WS20/21 Distributed Systems-Test protocol Program: „ConnectedCars“

Project team: Date: 11.01.2021

Harijan, Stephan;
Kraus, Andreas;

The following tests were undertaken as part of the second milestone;

1. Functionality tests

a) Tested subject: Data arrives in the correct order
b) Tested subject: Correct data is received
c) Tested subject: Data packets are complete
d) Tested subject: Data transmission via MQTT protocol functions correctly
e) Tested subject: Data transmission via RPC functions correctly

- Method:

  - In order to test whether the sent and received data is complete, correct and in the right order as well as the correct functioning of the MQTT-RPC-data transmission, the system was run 20 minutes and its written database entries were analysed for discrepancies between identifier order and timestamp values as well as the values of the column "value".
  - The test data can be found under: Test/distance.csv || Test/fuel.csv || Test/speed.csv || Test/traffic.csv

- Results :

  - All data was received in the correct order, which can be seen by the order of the identifieres of entries in the database. Thus confirming the correct transmission of the data via the MQTT-protocol and RPC data transmission.
  - The comparison of the expected values of the received data showed no unexpected occurrences thus confirming the correctness of the received data.
  - Each data entry should consist of the sensor type, its identifier, value and the timestamp (here: receiving time). The comparison of the expected values of the received data showed no unexpected occurrences thus confirming the completeness of the received data.

f) Tested subject: Unexpected sensor failure or sensor disconnection and subsequent sensor restart

- Method:

  - The process of the sensor was killed via the command line command "sudo kill <PID>". After 2 minutes the killed docker image was restarted.

	type	identifier	value	timestamp
1	TRAFFIC	1	LOW TRAFFIC	2021-01-08 17:37:00
2	TRAFFIC	2	JAM	2021-01-08 17:37:03
3	TRAFFIC	3	LOW TRAFFIC	2021-01-08 17:37:06
4	TRAFFIC	4	NO TRAFFIC	2021-01-08 17:37:09
5	TRAFFIC	5	JAM	2021-01-08 17:37:12
6	TRAFFIC	6	LOW TRAFFIC	2021-01-08 17:37:15
7	TRAFFIC	0	JAM	2021-01-08 17:39:36
8	TRAFFIC	1	NO TRAFFIC	2021-01-08 17:39:39
9	TRAFFIC	2	HIGH TRAFFIC	2021-01-08 17:39:42
10	TRAFFIC	3	LOW TRAFFIC	2021-01-08 17:39:45
11	TRAFFIC	4	HIGH TRAFFIC	2021-01-08 17:39:48
12	TRAFFIC	5	JAM	2021-01-08 17:39:51

Screenshot of the database entries after sudden disconnection and restart of the sensor traffic

- Results:

  - The restart of a sensor leads to the reinitialization of the identifier thus leading to confusing and double identifier values.

--> Reaction/Consequence:

