

Information freshness in Smart Cities

ALOHA Environment

ST7: Optimization of network infrastructures for Smart Cities

Written by:

Sara ABKARI

Anouar BOUMEFTAH

Paul BOUYÉ

Erhun ÇETINER

Maxence CUMER

Introduction

Smart cities are the future. They are basically cities that can change responses according to varying conditions such as environmental, natural, non-ordinary, ordinary and social changes. We call them smart because they are made to respond accordingly to changing conditions automatically or with accessible human intervention, just as a smart individual would. They can be quite useful as they free up resources and materials. Smart cities offer an important prospect for the future of city management.

The Aloha process, also known as the Aloha protocol, is a communication protocol used for wireless networking. It was first developed at the University of Hawaii in the early 1970s for the purpose of providing reliable and efficient communication between remote computer systems. The Aloha process is a random-access protocol that allows multiple devices to share a single communication channel without the need for any central coordination. This makes it particularly useful for situations where many devices need to communicate over a network, such as in satellite communication, mobile networks, and local area networks (LANs).

Smart cities need automation and digitalization to work properly, this means that a lot of information must be transferred constantly to keep all the systems and responses in good order. This requires a huge workload for the networks however. For a functioning smart city, many sensors would be needed. This many sensors and the messages, the responses, the measurements etc. that they are responsible for would load the whole smart city network. This load would affect the overall performance of the whole system, possibly causing unwanted results. The responses of the smart city would be slower, badly timed and even harmful in some cases. This is mainly because of the “Age of Information (AoI)” .That is the latest information signal sent by sensors to simply be too old for the response mechanisms to reply in the desired manner. In order to solve this problem, the paper of Rajat Talak, Sertaç Karaman, and Eytan Modiano titled *Distributed Scheduling Algorithms for Optimizing Information Freshness in Wireless Networks* ([2]) was examined.

According to [2], the main difficulty that multiple sensors and a complicated network presents is that forming an optimization problem that is convex is usually impossible and it causes eliminating the extra AoI basically impossible. For that, they proposed a solution which lets us solve the same problem in the exponential form and which allows us to find an optimal solution. That is why in this project, we have tried to implement the proposed mathematical improvement. For this purpose, we have used Python to solve the optimization problem, simulate a network respecting the solution, and then observe the results. The mathematical operation allowed us to decrease the AoI. In practical applications, this would mean that the last received information from a group of sensors to be arranged in such a manner that the overall performance is simply better resulting in a more efficient network - a smarter city, if you will.

Chapter 1

Optimization problem

1.1 Mathematical tools development

1.1.1 Age of Information

As explained in the introduction, we are first going to introduce a metric called "*Age of Information*" (which we will be referring to as "*AoI*"), that is a lot less used than the other metrics like the throughput and the delay. Yet, it brings other pieces of information that are really interesting to design a network. Indeed, it corresponds to the time elapsed between two successive received information packets emitted by the same device. To check the utility of this metric, we will consider a network with a central monitor receiving information from a set of devices, such that each device can only communicate with the monitor, using an ALOHA protocol. This latter corresponds to a communication protocol where a device do not take care about the state of the others when transmitting, it only avoids transmitting if a device is already occupying the canal. We will be considering only a network with one canal, that means that each communication device-monitor can collide with any other one.

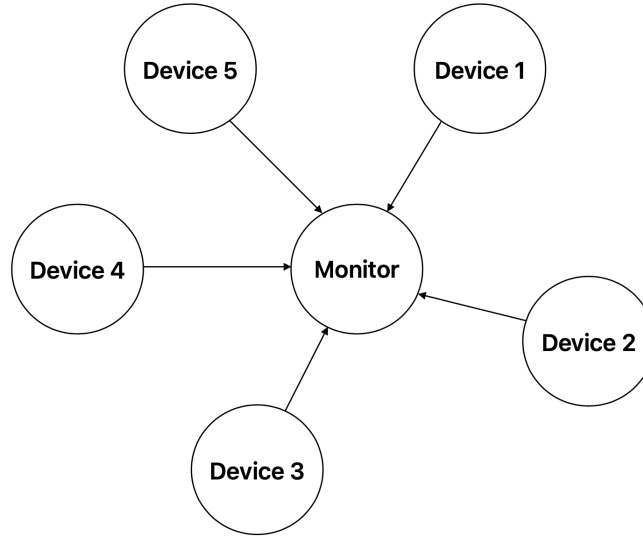


Figure 1.1: Form of the considered network

So, first, we have to model, as explained in [2], a wireless communication network as a graph $G = (V, E)$, where V is the set of nodes and E is the set of links between the nodes. Each link $e \in E$ is associated with a source-destination pair. In our case (corresponding to a central monitor with n devices), we have:

- V will contain the nodes corresponding to the central monitor and the devices,
- E will contain all the links (m, d) where $m \in V$ is the node corresponding to the monitor and $d \in V$ is the node corresponding to a device.

Transmission of update packets cannot occur simultaneously over 2 links due to wireless interference constraints. Otherwise, there would be some interference, making the information delivered impossible to be understood by the receiver. So for each link $e \in E$, there is a subset of links $N_e \subset E$ that can interfere with link e . In other words, if link e and link $l \in N_e$ attempt simultaneous transmission, then both transmissions fail

due to the collision of the packets. In our simplified model, each link can interfere with every other link, hence $\forall e \in E, N_e = E \setminus \{e\}$.

As there is a bijection between the set of links E and the set of all devices $(V \setminus \{m\})$, we can define for all links $e \in E$ the age $A_e(t)$ as the time elapsed since the last successful transmission over e , it is called the *Age of Information* or AoI. We also define link e 's average age A_e^{ave} and its peak age A_e^p (the average of all its age peaks) to be:

$$A_e^{ave} = \limsup_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T A_e(t) \quad (1.1)$$

$$A_e^p = \frac{1}{\mathcal{N}_e(T)} \limsup_{T \rightarrow \infty} \sum_{i=1}^{\mathcal{N}_e(T)} A_e(T_e(i)) \quad (1.2)$$

where $T_e(i)$ denote the i -th instance of successful transmission over link e and $\mathcal{N}_e(T)$ is the number of successful transmissions over link e until time T .

We can finally define average and peak AoI as the average of each device's average age and peak age, with a given weight w_e for each source, as in [2]. In our next simulations (cf Chapter 3), we will be taking all w_e equal (which consists in taking $w_e = \frac{1}{|E|}, \forall e \in E$).

$$A^{ave} = \sum_{e \in E} w_e A_e^{ave} \quad (1.3)$$

$$A^p = \sum_{e \in E} w_e A_e^p \quad (1.4)$$

Those are the two parameters that we want to minimize, in line with our first intuition.

1.1.2 Distributed stationary policies

We consider policies in which all decisions are made by the source of each link, and not by a centralized monitor. That means that each device must calculate itself its probability of emission. As in the article, each link e will be attempting transmission with probability $p_e > 0$, which will depend on the weight accorded to the link. In case of equal weights, each p_e will be equal. Hence the name of *Distributed stationary policies*.

The probability that link $e \in E$ achieves successful transmission is therefore given by f_e , defined in Equation 1.5, called the *link activation frequency*.

$$f_e = p_e \prod_{e' \in N_{e'}} (1 - p_{e'}) \quad (1.5)$$

The space of all link activation frequencies \vec{f} attainable with distributed stationary policies is given by :

$$\mathcal{F}_D = \left\{ \vec{f} \in \mathbb{R}^{|E|} \mid f_e = \prod_{e' \in N_{e'}} (1 - p_{e'}) \text{ and } \forall e \in E, 0 \leq p_e \leq 1 \right\}$$

It can be shown that \mathcal{F}_D is non-convex, making the optimization problem a little harder. The next step, as detailed in [2], consists in getting an equivalent convex optimization problem.

