Beautifull title

Olivier Flauzac
???
name of organization (of Aff.)
Reims,France
olivier.flauzac@univ-reims.fr

Florent Nolot
????
name of organization (of Aff.)
Reims,France florent.nolot@univ-reims.fr

Joffrey Hérard
???
name of organization (of Aff.)
Reims,France
joffrey.herard@etudiant.univ-reims.fr

***Abstract*—Machine to Machine (M2M) communication has gained much interest in the recent past and the issues related to energy efficiency are central to all wireless networks, including wireless sensor, for an energy economy, we need to limit emissions so some nodes will make requests to their neighbors. This article will show how to make sure neighbors are woken up during queries in a hybrid network architecture or how to make sure that the nodes that will receive the information will be woken up ?**

***Index Terms*—M2M, hybrid network,hybrid network, wifi, ad-hoc**

I. Introduction

This article is based on a particular architecture choice: that of the relay. Indeed it will be question of the establishment of this architecture to solve the problem of isolated nodes. An isolated node is a node that finds itself totally unable to join a central access point / data concentration unit to respond to survey problem on sensors. If a node is said to be isolated, the means must be found to be able to acquire the information it holds, but also to be able to ensure that it has enough energy to communicate over a long period of time. To do this, the track that will study this article is about the redefinition of actors. Indeed we would have two more actors: Main concentrator, sensors connected to the Main Concentrator but we would have four: Main concentrator, Sensor connected to the Concentrator, Sensor connected to the Concentrator and able to relay the information of an isolated node, An isolated node . To explain our solution we will describe how was the solution algorithmically thought out with chronograms. Then we will see with a more formal algorithmic writing. As well as the resulting simulation of the implementation of our algorithm to evaluate the loss of energy as well as the cost of sending messages. This is a list of term findable inside this article :

- IN= Isolated Node.
- WGW=Wifi Gateway.
- GW= Gateway using wireless communication.
- WELL= Concentrator, center of the network sensors.
- A mode = Agregation mode.
- NA mode = Non agregation mode.

II. Operation

Phase 1: Discovery Isolated nodes broadcast messages to discover(message 1) a Local Gateway. The end-nodes send

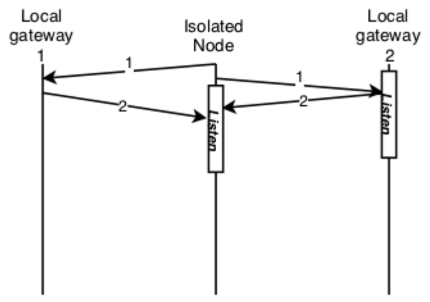

Figure 1. Discover Phase

messages every (X + random Integer) seconds. If a Gateway receive a discovery message, the Local Gateway will accept (message 2)and communicate to the end-nodes the communication slots.

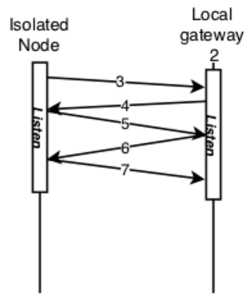

Figure 2. Registering/pairing phase

Phase 2: Registration

The end-node will confirm the pairing to the Local gateway (message 3) . The end-node is registered to the Local gateway. There might be several Local gateway inside the system. However, there must never be multiple registrations on the same one. After the local gateway receive the message 3, gateway send to the isolates node the message 4 to give his acknowledgment. Message 5 is the first message from the isolated node, which contains datas to put an end the pairing phase. When all this stuff is done, the local gateway is able to send the message 6 : a request of data, then the node will answer his data.

Phase 3: Collection The Local gateway will ask data to the

Identify applicable funding agency here. If none, delete this.

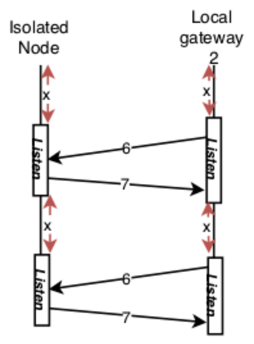

Figure 3. harvest phase

end-nodes. If so, the isolated node sends the data to the Local gateway. This one will then forward it to another Gateway.

1) Accuracy: Each message contains information on the next slot, the time of listening on this one, as well as the id of the senders and receivers. If a message is lost or collapsed, it will be re-sent managed by timers.