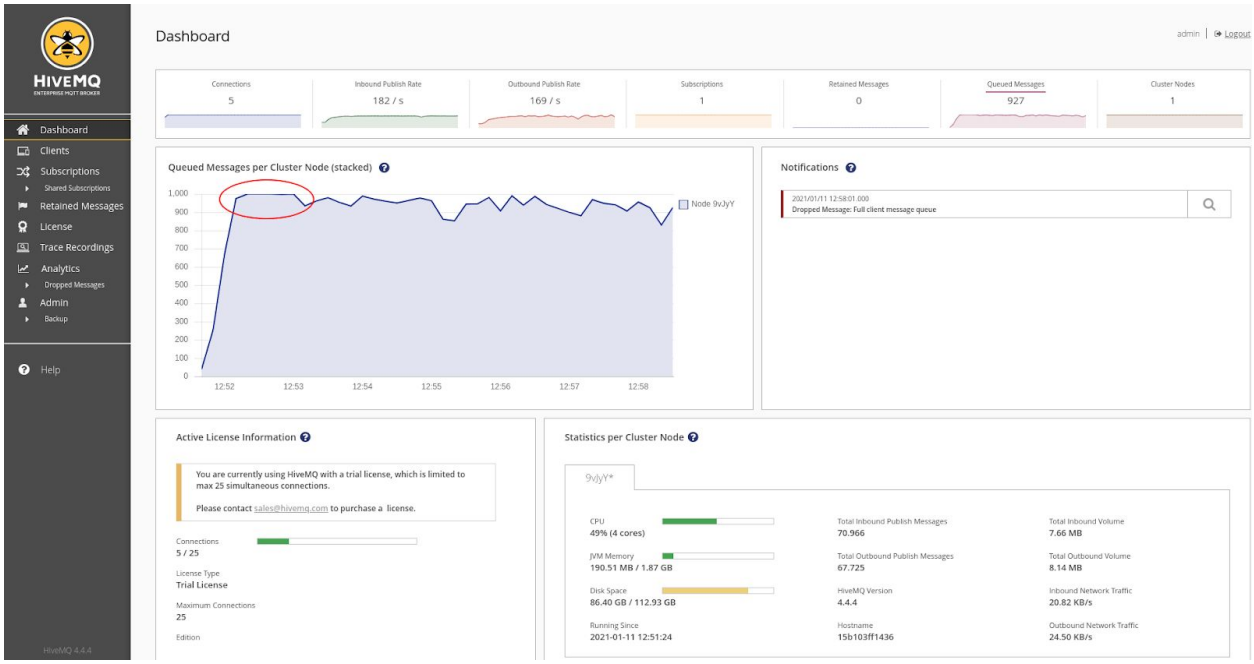
  - The currently used identifier is proven to be faulty or rather misleading if the sensor restarts unexpectedly. At the same time for the purposes of this course an identifier that is incremented by the value of one is an easy to handle method for the testing of correctness of transmitted data. As a consequence of these opposing benefits there was no refactoring undertaken, but the primary key was changed to a combination of identifier and DB-timestamp thus making the PK unique and at the same time keeping the easily accessible identifier.

2. Performance test

Tested subject: Transmission quality in dependence of sending interval of the sensors.

- Method:
  - Varying the sending interval of the sensors the data entries in the database were analyzed for completeness and correct order and data integrity.
  - Collected data was saved as .csv file with the filename describing the sample parameters (e.g. distance_20ms_SendIntervall: Values are from table distance, sending interval was 20ms); Additionally the screenshot SendIntervall_20ms_BrokerLimit.png shows the data transmission from the viewpoint of the broker.
- Results:

- All measurements showed correct order and data integrity regardless of the sending interval (e.g. see fuel_1ms_SendIntervall.csv
- While with a sending interval of 25ms all data was received in full, this wasn't the case with a sending interval of 20ms. Here it could be observed that data deletions occurred. In fuel_20ms_SendIntervall.csv of the 1014 data packets sent only 1000 were received and submitted to the database. This result can be explained by analysing the broker statistics (see: SendIntervall_20ms_BrokerLimit.png). Here the marked interval of queued messages per cluster node shows the period in which the limit was reached and messages were subsequently dropped (as can also be seen in the Notifications section on the right).



Screenshot SendIntervall_20ms_BrokerLimit.png showing the data transmission from the viewpoint of the broker.

--> Reaction/Consequence:

- One possible way to circumvent the sending limit of 20ms interval could be to increase the maximum capacity of the Mqtt-Broker message queue. This would not increase the transmission speed but insure the completeness of submitted data into the database, because the increased limit of buffered data would prevent the dropping at this sending interval. One side effect though could be a noticeable delay between data transmission by the sensors and data writing into the DB.
- Because there is no sending interval specified in the requirements and the measured limit of 25ms is within the self set interval range (3s) no changes to the maximum queuing capacity of the Mqtt-Broker were undertaken. Instead the sending interval of 25ms was noted as the maximum sending interval which still guarantees data transmission without any loss of data packets.