1.2 Make the problem convex

It can be established that for any distributed stationary policy and with the same weight for each source, the average and peak AoIs are given by Equation 1.6, where γ_e is the channel success probability for link e (so $\frac{1}{\gamma_e f_e}$ the mean of the geometrical distribution of the time since the last activation of the device).

$$A^{ave} = A^p = \frac{1}{|E|} \sum_{e \in E} \frac{1}{\gamma_e f_e} \quad (1.6)$$

The minimization problem can then be written:

$$\min_{\vec{f} \in \mathcal{F}_D} \sum_{e \in E} \frac{1}{\gamma_e f_e}$$

Notice that, although the objective function is convex, the constraint set \mathcal{F}_D is non-convex. This problem allows us to obtain optimal link activation frequencies f^* , but it does not suffice as the distributed policy is characterized by the attempt probabilities \mathcal{P}_e . The optimal values of these attempt probabilities \mathcal{P}_e^* are given by:

$$\mathcal{P}_e^* = \frac{\mathcal{W}_e \mathcal{A}_e}{\mathcal{W}_e \mathcal{A}_e + \sum_{e \in N} \mathcal{A}_e \cdot \mathcal{W}_e}$$

According to [2], the previous minimization problem is equivalent to the following convex optimization problem:

$$\max_{\lambda_e \geq 0} \sum_{e \in E} \left(\lambda_e + \sum_{e' \in N_{e'}} \lambda_{e'} \right) H \left(\frac{\lambda_e}{\lambda_e + \sum_{e' \in N_{e'}} \lambda_{e'}} \right) + \sum_{e \in E} \lambda_e \left[1 + \log \left(\frac{1}{\mathcal{A}_e} \right) \right] \quad (1.7)$$

with $H(p) = p \log \left(\frac{1}{p} \right) + (1 - p) \log \left(\frac{1}{1-p} \right)$.

1.3 Algorithm

It can be hence established that the following algorithm gives a solution of the previous problem:

With $p_e(m)$ the attempt probability of link e in frame m , $\lambda_e(m), \theta_e(m)$ the dual variables in frame m , having set $\lambda_e(0) = 1$ and $\theta_e(0) = |N_e|$ for all $e \in E$, $p_e(0) = \frac{1}{2}$ and $m = 0$. For each frame m we must do :

Send $\theta_e(m)$ to all $e' \in N_e$. Compute $\lambda_e(m+1)$ so that :

$$\lambda_e(m+1) \leftarrow \Pi_e \left[\lambda_e(m) + \eta_m \left\{ \log \left(\frac{w_e}{\lambda_e(m)} \right) + \log \left(1 + \frac{\theta_e(m)}{\lambda_e(m)} \right) + \sum_{e' \in N_e} \log \left(1 + \frac{\lambda_{e'}(m)}{\theta_{e'}(m)} \right) \right\} \right]$$

Send $\lambda_e(m+1)$ to all $e' \in N_e$ and update $\theta_e(m+1)$: $\theta_e(m+1) \leftarrow \sum_{e' \in N_e} \lambda_{e'}(m+1)$

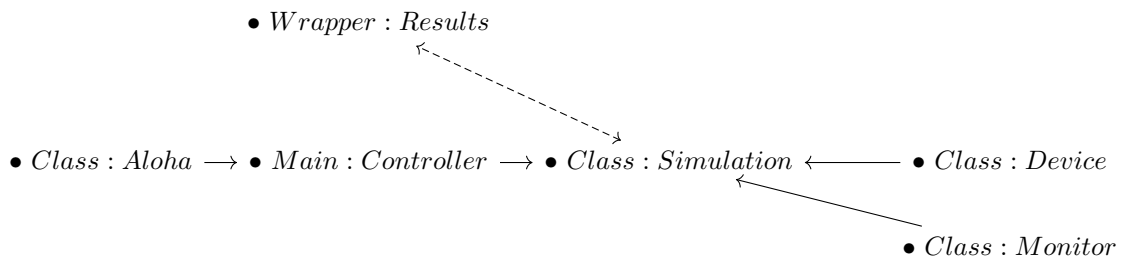
Update $p_e(m+1)$: $p_e(m+1) \leftarrow \frac{\lambda_e(m+1)}{\lambda_e(m+1) + \theta_e(m+1)}$

Chapter 2

Code

The program¹ splits into two parts : The first part simulates the ALOHA algorithm and computes the parameters. These parameters are then used in the second part where the network is simulated.

The simulation is based on multiple devices linked to a single receiving monitor. The time is based on epochs or number of iterations elapsed since inception.



