ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DEPARTMENT OF

ELECTRICAL, ELECTRONIC AND INFORMATION ENGINEERING

"Guglielmo Marconi" DEI

MASTER DEGREE IN TELECOMMUNICATIONS ENGINEERING

Master Thesis In

Radio Network

UAV BASED NARROWBAND – INTERNET OF THINGS

Candidate:

SALVATORE BARRETTA

Supervisor:

PROF. ING ROBERTO VERDONE

Co-supervisor:

PROF. ING CHIARA BURATTI

ING. SILVIA MIGNARDI

Academic Year 2019/2020

Table of contents

Premise

Nowadays Internet of Things (IoT) devices are becoming more and more present in our everyday life. As a matter of fact, IoT devices are different kind of chipsets enabling machines to communicate wirelessly and interacting with the real world through sensors and actuators. They offer a variety of solutions in many fields and applications, thanks to the cheapness of the devices utilized, low power consumption and large coverage offered. Not exhaustive examples can be the deployment in the so-called smart house, smart cities, industry 4.0, smart agriculture, and smart health.

At the same time, to enhance the flexibility of use and further enhance coverage, in the recent years the interest is also growing toward unmanned aerial vehicles (UAVs) over which a base station (BS) is mounted, often referred to as unmanned aerial base station (UAB). Novel studies referred to such a kind of moving aerial BS are frequently showing up, due to the drone flexibility and mobility, that is able to enlarge the field of applications of the existing infrastructure scenario. UABs can especially be employed to solve coverage issues in areas where terrestrial base stations lack, like in emergency situations to rescue people in inaccessible places, or during occasional peaks in the traffic demand, such as during events where big crowds are gathered around (concerts, football matches, etc.) directing the UAV to the target area and so enhancing the capacity of the cell.

This work originates from the union of these two innovative concepts, analyzing a scenario in which a UAV serving as base station flies over a known area collecting data from clusters of IoT nodes. Among the set of standards that in the recent years have been conceived to work with this type of devices, we focused the attention on the so-called Narrowband - Internet of Things (NB-IoT).

Introduction

Narrowband IoT was born within the context of low-power wide area networks (LPWAN), with the 3GPP Release 13 (part of the LTE specifications), and pursuing mainly three goals: serving a massive number of devices in a cell, while reducing as much as possible the power consumption and improving coverage with respect to the previous cellular standards. These achievements are finalized to the aim of serving a very large number of IoT devices, being conceived in the context of the so-called *massive* Machine Type Communication (mMTC): NB – IoT is indeed theoretically able to support the connection of up to one million devices per km^2.

Peculiar characteristic of this standard is the deployment flexibility that it allows, under the point of view of the integration with standards for cellular networks too: it can coexist both with older and more recent generations, i.e. with 2G, 3G, 4G and the incoming 5G. NB-IoT is indeed developed as part of the LTE specifications and maintains as a consequence the same numerology, while is compatible with the GSM simply through an update of the devices, drastically reducing the costs with respect to the requirement of brand-new hardware. More in details, this is made possible by means of its three modes of operation: stand-alone, which provide the possibility to deploy NB-IoT in the GSM refarmed band thanks to it extremely reduced bandwidth (180kHz); in-band, meaning its deployment within the LTE useful band, utilizing one of its physical resource blocks (PRBs); guard-band, again exploiting one LTE PRB, this time exploiting the guard band, as the name suggests.

Beside the main features of the standard itself, we have to underline that in this study we simulated via Java and Matlab environments a scenario in which a single drone is exploited to carry the NB-IoT base station over an a-priori known area. More specifically, we examined the performance of a drone flying over a area where IoT devices are grouped in clusters and trying to serve as many nodes as possible, with the trajectory solved exploiting the well-known solution of the Travel Salesman Problem (TSP), through which the UAV, that in this study we consider starting from the center of a cluster, finds the minimum path to subsequently reach the center of all the other clusters before reaching again the starting point, where the flight is considered ended. Starting from this assumption we wanted to find the scenario that can best benefit from joint use of NB-IoT and UAV, analyzing both its network throughput and how resources are scheduled by this standard. Then we want to compare the previous results with the new ones obtained changing the way random access is simulated in the code.

This premise done, the work is organized as follows:

- In the first chapter an overview of the state of the art is given, introducing technical explanation of the standard and reporting some of the latest publications on the argument;
- In the second chapter I analyzed the network throughput of the UAV base station varying the cycle time duration in a range of values that at the best of my knowledge had not been deeply investigated yet. Then, we wanted to further investigate how the resources are assigned by the drone while the UAV fulfill its trajectory over the clusters of the considered scenario, in terms of given RUs among the available amount. We also evaluated how the node distribution and the dimension of the area considered affects that;
- In the third and last chapter, the study goes in the direction of implementing a simulation of the random access in a faithful way with respect to the mechanism described in the standard, substituting the simpler access probability that we used until this point.

Chapter 1 – NB-IoT in 3GPP Realease 13

Before describing the simulations performed in this first chapter, a contextualization of what we are going to examine and simulate is mandatory.

To the end of this chapter, that is to analyze the network throughput, the description of NB-IoT is limited to the uplink phase for the sake of simplicity.

1.1 Uplink physical resource grid

A crucial aspect to specify before illustrating the first simulations, is the description of the time/frequency resources proper of the uplink, called Resource Unit (RU) and peculiar of the NB-IoT standard. When 12 tones are allocated for the uplink, and with a subcarrier spacing of 15 kHz, one RU corresponds to a LTE Physical Resource Block (PRB) pair, i.e. a device scheduled bandwidth of 180 kHz, and in time it lasts the duration of one frame (1 ms) as regarding the frequency In the case under examination of NPUSCH format 1 with 3.75 kHz device scheduled bandwidth and with a slot length of 2 ms, one RU has a duration of 32 ms, with this extended duration due to the compensation of the thinner bandwidth.

Table I - RU Duration and number of subcarriers per transmission mode

Transmission Mode	Subcarriers	Subcarrier Spacing [kHz]	RU Duration [ms]
Single-Tone	1	3.75	32
	1	15	8
Multi-Tone	3	15	4
	6	15	2
	12	15	1

It is important to underline this feature since it will have a deep impact on the performance achieved with the chosen cycle time (or, that is the same, NPRACH/NPUSCH periodicity), reminding that we consider a set of four possible values (80, 160, 320 or 640 ms).

Therefore the cycle time in this study is composed by one NPRACH plus one NPUSCH.

Considered that NPRACH basic TTI is in this study of a fixed duration of 7 ms, as a consequence the duration of NPUSCH is equal to the duration of the duration of the cycle time minus the NPRACH, i.e. 63, 153, 313 and 633 ms for our reference values.

1.2 Uplink physical channels

NB-IoT uplink uses single-carrier frequency-division multiple-access (SC-FDMA) and it supports both multitone and single-tone transmission, the former with a subcarrier spacing of 15 kHz and a larger capacity (to extend the coverage domain) with respect to the latter that supports both a 15 kHz or 3.75 kHz subcarrier spacing. In this last case we have a larger capacity, being able to serve a lot of devices at the same time, This is the case we focus on in this study. Regarding the single-tone transmission mode with 3.75 subcarrier spacing, SC-FDMA is mathematically identical to OFDM. The set of NB-IoT uplink channel is composed by three signals, that are: NPRACH NPUSCH and DMRS, with the last one that fulfills signaling functions and is always strictly related to the corresponding NPUSCH characteristics, being transmitted inside a NPUSCH slot, so we will omit it in the rest of our examination since it is not relevant for the aim of this study.

1.2.1 NPRACH

In order to serve NB-IoT devices in different coverage domains, there are three format of NPRACH channel (Format 0, 1, 2), presenting always the same structure but with different frequency and time resources allocation. For our purpose, Format 0 is the only one utilized and so we focus on this one. For Format 0, each symbol group is composed by a cyclic prefix (CP) of 66 μs plus five OFDM symbols, that being single-tone with subcarrier spacing of 3.75 kHz, have a duration of 266.67 μs each, and can support users at a distance up to at least 10 km from the base station, that can be even larger when deployed in rural areas.

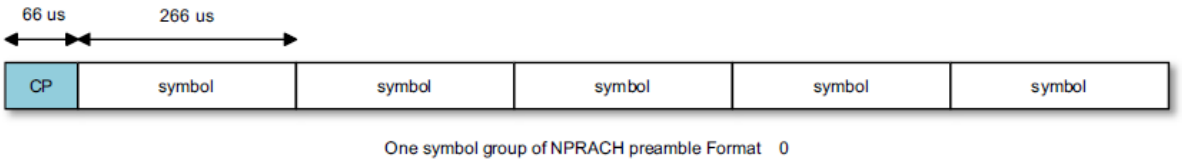

Fig.1: NPRACH symbol groups for Format 0 (FDD)

However, a single symbol group repetition would not guarantee a sufficient probability of correct reception at the receiver base station, so it has been established that a basic NPRACH preamble unit must be constituted by a repetition of four symbol groups. Each symbol group is transmitted on a different subcarrier and following a specific frequency hopping pattern to select the subsequent tones (as described in the figure x),

Table II - Deterministic hopping patterns within a NPRACH repetition unit (FDD)

	Index of the tone used by the first symbol group	Deterministic hopping patterns within a repetition unit
Format 0 & Format 1	0, 2, 4	{+1, +6, −1}
	1, 3, 5	{−1, +6, +1}
	6, 8, 10	{+1, −6, −1}
	7, 9, 11	{−1, −6, +1}

after that the first one is randomly chosen among a set of 12 continuous tone, in a range whose starting frequency is primarily decided by a frequency offset. In this way, the index of the subcarriers occupied after the first one is known and deterministic, and thus two devices that at the beginning of a certain cycle choose the same initial subcarrier, will collide for the entire NPRACH duration. As an example, the following figure shows a frequency hopping pattern of (+1, +6, -1).

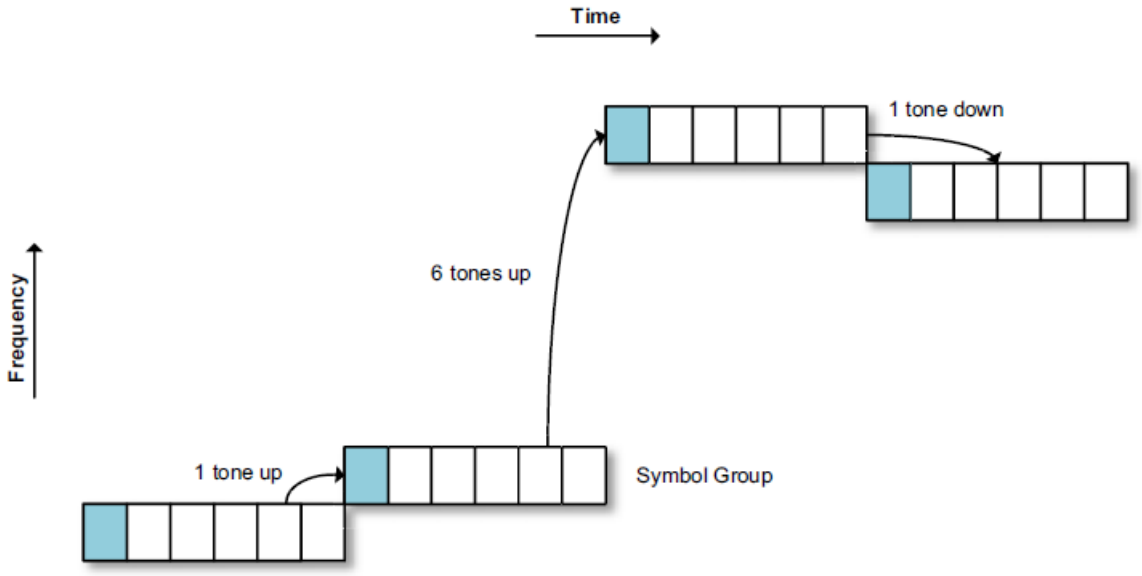

Fig.2: One NPRACH preamble repetition unit and an example of the tone relationship between the four symbol groups

1.2.2 - NPUSCH

NPUSCH is the signal that actually allows the data transmission in uplink, besides carrying information from the higher layer. With 3GPP Release 15 (2018), it also allows the device to alert the base station of its will to transmit data (scheduling request), and, on the contrary of LTE, with NB-IoT as already mentioned it’s possible to schedule also sub-portion of a PRB.

