NavUP System

End user and technical documentation

Group name: Broadsword Data

Jacques Smulders - u15003087
Keegan Ferret - u15132847
Lesego Makaleng - u15175716
Linda Potgieter - u14070091
Lyle Nel - u29562695
Seonin David - u15063021

Contents

1 Data streaming module

1.1 Scope and responsibilities

The data streaming module serves as an intermediary between the Aruba analytics engine and the rest of the subsystem. The module is mainly concerned with serving requests about the location of a device visible to Aruba.

1.2 End User

This section is intended for system administrators and anyone that needs to deploy this subsystem.

1.2.1 Installation and dependencies

To satisfy dependencies run the following shell command:

```
$ ./integration_setup.sh
```

1.2.2 Configuration

All configuration of the data subsystem is done as command line arguments to the main entry point of the system, query_resolver.py. The arguments are as follows:

```
usage: query_resolver.py [-h] [--nsqlookupd_hostname] [--nsqlookupd_port]
                         [--nsqlookupd_polling_interval] [--nsqd_hostname]
                         [--nsqd_port] [--subscribe_topic] [--nsq_channel]
                         [--aruba_hostname] [--aruba_port] [--aruba_username]
                         [--aruba_password]

Serving location requests on the an NSQ topic

optional arguments:
  -h, --help            show this help message and exit
  --nsqlookupd_hostname
                        The http hostname of the NSQ lookupd daemon (default:
                        http://127.0.0.1)
  --nsqlookupd_port     The port number of the NSQ lookupd daemon (default:
                        4161)
  --nsqlookupd_polling_interval
                        The amount of time in seconds between querying all of
                        the supplied nsqlookupd instances. A random amount of
                        time based on this value will be initially introduced
                        in order to add jitter when multiple readers are
                        running (default: 15)
  --nsqd_hostname       The http hostname of the NSQ daemon (default:
                        http://127.0.0.1)
```

```
--nsqd_port           The http port of the NSQ daemon (default: 4150)
--subscribe_topic     The nsq topic to listen for requests (default: data)
--nsq_channel         The channel within an NSQ topic to listen for requests
                      (default: navup)
--aruba_hostname      The hostname of the aruba location engine (default:
                      https://137.215.6.208)
--aruba_port          The port of the aruba location engine (default: )
--aruba_username      The username to use the aruba location engine
                      (default: )
--aruba_password      The username to use the aruba location engine
                      (default: )
```

1.2.3 Troubleshooting

Any malformed queries made to the data subsystem are logged in a file called error.log for diagnostic purposes. In addition, any connection issues to either the ALE or NSQ server is also be logged.

1.3 Technical

1.3.1 API

The API provides a response to a getLocation request made to the data topic on the NSQ Server. Furthermore, the request can be made by any subsystem. The message contained in both the request and response is a JSON String with the formats provided below.

The Request

```
{
    "src"  : <any subsystem can be the source of the request>,
    "dest" : "Data",
    "msgType" : "request",
    "queryType" : "getLocation",
    "content" : {
                    "mac" : "FF:FF: . . . . "
                }
}
```

The Response

```
{
    "src"  : "Data",
    "dest" : <the same system that made the request>,
    "msgType" : "response",
    "queryType" : "getLocation",
    "content" : {
```

