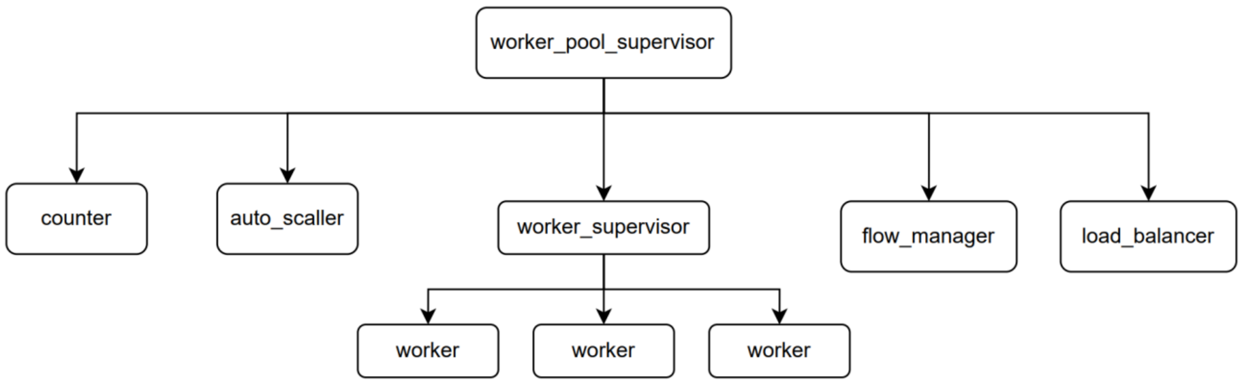
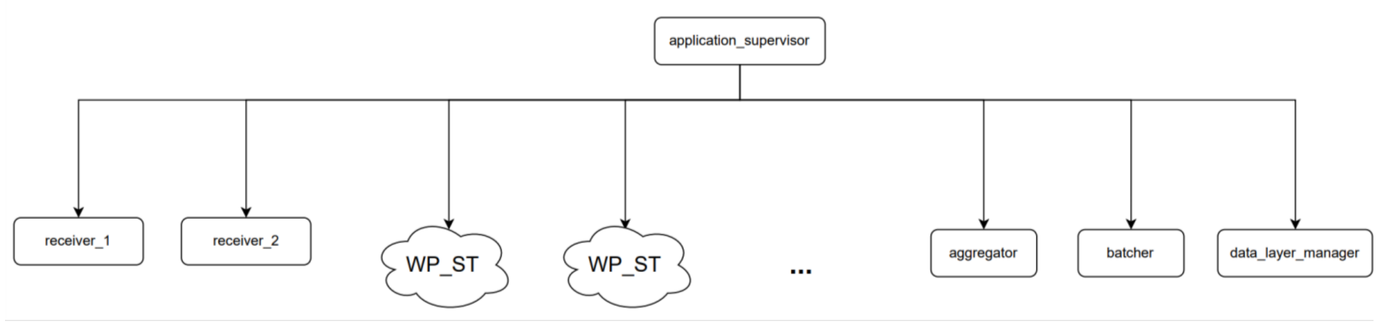
PTR checkpoint nr.1

