Enabling QoS-Aware Task Execution on Distributed Node-RED Cluster for Fog Computing Environments

Project Definition

Internet of Things (IoT) evolves over all the world and the number of the connected devices to the Internet have been considerably increasing. The burden of the exponentially augmenting connections cannot be easily handled by the current cloud architectures when QoS of each service is considered. Fog computing is proposed to address these issues by extending the cloud technology towards end-users and aims at providing local resources as a computation, storage and network resources to the users in the close proximity. Fog networks are mostly composed of various devices that have different characteristics and resource capacities, which lead to not have a standard QoS parameters while running a service. Even some services cannot be easily run on some nodes due to the scarce of the resource. One of the solution would be the decomposition of services in small tasks, which can be executed on the various resource-capable fog nodes. The introduction of distributed Node-RED program [1], which enables the distributed execution of a tasks (functions in a flow) of a service (flow) on the different nodes, can help the developers in designing fog-enabled services/applications. The challenging problem is here to provide an acceptable QoS that meets the end-user requirements while running the small parts of the service on different hardwares. The main aim of this thesis is to distribute dynamically the tasks of a service based on service QoS requirements.

The modules of this work can be listed as below:

- Containerisation and Deployment of Distributed Node-REDs and Functions
- Service Composition Development Tool
- QoS-Monitor & -Orchestrator
- Implementation of A Resource Allocation Algorithm

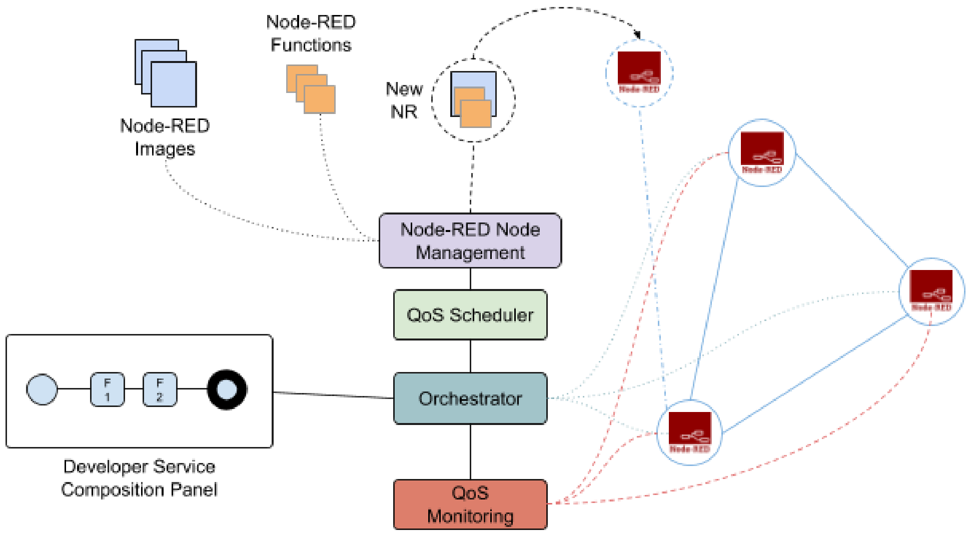

Figure1: QoS-Aware Distributed Node-RED Architecture

Containerisation and Deployment of Distributed Node-RED and Functions

The distributed and dynamic deployment of the node-RED instances plays a crucial role in case having a dynamic network nodes. A container-based approach [2] can enable this dynamic deployment. The function set of the Node-RED should be also stored in a database and in case of need those functions should be moved from the storage on top of a node [3]. The distributed node-RED[distributed node-red] should be able to communicate each other [4].

Service Composition Development Tool

A service composition panel that covers the management of distributed node-RED nodes is a requirement for the service developers. Using a similar concept as indicated in [dnr], services should be seen without their server-id labels. A service developer should be able to select functions regardless of knowing which function belongs to which node-RED. A list of services that are offered by the node-reds should be shown. Using them, a service composition should be constructed and the required QoS parameters such as bandwidth, delay should be given for the composed service.

QoS-Monitor & -Scheduler & -Orchestrator

In order to run efficiently the created composite services on the distributed node-red networks, a qos-orchestrator and qos-monitor are required. The former one would be responsible for the selection of the node-RED nodes and distribution of the node-RED functions if required. The latter one would be in charge of monitoring the identified QoS parameters between node-RED instances. The connectivity quality or the availability of the resources such as cpu, memory or storage can be seen as the required parameters.

Each node-RED node should also include a qos-agent that measures the resources and connectivity quality with its neighbours as well as sharing them with the central qos-monitor component.

Using QoS-Monitor data, QoS-Orchestrator ought to be able to change the place of the node-RED function or select another available node-RED function. This decision will be triggered by the observation of the continuous QoS parameter measurements in the network.

QoS-Scheduler determines which service and when it should be executed. This component plays a crucial role in the decision of QoS-Orchestrator

Implementation of A Resource Allocation & Scheduling Algorithm

As mentioned above, the selection of node-RED nodes is operated by the qos-orchestrator. The decision mechanism will be depends on an implementation of some of the resource-allocation algorithms. A thorough literature review for the available resource allocation algorithms in the distributed heterogeneous networks should be carried out.

The increasing number of the composed services may not be directly executed on the node-RED nodes due to either the scarce of the resources or the non-satisfying QoSes. The execution order of the services should be managed a scheduler that may consider the priority, the complexity, etc. The existing scheduling algorithms need to be analyzed and one of the suitable algorithm should be implemented.

Main Requirements

- Must: Java, Python and Javascript Programming Languages
- Desired: Having experience with Node-RED

References

1 - https://github.com/namgk/dnr-editor
2 - https://hub.docker.com/r/nodered/node-red-docker/
3 - https://randomnerdtutorials.com/exporting-and-backing-up-your-node-red-nodes/
4 - https://cookbook.nodered.org/mqtt/connect-to-broker