2) Particular case: When two gateways are in range from the isolated node and there both answer to him with the message 2. The isolated node pick one randomly. To manage multiple isolated node,

III. Algorithm and simulation

A. Algorithm

1) Messages format: The set of system messages are of the form:

```
< message_type , source , destination ,
                 data >
```

There are several messages in the system. It is considered that each message corresponds to a function carrying the name of the message, initializing the type and the source of the message, and taking in parameter the destination and the data.

2) Message from IN to GW:

- The `discover` message
  - message set up for the discovery of a `GW` by an `IN`
  - `destination = udef` : broadcast mode
  - `data = udef` : nothing
- Messages `pair`
  - Pairing message from an IN to a GW.
  - `data = udef` : nothing
- Messages `data_response`
  - Answer from the data request of a GW.

3) Messages from GW to IN: For all the messages coming from the `GW` the data part is structured as follows:

- `answer_frequency` : frequency on which the IN must respond
- `next_slot` : delay by the next listening window
- `next_duration` : fixed time of the next listening window
- `next_frequency` : Frequency of the next listening window
- `data` : data space specific to the exchange

The message is like that :

```
< message_type , source , destination ,
   answer_frequency , next_slot ,
next_duration , next_frequency , data>
```

The different messages `GW` *rightarrow* `IN` are thus:

- the messages `candidate`
  - texttt GW reply message after receiving `discover` from `IN`
  - texttt data = udef: no info
- messages `data_request`
  - data request message
  - `data = udef` if only one data available or `data = requested_data` in the case of multiple data

4) List of differents used functions:

a) Sending functions: `void send(frequency , message)`

b) Receiving functions: The function will `listen` on a particular `frequency` during time defined by `time` The function is decribed like this :

```
(message,time) listen(frequency , source ,
        message_type , time_listen)
```

the values of the parameters of this function are:

- `frequency` : listening frequency
- `source` : id of the sender of the message
  - `source = udef` : listen to all nodes on the defined frequency
- `message_type` : type of message expected
  - `message_type = udef` : listen to all types of messages
- `time_listen` : duration of the reception window
  - `time_listen = udef` : infinite window

Return values are :

- `message` message received
  - passage of the message in its totality
  - `message == udef` : no reception respecting the constraints
- `time` remaining time based on `time_listen`
  - `time == udef` :in the case of `time_listen = udef`

The subtlety to ensure that an isolated node is waiting for a request for data lies in the first data request message that contains the data: In how long I have to be awake and for how long I should be . So once the data is sent and the end of the slot arrives the node goes to sleep again (if it's the first time it works) and wakes up in time to the next window of data request. There are two conditions for these to pass correctly.

- Firstly the rest time must be long enough, indeed if the relay has a lot of node to collect, if the time is not enough it can never be in deep sleep state and therefore can exhaust its battery.
- Secondly the calculation of the slot sent in the request message. If there is once again a lot of node harvested for the relay. he must make sure to save the isolated nodes attached to him, he can calculate an offset corresponding to the order in which he will pick up the one he sends this slot value.

Algorithm 1 Initialization of communication variables of IN

```
1: procedure $init\_var$(msg)
2:     $gw \leftarrow msg.source$
3:     $next\_time \leftarrow msg.next\_slot$
4:     $timer \leftarrow msg.next\_duration$
5: end procedure
6:
7: procedure $flush\_var$( )
8:     $gw \leftarrow udef$
9:     $msg \leftarrow udef$
10:    $next\_time \leftarrow udef$
11:    $timer \leftarrow timer\_disco$
12: end procedure
```

Algorithm 2 Algorithm IN 1-1

```
1: while $(true)$ do
2:     $flush\_var()$
3:     while $(msg = udef)$ do
4:         $send(freq\_listen, discover(udef, udef))$
5:         $(msg, t) = listen(udef, candidate, timer + rnd())$
6:     end while
7:     $initVar(msg)$
8:     $send(freq\_send, pair(gw, udef))$
9:
10:    while $(gw! = udef)$ do
11:        $sleep(next\_time)$
12:        $(msg, t) = listen(gw, data\_request, timer)$
13:        if $msg! = udef$ then
14:            $initVar(msg)$
15:            $send(fdate\_response(gw, local\_data)$
16:        else$flush\_var()$
17:        end if
18:    end while
19: end while
```