Explanatory diagram of the project's structure

The following constants are user-defined :

- NB_DEVICES : for number of devices (or nodes) in the simulation
- SIM_LIFESPAN : for simulation's lifespan (in epochs)
- SIM_TF : for device's duration of transmission (in epochs)
- ITERATIONS_ALOHA : for ALOHA's algorithm number of iterations (m)
- SIM_WAITING_TIME : for device's rate of transmission (in epochs)

¹<https://github.com/N3CK5/Freshness2>

Figures 2.1 and 2.2 shows different network simulations for the following user-defined parameters : $NB_DEVICES = 5$, $SIM_LIFESPAN = 50, 200$, $SIM_TF = 2$, $SIM_WAITING_TIME = 10$. A packet is considered lost when a collision occur (i.e. when more than one device transmits at the same time). Successfully received packets are represented in green and lost packets in red.

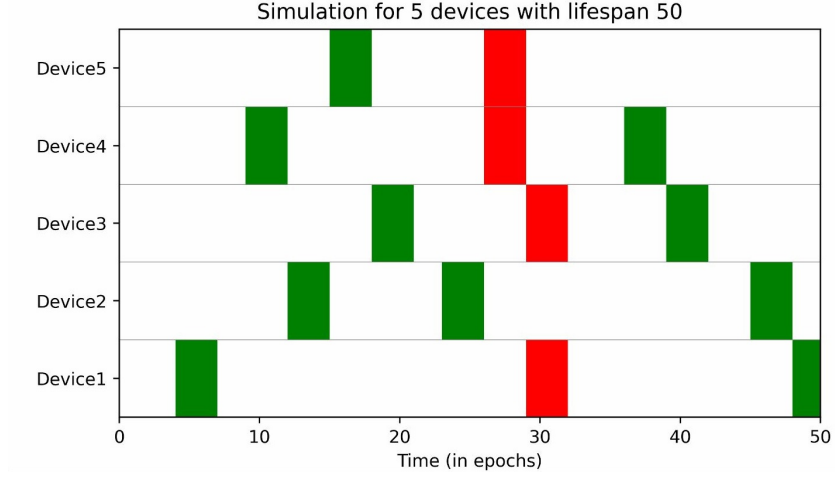


Figure 2.1: Simulation results for 5 devices with a lifespan of 50 epochs

We can observe in both of these figures that the duration of transmission SIM_TF and waiting time $SIM_WAITING_TIME$ is respected. It is important to note that the device's behaviour at the start of the simulation is different in term of waiting time. For that special case, the simulation implements a prior waiting time depending on a distribution limited by $SIM_WAITING_TIME$.