The rationale behind this choice is that this standard target ultra-low-end IoT devices, thus the assignment of an entire PRB would often be a waste of resources to serve a demand of small data packets. In fact in coverage limited scenario the devices constraints are in terms of power consumption rather than in the requested bandwidth. Moreover, in order to accomplish the requirement of extending as much as possible the battery life, having a low peak-to-average ratio (PAPR) in the waveform (that is a SC-FDMA in this case too) is fundamental, especially for nodes located at the boundaries of the cell, and the use of sub-PRB helps in this too. According to the data the device has to transmit, there are two format of NPUSCH. We adopt format 0, that allows single-tone transmission and 3.75 kHz subcarrier spacing, which implies a slot duration of 2 ms in this case. This slot format is constituted by seven SC-FDMA

symbols (lasting 275 microsec each), preceded as for the NPRACH by a cyclic prefix, this time shorter (8,33 microsec). For the type of devices under consideration provides a maximum Transmit Block Size (TBS) of 1000 bit.

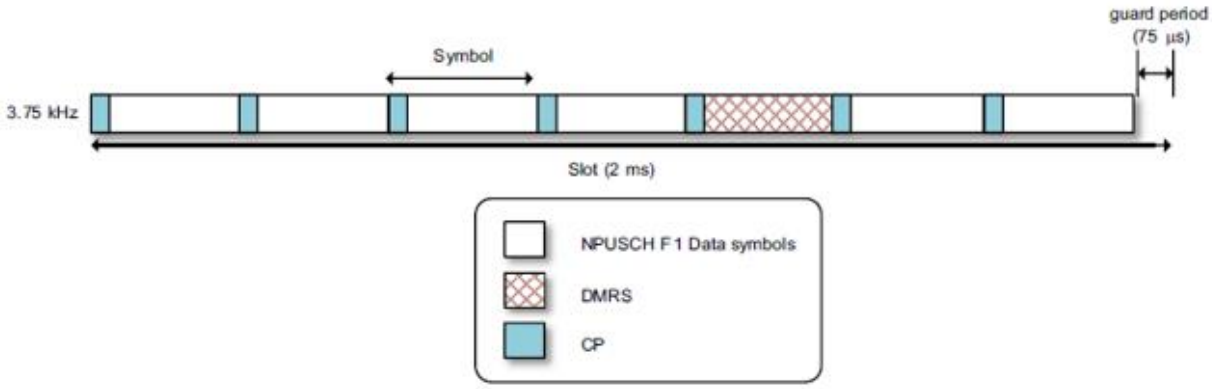

Fig.3: Slot format for NPUSCH Format 1

1.3 Coverage enhancement (CE)

Both NPRACH and NPUSCH can be transmitted with 1, 2, 4, 8, 16, 32, 64 or 128 repetitions to allow devices in poor and extremely poor coverage conditions to access to the channel/connect with the base station, but reducing in this way the time/frequency resources dedicated to the data transmission, therefore a trade-off is needed. About this, we must say that three range of coverage conditions are consented in NB- IoT: coverage enhancement (CE) 0, 1, or 2, and in the extreme case of CE 2, a Maximum Coupling Loss (MCL) of 164 dB is allowed, that represents a 20 dB coverage enhancement with respect to GSM/GPRS. The drawback is a lower data rate, so a trade-off is needed. In the scope of our study, the CE 0 only is taken into account, implying a MCL of 144 dB, therefore considering the IoT nodes

in good coverage, and no repetitions, reducing to the minimum the resources dedicated to signaling, with both these choices motivated by the aim of investigating the best achievable performance.

Table III - NPRACH link budget

Coupling loss	**144 dB**	**154 dB**	**164 dB**
Required UL SINR [dB]	14.3	4.3	-5.7
Receiver sensitivity [dBm]	-121	-131	-141

Table IV - NPUSCH single-tone physical and MAC-layer data rates

NPUSCH single-tone (3.75 kHz)	**Stand alone [kbps]**	**In-band [kbps]**	**Guard-band [kbps]**
Peak Phy layer data rate	5.5	5.5	5.5
Peak MAC layer data rate	4.8	4.8	4.8

1.4 Power saving

An important improvement in power savings (and therefore in the life of the battery) for mMTC, introduced in Release 13, is given by the presence of two different modalities of functioning during the idle mode, that are the extend discontinuous reception (eDRX) and the power saving mode (PSM). This attention does not have to arouse astonishment because IoT devices are meant to stay much part of their lifetime in this to ensure a battery life as long as possible. The choice between one or the other mode is mainly dictated by the reachability period, defined as the interval between paging occasion, required from the device (or, in other words, by the type of application for which is committed), since for the former this interval has to be within 3h, while for the latter the power consumption is reduced to the minimum thanks to the fact that this periodicity can exceed 1 year. The difference and at the same time the advantage between PSM and completely turning off the device is that it stays registered in the network, reducing the signaling overhead when it comes back available for paging. During eDRX or PSM only a timer is on to maintain knowledge of when is periodically moment to get off these states.

Then, when the device leaves the PSM as a consequence of data passed from the upper layers to be sent in UL, the random access procedure starts. This is in large part equal to the LTE RACH, so here

we underline the characteristic aspects, i.e. the peculiar exchange of messages illustrated in the following figure:

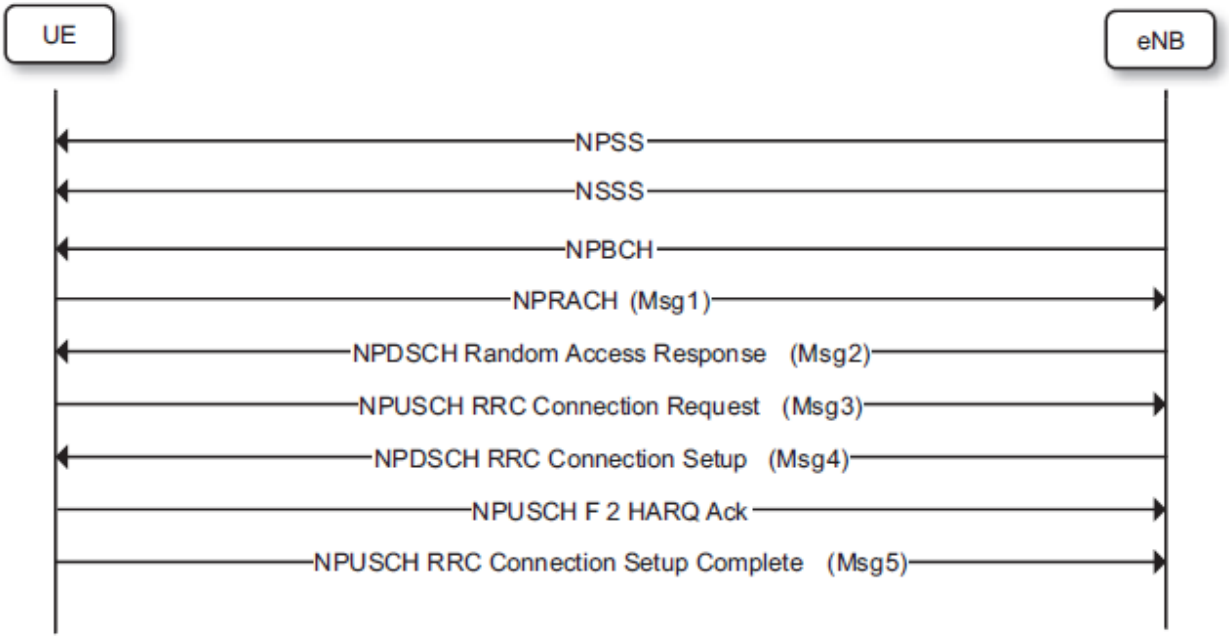

Fig.4: NB-IoT random access procedure

When with the Radio Resource Configuration (RRC) Connection Setup any possible contention among devices which selected the same initial preamble is resolved by the network, with the completion of the setup (Msg 5), the connected mode is established and the device will be able to send its data through the NPUSCH channel.

1.6 Recent studies

At the best of my knowledge there are still few study cases in literature where joint use of NB-IoT standard and UAVs is studied on the field or simulating real scenarios. Among the latters, a recent scientific publication in which IoT devices with sensor are used to collect underground soil parameters in potato crops, using also in this case a single drone in which is mounted a NB-IoT base station. It is shown that with 50 seconds of fligth time it could satisfy more than 2000 sensors deployed in 20 hectars.

Another study presents a networking level simulation where NB-IoT infrastructure is exploited to track containers transportated by cargo vessels operating near the coastline. Comparison are made between the direct sensor-to-onshore base station and the use of a UAB, acting as intermediate node between sensors and on-shore backhaul network. Resulting performance show that the UAV aided system improve the connectivity quality, considering a range of input parameters.

Chapter 2 – Scenario and simulation analysis

2.1 – Reference scenario

In this thesis Java environment is used to perform the simulations, imprinted at the networking level, while all the input settings and parameters both for the drone and the scenario itself are built in MATLAB, as well as the plotted charts.

Table V – Main Network parameters

Channel bandwidth	180 kHz
Subcarrier spacing	3.75 kHz
Available subcarriers	48
Carrier frequency	1747.5 MHz
RU duration	32 ms
Time slot duration	2 ms
Number of slots per RU	16
Number of OFDM symbols per slot	7
MCS Index	6

First of all we have to specify our assumption on users distribution, because until now we have only mentioned that they are located in clusters. We consider an urban scenario, where the devices can be located in whatever part of the city, both outside in parks, crossroads, squares, rooftops or in more specific location of application like for example can be the so called smart bin, and inside in buildings. The scenario is modeled using a particular Poisson Cluster Process, named Thomas cluster process (TCP), that is a stationary and isotropic Poisson process and is characterized by a set of child nodes independently and identically distributed (i.i.d.) around each point of a Parent Point Process (PPP). These last ones are randomly located in the space, with a rate λ_p that is their average density in the considered scenario. In this study we refer to these points as *cluster-heads*, but they have only a mathematical meaning, being the nodes located at the exact center of the cluster where the UAV flies over during its trajectory generated by the already mentioned TSP, and it does not exist any difference in the simulated behavior from the *child-nodes*, distributed around them according to a symmetric normal distribution with variance σ^2 and mean value n. Therefore the child-nodes rate is $\lambda = \lambda_p * n$.

In this study, like usual dealing with real scenarios, NB-IoT devices can be found in Line-Of-Sight (LOS) or in Non-Line-Of-Sight (NLoS) with the drone. In the latter situation the signal does not

arrive directly to the users but interacts with objects near the location of the devices before reaching them, resulting in a shadowing effect. Anyway we can generalize the path loss model as follows:

$$L(d)[dB] = 20\log\left(\frac{4\pi f_c d}{c}\right) + \xi + \eta$$

where the case of LoS or NLoS can be taken into account simply changing accordingly the value of the shadowing coefficient ξ, while f_c is the carrier frequency, c is the speed of light in the medium, d is the distance user-to-drone and η is the penetration loss, to take into account in case of indoor deployment.

Table VI – Main Radio parameters

P_{tx}	14 dBm	Uplink transmitted power
A_L	2.5 dBi	Antenna Loss
η	40 dB	Penetration Loss
P_n	$30*10^{-17}$ W	Noise Power
$P_{rx,min}$	-121 dBm	Receiver sensitivity

The area over which the drone flies and in which all the users are located is a square area of dimension L x L m^2, where in the first simulations done we considered L = 500 m but the results we obtained were not satisfying since our assumptions would have been denied, especially related to the range of the UAV, i.e. its radius of visibility of the users, that was fixed at 300 m as we considered at the beginning, because with these settings the concept of cluster was lost. Indeed the drone was able to see on average the 75% of nodes in each instant, so the concept of clusters was not true. After the side of the square area was set to L = 1000 m, according to the evolution of this study, and the range of the drone reduced to 30m, in order to well distinguishing each cluster during its flight. Similar, for the height of the UAV we firstly considered a range of values between 50 m and 100 m, as shown also in previous figures, in order to find the optimal in term of network throughput, that has proved to be 50 m.

Table VII – Main scenario parameters

Side of the squared area	1000 m
Speed	20 m/s
UAV height	50 m
UAV range	30 m
Average number of cluster-head number	5
Average number of node per cluster	101
Variance σ^2	100 m^2

2.2 – Users nomenclature

In order to perform the simulations for the network throughput output, it must be explained which are the input and how they are calculated. The only users that contribute to the network throughput are the ones that complete the upload of the demand, fixed for all to 500 bit in this study case, and that will be named served or satisfied users.