Algorithm 3 Initialization of communication variables of GW

```
1: procedure $init\_var$( )
2:     $freq\_send \rightarrow chose()$
3:     $timer \leftarrow chose()$
4:     $freq\_listen \leftarrow chose()$
5:     $freq\_next \leftarrow chose()$
6: end procedure
7:
8: procedure $flush\_var$( )
9:     $timer \leftarrow timer\_disco$
10:    $freq\_listen \leftarrow freq\_disco$
11:    $freq\_send \leftarrow freq\_disco$
12:    $in \leftarrow udef$
13: end procedure
```

Algorithm 4 Algorithm gw 1-1

```
1: $flush\_var()$
2: while $(true)$ do
3:     if $(in == udef)$ then
4:         $(msg, t) = listen(, udef, discover, timer)$
5:     end if
6:     if $(msg! = udef)$ then
7:         $in \leftarrow msg.source$
8:         $init\_var()$
9:         $send(candidate(in, slot, duration, udef)$
10:        $(msg, t) = listen(, in, pair, timer)$
11:        if $(msg == udef)$ then
12:            $flush\_var()$
13:        end if
14:    end if
15:    if $(in! = udef)$ then
16:        $init\_var()$
17:        $send(data\_request(in, slot, durationt, udef)$
18:        $(msg, t) = listen(in, data\_response, timer)$
19:        if $(msg! = udef)$ then
20:            $send\_data(id + " : " + local\_data + ";" + in + " : " + msg.data)$
21:        else
22:            $send\_data(id + " : " + local\_data + ";" + in + " : " + udef)$
23:            $flush\_var()$
24:        end if
25:    end if
26: end while
```

B. Simulation

We simulated 100 graphs with each 1000 nodes and for each configuration:

- Variation of the number of nodes isolated by gateway between 1 to 4.
- Variation of the maximum gateway number per well between 1 to 4.
- Variation of the number of data to recover per day between 1 and 24 messages.

- Variation on the method of reassembly of the data collected by the isolated nodes (ie, sent to the well at each reception (NA method) or reassembled after merging of all the messages (A method))
- Variation of the speed of each link between two nodes.

Which makes us a total of 12800 graphs to create. In order to realize this creation we have written an algorithm based on the probability of existence of a link between two nodes.That is to say that at the beginning we create the first well. At that one looks at its degree (which is zero at the beginning) of gateway hooked to it. One draws a probability and if it is greater than that established in input of the algorithm one adds a gateway and the link between the well and this one. We do the same with the isolated nodes. In order to summarize we had: one hundred graphs with a maximum of one isolated node per gateway and a maximum of one gateway per well. This configuration will be abbreviated 1 (IN) -1 (GW) $\rightarrow$ 1-1.

So we used the Omnet ++ tool to simulate all these graphs and each case. In these we have noted for isolated node the number of messages sent as well as the remaining battery at the end of the simulation. For the battery we assumed that it had 6600 mAh or 23760000 mAs. The consumed electric energy was taken from the LoPy documention, WiFi section. Our forecast calculations are realized as follows:

The device is considered to send X msg (#msg) per day (over 24 hours). The current of an emission is Ce = 107.3 mA. Each emission lasts Te = 2s. The current in reception is Cr = 37mA. The time in reception is Tr The standby current is Cv = 0.531 mA. The device is idle for Tv = (24 * 3600- #msg * Te-Tr * Cr) The current consumed on a day is therefore: CDay (mA.s) = (#msg * This * Te + Cr * Tr + Cv * Tv)

So here is a graph showing the difference between aggregation and non-aggregation mode at the message level. We can

