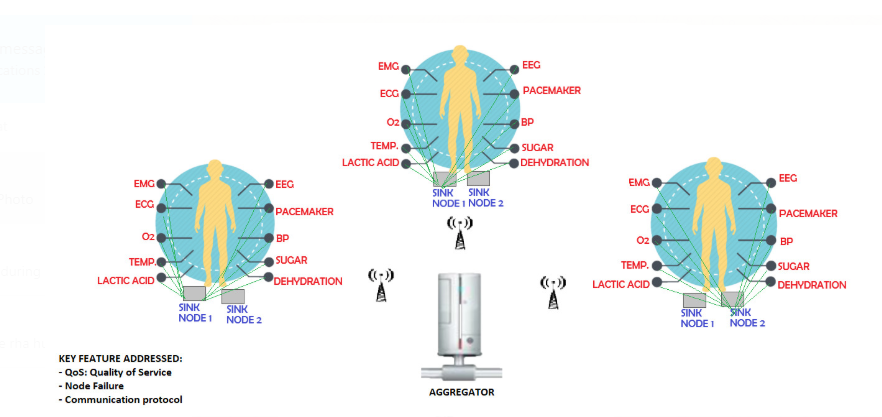
**Body Area Networking for IOT -**

**Tools used:** Python latest version, IDEs like PyCharm, Spyder can be used.

**Libraries used**: Socket , threading ,time, json

**Architecture:**



**Sinks:**

* 2 sinks defined in the files receiver.py, receiver2.py are assumed to be located on the body.
* Both the receivers are programmed to receive dynamic data generated from sensors located on the body through socket programming.
* 2 sinks are defined for availability ,i.e, incase one sink goes down the other one is available to receive the data from the sensors.
* Also the sinks are programmed to be available to receive the data only when they are charged enough or having sufficient battery/energy level. Otherwise receiver is in sleep mode. This is to demonstrate the energy efficiency performance of the computing systems.
* Both these files would have local machine ip defined, on which they are running.

**Sensors**:

* Different sensors on the body are defined in the following .py files:
* Bloodoxylevel\_sensor.py, bloodpressure\_sensor.py, dehydration\_sensor.py, ecglevel\_sensor.py, eeg\_sensor.py, emg\_sensor.py, lacticacid\_sensor.py, pacemaker.py, sugarlevel\_sensor.py, temeperature\_sensor.py
* Each sensor code contains-
  + Function for generating the data (for eg. Generating body temp, heart beats, blood pressure etc.).
  + Contains feature for generating alert message for any abnormal data generated.
  + Function to send the sensor generated data through one of the available sinks.
  + Files generated locally,i.e, file generated by the sensors in the program is received by the sinks and stored in the Receiver folder with the filenames defined in the receiver.py/receiver2.py files.

**Sender**:

* Sender.py file contains a sender function defined which will receive the data and port as arguments.
* Data is then sent through the port on the network through socket programming.
* This file would have the local ip of the machine defined ,on which this program is running.

**Energy**:

* Energy.py file has battery defined along with functionalities for energy decrease while sending/receiving data, idle mode, charging/discharging conditions and for throwing alerts when the battery runs out.
* This function is called by the sensors and sinks to evaluate and compute their energy levels while sending/receiving data, while in idle mode, charging/discharging mode,etc.

**Constants:**

* config.py file contains ports of the sinks defined.

**Sink to Aggregator communication:**

* client.py file contains program to send the files stored in the sink(Receiver folder) to Aggregator(configure separate machine with same set up of above files and folders and treat it as aggregator).
* Once the file is sent from the sink to the Aggregator, files are deleted from the sink as the memory of the sink needs to be used efficiently. If Aggregator is unavailable , then the files keep accumulating at the sink(Receiver folder) until connection is re-established with the Aggregator. This prevents Loss of Data.
* Client.py file would have the ip address of the aggregator defined.
* Server.py has the program to receive the data from multiple sinks. It must have the port defined through which the sinks would be communicating to it.

**Steps performed:**

1. Run the receiver.py and receiver2.py files to turn on the sinks. You will notice that the receivers are ready to listen on the network.

Please note that ports are available on your network before running the receiver files.

1. Run the All\_Sensors.py file. You will see the data getting generated in json format and getting transmitted to the one of the available receivers. You can try stopping one of the receivers to notice that the other receiver starts receiving the data of the sensors.
2. Files are stored under Receiver folder.
3. Then run the server.py file on the aggregator so that aggregator is ready to receive data from sink.
4. Run the client.py file on your local machine. This would send the files from your Receiver folder to Aggregator and you will see the files on the Aggregator end in the ‘uploads’ folder. You will see the files getting deleted from the Receiver folder on your local machine and appearing at the Aggregator end.
5. If the Aggregator is unavailable to receive the data from the sink , the files keep accumulating in the sink(Receiver folder).