Before reaching the phase in which they effectively send their data, NB-IoT users must satisfy certain requirements that are depicted as follows:

- Users in range: the first condition is that the users location is within the range of the drone, that is fixed, and users have the chance to be seen by UAV only for a smaller or bigger period of the trajectory time, according to parameters like the dimension of the range itself and of the area considered
- Users in coverage: accomplished the first requirements, now to go on with the procedure and transfer the data, the drone received power ha to be above a threshold that is its receiver sensitivity P_{r_min}, that we put at -121 dBm, while the IoT nodes transmitted power is fixed at 14 dBm.

$$1) meanUsersInCoverage = \frac{(\sum_t UsersInCoverage(t))}{Flight\ Time}$$

$$2)\ if\ (User\ is\ active) \rightarrow if\ (Pr > Pr_{min}) \rightarrow User\ in\ coverage$$

- Users trying the random access (RA): at this point, users have to competitively participate to the RA procedure, that in the initial simulations is calculated according to the probability of success, i.e. $P_{acc,u} = e^{-\frac{U}{N_{RU}}}$, where Nru is the number of available subcarriers and U is the number of users entering the RA. The probability of collision is directly derived [1- $P_{acc,u}$]

$$1) meanUsersConnecting = \frac{(\sum_t UsersConnecting(t))}{Flight\ Time}$$

$$2)\ if\ (User\ in\ coverage) \rightarrow if(User\ succeed\ the\ RA) \rightarrow UserConnecting$$

- Users getting RU: users that success the RA are connected and have finally the right to get resources, that is they are assigned to a subcarrier and send Rus. cycle by cycle RUs. If the cycle time duration does not allow to complete the upload within one round, connected nodes keep a reservation on the given subcarrier, and can automatically continue the transfer in the subsequent cycles until they complete the transfer of the 500 bit of the initial demand, assuming that they are still in the range of the drone and in coverage.

$$1) meanUsersGettingRu = \frac{(\sum_t UsersGettingRU(t))}{Flight\ Time}$$

$$2)\ if\ (User\ in\ coverage) \rightarrow if(User\ connected\ \&\ not\ satisfied) \rightarrow if(any\ tone\ has\ residual\ RUs) \rightarrow UserGettingRU$$

- Users satisfied: at the end, NB-IoT devices that manage to send all the bit of demand, under the form of RUs, will be the ones counting for the throughput, that is calculated as follow:

$$Throughput = \frac{Demand\ [bit]}{Delay\ [s]}$$

where $Delay = Time\ at\ current\ cycle\ [s] - Device\ arrival\ time\ [s]$

2.3 Network throughput and served users

2.3.1 Demand – RUs relationship

Reminding that each user requests the upload of 500 bit, we have to specify the relationship between RUs and maximum Transport Block Size (TBS) set by the NB-IoT standard, shown in Table VIII.

Table VIII

NB-IoT UL TRANSPORT BLOCK SIZE (TBS) IN BITS

RUs number	2	4	5	6	8	10
Max. packet size for $I_{MCS} = 6$ [bits]	176	392	504	600	808	1000

This clearly shows that with our assumption, 5 RUs are needed for the completion of the demand .

Knowing that in our case study a RU is 32 ms long, the number of RUs made available by each subcarrier for the different cycle duration is trivial, given by simple math division:

(CycleTime - NPRACH)/32ms

From the output of this formula we derive the RUs each subcarrier make available with the corresponding cycle duration:

80ms → 2 RUs

160 ms → 4 RUs

320 ms → 9 RUs

640 ms → 19 Ru

2.3.2 Throughput trend analysis

Figure 5 shows the network throughput for the different cycle times taken into account, while figure 6 the corresponding served users.

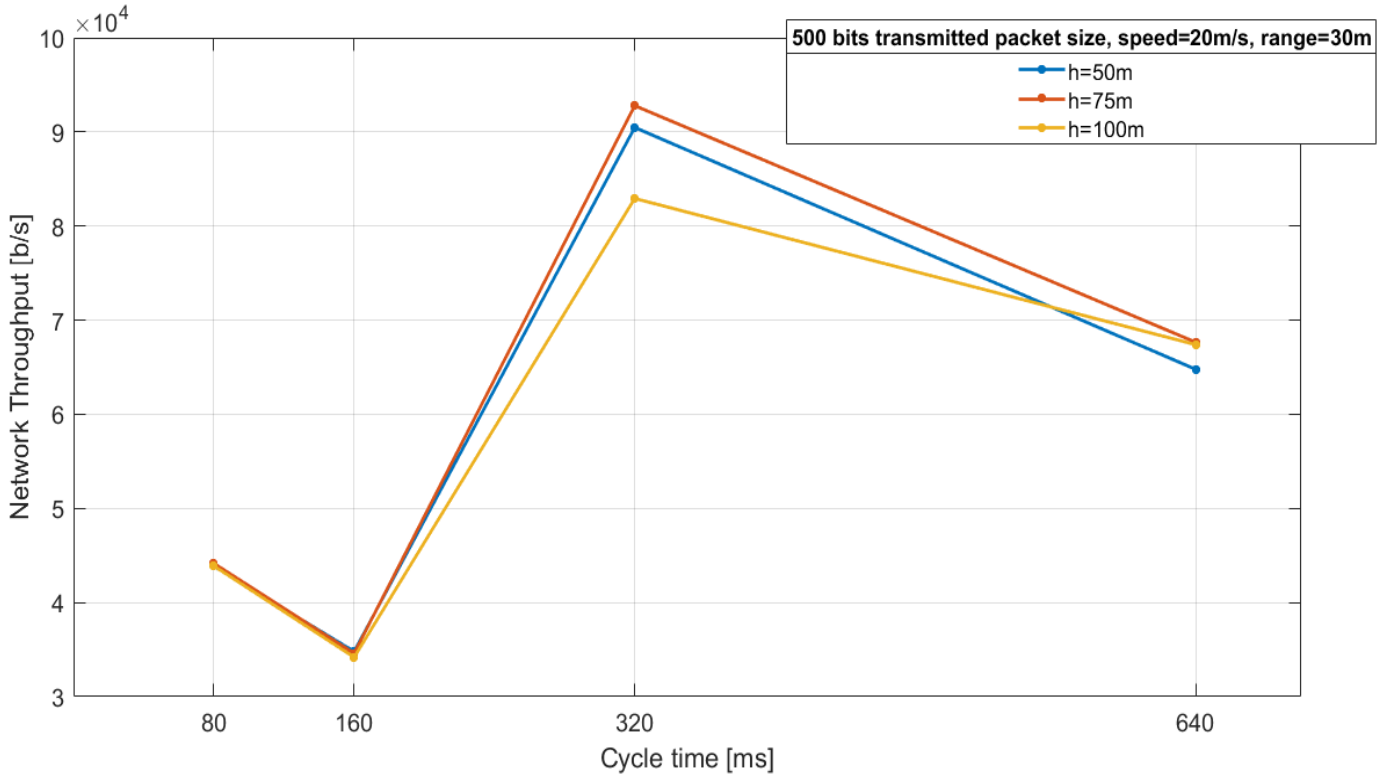

Fig.5: Network throughput per cycle time duration

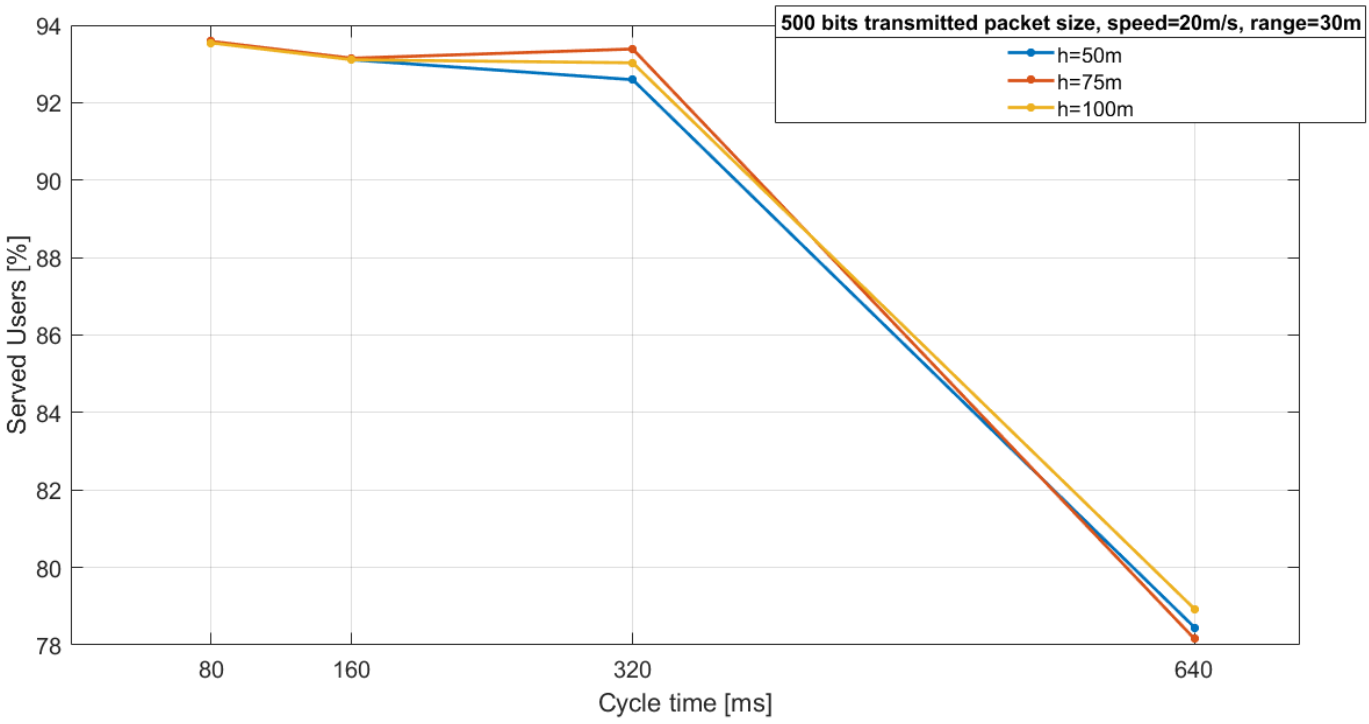

Fig.6 Served NB-IOT devices (%) per cycle time duration

Seeing now the resulting throughput, a question may arise: at first glance we would expect an increasing trend from 80 ms up to 320 ms concurrently with the increase of the available RUs, and then a drop down to 640 ms when resources are in excess with respect to the need. In order to understand the reasons behind this behavior, I chose to indagate what happens for other values of the cycle time beside the default ones. These new additive values exactly correspond to the spurt of an extra Rus, starting from 1 and going up to 5, focusing in the range of 160 ms because this is the point in the graph not immediately clear. We must underline that certain values of the cycle we examined have only analytical relevance; indeed only certain number of RUs are allowed to be assigned within a cycle by the standard and, as shown in the previous table, 3, 7 and 9 are not permitted numbers. Hence for the first point, corresponding to 1 Ru, we will have a cycle time that is equal to the length of the RU itself (32 ms) and representing the NPUSCH, plus the duration of the NPRACH (7 ms), that is 39 ms; for the case of 2 Rus, it will be 32+32 = 64 ms of NPUSCH + 7 ms of NPRACH, thus 71ms of cycle time, and so on and so forth up to 5. The resulting step behavior, with a decreasing trend from the interval between a new available RU and the subsequent, is predictable since in these ranges only the delay increases as we increase the cycle time. What may give rise to questions is the point in which 4 RUs become available, that is 135 ms. Here we don’t have as usual a peak but an extension of the decreasing trend that starts at 103ms / 3 RUs. This is well explained as the number of cycles required to satisfy the 500 bit of demand remains the same, that is two cycles, so we are still increasing the delay. Then, at 167 ms, value that corresponds to the availability of 5 RUs with no additional overhead, there is the max of the network throughput since the demand would be satisfied in only one cycle, while after that the function becomes monotonically decreasing, with an explanation similar to the analogous reasoning made for the case of 4 RU.