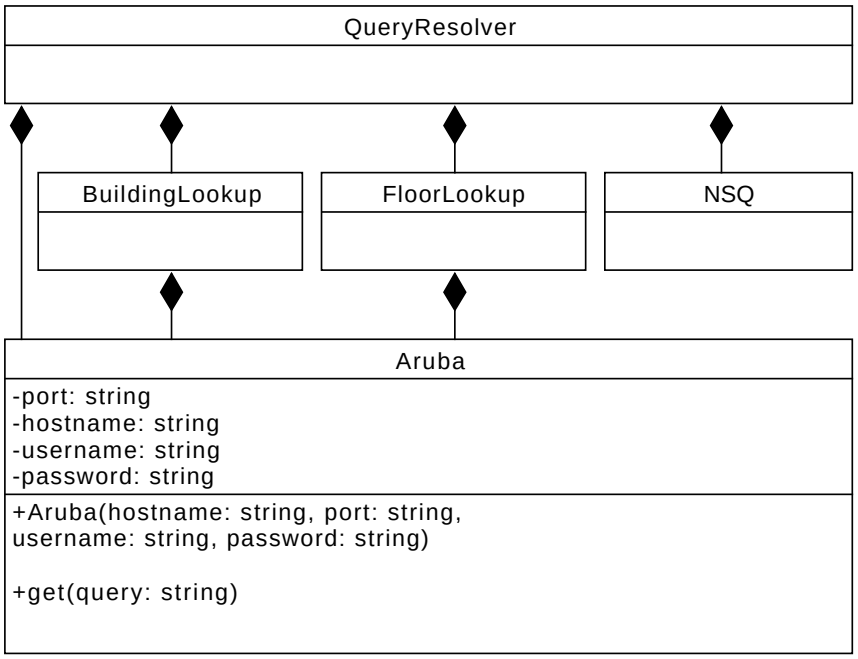

Figure 1: How it fits together

```
                    "success" : <boolean>,
                    "error" : <reason for error>,
                    "coordinate" : <X,Y coordinates relative to the access point>,
                    "floor_name" : <floor identifier>,
                    "building_name" : <building identifier>
                }
}
```

1.3.2 Implementation details

Due to the change in requirements from a streaming subsystem to a request response subsystem, Apache Flink is no longer a suitable techonology. Instead, the requests are receieved and served through NSQ. The data subsystem consults the aruba location engine as needed to give up to date information to the relevant subsystems. The structure of the subsystem is purely composition as can be seen in figure 1 on page 4. In addition there is a very clean push pull relationship between interfaces of different components within and outside the subsystem as can be seen in figure 2 on page 5.

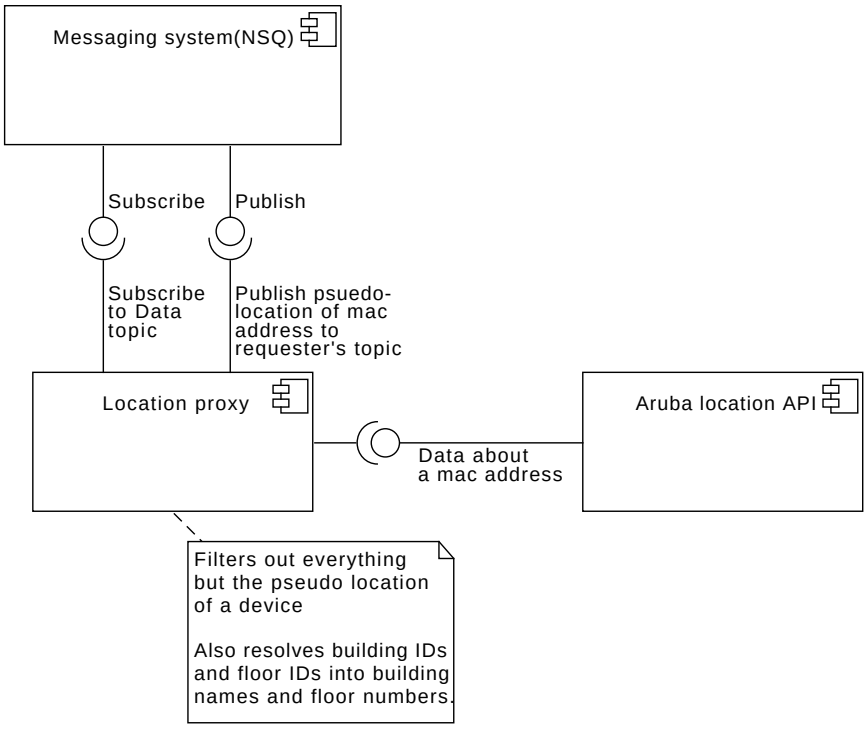

Figure 2: Overview of components