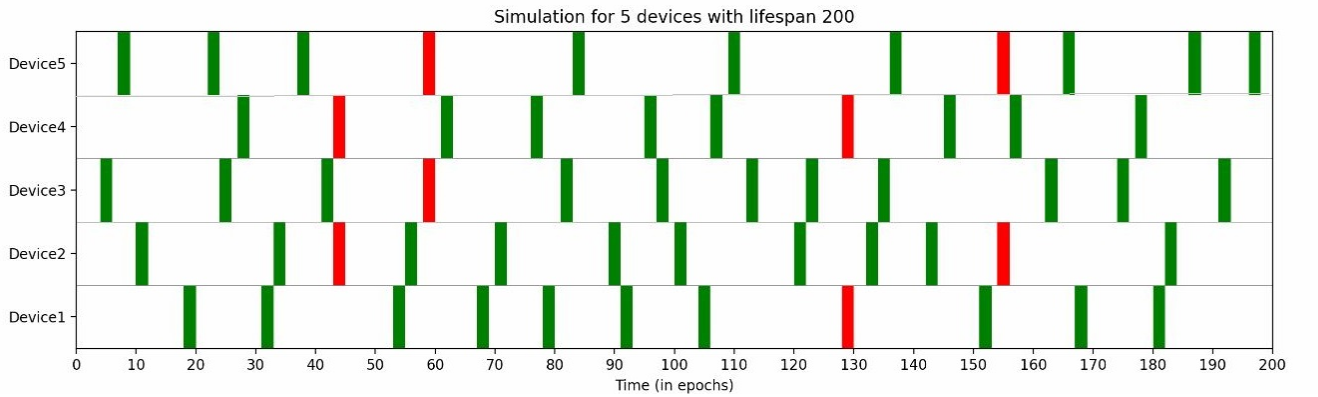


Figure 2.2: Simulation results for 5 devices with a lifespan of 200 epochs

Chapter 3

Results

After having tested our code to check that it works well (e.g. by checking that it gives a good value of p_e when each link has the same weight), we can launch some simulations so as to design a network for a specific use case. We will be considering in this chapter two different situations, for each of which we are going to measure, through simulations, the maximum number of transmitters, respecting a constraint in terms of latency. The two situations (taken from [1]) that we are going to analyse are:

- **Indoor climate control:** each device generates a packet every 15min and the latency should be inferior to 5s. One second is needed to send a packet. In that situation, the number of devices must be in the order of several hundreds.
- **Smart healthcare:** each device generates a packet every 10s and the latency should be inferior to 3s. One second is also needed to send a packet. In that case, the number of devices must be in the order of several dozens.

3.1 Indoor climate control

To simulate the corresponding network of this situation (that is described above), we need to change the values of SIM_LIFESPAN, SIM_TF and SIM_WAITING_TIME. And we are going to take a high value of NB_DEVICES and to lower it until we get an average AoI respecting the constraints of latency. In our case, we are going to take the following *time - number of epochs* equivalence: $1s \Leftrightarrow 1$ epoch. In that case, we should set SIM_WAITING_TIME to $15 \times 60 = 900$, SIM_LIFESPAN to $15 \times \text{SIM_WAITING_TIME} = 13500$ and SIM_TF to 1.

We launched the algorithm (explained in Section 1.3) to find the optimal values of p_e so as then to use them in the simulation. Yet, as we wanted to run several times the simulation, we did not want to benefit from too many iterations in this algorithm. As a consequence, we decided to set the number of iterations to 100. If with this number, for all $e \in E$, the sequence composed of all the consecutive values of λ_e is not converging, as we can notice on Figure 3.1.

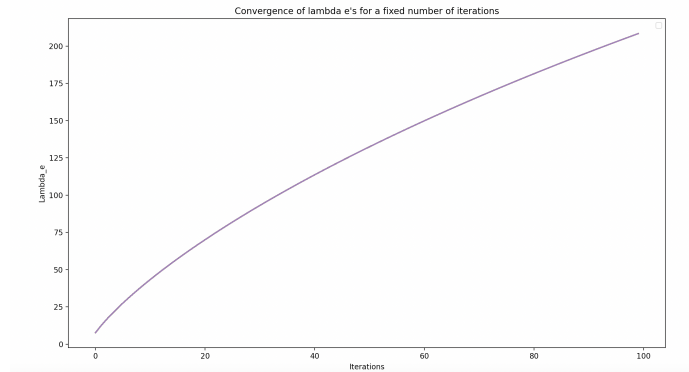


Figure 3.1: Graph of the evolution of λ_e considering 100 iterations of the algorithm

However, we also launched the algorithm with 5000 as the number of iterations. We find that the sequence converges, as we can notice on Figure 3.2. Yet, is it really important in this situation that the sequences of the consecutive values of λ_e converge? Indeed, we find, with 5000 iterations, a value $p_e = 0.003389830508474528$, whereas for 100 iterations, we find $p_e = 0.0033898305084745675$, giving a difference of 1.47×10^{-17} , which is

negligible. That means that, so as to run several times the simulation (to get a better average value of the AoI), we just need to run the algorithm with 100 iterations.

