Messages between servers in the system:

* A server s1 can send an upsert a ride to another server s2 (using gRPC): s1 will send a ride info to s2, s2 will insert / update this ride in his memory, it will insert it if it didn’t have in his memory earlier, and otherwise will update it.
* A server s2 will respond to a message that he received from server s1 with success / failure.

Information reserved in each server:

* Server name.
* Cities: An array that contains all of the cities in the system.
  + Each city contains the name of the shard it resides in and its coordinates.
* Shards: An array of shards that contains the shards that will be saved in this server.
* Shard leaders: A map that maps from a shard name to the name of the leader server for this shard.
* An array that contains all the rides that are saved in this server. A ride will be saved

Subtree `cities` contains each city as a znode directly connected to the root called `cities`

Cities: {A, B, C, D}

S1={A, B}, S2={C}, S3={D} (leading for those cities)

AddRide:

* Add ride REST request was received in some server s.
* If s is leader for the start\_city of the requested ride, then s will take care of saving the ride by doing the following:
  + Take a lock on start\_city, and all of the neighbors of the line start\_city->end\_city.
  + Add a Znode in zookeeper in subtree `cities` as a child for each neighbor (and a child of himself). This procedure is synchronized, we needed to acquire a lock in order to invoke it. This call does not return unless all the locks were acquired.
  + Release locks acquired earlier on all of the neighbors.
  + Add the ride to the internal data structure containing all the rides.
  + Broadcast the ride to all of the followers for the shard containing city start\_city.
  + Return success.
* If s isn’t leader for start\_city, find the leader for this city (from a mapping saved locally), and send a blocking grpc request to the leader and wait for the result.
* Once done, return answer to client from the server that got the REST request.

reserveRide:

* Reserve ride REST request was received in some server s.
* If the path requested was of length 2 (start\_city->end\_city):
  + Send ZK getChildren request on path `/cities/start\_city`.
  + For each child (city), try to reserve the ride in this city by sending a blocking grpc request to the leader of the city.
  + Return the info of the ride from the first request that was accepted, otherwise return failure.
  + A leader server receiving a grpc request to reserve a ride would do:
    - Acquire the lock of start\_city.
    - Check if a suitable ride was found, this should be synchronized on the local rides Array.
    - If a suitable ride was found:
      * Acquire lock on all of the neighbors in order to update them – don’t acquire lock on start\_city again. (might need also to acquire the global lock, should think about it…).
      * Update all of the neighbors (can be calculated locally) in ZK subtree `cities` (counters, add/remove znode etc…).
      * Release lock on all neighbors.
      * Release lock on start\_city.
      * Broadcast reservation for all of the followers on the shard that contains start\_city.
      * Return the reserved ride.
    - Otherwise return that on suitable ride was found.
* If the path requested was of length more than 2, coordinator is managing the path:
  + Acquire locks on all cities atomically (using global lock) that are a start\_city of a subpath in path. Release global lock afterwards.
  + Start a 2PC protocol in order to reserve the path:
    - Send query request to the servers that a potential ride might be in it, in order to get all of the rides with vacancies > 0 (stream them). In this request all of the servers will add a watched to a Znode that the coordinator is responsible for, and the coordinator will use it to broadcast a reservation request to them.
    - Once we get a response from all of the requested servers, we start a backtracking algorithm to try and build a valid path from the rides.
    - If such path of rides was found, reserve all of the rides by broadcasting in ZK in the Znode that the servers are watching on, we broadcast the reserving information (it is guaranteed that they didn’t reserve these rides meanwhile because we acquired the locks).
    - release all of the locks.
    - Return the path of rides reserved to the client.
  + If the coordinator failed while doing this, we prefer that the other servers don’t reserve the rides, each server can check if the grpc client of the coordinator is alive, and only then reserve the ride.

CHECK SERVER / GRPC CLIENT FAILURES CORRECTNESS OF LOCKS!!