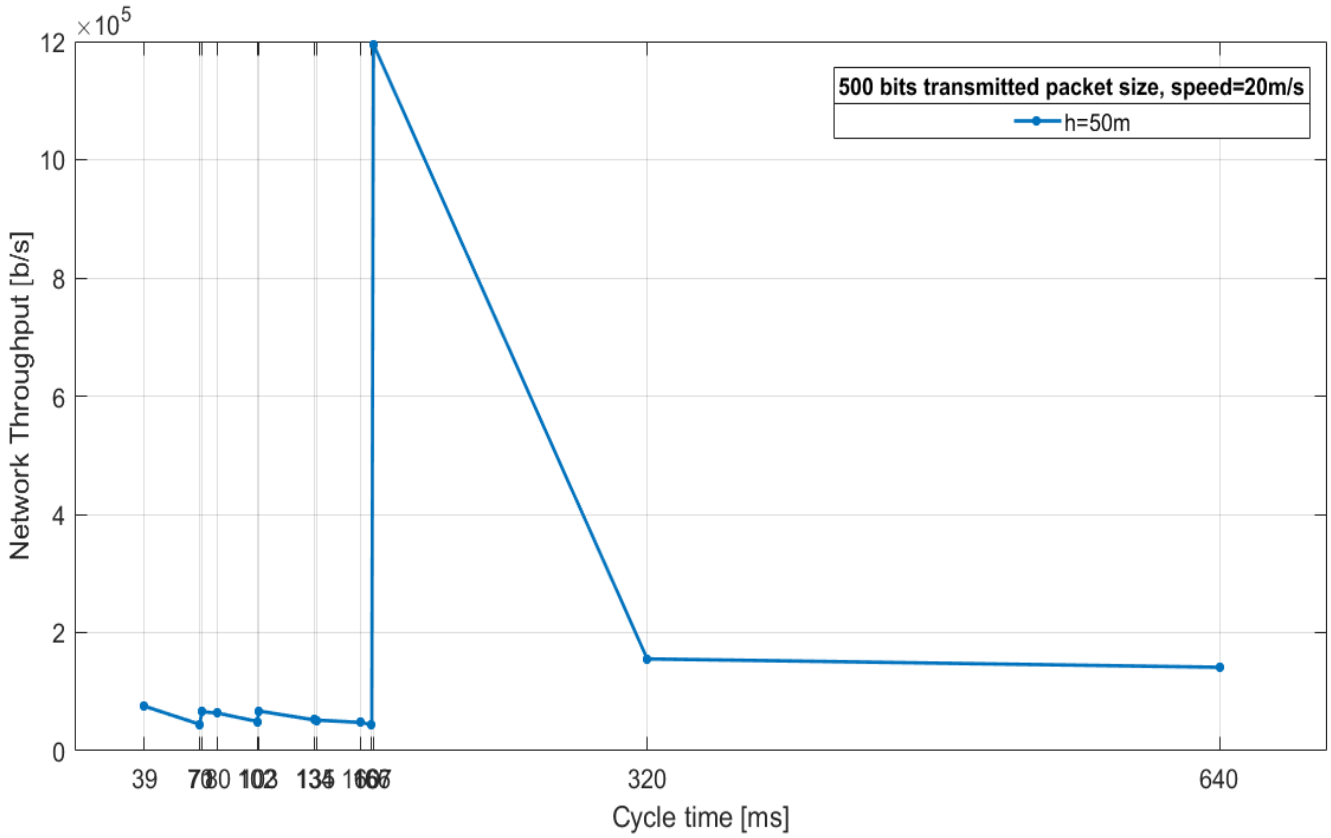

Fig.7 Network throughput increased grid

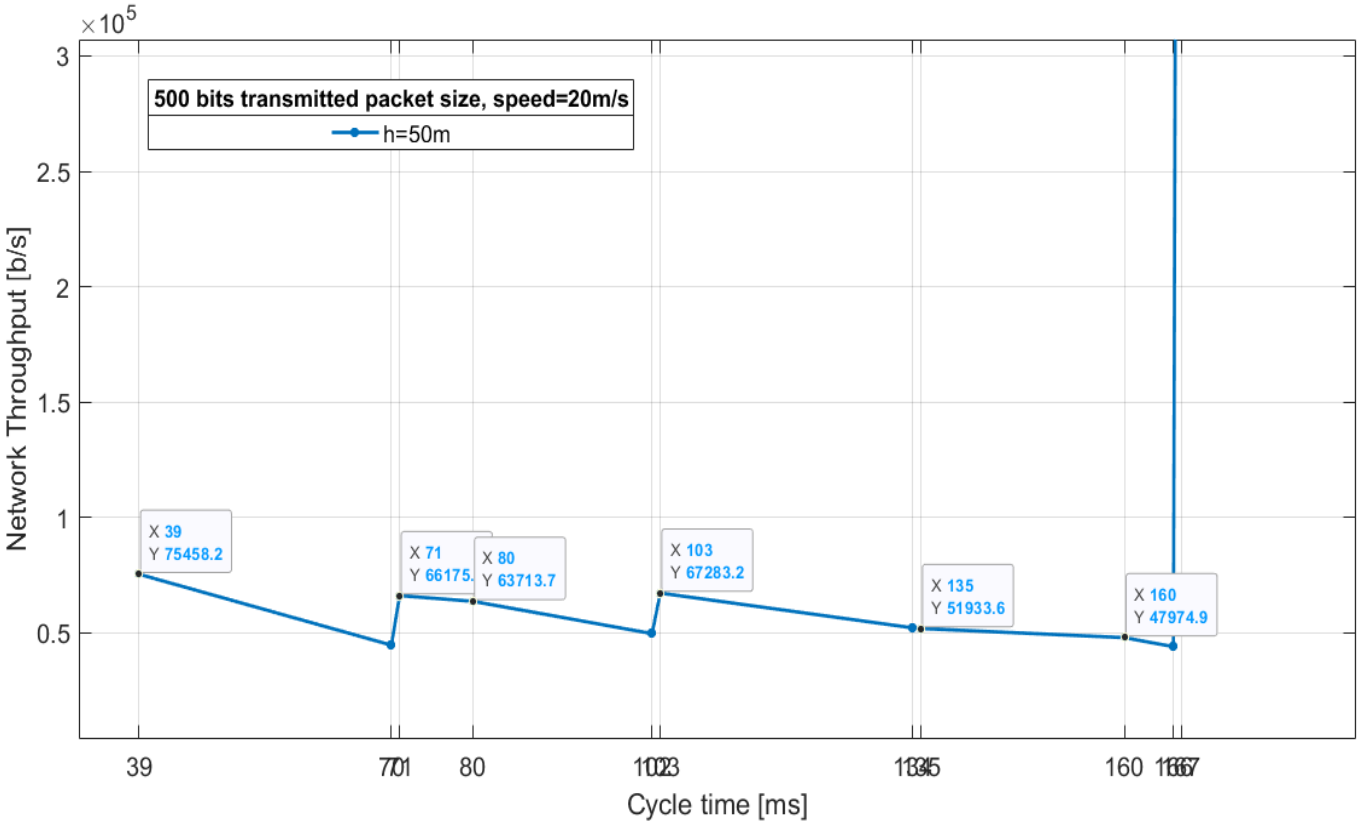

Fig.8: Network trhoughput zoom on increased grid

2.4 – A deeper insight: NB-IoT devices trend

We want now to have a deeper insight of how those values of network throughput and delay are obtained, checking for each of the four values chosen for the cycle time the mean numbers of the users in coverage, connecting and connected/getting resources (these two nomenclatures are in fact referred to the same condition of the users, since only devices willing to transmit data compete for the access, and once they succeed, for sure they are given resources and stay in this state until the demand is satisfied or they loose connection because the drone moves too far from them or again because the channel condition make the received power below the threshold) averaged for cycle, according to the definition previously given of these parameters.

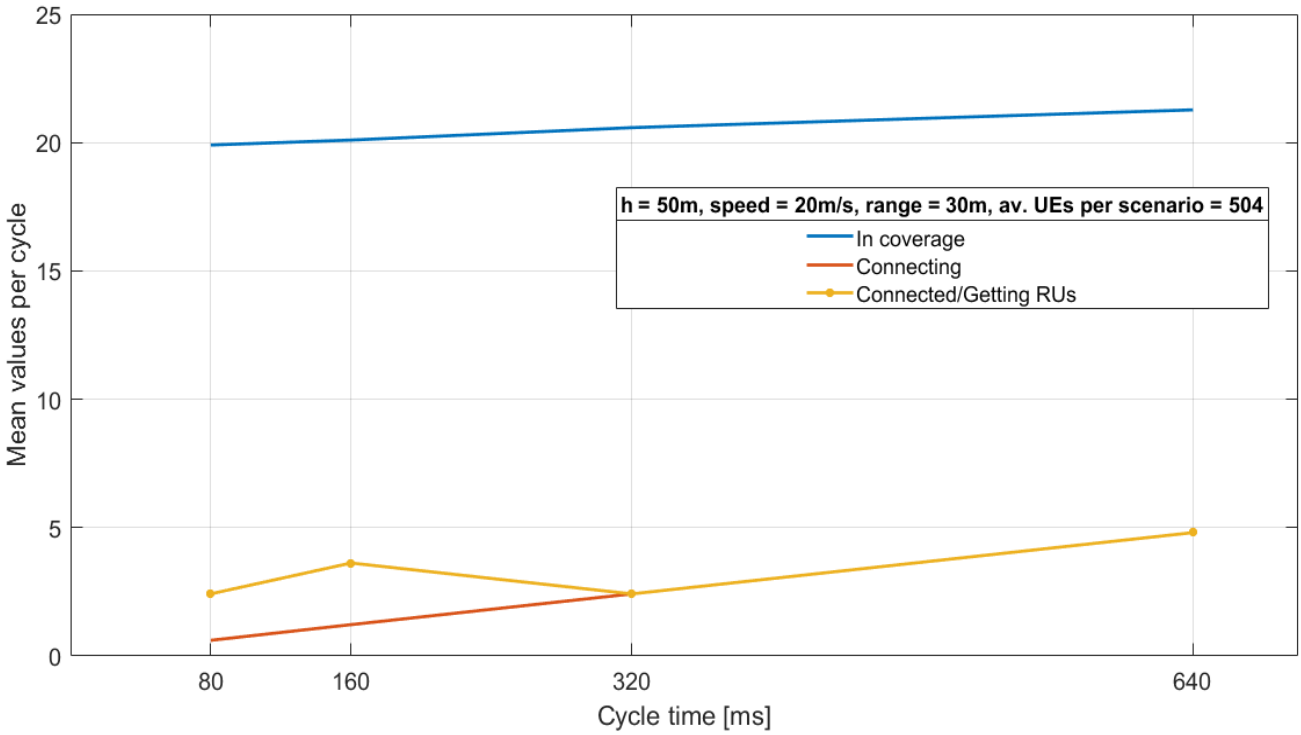

Fig.9 Mean values of users state per cycle

The average per instant is calculated along all the drone trajectory and not only while it is above a cluster, resulting in low absolute values since we are taking into account many cycles in which there is no user in the range of the drone. At first sight it appears immediately clear that the number of users in coverage is the highest one and with a substantial margin with respect to the other two, since only a fraction of users can at the end access the channel. Moreover it shows correctly a flat trend because the UAV received power, which determines the number of users in coverage, is not function of the cycle time duration but of the channel impairments. What instead needs to be investigated is the

general non-linear trend of connecting and connected users. Especially, for both, the local maximum at 160 ms, that was the point of minimum in the throughput. Thus again, similar to the method previously adopted, I plotted the same graph adding extra cycle time values finding out that the maximum stands for 166ms, that is the longest possible duration (using millisecond as unit of measure of the time) before that 5 RUs become available, and then abruptly decreases again.