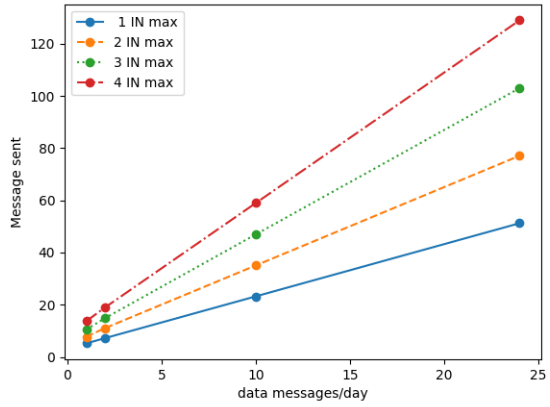

Figure 4. Mode : Agregation

notice that in the following figures that a linear aspect stands out clearly on the variation of the number of messages on the part of a gateway whatever the mode (with aggregation or without aggregation). What we can also notice is the number that as expected is bigger and grows much faster without

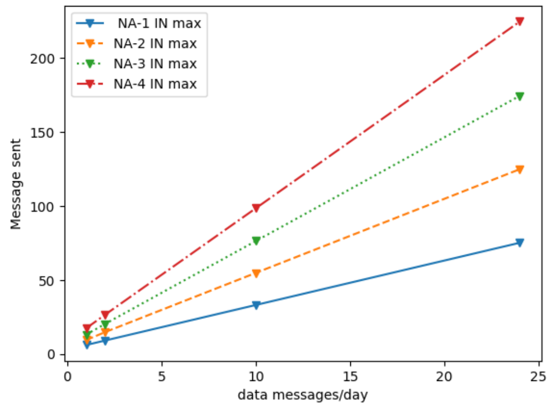

Figure 5. Mode : No agregation

the aggregation mode (Fig 4, 5, 6). What can therefore be considered significant enough. This is because adding the data of the gateway to those of the isolated nodes makes the number of messages in cluster mode to the same numbers as if the INs were GWs and thus connected nodes.

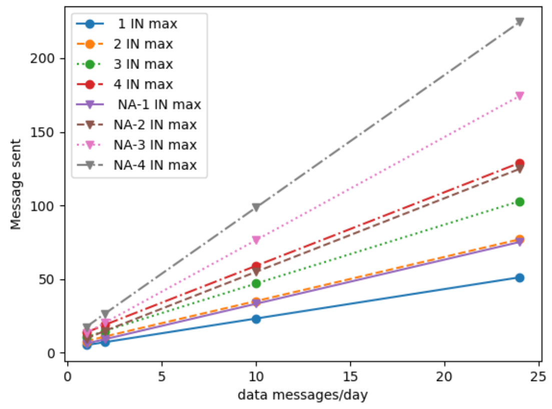

Figure 6. Mode : Agregation vs No Agregation

The objective was to check if the intuitive idea is right: the architecture with intermediary relay and aggregation returns, in number of messages and thus lifespan of the network, to the model without relay, the model or each client speaks live with antennas always UP.

Figures 7 to 10 are more complex. Indeed these graphs represent by their colors the number of maximum Gateway related to a well. Each point is a simulation recall described by the maximum degree of gateways and isolated nodes as well as the number of messages requested by the well per day. Here these four figures are in aggregation mode.

First on each graph, each point is an average of the number of messages sent or the state of the battery if any, by an isolated node, or by a gateway if necessary. For each graph, the block of 4 colors is therefore the set of graphs with degree

1 IN. it should be understood that the simulations from 0 to 399 are those called ”1-1, 1-2, 1-3, 1-4”. from 400 to 799 ”2-1, 2-2, 2-3, 2-4” etc. We realized these simulations with as previously said a number of message requested by the different well. Indeed we chose: 1, 2, 10, 24 messages per day.

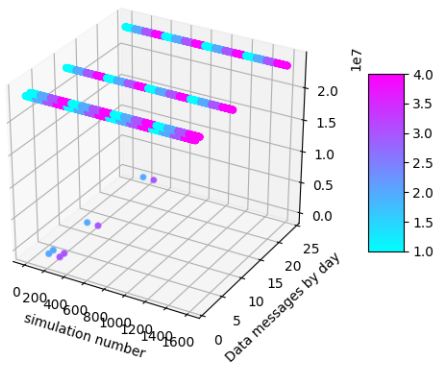

Figure 7. State of the battery on a Day / data message required/day for a gateway

