CPS Humidity Homework

Janos Benjamin Antal

January 14, 2018

Prerequisites

To build the application on Linux Mint 18.3 the following steps are needed:

- Install required packages with apt:
  - `libboost-all-dev`
  - `libcurl4-openssl-dev`
  - `libssl-dev`
  - `uuid-dev`
  - `rapidjson-dev`
  - `build-essential`
  - `cmake`
  - `g++`
  - `git`

  Use the following command:

  ```
  $ sudo apt install libboost-all-dev libcurl4-openssl-dev
      libssl-dev uuid-dev rapidjson-dev build-essential cmake g
      ++ git
  ```

- Install cURLpp with the steps described here:

  ```
  $ sudo apt-get remove libcurlpp0
  $ mkdir curlppbuild
  $ cd curlppbuild
  $ git clone https://github.com/jpbarrette/curlpp.git
  $ cd curlpp
  $ cmake .
  $ sudo make install
  ```

- Install AzureIoT C SDKfrom source. After building the SDK use `sudo make install` to copy the header and the lib files to the system include path.
- Download and install RTI Connext DDS 5.3
- Set the `NDDSHOME` environment variable to the root of RTI Connext DDS, e.g.: `/opt/rti_connext_dds-5.3.0`

Build

1. Clone git repository

   ```
   $ git clone https://github.com/antaljanosbenjamin/
       cps_homework.git
   ```

2. Generate source code from .idl files

   ```
   $ cd cps_homework
   $ $NDDSHOME/bin/rtiddsgen -language C++11 -stl -d DDS/Config
       /common -replace idl_files/Config.idl
   $ $NDDSHOME/bin/rtiddsgen -language C++11 -stl -d DDS/
       Decision/common -replace idl_files/Decision.idl
   $ $NDDSHOME/bin/rtiddsgen -language C++11 -stl -d DDS/
       Schedule/common -replace idl_files/Schedule.idl
   $ $NDDSHOME/bin/rtiddsgen -language C++11 -stl -d DDS/
       Humidity/common -replace idl_files/UvegHaz.idl
   $ $NDDSHOME/bin/rtiddsgen -language C++11 -stl -d DDS/
       Weather/common -replace idl_files/Weather.idl
   ```

3. Build

   ```
   $ mkdir build
   $ cd build
   $ cmake ..
   $ make [-j 8]
   ```

Run demo

1. Start humidity publisher

   ```
   $ ./cps_main h <humidityDataFilePath>
   ```

   As result of this command the application will read the data file and start to publish a humidity value every 4 minutes. The file shall contains a humidity value per line. Each humidity value is a decimal number. See example file.

2. Start IoTEdge

   ```
   $ ./cps_main e <weatherApiKey> <azureConnectionString> <
       scheduleFilePath>
   ```

   The meaning of parameteres are the following:

   - `weatherApiKey`: an API key for http://api.airvisual.com
   - `azureConnectionString`: the connection string of the device used Azure IoT Hub
   - `scheduleFilePath`: path to a CSV file which stores the schedules time intervals. See example file.

   The IoTEdge module is responsible to communicate with Azure IoT Hub, the weather information system and also to send schedule information through DDS topic.

3. Start humidity controller

   ```
   $ ./cps_main c
   ```

   The controller receives the required informations and sensor values and also make decisions based on the collected data. It also sends the decision input and output to the IoTEdge in order to store them in the cloud.

System architecture

The system consists three modules:

- IoTEdge
- Controller
- Publisher

The publisher module exists only for testing purposes, so this documentation doesn't contains it's details.

IoTEdge

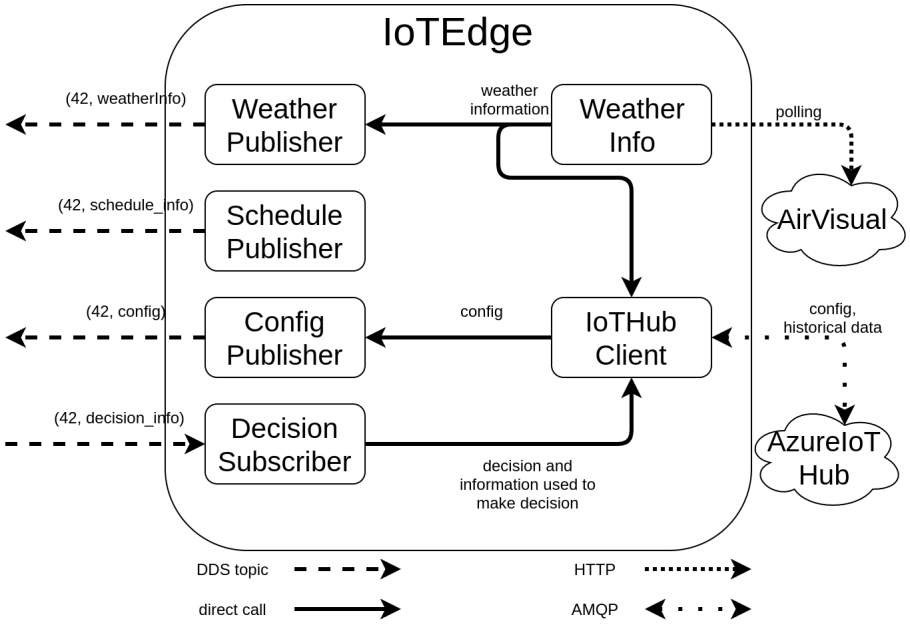

Figure 1: Architecture of IoT Edge

Figure 1 shows the semantic architecture of IoT Edge.