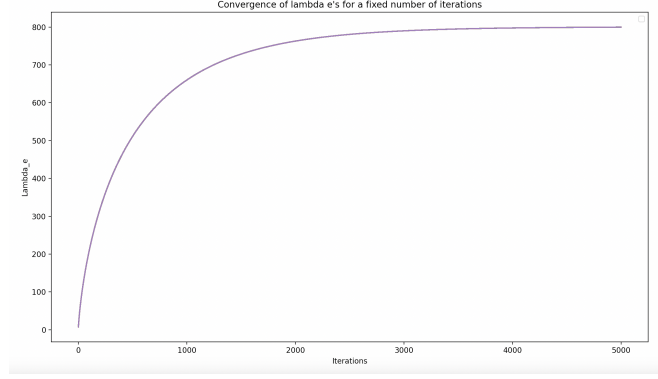


Figure 3.2: Graph of the evolution of λ_e considering 5000 iterations of the algorithm

With these parameters, we get a maximum number of devices of around 295. That means that, to get a latency inferior to 5s, only 295 should be used on this network. Yet, because there are some probabilities in the Aloha protocol, this result is not exact. Indeed, by launching the simulation a lot of times, we get this result, but that does not mean that we will get at each simulation this number of devices. For example, we can get at some simulations an average AoI greater than 905, but in average, we get an AoI between 900 and 905 epochs (which means between 15 min and 15 min 5 s).

We tested 295 devices, we launched the simulation 5 times and we obtained an average AoI over our 5 simulations of $904.6345163164713 < 905$, as we can notice on Figure 3.3. So we obtain the same order of magnitude of the number of devices (hundreds of them) as the one proposed in [1].

#CONSTANTS	0.0033898305084745675
NB_DEVICES = 295	0.0033898305084745675
SIM_LIFESPAN = 15*60*15	0.0033898305084745675
SIM_TF = 1	0.0033898305084745675
ITERATIONS_ALOHA = 100	0.0033898305084745675
SIM_WAITING_TIME = 15*60	0.0033898305084745675
	904.6345163164713

Figure 3.3: Screenshot of the average of the results of the simulations, given the previous parameters

In a simulation, we plotted the evolution of the Age of Information of three of the devices and we obtained the graph on Figure 3.4, where one epoch is equivalent to one second.

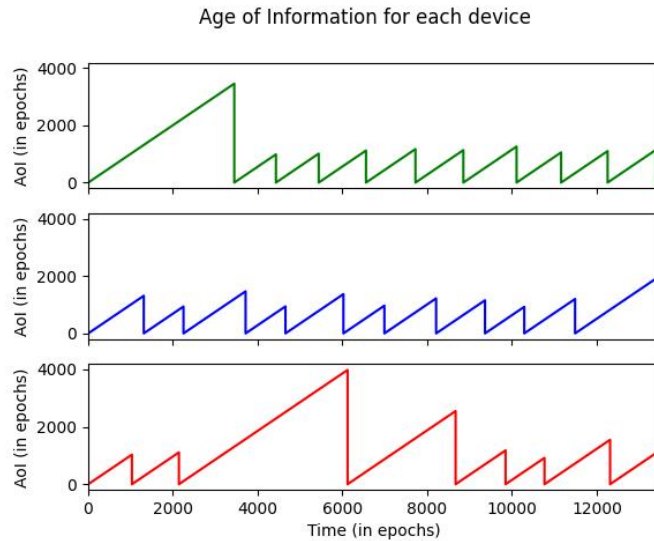


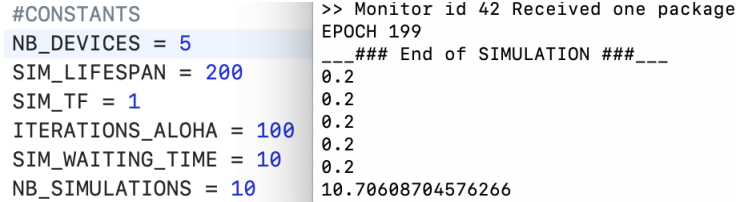
Figure 3.4: Evolution of the AoI of three of the devices in the first situation

3.2 Smart healthcare

For the network of this situation, we need to take new values for the different constants. The values that we used are (with 1 epoch corresponding to 1s):

