IOT Bridge Platform

By: Mariana Oliveira Ramos

University of Bologna, Internet of Things, 2021/2022

Requirements:

In the command shell run:

```
npm install yargs --save
npm install coap --save
npm install mqtt --save
npm install axios --save
npm install express --save
npm install --save @influxdata/influxdb-client
```

Run the project:

```
node ./bridgeIoT.js --help
```

Features:

Visualization

1. Initialize the data simulation:

```
node ./<protocol>_simulator.js
```

2. Visualize responses:

```
node ./bridgeIoT.js -p <protocol> -h <host_address> -t <topic> -c visualize
```

Note:

- Simulator of HTTP runs on port: localhost/3001
- Simulator of COAP runs on port: localhost/5683
- Simulator of MQTT runs on port: localhost/1883, default of Mosquitto

Examples:

Visualize MQTT protocol

```
> node ./mqtt_simulator.js
> node ./bridgeIoT.js -p "mqtt" -h "localhost" -t "temperature" -c visualize
```

Visualize COAP protocol

```
> node ./coap_simulator.js
> node ./bridgeIoT.js -p "coap" -h "coap://localhost/temperature" -c visualize
```

Visualize HTTP protocol

```
> node ./http_simulator.js
> node ./bridgeIoT.js -p "http" -h "http://localhost/temperature" -c visualize
```

Aggregation

1. Initialize the data simulation:

```
node ./<protocol>_simulator.js
```

2. Visualize statistics computed every *n* observations:

```
node ./bridgeIoT.js -p <protocol> -h <host_address> -t <topic> -c aggregate -n <n>
```

Examples:

Visualize statistics MQTT protocol - every 5 observations

```
> node ./mqtt_simulator.js
> node ./bridgeIoT.js -p "mqtt" -h "localhost" -t "temperature" -c aggregate -n 5
```

Visualize statistics COAP protocol - every 5 observations

```
> node ./coap_simulator.js
> node ./bridgeIoT.js -p "coap" -h "coap://localhost/temperature" -c aggregate -n
5
```

Visualize statistics HTTP protocol - every 5 observations

```
> node ./http_simulator.js
> node ./bridgeIoT.js -p "http" -h "http://localhost/temperature" -c aggregate -n
5
```

Storage

- Inicialize InfluxDB: C:\Program Files\InfluxData> influxd. http://localhost:8086/
- Inicialize Grafana: http://localhost:3000 (Docker container), URL IP of Influx: 192.168.32.1, PORT: 8086

1. Initialize the data simulation:

```
node ./<protocol>_simulator.js
```

2. Store responses:

```
node ./bridgeIoT.js -p <protocol> -h <address> -t <topic> -i <file with influx
configuration> -c save
```

Examples:

Store MQTT protocol data

```
> node ./mqtt_simulator.js
> node ./bridgeIoT.js -p "mqtt" -h "mqtt://localhost" -t temperature -i
influx_conf.json -c save
```

Store COAP protocol data

```
> node ./coap_simulator.js
> node ./bridgeIoT.js -p "coap" -h "coap://localhost/temperature" -i
influx_conf.json -c save
```

Store HTTP protocol data

```
> node ./http_simulator.js
> node ./bridgeIoT.js -p "http" -h "http://localhost/temperature" -i
influx_conf.json -c save
```

Query example for Grafana in Flux:

```
from(bucket:"iot_bucket")
|> range(start: -1h)
|> filter(fn: (r) => r._measurement == "testmeasure" and r._field ==
"temperature")
```

Bridging

1. Call translate command to translate from Protocol1 to Protocol2:

```
node ./bridgeIoT.js -p <Protocol1> -h <Protocol1_address> -c translate -d
<Protocol2> -f <Protocol2_configuration_file>
```

2. Start data simulator of the Protocol1

```
node ./<Protocol1>_simulator.json
```

3. Make a request to Protocol 2 - for example trying to visualize data.

```
node ./bridgeIoT.js -p <Protocol2> -h <Protocol2_address> -c visualize
```

Examples:

Translate from COAP to HTTP

```
> node ./bridgeIoT.js -p coap -h coap://localhost/temperature -c translate -d http
-f "protocol_conf_files/http_conf.json"
> node ./coap_simulator.js
> node ./bridgeIoT.js -p "http" -h "http://localhost/temperature" -c visualize
```

Translate from HTTP to COAP

```
> node ./bridgeIoT.js -p http -h http://localhost/temperature -c translate -d coap
-f "protocol_conf_files/coap_conf.json"
> node ./http_simulator.js
> node ./bridgeIoT.js -p "coap" -h "coap://localhost/temperature" -c visualize
```

Translate from MQTT to COAP

```
> node ./bridgeIoT.js -p mqtt -h mqtt://localhost -t temperature -c translate -d
coap -f "protocol_conf_files/coap_conf.json"
> node ./mqtt_simulator.js
> node ./bridgeIoT.js -p "coap" -h "coap://localhost/temperature" -c visualize
```

Translate from COAP to MQTT

```
> node ./coap_simulator.js
> node ./bridgeIoT.js -p coap -h coap://localhost/temperature -c translate -d mqtt
-f "protocol_conf_files/mqtt_conf.json"
> node ./bridgeIoT.js -p "mqtt" -h "mqtt://localhost" -t "temperature" -c
visualize
```

Translate from MQTT to HTTP

```
> node ./bridgeIoT.js -p mqtt -h mqtt://localhost -t temperature -c translate -d
http -f "protocol_conf_files/http_conf.json"
> node ./mqtt_simulator.js
> node ./bridgeIoT.js -p "http" -h "http://localhost/temperature" -c visualize
```

Translate from HTTP to MQTT

```
> node ./http_simulator.js
> node ./bridgeIoT.js -p http -h http://localhost/temperature -c translate -d mqtt
-f "protocol_conf_files/mqtt_conf.json"
> node ./bridgeIoT.js -p "mqtt" -h "mqtt://localhost" -t "temperature" -c
visualize
```

Documentation:

- Report: https://pt.overleaf.com/project/62801b4af7dde22663c18e79
- Presentation: https://www.canva.com/design/DAFAtSgiSEY/ZcONNsmoU42DTMfDZmNSJA/edit
- Diagrams: https://app.diagrams.net/#G1Pz5bcnLAZsLoNUaJe29xQhDJ8Yi4LoKf