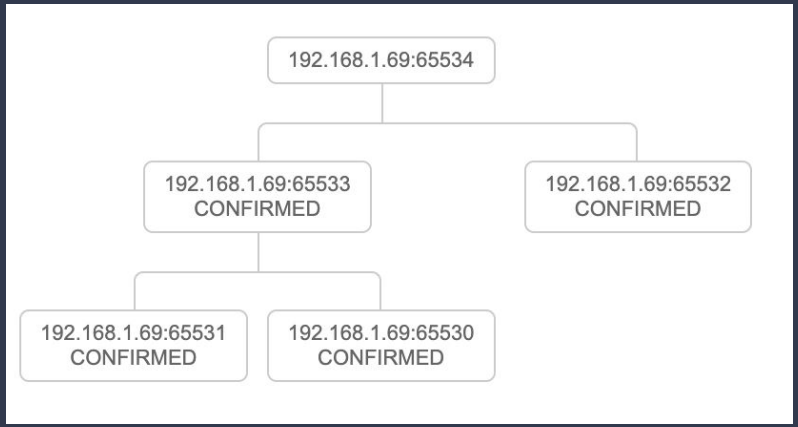
Broker tree management

Exam project for: Architetture dei Sistemi Distribuiti

Tuesday, 7th June 2022

powered by Simone Pio Caronia & Paolo Ruggirello

General idea



The purpose of the project is to handle a tree network of brokers.

Characteristics:

SCALABLE

FAILURE RESISTANT

TRANSPARENT FOR CLIENTS

Main entities

To reach the characteristics presented before, 3 different entities have been used:

- **SUPERVISOR**: it's the entry point of the system. It handles and manages the tree and lets clients connect to brokers;
- **BROKER**: it takes part of the tree network and lets clients communicate to each others, subscribing to topics and sending messages;
- **CLIENT**: developed friendly CMD and GUI clients that interact with supervisor and brokers.

Supervisor

Supervisor is responsible for handling the entire network. It's the entry point both for brokers and clients.

It exposes a REST server with the following endpoints:

- GET /tree - /jtree: they return a well designed and json version of the current tree;
- POST node/register: it's called by new broker that wants to join the network. Father <ip:port> is returned;
- POST node/confirm: it's called by a father node that notifies to supervisor new node;
- POST node/down: it's called by node down's neighbours and let it be removed by tree;
- GET /broker: endpoint called by clients in order to retrieve an available broker to communicate with.

Supervisor also exposes a TCP server that listens on root tree state changes.

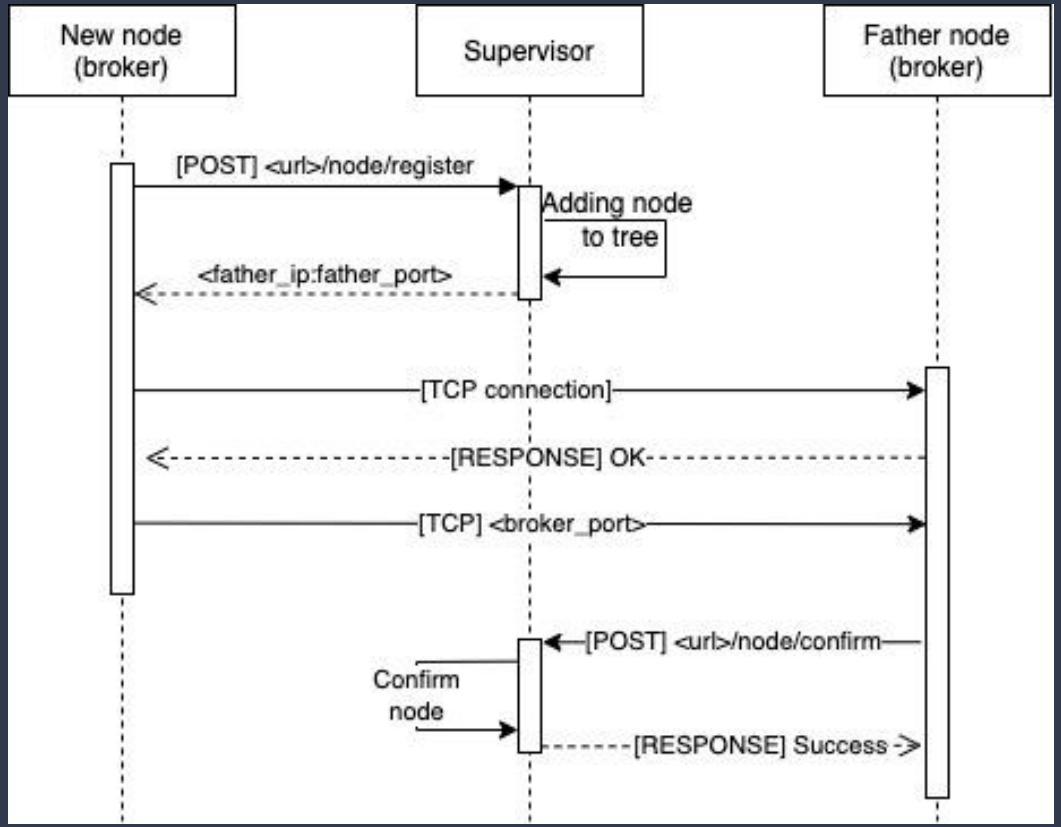
How a new node is added to network?