- SIM_WAITING_TIME = 10 epochs
- SIM_LIFESPAN = 200 epochs
- SIM_TF = 1 epoch

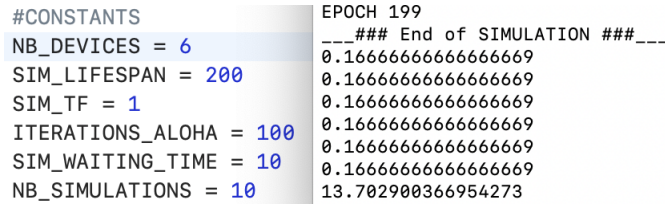
Yet, if we consider this, we get that only 5 devices can be connected to the network so as to respect the maximum latency (3s), as shown on Figures 3.5 and 3.6.



```
#CONSTANTS
NB_DEVICES = 5
SIM_LIFESPAN = 200
SIM_TF = 1
ITERATIONS_ALOHA = 100
SIM_WAITING_TIME = 10
NB_SIMULATIONS = 10

>> Monitor id 42 Received one package
EPOCH 199
---### End of SIMULATION ###---
0.2
0.2
0.2
0.2
0.2
10.70608704576266
```

Figure 3.5: Screenshot of the average of the results of the simulations, given the previous parameters with 5 devices



```
#CONSTANTS
NB_DEVICES = 6
SIM_LIFESPAN = 200
SIM_TF = 1
ITERATIONS_ALOHA = 100
SIM_WAITING_TIME = 10
NB_SIMULATIONS = 10

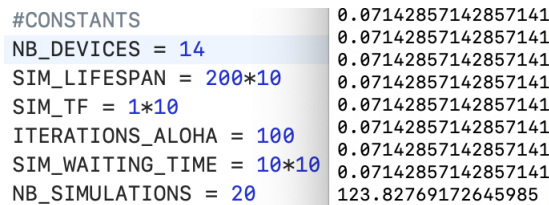
EPOCH 199
---### End of SIMULATION ###---
0.16666666666666669
0.16666666666666669
0.16666666666666669
0.16666666666666669
0.16666666666666669
13.702900366954273
```

Figure 3.6: Screenshot of the average of the results of the simulations, given the previous parameters with 6 devices

Yet, so as to be closer to the reality, we have to take a ratio *time - number of epochs* really inferior to 1s/epoch. As a consequence, we chose the following parameters:

- SIM_WAITING_TIME = 100 epochs
- SIM_LIFESPAN = 2000 epochs
- SIM_TF = 10 epoch

We found with these new numbers the that for NB_DEVICES = 14 devices, the average of the AoI is approximately 123 epochs < 130 epochs = 13s, as shown in Figure 3.7.



```
#CONSTANTS
NB_DEVICES = 14
SIM_LIFESPAN = 200*10
SIM_TF = 1*10
ITERATIONS_ALOHA = 100
SIM_WAITING_TIME = 10*10
NB_SIMULATIONS = 20

0.07142857142857141
0.07142857142857141
0.07142857142857141
0.07142857142857141
0.07142857142857141
0.07142857142857141
0.07142857142857141
0.07142857142857141
0.07142857142857141
0.07142857142857141
123.82769172645985
```

Figure 3.7: Screenshot of the average of the results of the simulations, given the previous parameters with 14 devices, where a slot equals 1s

To understand the reasons of this difference of maximum numbers of devices, we need to go back to how an Aloha network works. Indeed, after having scanned the canal, a device will emit at the first slot with a probability p_e . If it does not, it will redo the same process next slot,... until it emits. So by dividing an initial slot into 10 "new" slots, we give the device ten more chances to start emitting between a first initial slot and a second one. That is closer to the reality, where the devices do not really work with slots but with the continuity of time. By increasing the number of slots per second, we tend to the real model. That is why the second figure is more precise. We could go on by dividing the slots by 10 once more. This gave us an optimal number of 17 devices. Once more, we can display the evolution of the AoI for three devices (see Figure 3.8).