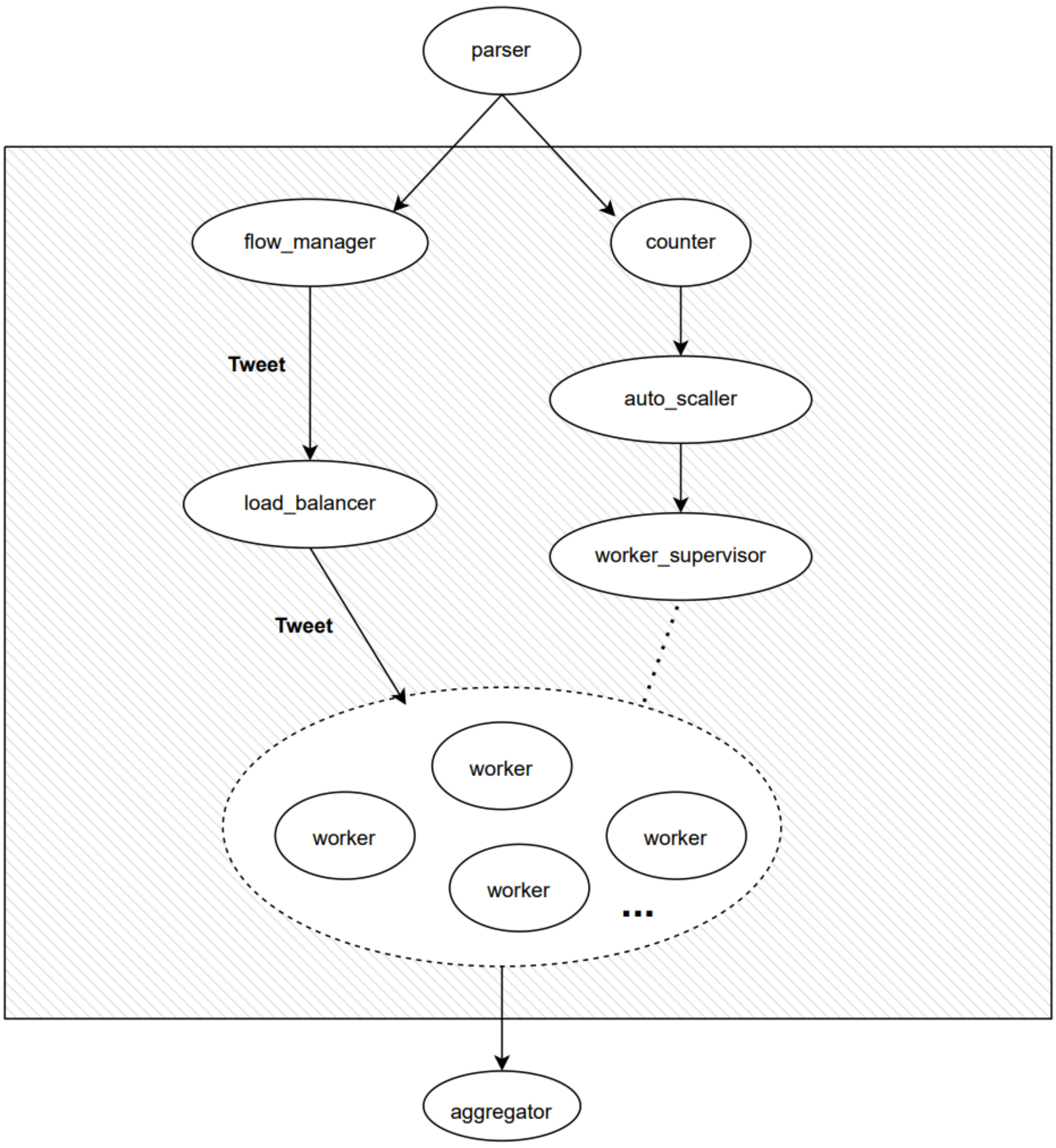
Supervision tree – worker pool (WP_ST):



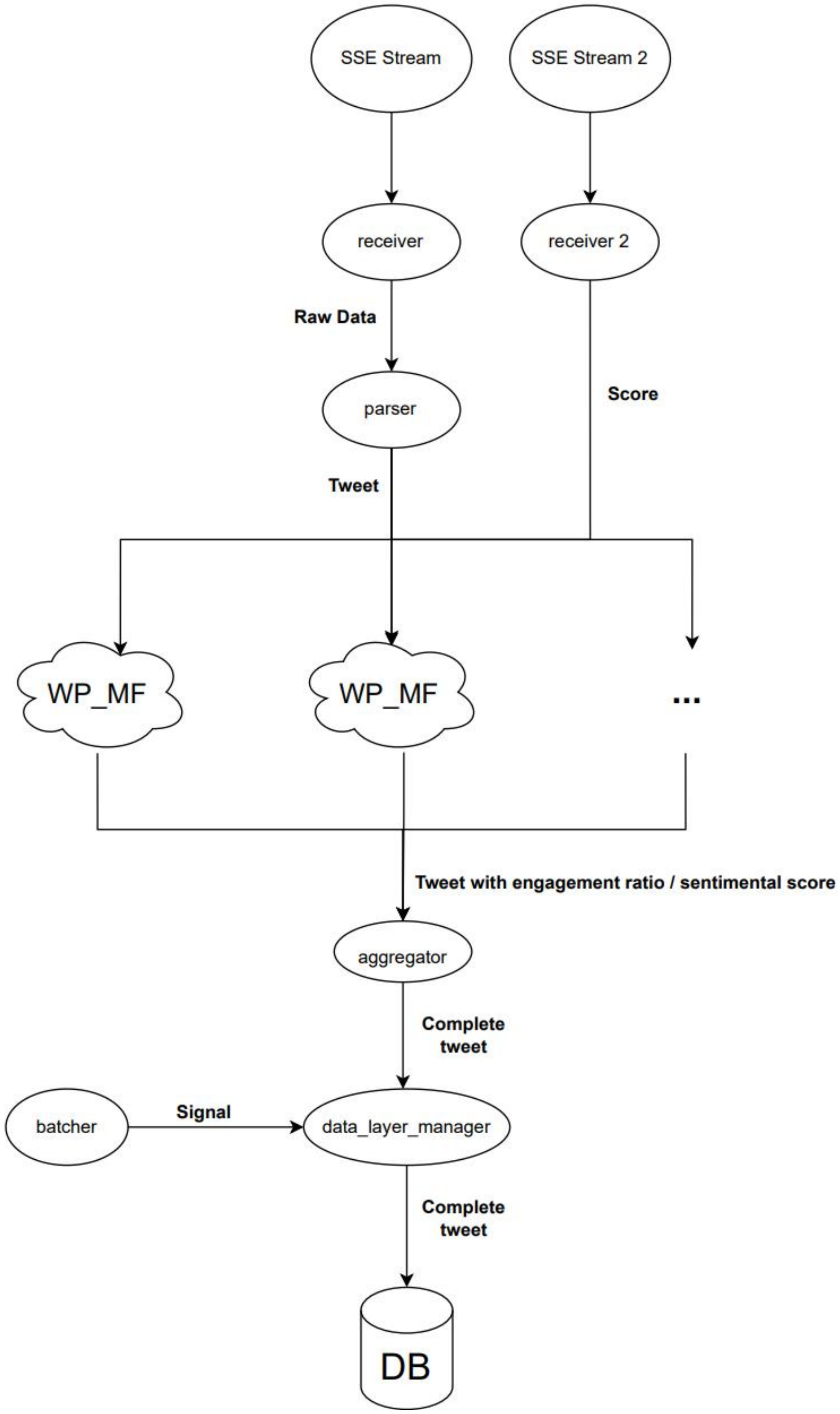
Supervision tree – whole system:



Message flow - worker pool (WP_MF):



Message flow – whole system:



Actors:

receiver:

- Receives data from the first SSE stream which provides tweets.

receiver 2:

- Receives data from the second SSE stream which provides scores.

parser:

- Converts raw data from receiver to structured map, before sending the tweets to each worker pool (most probably it will send the tweets to each pool using Round Robin).
- Notifies the counter about every new tweet.

worker_pool_supervisor:

- Supervises all the existing actors inside an worker pool.

flow_manager:

- Stores the PID's of all the existing workers from the worker pool.
- Acts as an intermediar between the parser and the load balancer.

counter:

- Counts the number of tweets per second.
- Based on the number of tweets per seconds and the current number of workers it computes the required number of workers to be added or removed. Implements that by communicating with the auto_scaller.

load_balancer:

- Distributes the tweets to the existing workers using Round Robin. It does that by storing the current index inside it's internal state. At each distribution, the index is increased and so on.

auto_scaller:

- Provides the functionality for adding or removing workers from the worker pool, by communicating with the worker_supervisor.
- Provides info about the current number of workers.

worker_supervisor:

- Supervises the workers inside an worker pool.

worker:

- Responsible for computing the sentimental score or the engagement ratio of a tweet, based on how it is configured.
- Can be of two types: engaged or sentimental.
- Sends the processed tweet to the aggregator.

application_supervisor:

- Supervises all the actors from the system.

aggregator:

- Creates the final tweets with both engagement ratio and sentimental score, ready to be stored.
- Extracts the existing users.

batcher:

- Signals the data_layer_manager based on the configured time window.

data_layer_manager:

- Stores the received tweets and users in the database.

Used language: *Elixir*

Short description + reflections:

In the process of doing the laboratory work nr.1 a lot of new actors have been added to the system, because I started to understand better the role of an actor. In the previous design there were actors with too many functionalities assigned to them. Now, I approve more the „principle" of many actors with less functionalities for each.

Talking about the flow in the system, now there will be 2 SSE streams - one providing the actual tweets and another one providing the score for the words. Thus, I have defined two receivers – one for each stream. The same parser remains, but now instead of sending the tweets to one actor, it will distribute with Round Robin to multiple actors, because currently the system must support multiple worker pools. Each worker pool will have it's own components for processing tweets and it will be possible to configure the worker pool for a specific type of worker: engaged or sentimental, by passing as a parameter the type.

The actual functionality for tweet processing, load balancing, auto scalling remains pretty much the same for each worker pool. The only difference is that now there will be multiple such components that will do this work in parallel. Component wise, there are also added the aggregator, the batcher and the daya_layer_manager. Their roles are specified in the actors section, but they mostly deal with the data layer of the system.