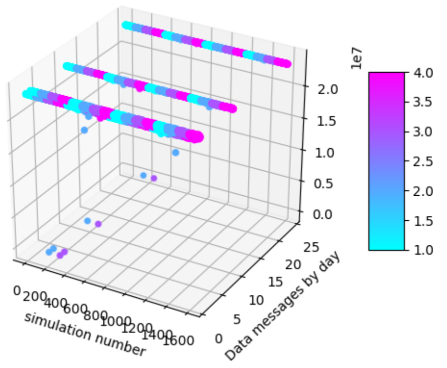

Figure 8. State of the battery on a Day / data message required/day for an IN

The previous remark made on the message number which remains coherent with respect to an architecture without relays it also on the state of the battery on the side of the isolated nodes, from the moment when the pairing is carried out. Of course, in the different figures such as FIG. 10 or FIGS. 8 and 7, there are very distinct points of all the others, these can be explained by a set of things:

- The connection / link is too bad / noisy so the device may have fallen out of battery / have sent too many messages. and therefore can never be recognized across the entire network.
- Corruption on identifiers.
- bad calibration of the device on its WiFi antenna

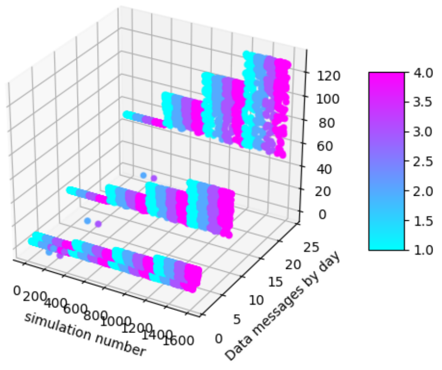

Figure 9. Number of message on a Day / data message required/day for a gateway

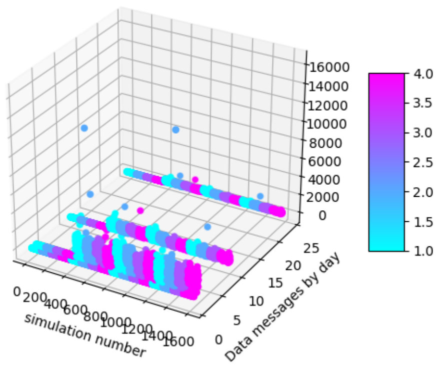

Figure 10. Number of message on a Day / data message required/day for an IN

IV. Conclusion

To conclude, we have therefore shown that it is possible to rally isolated nodes directly by the network concealer in order to retrieve despite all the data it has. All with a proof that the number of messages does not explode in aggregation mode compared to an architecture where the nodes would be connected directly to the hubs. The battery therefore remains in the same state whatever the architecture from the moment the pairing is done. We’ve also seen how to make sure a node is awake and waiting to send that data to its relay. It is expected to know how to handle the system once pairing is complete when an error / problem occurs. For example, what happens if the relay goes down? How to detect it from the IN and the well? How to know that an IN has failed from a gateway?

[6], [5], [2], [3], [4], [1]

Acknowledgment

References

[1] E. C. Haddad and J. C. Gregoire. *Implementation issues for the deployment of a WMN with a hybrid fixed/cellular backhaul network in emergency situations*, pages 525–529. 2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology. 2009.

[2] Ming Ann Ng and K. L. A. Yau. *An energy-efficient Hybrid Wireless Mesh Protocol (HWMP) for IEEE 802.11s mesh networks*, pages 17–21. 2013 IEEE International Conference on Control System, Computing and Engineering. 2013.

[3] N. G. Palan, B. V. Barbadekar, and S. Patil. *Low energy adaptive clustering hierarchy (LEACH) protocol: A retrospective analysis*, pages 1–12. 2017 International Conference on Inventive Systems and Control (ICISC). 2017.

[4] G. Sharma, N. B. Shroff, and R. R. Mazumdar. *Hybrid sensor and mesh networks: paradigms for fair and energy efficient communication*, pages 83–92. 2006 2nd IEEE Workshop on Wireless Mesh Networks. 2006.

[5] S. Singh, K. L. Huang, and B. S. P. Lin. *An energy-efficient scheme for WiFi-capable M2M devices in hybrid LTE network*, pages 126–130. 2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS). 2012.

[6] K. Tanaka, M. Murase, and K. Naito. *Prototype implementation of BLE based automated data collection scheme in agricultural measurement system*, pages 1–2. 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC). 2018.