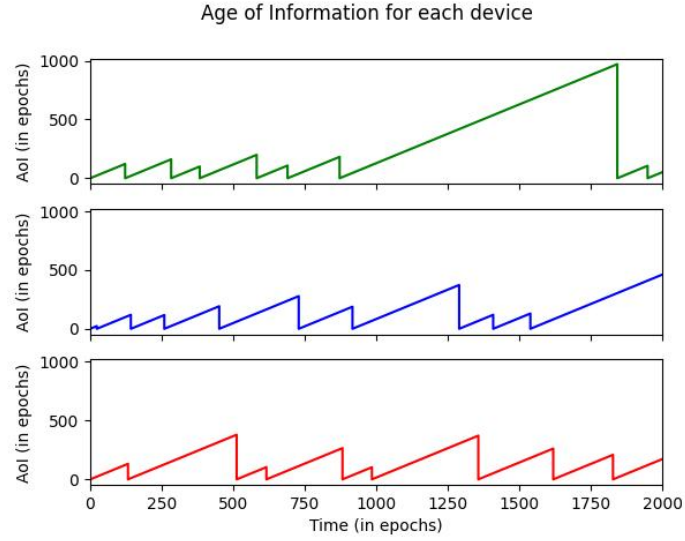


Figure 3.8: Evolution of the AoI of three of the devices in the second situation

To finish with, we can plot once again the evolution of the successive values of the λ_e with 150 iterations (see Figure 3.9). We will see that it converges in around 150 iterations. So why did we take only 100 iterations previously? That is because we get exactly the same values of p_e (with 17 significant digits). So we could run the algorithm using only 100 iterations, its results would be the same as in 150 iterations. The convergence of p_e interests us more in these cases than the one of λ_e .

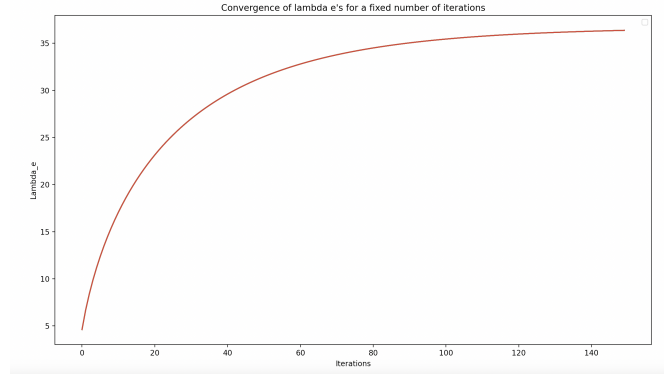


Figure 3.9: Graph of the evolution of λ_e considering 150 iterations of the algorithm

Conclusion

In summary, the Aloha process is a random-access protocol that allows multiple devices to share a single communication channel without the need for any central coordination. It works by allowing devices to transmit their data packets at any time, and in the event of a collision, each device waits for a random period before attempting to transmit its data packet again.

As more and more devices are using higher bandwidth and new wireless standards like 5G, measuring the freshness of the data becomes increasingly central in designing networks. By testing the effect of using a distributed approach on - at first glance - a convex problem, this study established a sufficient and acceptable standard for Indoor Climate Control situations and Smart Healthcare. The former's results are satisfying on around 300 devices and the latter on approximately 15. These numbers are backed by the clear convergence of the algorithm's parameters that guarantees us a shared transmitting and collision-avoiding policy. It is however important to note that these simulations are based on a user-defined timespan as we can see it on the healthcare situation where the latter had to be well defined to obtain an acceptable result. This suggests that every situation should have an adequate approach in terms of simulating time and latency (epochs in seconds, minutes, etc..).

Aloha distribution includes lots of limitations: First, Low efficiency: The protocol is not very efficient as a lot of time is wasted in retransmitting due to collisions. Then, The protocol is susceptible to interference from other sources, such as other wireless networks, which can cause collisions and further reduce the efficiency and throughput of the protocol. The lack of priority is one of the negative points of Aloha: the protocol does not provide any priority mechanism to ensure that more important or time-critical transmissions are given priority over other transmissions. Finally, Aloha is susceptible to interference from other sources, such as other wireless networks, which can cause collisions and further reduce the efficiency and throughput of the protocol.

(Ouverture à discuter??)

Bibliography

- [1] Jozef Mocnej, Adrian Pekar, Winston K.G. Seah, and Iveta Zolotova. Network traffic characteristics of the iot application use cases.
- [2] Rajat Talak, Sertac Karaman, and Eytan Modiano. Distributed scheduling algorithms for optimizing information freshness in wireless networks. *arXiv:1803.06469v1*, 2018.