University of Vienna
Faculty of Computer Science
Research Group Software Architecture

Assignment 3

Distributed Systems

2016S – 050054 PR – Software Architectures

May 26, 2016

General Remarks

- You can get **15 points** for this exercise. If you do not attend the presentation talk, you will get **0 points** for the exercise. This means that you have to hand in this exercise **and** attend the presentation talk in order to pass this assignment.
- The **deadline** for this assignment is **June 13, 2016 at 23:59 CET**. No deadline extensions are given.
- This is a **group work**. You and your work group members are allowed to work together in solving this assignment.
- If you copy code or other elements from sources other than the lecture slides, please provide a reference to it in a comment above the corresponding entry.
- If you encounter problems, please post your question in the Moodle[1] discussion page. Alternatively, you can contact the tutors via swa.tutor@swa.univie.ac.at. As a last resort you can contact the course lecturer directly via swa@swa.univie.ac.at.

Submission Guidelines

All files required by this assignment have to be submitted to our GitLab[2] server into the proper project (repository) in the Submission and Feedback System[3]. If this task is an **individual work** assignment, you have to submit (commit,push) your changes and solutions in your **personal project** to the **2016s_swa_task3** branch. Otherwise, if this task is a **group work** assignment, you have to submit (commit,push) your changes and solution to your **work group project** to the **2016s_swa_task3** branch. For any questions regarding the **GitLab**-based task submission please refer to the Git[Lab Submission] Tutorial[4].

[1] https://moodle.univie.ac.at/course/view.php?id=50738
[2] https://gitlab.swa.univie.ac.at
[3] https://gitlab.swa.univie.ac.at/submission
[4] https://gitlab.swa.univie.ac.at/submission/tutorial

Assignment 3: Distributed Systems

Description

There are upcoming elections in the Federative Republic of Ruritania, and you have been hired to create an electronic system to monitor the course of the election and gather and tabulate the results. The election takes place on a single day, with polling stations opening at 7:00 and closing at 17:00. It is overseen by the Central Election Commission (CEC), which receives data from the electoral districts into which the country is divided. Each district in turn comprises several hundred individual polling stations. **Before** the voting process ends, the CEC publishes updates on voter participation – i.e. number of people who have voted vs. registered voters – for each district and nation-wide every 15 minutes. **After** 17:00, it gathers the election results – number of votes, votes for each party, blank and invalid votes – as they come in, and publishes them per district (including the percentage of polling stations these results represent) and per polling station, while also updating the nation-wide tally. News services, polling institutes, private citizens, etc. can subscribe to the CEC and receive its updates as they are published.

The system is composed of three main elements: the individual polling station, the electoral district, and the CEC. Because Ruritania is a large country and its communication infrastructure is poor, the individual polling stations communicate their data to the district server, which must then send a response that it received the data within 30 seconds. If the response does not come in that time, the polling station must re-send its data and continue doing so until their receipt is acknowledged. The district server bundles the data messages from the individual polling stations and forwards them to the CEC server, who in turn must acknowledge receipt, etc. The CEC must split the message bundles, aggregate the data from the polling stations per district, and publish an update for all of Ruritania, every district, **and** every polling station.

Take note that **before** 17:00, the system requires **all** polling stations and districts to report in at regular intervals of 15 minutes. If for whatever reason new data from a polling station fail to arrive in time, the district will re-send the previous message; if this happens more than two consecutive times, an error message must be displayed. The same applies to the CEC – district communication. **After** 17:00 however, the polling stations report only **once**, i.e. when they have finished their vote count. The district server will therefore only expect and aggregate the messages from the stations that have finished their count, and forward only them, again every 15 minutes, to the CEC. Naturally, the district server will also have to make sure that there are no network problems, so it will have to regularly ping the stations that have not yet reported in. The CEC will then have to tally the votes and publish the results nation-wide, district-wide, and per polling station, until all districts have reported in, when the final result is published.

Implementation

Implement the above-described system. Use **Message Queues** for the communication between individual polling stations, the electoral district, and the CES. The districts must use an **Aggregator** to combine the incoming messages and send a single aggregated message to the CEC. The CEC must decompose this message by using a **Splitter**. After making its calculations, the CEC must publish its results according to the **Publish-Subscribe** pattern,

with subscription topics being nation-wide data, district data, and individual station data, for both electoral participation and vote counts.

For the concrete implementation of the required Enterprise Integration Patterns (EIPs) you can use Apache ActiveMQ open source messaging server. Apache ActiveMQ supports all required patterns via the Apache Camel library[5]. Please study the online documentation on how to create an Apache Camel application[6], create the required communicating components as endpoints, add routes to connect those endpoints, and start Camel-internal threads to process messages. If you need any additional details to implement the required EIPs that are not given in the text above, you can adopt any reasonable logic.

Your final solution must contain an example instance with *at least* two districts, each having at least three polling stations, as well as *at least one* subscriber who can un/subscribe to and receive the CEC results. Give careful thought to your data models. The exercise is about the integration and interaction of various EIPs in a distributed system. Neither any elaborate visualization of the results nor persistent storage of data in databases etc. are required.

Submission

Your submission will consist of:

- All source code files directly in your GitLab Work Group Project in the **2016s_swa_task3** branch. Please **do not** use any archives like `.zip`, `.7z`, `.rar` etc.
- The submission must also contain a document file (`.txt`, `.md`, or `.pdf`) documenting and explaining your assumptions, thought process, and decisions while working through the assignment, as well as how to initiate the program and check the instances. Lack of the documentation file will result in the **immediate deduction of 3 points** from your final score for this assignment.
- Keep in mind that this is a **group work** assignment, but each member of the **work group** should know all the details about the submitted solution. If the solution of this task required a division of labour between the group participants, please note this in the documentation file.

[5] http://activemq.apache.org/enterprise-integration-patterns.html
[6] http://camel.apache.org/book-getting-started.html