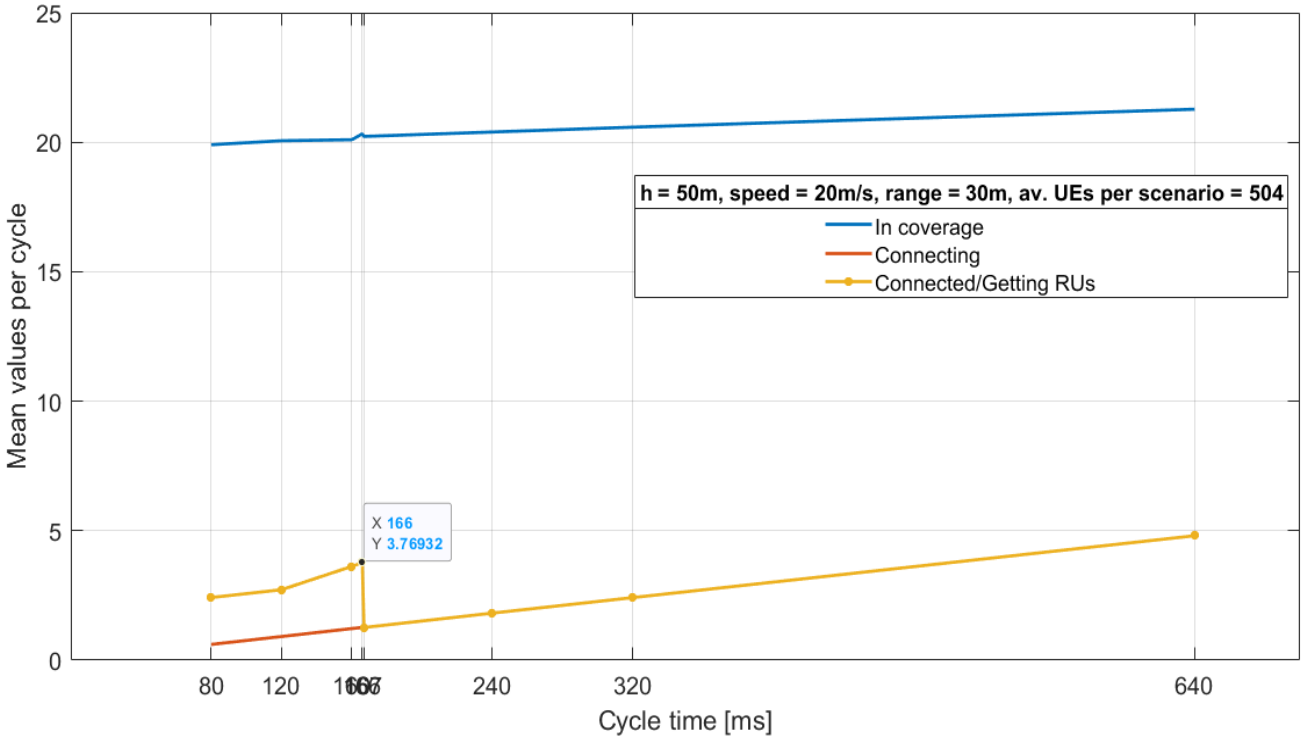

Fig.10: Mean values of users state per cycle increased grid

This can therefore be explained by the fact that until that moment users need more cycles to satisfy the 500 bit of demand, so passing more time active in the network before becoming silent. Anyway, limiting the discussion to what said up to now, someone would perhaps expect

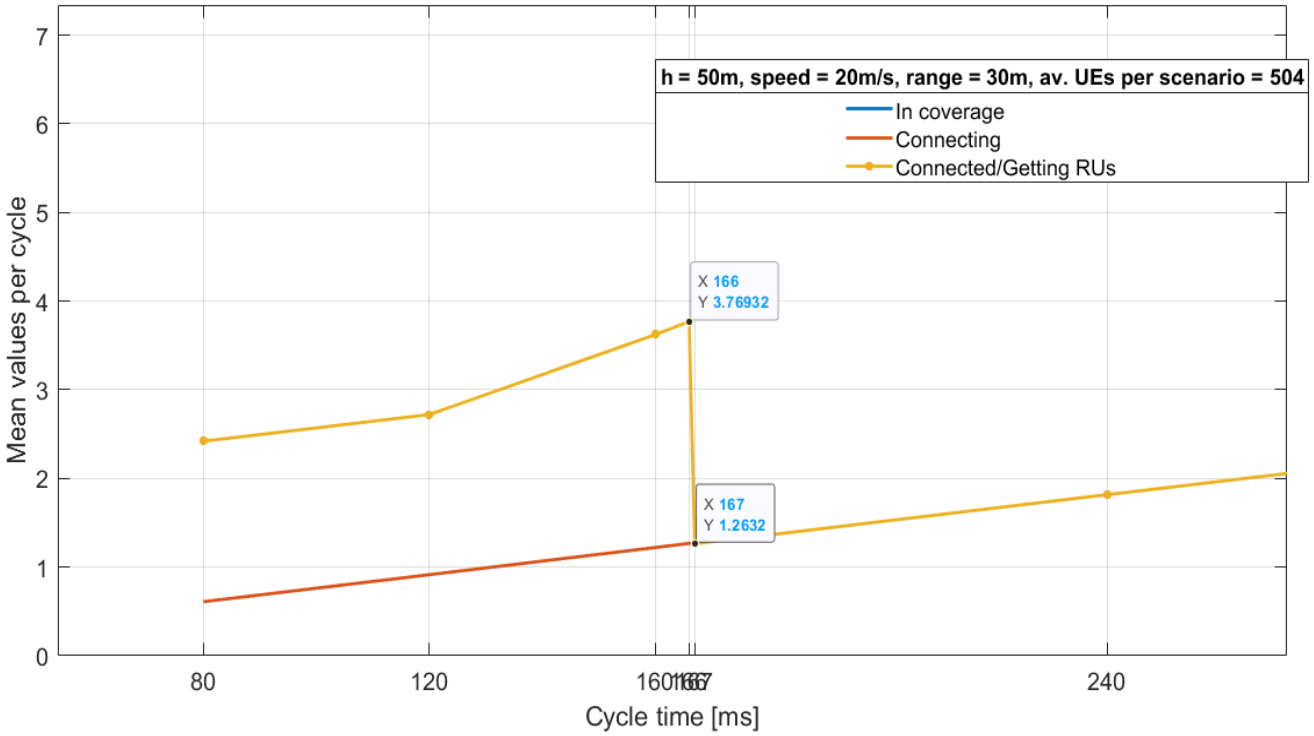

Fig.11: Mean values of users state per cycle, zoom on 166 ms cycle time duration

to find the maximum at 80 ms, having less resources per tone, which translates in a longer time attached to the network for each user (in terms of number of cycles, 3 with 80 ms vs 2 with 160 ms). But the increasing trend of the number of connected users in the first stretch (from 80 ms to 160 ms) is the result of a trade-off between the time passed being connected, that is reduced by a factor $\frac{1}{3}$, and the frequency of random access occasions, that is halved.

Starting from the value of cycle time of 320 ms and going forward, the number of users connecting and connected overlaps. This behavior should not be surprising because we must remember that the 500 bit of users' demand is now on satisfied in just one cycle (with 320 ms each subcarrier can assign up to 9 RUs to the same user, 4 remaining unused in our assumption. A further study could go in the direction of reassigning the ones left) and thus no user holds the assigned subcarrier for more cycles. For this reason, this behavior is reasonable and correct (strictly speaking the same behavior would come out starting from 167 ms, but our grid is clearly less dense).

The linearly incresing trend between 320 ms and 640 ms in instead due to the halved frequency of access occasions, for which at the beginning of each new cycle there is a larger accumulation of users that have data to transmit: indeed the connecting/connected users are doubled.

The previously shown outcomes on network throughput and delay were obtained averaging among ten scenarios, each characterized by a different position and number of both clusters and nodes, while maintaining the same distribution and channel characterization. Now, to have a deeper level of detail, let's check what happens inside a particular scenario, taking as example one among those ten, in which there are four clusters and 380 IoT nodes.

2.5 – Study case: a particular scenario

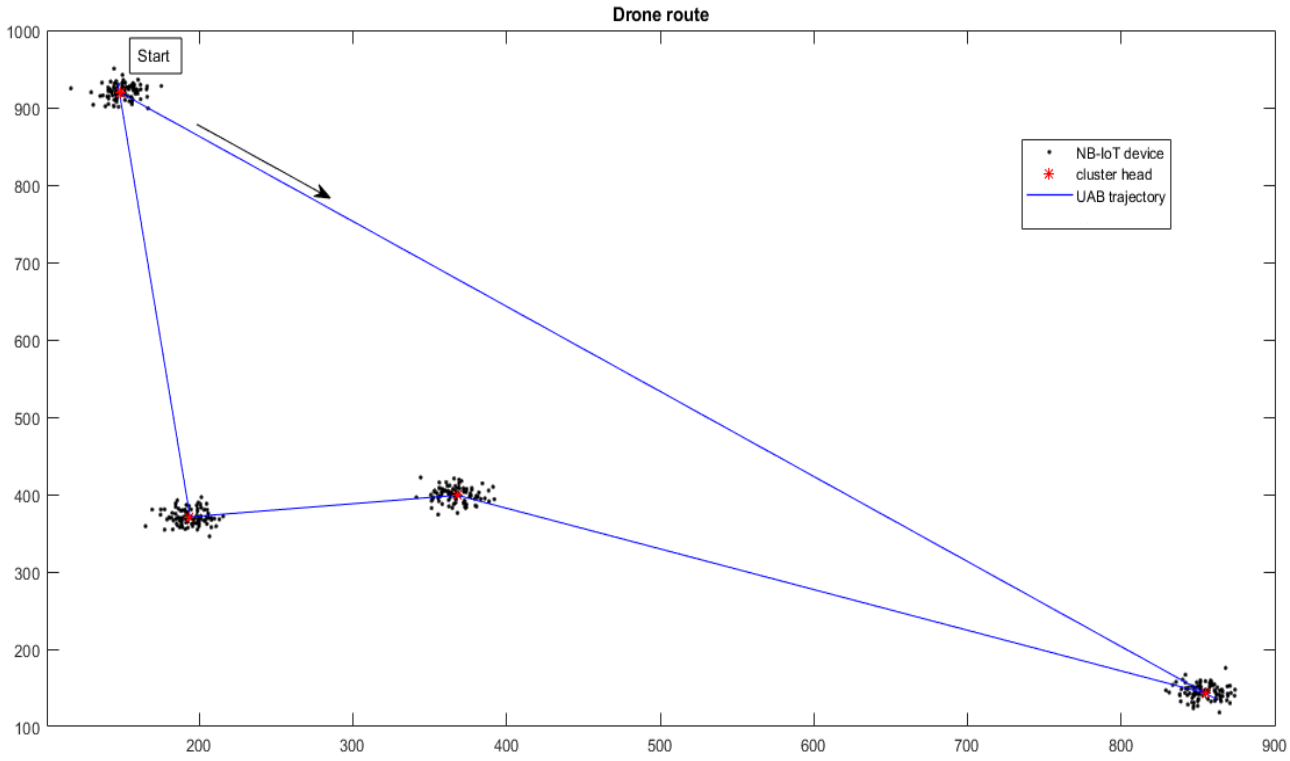

Fig.12: Examined scenario

We are particularly interested in how resources are assigned inside the clusters when the drone flies over them, and consider as an example the case of 80 ms duration of the cycle. The following figure show the behavior:

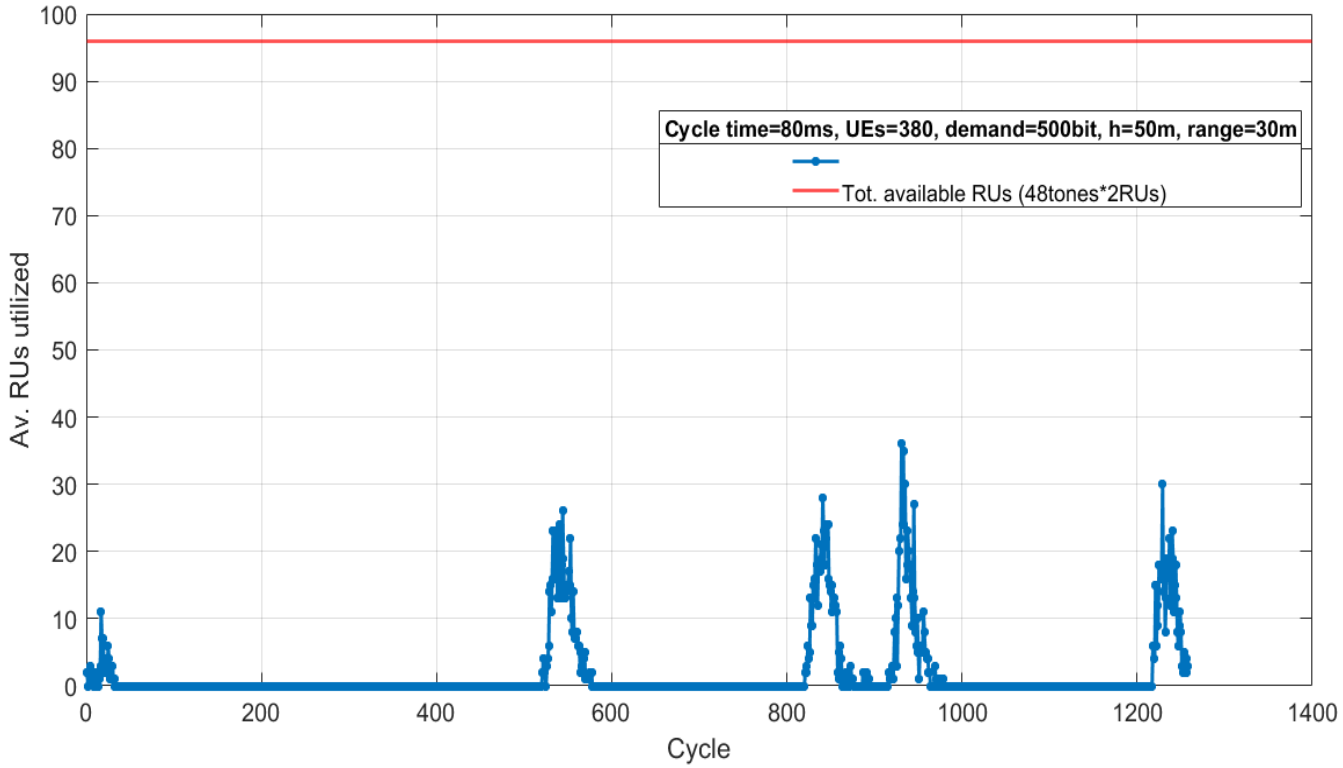

Fig.13: RUs assignment per cycle

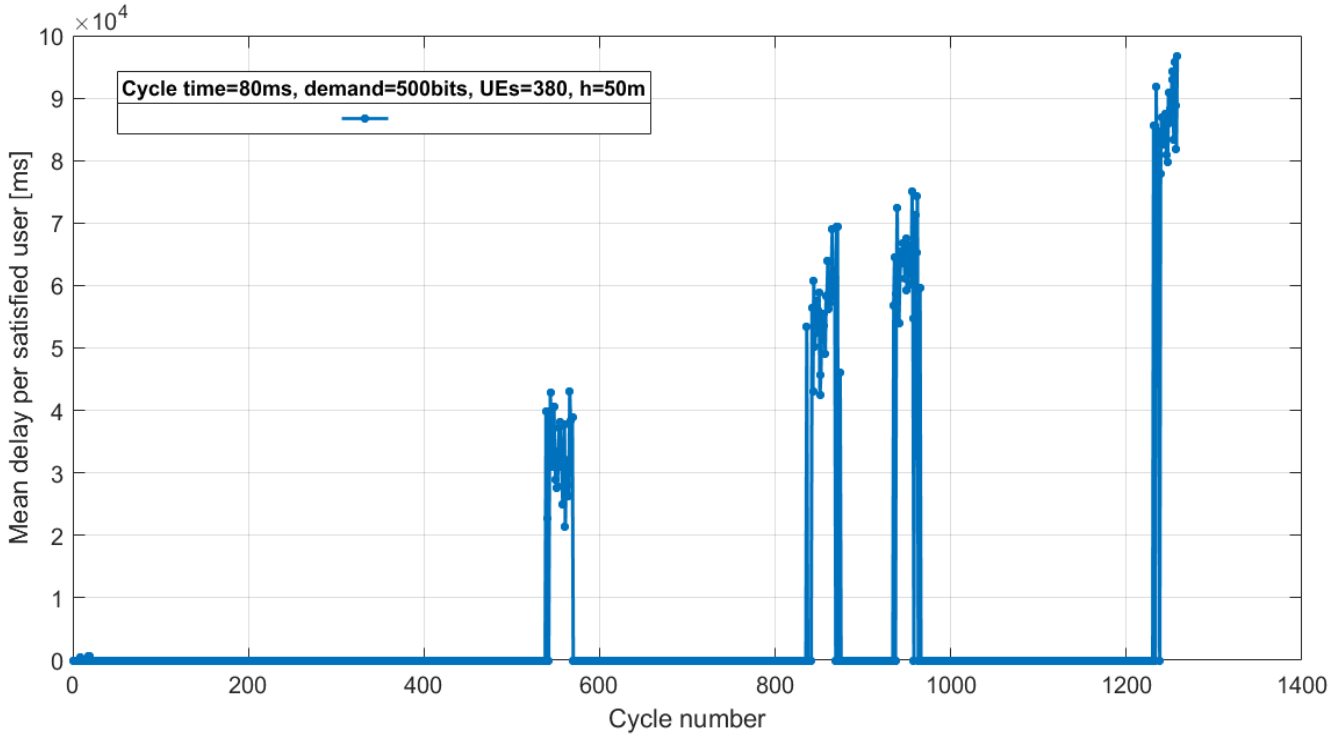

Fig.14: Mean delay of the satisfied users per cycle

The passage over the four clusters is clearly distinguishable, in fact as soon as the IoT devices enter in the range of the drone is visible the utilization of the RUs, through the assignment of subcarriers to users who succeed the RA. Remarking the flight starts at the center of a cluster, we could ask ourselves why during the initial cycles the resources utilized are so few. This comes from the random arrival of the users in the network, that is set in the range [0;13] s, for a scenario in which the flight duration is ~100 s (for a UAV speed of 20 m/s); then nodes stay active for 40s before expiring, i.e. this is the period of time within nodes can be served. For this reason most of resources for the first cluster are exploited when the UAV returns to the initial cluster head to end the journey, being initially silent. On the other hand, if we set the arrival time to 0 s for all the users, letting the devices of the first cluster being served from the beginning, this would result in a lower network throughput because we would increase for all of them the time between the wake-up and the moment in which they are served, i.e. the delay, which stands in the denominator for the formula of the throughput. However, in all the clusters the peaks of the given RUs are much below the available amount, that in the 80 ms case are 96 RUs for cycle. The explanation stands in different factors: first of all the UAV progressively see more users while it passes over one clusters, and furthermore we have to take into account the randomness of the drone received power, that must be over the threshold to initialize the procedure of data upload for each user but the path loss can be excessive preventing it, and the randomness of the RA outcomes.

2.5.1 – Test on the scenario

These considerations done, we wanted to prove the fairness of the Java code eliminating temporarily these randomness. Summing up, I run the code eliminating the randomness on:

- The nodes arrival time, set to 0 s for all;
- The drone received power, setting $P_r > P_{r\_min}$ always true;
- The RA outcome, always succeeded:
- The position of the nodes, placing the ones of a same cluster superimposed to the position of the cluster head, in order to be seen by the UAV everyone at the same time.

Moreover I placed exactly 48 users per cluster, same number of the available subcarriers, to check that in each cluster the RUs are exploited just for 3 cycles, the needed number in case of 80 ms cycle time to satisfy the 500 bit demand, plus 1 extra cycle because tones are not reassigned to other users within the same cycle, so as we can see from the next figure, cycle number 3 and 4 exploit the half of the resources; otherwise, considering higher amount of users would be impossible to check since we should have a flat blue line in correspondence of 96.

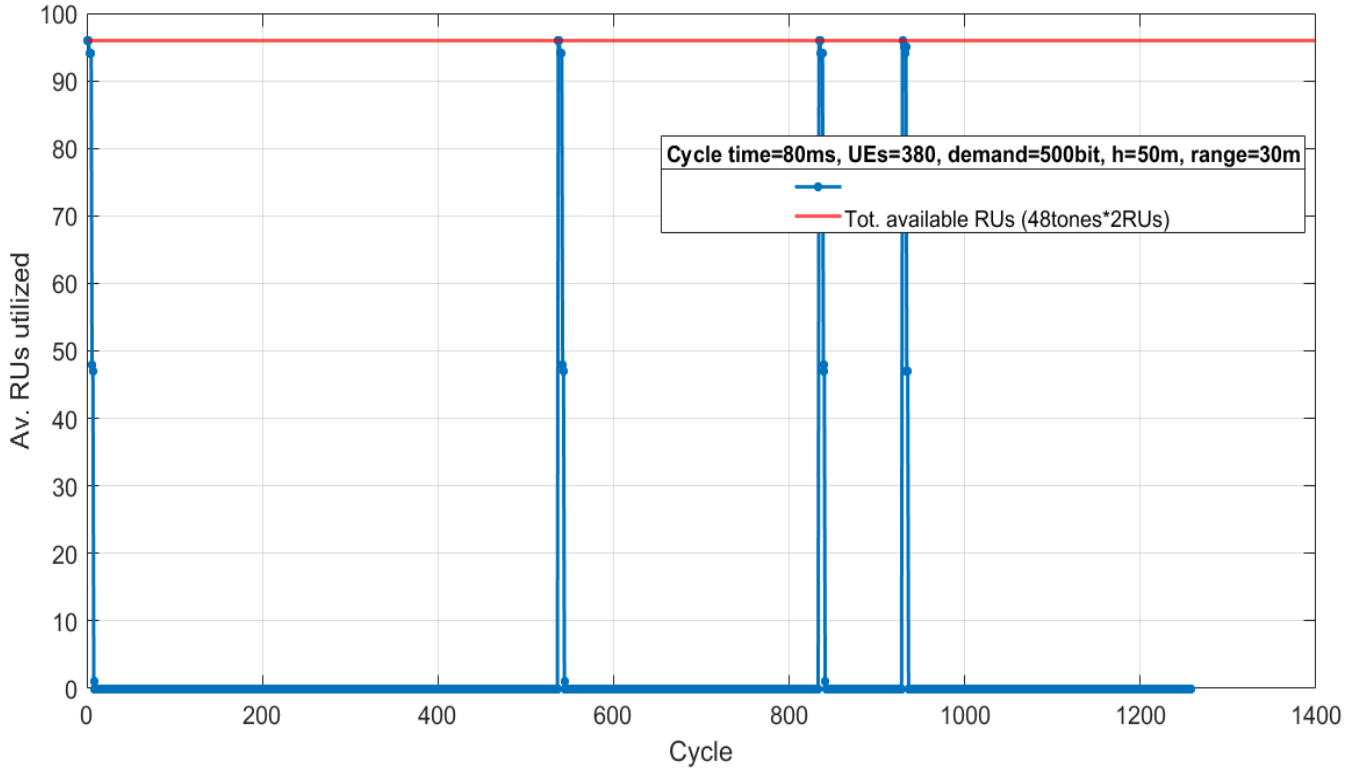

Fig.15 Test on the RUs assignment

With the previous assumptions we can see how that the resources are completely exploited as expected in this case, and this proves the correctness of the code.

Chapter 3 – Random Access implementation

3.1 Access probability

In the second chapter we reported the access probability for users willing to transmit data, according to their 500 bit of demand, that we again report here for the sake of convenience:

$$P_{acc,u} = e^{-\frac{U}{N_{RU}}}$$

The same probability has been used in the Java code through which we run all the simulation and showed the previous graphs and results: this access probability and the complementary collision/failure probability were associated to a binary random variable, that was then given as input of a random number generator according to the well-known Bernoulli distribution. Then, for each user, if the outcome of this generator was '1' the random access was considered succeed by that user, otherwise that device was prevented from connection. Of course that method was fair for the reliability of our results, but now as a further step we want to change it with a procedure more faithful to what physically happens when an IoT device desires to connect to the NB-IoT base station, according to what described in the 3GPP Release 13 standard, and that we summarize below.

As already mentioned, during the uplink phase NPRACH is used for the preamble transmission, and with its mandatory 3.75 kHz subcarrier spacing, given the 180 kHz total bandwidth it is directly derived that 48 subcarriers are available for the data transmission. Considering that the NPRACH transmission itself can be configured with a repetition value from the set {1, 2, 4, 8, 26, 32, 64, 128} according to the coverage class but we consider in this study the case of no repetitions supposing the devices at the best coverage level, we have that each active user will contend on all the 48 subcarriers, resulting in an equal probability for them all to be chosen (1/48); so in the previous formula we have that $N_{RU} = 48$, a-priori known, while U depends on the number of users among the ones in the UAV range that have data to transmit. Recalling that each preamble is constituted by four symbol groups, with the first determined by a pseudo-random hopping and the following ones deterministically selected based on the known rule, two devices who select the same first subcarrier, will have the same correspondence on the other three too.

