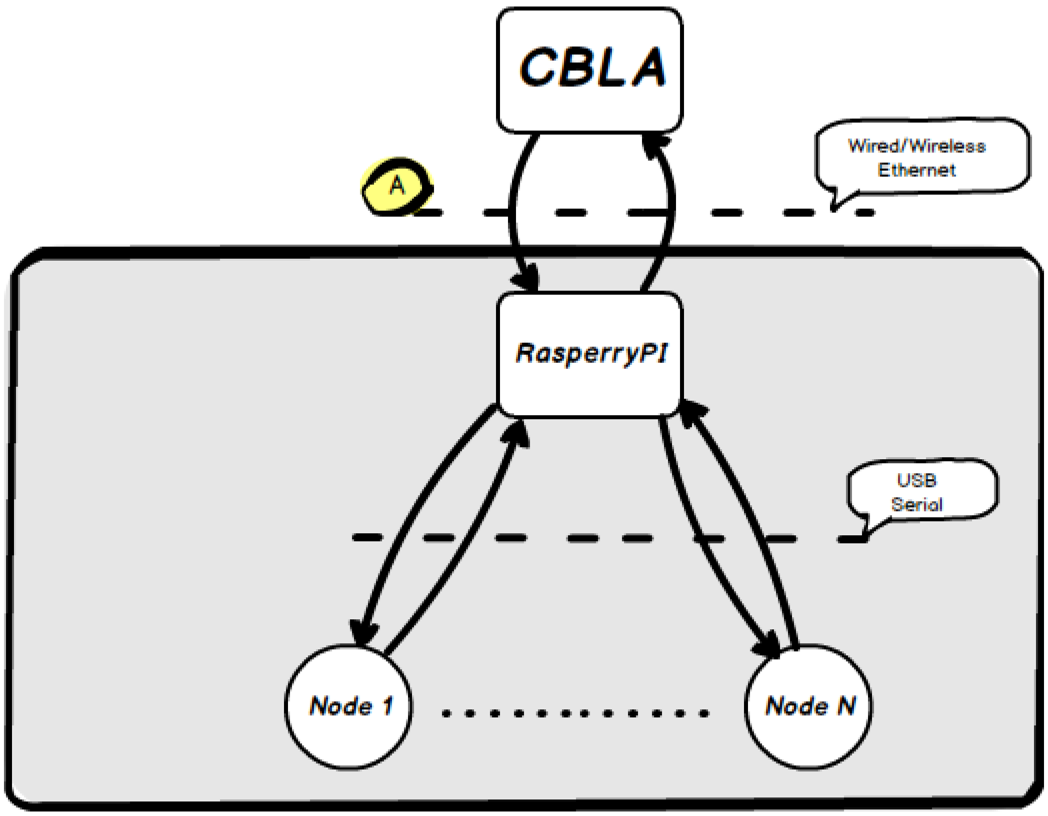
REST protocol proposal

Proposal for a REST-based communication protocol between CBLA and RaspberryPI/simulator

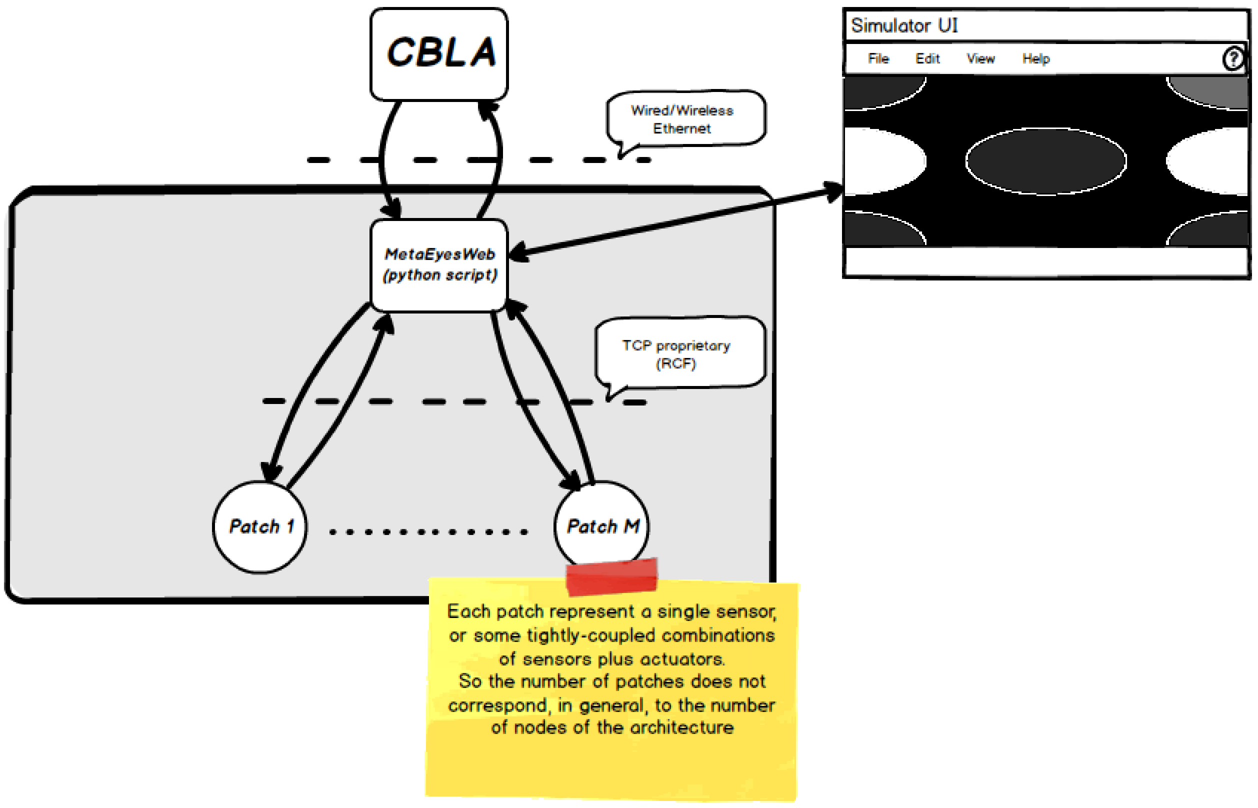
The following picture depicts the current system architecture (as I understood it), where the CBLA algorithm communicates with the software running on the Raspberry PI to get the system state and to provide control inputs:



And the communication I'm focusing on now is the one marked with the A symbol.

The communication protocol between CBLA and Raspberry, which is now undergoing some changes mainly due to code clean-up, is basically made of a set of commands sent by CBLA to get the status of sensors and to set the reference values of the actuators.

The simulator architecture is similar:

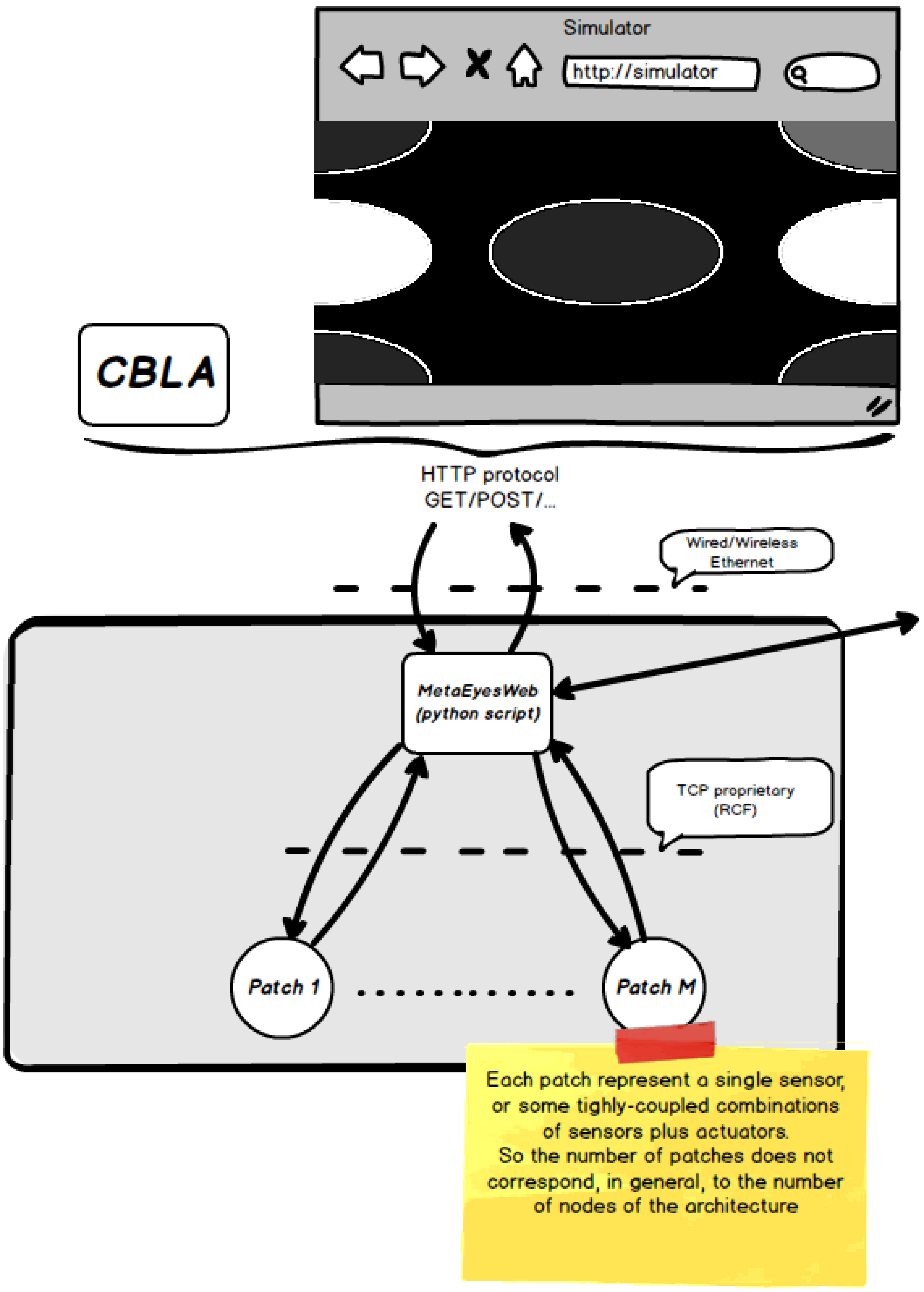

CBLA
Wired/Wireless Ethernet
Simulator UI
File Edit View Help
MetaEyesWeb (python script)
TCP proprietary (RCF)
Patch 1
Patch M
Each patch represent a single sensor, or some tightly-coupled combinations of sensors plus actuators. So the number of patches does not correspond, in general, to the number of nodes of the architecture

The simulator is based on a python application which coordinates several EyesWeb patches, each one implementing the simulation of a sensor or a group of sensors and actuators.

One of the requirements of the simulator is that it must provide a user interface. The interface should provide both access to graphs plotting the behavior of the state variables, or a more abstract interface showing a 2-dimensional representation of the sensors/actuators positions and states.

In the first simulator implementation this simple 2-dimensional interface is embedded in the python script. This implementation assumes that no communication protocol changes will occur. However, since the protocol is already undergoing some changes, a different architecture might be envisioned.

One proposal is the following:


Simulator
http://simulator
CBLA
HTTP protocol
GET/POST/...
Wired/Wireless
Ethernet
MetaEyesWeb
(python script)
TCP proprietary
(RCF)
Patch 1
Patch M
Each patch represent a single sensor, or some tighly-coupled combinations of sensors plus actuators.
So the number of patches does not correspond, in general, to the number of nodes of the architecture

The main difference are the followings:

1. UI is not embedded in the simulator but communicates with the simulator (or the real system) with the same protocol as CBLA
2. The protocol is based on http and relies on the REST guidelines