The addition of new node is composed by 2 parts:

1. The new broker asks for a father node to connect. It invokes the endpoint POST node/register exposed by supervisor.
   In this step the supervisor knows that a new node wants to connect to the network but it isn't yet connected to any of existing nodes;

2. The father node, after it receives the connection of broker and [PORT] command, confirms to supervisor that new node is actually connected to network, calling the endpoint POST node/confirm.
   In this step, the child's status is changed to CONFIRMED and it's ADDED also to TREE STRUCTURE, making it available for new brokers and clients.

New Node – Sequence diagram



Broker

```
Connection established with: ('172.20.10.2', 57803)
Started connection manager for 172.20.10.2:57803...
Waiting for connections ...
Broker 172.20.10.2:65530 confirmed successfully!
Received status code: 200
Started connection manager for 172.20.10.2:65533...
Broker UP (port: 65530 - PID: 26945)
Waiting for connections ...
```

Broker is the component of the entire system that is able to accept new clients and brokers connection, sending them a confirmation message => *[RESULT] OK*

It owns a TCP server, but each broker can also be seen as client of another broker.

An active connection is marked as BROKER if it sends the PORT command indicating its server port. At this point, the father broker also confirms to supervisor the new child broker.

Other available commands:

SEND - USER - SUBSCRIBE - UNSUBSCRIBE

What happens when a broker node crashes?

If a broker node crashes, the system react in the following way:

- its father broker node notifies that the broker crashes and supervisor removes the relationship between them, calling endpoint POST node/down;
- if the broker has children, they notify as well (same endpoint) the dead node and relationship is removed. The children request for a new father to supervisor and reconnect to network;
- When all relations are removed from dead broker, it is completely erased from tree structure.

Clients

Clients implemented communicate with brokers and supervisor and give user a friendly experience. They wrap correctly the values given by user using the protocol created.

Each client message sent to broker follows this pattern:

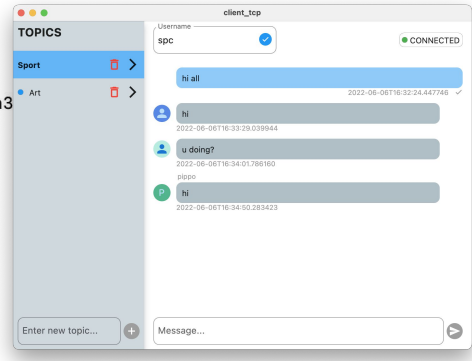
[command] {“key”: <value>}

Interfaces of the 2 versions:

```
(venv) caroniasi@mac-caroniasi adsd-spc-pr % python3

Welcome on client. Make sure SUPERVISOR is up!
To start connection invoke – connect
Documentation is available typing <help>

>
```

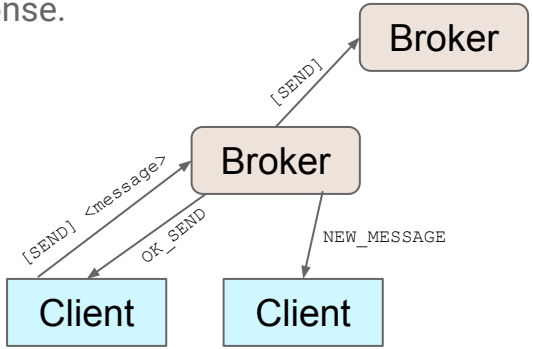


How do they communicate each other?

Each client asks for an available broker to supervisor, which distributes clients between all brokers in the tree.

After subscribing to a topic, client sends message specifying to which topic it has to be published. To let clients connected to different broker receive correctly the message, this one is forwarded in the network tree.

Each broker notifies client's connection if client is subscribed to that specific topic with a NEW_MESSAGE notification type and all brokers' connections, wrapping in this case again the message with a SEND command. After every command received, the broker sends a confirmation/error response.



… so, summing up

So, let's analyse again the 3 characteristics presented at the beginning:

SCALABLE: system developed is scalable because each broker can accept new broker's connection and so tree structure can increase its dimension.

FAILURE RESISTANT: even if some brokers fail, the children brokers are able to reconnect to network. The system is reliable and able to resist to crash failures. In order to increase the availability of the system, some improvements have been developed such as release locks during arbitrary fails, intercepting all exceptions.

TRANSPARENT FOR CLIENTS: the reconnection system is hidden to clients and also clients connected to different broker can communicate each other, as well as they do within the same broker.

Thanks for your attention!

Simone Pio & Paolo