Distributed Node-red Flows

Middleware Technologies for Distributed Systems

Chiara Marzano
Massimiliano Nigro
Daniele Paletti

29/09/2021

Contents

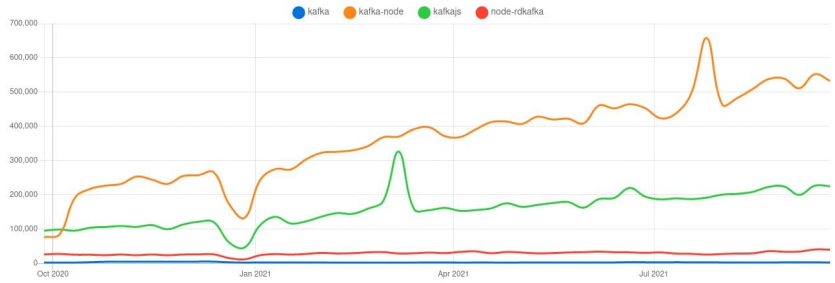


Figure 1: Usage of Kafka clients over time.

1 Overview

This project implements two custom Node-red nodes, Producer and Consumer, that allow distributed flows between two or more devices.

2 Technology choices

Following the specifications of this project, we decided to develop this project using **Apache Kafka** for the backend and **KafkaJS** for the custom nodes. The decision to use MPI against Apache Kafka is based mainly on the following reasons:

- Space decoupling allows to dynamically add components without needing to restart the configuration. As stated in the project requirements the set of running Node-red installations to be dynamic, this solution fits the guidelines perfectly.
- On the other hand, AKKA provides best-effort communication as messages are sent at most once. This type of communication does not match the requirements of the project of the previous point.
- Actors in AKKA provide a wide range of actions that are not required nor needed in this project such as editing the internal state or behavior.
- Kafka stores topics centrally and durably, therefore providing fault tolerance, while AKKA only stores the state of each Actor in distributed fashion.

Specifically, we decided to employ KafkaJS as client as it is among the most recently updated and widely used as shown in Figure 1.

3 Assumptions

Besides the assumptions listed in the project specifications, we assume to **know the address of the broker**, which is passed to our custom nodes upon creation.

Brokers employ ngrok to expose themselves publically and be reachable by flows.

4 Structure

To allow distributed flows between two or more devices, we built two custom nodes, Producer and Consumer, that employ Kafka's event-based communication to communicate over multiple distinct flows. A broker receives the publishes of producers and sends them to all the subscribed consumers.

4.1 Producer

Producers are nodes with one input and no outputs that forward messages received from the flow to the broker.

4.2 Broker

The **broker** receives messages from a producer and publishes them, i.e. forwards the to all consumers subscribed to the related topics.

4.3 Consumer

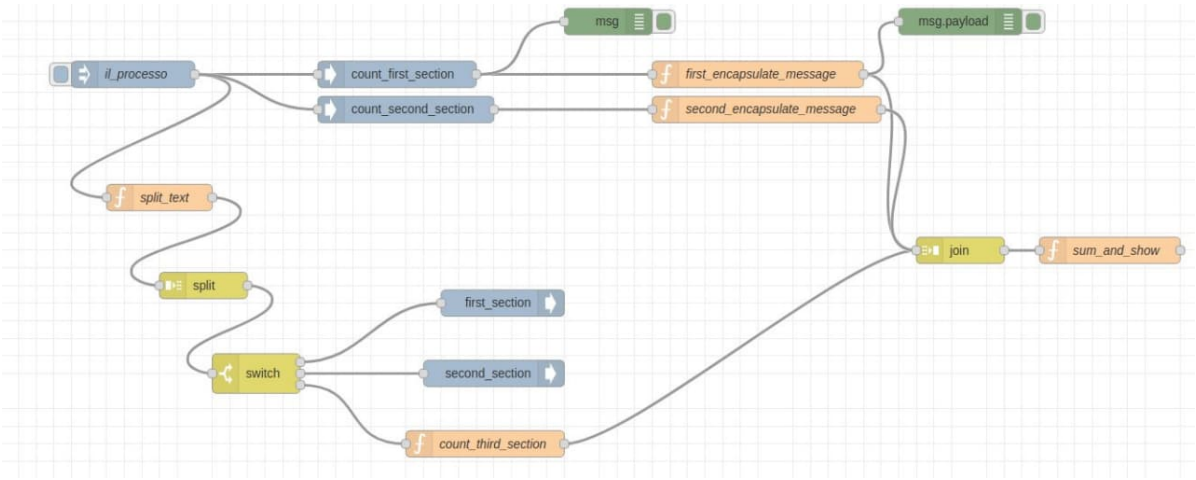
Consumers are nodes with one input and one output that subscribe to one or more topics and receive messages of interest from the broker.

5 Design choices

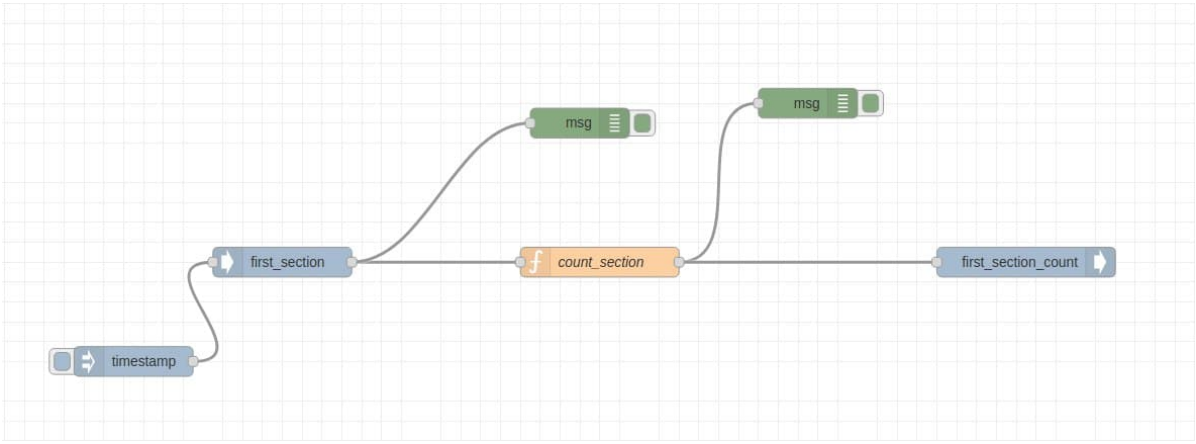
We decided to develop custom nodes instead of employing existing ones mainly for educational reasons as we were interested in learning how to work with such a flexible and adaptable tool. Our decision was reinforced by the fact that using existing nodes would have complicated the generation, readability and usability of the solution.

6 Example

To provide a working example of our solution, we built a simple distributed application in which we **count the occurrences of a letter** in the chapter of a book. The coordinator receives as input the chapter and divides it among the workers by publishing it. Each worker has a consumer that receives a substring, it computes the number of occurrences and then publishes with topic "count". The coordinator collects the local counts through a Consumer subscribed to "count", joins the data and computes the final result.



(a)



(b)

Figure 2: (a) Average absolute execution time. (b) Average execution time per day in simulation.