to Python limitations. A possible way to solve this is by encapsulating all startup code inside the Rust application, and inserting a larger barrier between the Rust and Python programs, for example with an RPC or ZMQ-based border.

- finding the perceived length of a route doesn't work. It doesn't consider ratings, water or parks, only the highway value. As this feature is only meant for testing, this should not be a problem, but consider fixing it once you got rid of the FFI border.
- Speaking about length: one of the defined server tests specified that for any two configurations, the one that generated a rout should perceive that route as shorter as the other. This is true given that the routing algorithm returns the shortest route possible. However, as the routing algorithm is heuristic, this cannot be guaranteed, rendering the test useless.