Point 1. enables using the same interface both for the simulator and for the real system, and could be useful as a sort of monitor tool of the ongoing activities (in the sense of sensors activations) occurring in a working installation.

Point 2. enables the use of very widespread techniques for the development of the UI, as it relies on standard techniques such as html/javascript. Moreover, if let a developer test the protocol even before developing the UI, as the http GET/POST methods can be easily invoked by the browser (the POST requires plugins available in all browsers), or by the linux CURL command.

As an example, pointing the browser to the following url:

```
http://localhost/api/sensors
```

could give the following response, i.e., the positions of all configured sensors/actuators.

```
{

 • infrared:

    [

     ○  {
         ▪ name: "ir1",
         ▪ type: "infrared",
         ▪ x: 0,
         ▪ y: 0,
         ▪ z: 0

        },

     ○  {
         ▪ name: "ir2",
         ▪ type: "infrared",
         ▪ x: 1,
         ▪ y: 0,
         ▪ z: 0

        },

     ○  {
         ▪ name: "ir3",
         ▪ type: "infrared",
```

        - **x**: 0,
        - **y**: 1,
        - **z**: 0

      },

    - {
        - **name**: "ir4",
        - **type**: "infrared",
        - **x**: 1,
        - **y**: 1,
        - **z**: 0

      },

    - {
        - **name**: "ir5",
        - **type**: "infrared",
        - **x**: 0.5,
        - **y**: 0.5,
        - **z**: 0

      }

    ],

- **sma-infrared**:

  [

    - {
        - **name**: "sma-ir1",
        - **type**: "sma-infrared",
        - **x**: 0,
        - **y**: 0.5,
        - **z**: 0

      },

    - {
        - **name**: "sma-ir2",
        - **type**: "sma-infrared",
        - **x**: 1,

```
            ▪ y: 0.5,
            ▪ z: 0

          }

      ]

  }
```

Similarly, the following url

http://localhost/api/sensors/status

could return the following result

```
{

 • infrared:

     {

      ○ ir1: 1,
      ○ ir2: 0,
      ○ ir3: 0,
      ○ ir4: 0,
      ○ ir5: 0

     },

 • sma-infrared:

     {

      ○ sma-ir1: 0,
      ○ sma-ir2: 0

     }

}
```

which gives the status of all sensors/actuators (in the above case, sensor ir1 is activated, the other sensors are not activated).