3.2 – Random access algorithm

In this study we consider that a preamble is successfully transmitted toward the base station if it experience a SINR above a threshold, and among all the users who select the same subcarrier, only one at most can succeed.

In this following section is reported in pseudocode the main steps of what realized in Java to implement this functionality:

As a first step, for each user that is in coverage, the corresponding drone received power is stored as an internal variable of the objects of the user class. Moreover, if that user is neither connected yet nor satisfied, a random number between {0, 47} is associated to it, representing the index of the subcarrier through which the user will try to access and send the first preamble symbol group.

```
for (each Active user u) do
    if (user u is within the range of UAV) then
        if (P_r,drone > P_r,min) then
            Add u to Users in coverage;
            Store P_r,drone associating it to an internal variable of the user u;
            if (u is not connected AND not satisfied) then
                Generate a random number rn in the interval {0, 47};
                Set user u chosen tone = rn;
            end
        end
    end
end
```

Then we check among all the users in coverage the ones neither connected nor satisfied to add them to the sub-group of users that will try to connect.

```
for (each User in coverage u) do
    if (user u is not connected AND not satisfied) then
        Add u to Users trying RA;
    end
end
```

At this point the core of the random access procedure is introduced. For each of the 48 subcarriers we first check if there is already any user attached to it, and in this case we skip to the next tone. After, among the users trying to connect during that cycle and which selected the same tone, we sum each relative UAV received power to the interference, while looking for the highest drone received power; once we find this value, it will be elected as the useful signal and therefore is subtracted from the total interference. In the case of only one user selecting a certain tone, there is clearly no interference and the devices from which the UAV received the signal (that at this point we already know from previous steps being for sure above the $P_{r,min}$ threshold) will be set connected. Otherwise we have first to verify if the corresponding Signal-to-Interference Ratio (SIR) is above the threshold in order to go on: if so, the user is elected among the connected ones, and the random access procedure is terminated.

```
for (each Tone t) do
    for (each User connected u) do
        if (user u chosen tone == index of the tone t) then
            Switch to the next tone;
        end
    end
    P_r,drone = 0;
    Interference = 0;
    for (each User trying RA)
        if (user u chosen tone == index of the tone t) then
            Interference = Interference + (get P_r,drone related to the user u);
```

```
            if (get P_r,drone related to the user u > P_r,drone) then
                P_r,drone = get P_r,drone related to the user u;
            end
        end
    end
    Interference = Interference - P_r,drone;
    SIR = 0;
    if (Interference > 0) then
        SIR[dB] = P_r,drone[dBm] – Interference[dBm];
    else (SIR[dB] > 14.3[dB]) = true ;
    end
    if (SIR[dB] > 14.3[dB]) then
        for (each User trying RA) do
            if (user u chosen tone == index of the tone t) then
                if (get P_r,drone related to user u  == P_r,drone) then
                    Add u to Connected users;
                end
            end
        end
    end
end
```

3.3 – Comparison

We want now to compare the metrics obtained with the previous calculation of the access probability with the new one, after the implementation of the random access procedure. Following the order in which the results are presented in this study, the first comparison to comment is with the network throughput and the served users. The performance are pretty similar, with the exception of 75 m and 100 m as height of the drone that show better performance relatively to the 640 ms duration of the cycle, even better than the 50 m case. Checking the relative percentage of served users, the behavior is stackable with respect to the previous code.

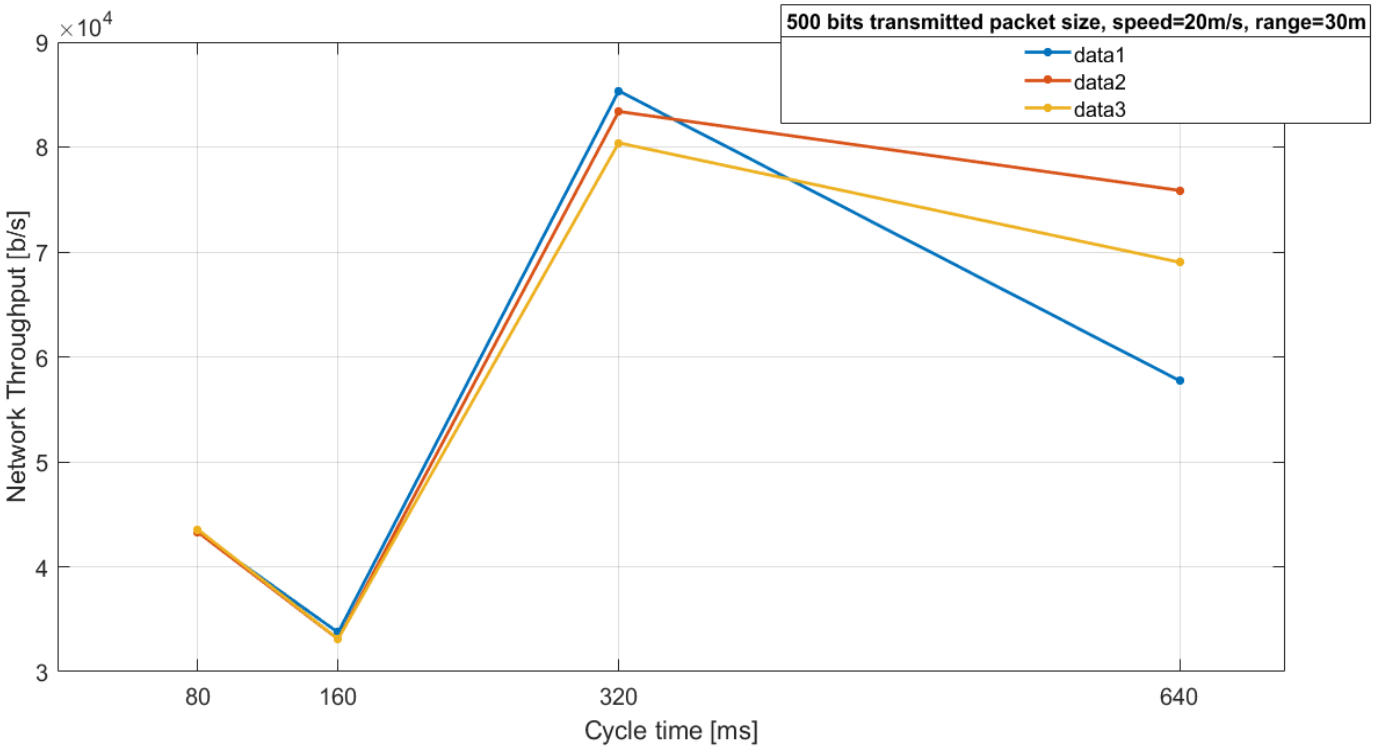

Fig.16: Network throughput with RA implementation

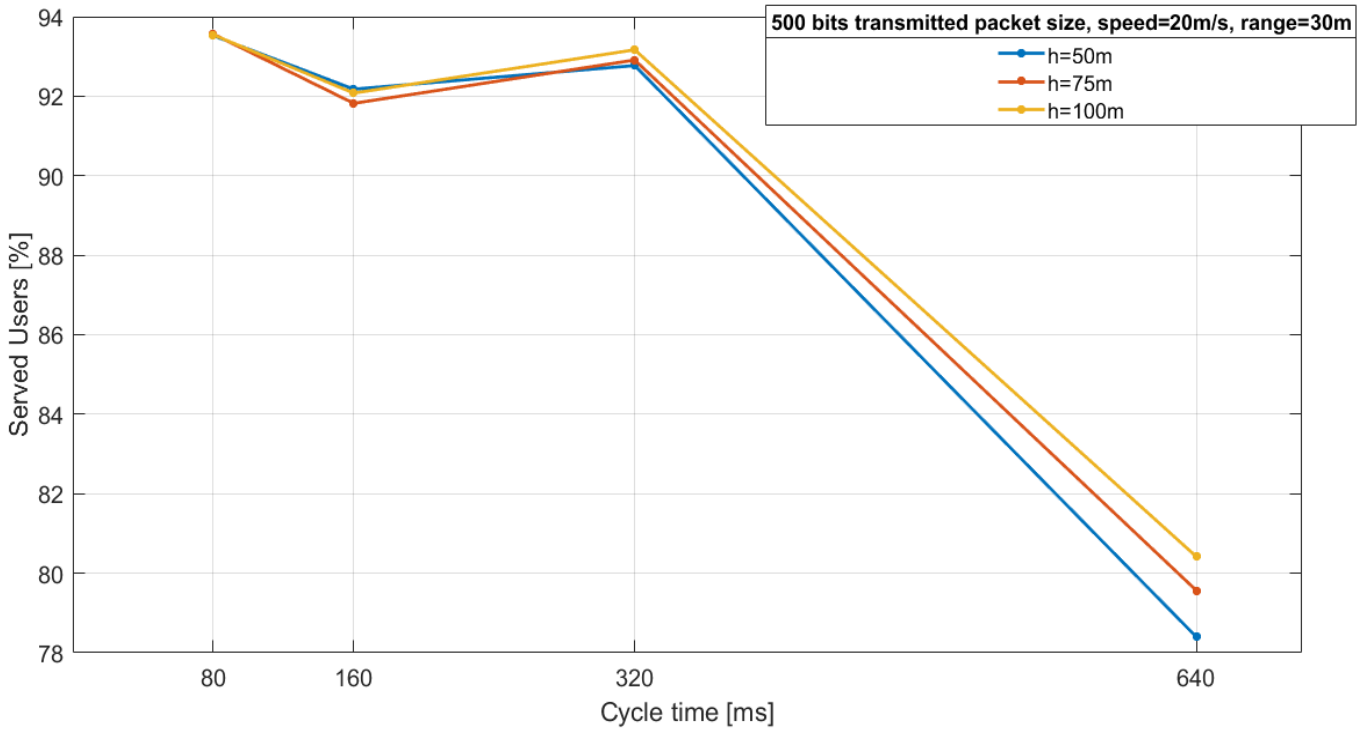

Fig.17: Served NB-IoT devices (%) with RA implementation

What remains unchanged is instead the average values of users in coverage, connecting and connected, as expected.

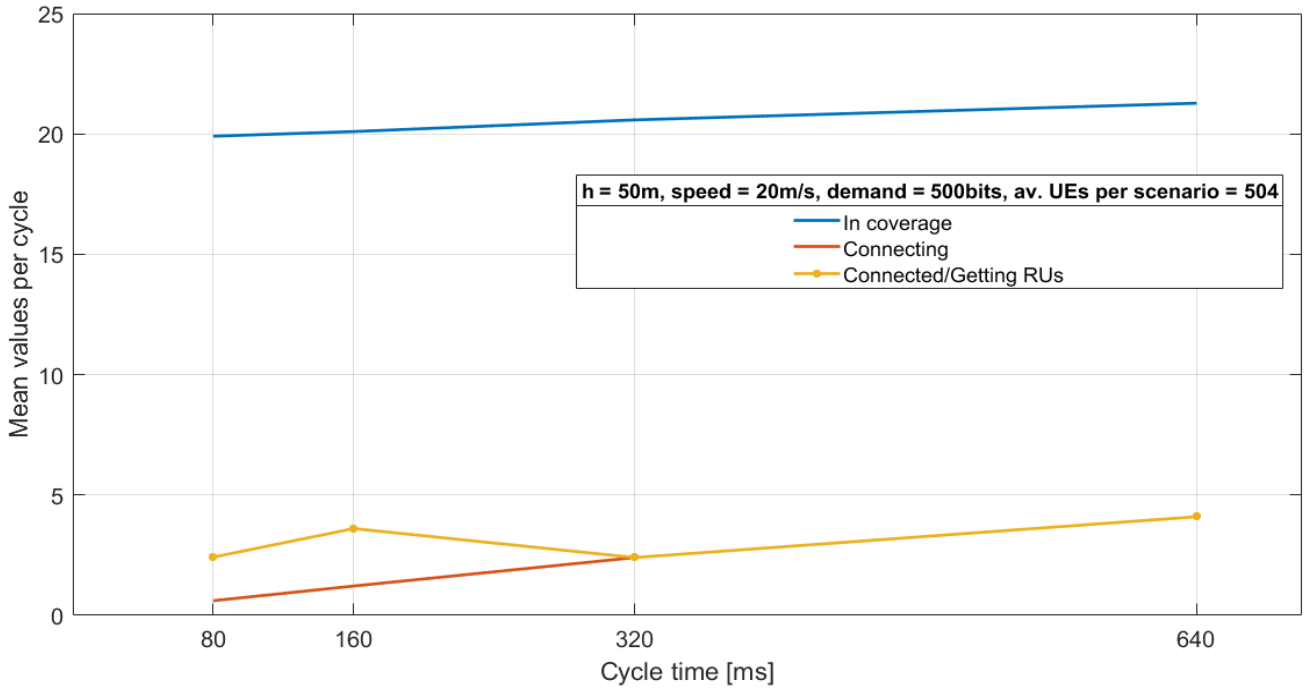

Fig.18: Mean values of users state per cycle with RA implementation

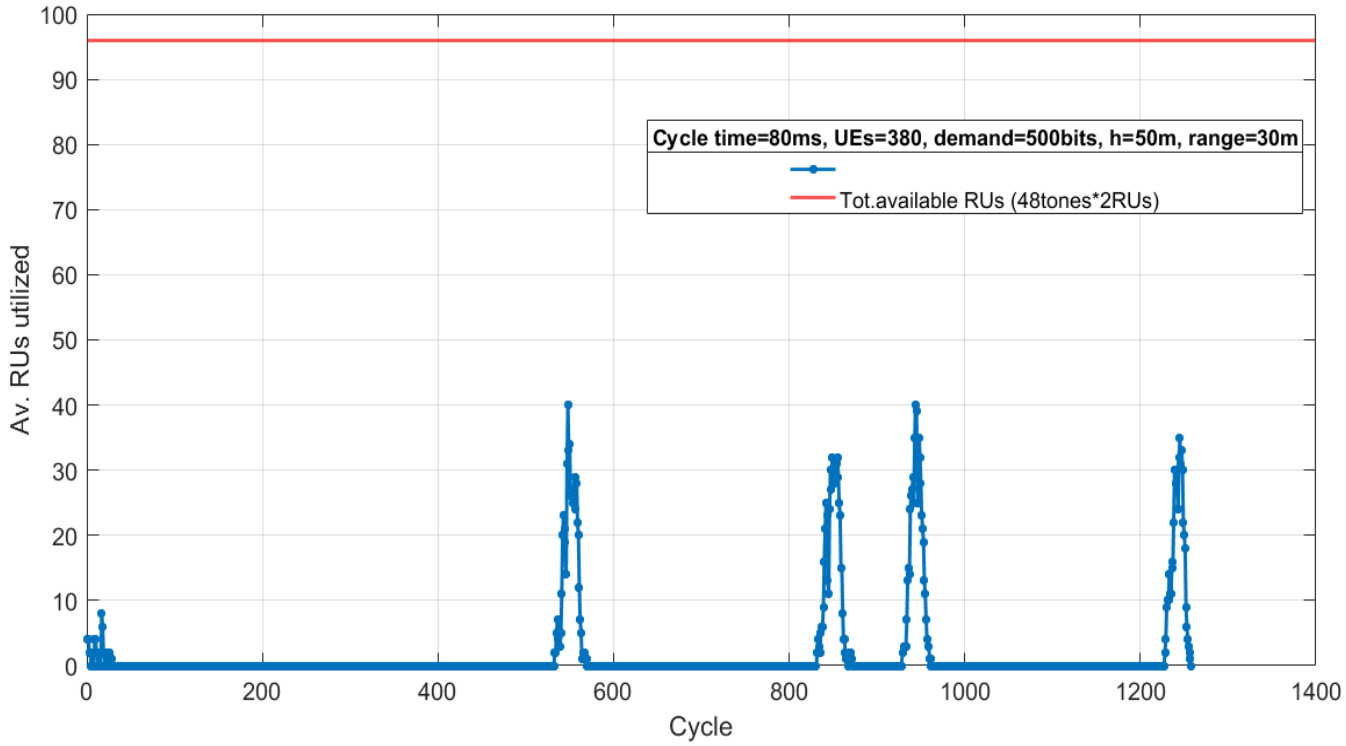

Fig. RUs assignment per cycle with RA implementation

Finally, let's consider again the reference scenario with 380 users and four clusters. We can see that the bells in the figure representing the resource assignment are now narrower and show higher peaks with respect to the previous condition. This is appreciable since the behavior is slightly more similar to the ideal case mentioned in chapter 2.

Conclusions

NB-IoT and its joint use with UABs represents an emerging solution to overcome traditional connectivity problems and serve a large number of devices characterized by small and infrequent data transmission.

As a first step I presented a summary of the 3GPP Release 13 documents of the standard, focusing on aspects that most concern the objective of this thesis. Furthermore, it follows a discussion of the most recent studies as well.

After the initial studies, starting from the work of Professor Roberto Verdone, Professor Chiara Buratti and PhD Silvia Mignardi, I worked on a Java/Matlab environment to simulate a real scenario in which a UAV acting as base station flies over clusters of IoT nodes requiring to send data, whose distribution and position were a-priori known. Analyzing the network throughput while varying height of the drone and duration of the uplink-phase cycle, its trend was explained in detail. The percentage of served users discussed before shows a satisfying trend being above 90% in most of the cases. The study moved then to the behavior of the NB-IoT users during the UAV flight over the clusters. Consequentially, the analysis has improved further in the direction of considering a particular scenario analyzing the “real-time” assignment of the resource and proving its fairness.

Later, the work on NB-IoT analysis was further enriched with the implementation of an improved user access mechanism, which lets users connect to the network in a more practical and closer to the standard way. Comparing the former analyzed metrics with the ones obtained by means of the new procedure, a similarity in the obtained results is observable. This is expected because the two methods describe at different levels the same physical event. Further investigations on the small differences between the two are left to future studies.

References

- Olof Liberg; Mårten Sundberg; Y.-P. Eric Wang; Johan Bergman; Joachim Sachs; Gustav Wikström – Cellular Internet of Things from massive deployments to critical 5g applications – Academic Press; 2020
- S. Mignardi; R. Verdone - On the Performance Improvement of a Cellular Network Supported by an Unmanned Aerial Base Station - IEEE 29th International Conference on Teletraffic Congress – Italy; 2017
- S. Mignardi, K. Mikhaylovy; V. Cacchiani, R. Verdone; C. Buratti - Unmanned Aerial Base Stations for NB-IoT: Trajectory Design and Performance Analysis – IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications – 2020
- V. Savkin; H. Huang - Deployment of Unmanned Aerial Vehicle Base Stations for Optimal Quality of Coverage - IEEE Wireless Communications Letters, Vol. 8, No. 1, february 2019
- L. Feltrin; G. Tsoukaneri; M. Condoluci; C. Buratti; T. Mahmoodi; M. Dohler; R. Verdone - Narrowband IoT: A Survey on Downlink and Uplink Perspectives - IEEE Wireless Communications, Vol. 26, Issue: 1, february 2019
- G. Castellanos; M. Deruyck; L. Martens; W. Joseph - System Assessment of WUSN Using NB-IoT UAV-Aided Networks in Potato Crops - IEEE Access (Volume: 8); 2020
- S. Kavuri; D. Moltchanov; A. Ometov; S. Andreev; Y. Koucheryavy - Performance Analysis of Onshore NB-IoT for Container Tracking During Near-the-Shore Vessel Navigation - IEEE Internet of Things Journal (Volume: 7, Issue: 4, April 2020)
- N. Jiang; Y. Deng; M. Condoluci; W. Guo; A. Nallanathan; M. Dohler - RACH Preamble Repetition in NB-IoT Network - IEEE Communications Letters (Volume: 22, Issue: 6, June 2018)
- https://www.gruppotim.it/tit/it/notiziariotecnico/edizioni-2016/n-3-2016/capitolo-4.html
- 3GPP TR 45.820, v13.0.0 Cellular system support for ultra-low complexity and low throughput Internet of Things; 2016.

Appendix

1. Java code for the access probability

```
        for(User u:users) {
            if(u.isActive(t*cycleTime + simulationTime)) {
                activeUsers.add(u);
                active++;
            }
            else {
                if(u.isConnected())
                    u.setConnected(false);
            }
            if(posDrone.distance(u.getP())< drone.getRange()) {
                userInRange++;
            }
        }
        int countConnectingPerTime = 0;
        int countConnectedPerTime = 0;
        int usersInCoverageTot = 0;
        int usersGettingRu = 0;
        double sumDistances=0;
        double meanDistanceperCycle = 0;
        int usersNotAccessing = 0;
        List<User> usersInCoverage = new ArrayList<User>();
        for(User u:activeUsers) {
            // COUNT and SET the number of active users inside the drone footprint
            if(posDrone.distance(u.getP())< drone.getRange()) {
                    usersInCoverage.add(u);
                    usersInCoverageTot++;

            }else {
                if(u.isConnected())
                    u.setConnected(false);
            }
        }
```

```
for(User u:usersInCoverage) {

    if(!(u.isConnected()) & !(u.isSatisfied())) {


        double P_i =  CoverageClassBased.accessProbability(u.getCoverageClass(), usersInCoverageTot - usersNotAccessing, Rc - tones_occupied);

        DistributedRandomNumberGenerator drng = new DistributedRandomNumberGenerator();
        drng.addNumber(1, P_i);
        drng.addNumber(0, (1-P_i));

        int UEsucceed = drng.getDistributedRandomNumber();
        if(UEsucceed>0) {
            u.setConnected(true); // waits a - time unit - to let the user complete the random access procedure

            countConnectingPerTime++;
            u.setSimultime(simulationTime);
            u.setRAtime((t-1)*cycleTime + simulationTime);
        }
    }
```

2. Java code for RA implementation

```
List<User> usersConnected = new ArrayList<User>();
for(User u:activeUsers) {
    // COUNT and SET the number of active users inside the drone footprint
    if(posDrone.distance(u.getP())< drone.getRange()) {
        if(drone.getChannel().mcl2(Pt, drone.distance3D(u.getP(), 0), u.getCoverageClass(), rand)) {
            u.setPr_drone(drone.getChannel().getPr());
            usersInCoverage.add(u);
            usersInCoverageTot++;
            if(!(u.isConnected()) && !(u.isSatisfied())) {
                Random rn = new Random();
                int choosentone = rn.nextInt(48);
                u.setChosentone(choosentone);
            }
        }
        else {
            if(u.isConnected())
            u.setConnected(false);
        }
    }else {
        if(u.isConnected()) //save energy
            u.setConnected(false);
        }
}
List<User> usersTryingRa = new ArrayList<User>();
for(User u:usersInCoverage) {
    double distance3D = drone.distance3D(u.getP(), 0);
    distances3D.add(distance3D);
    sumDistances = sumDistances + distance3D;
    if(u.isConnected() | u.isSatisfied()) {
        usersNotAccessing++;     // for previous access probability
    }
    if(!(u.isConnected()) && !(u.isSatisfied()))
        usersTryingRa.add(u);
    if(u.isSatisfied()) {
        u.setConnected(false);
    }
    if(u.isConnected())
        usersConnected.add(u);
}
```

```
label: for(Tone tone:drone.getTones()) {
            for(User u:usersConnected) {
                if(u.getChosentone()==drone.getTones().indexOf(tone))
                    continue label;
            }
            double Pr_drone = 0;
            double Interf = 0;
            for(User u:usersTryingRa) {
                if(!(u.isConnected())) {
                    if(u.getChosentone()==drone.getTones().indexOf(tone)) {
                        Interf = (double) (Interf + u.getPr_drone());
                        if(u.getPr_drone()>Pr_drone) {
                            Pr_drone = u.getPr_drone();
                        }
                    }
                }
            }
            Interf = (double) (Interf - Pr_drone);
            Pr_drone = (double) Channel.pow2db(Pr_drone);
            double SIR = 0;
            if (Interf>0) {
                Interf = (double) Channel.pow2db(Interf);
                SIR = (double) (Pr_drone-Interf);
            }
            else SIR=999;   // 0 interference
            if(SIR>14.3) {       //14.3 from standard (UL SINR)
                for(User u:usersTryingRa) {
                    if(u.getChosentone()==drone.getTones().indexOf(tone)) {
                        if(Channel.pow2db(u.getPr_drone())==Pr_drone) {
                            u.setConnected(true);
                            countConnectingPerTime++;
                            usersConnected.add(u);
                        }
                    }
                }
            }
        }
```