

# Contention Management for 5G

DB,CC,MG,OM,YS

February 14, 2017

## Abstract

This article treats about Contention Management for 5G.

## 1 Introduction

- Context and problematic
- Related works
- Article contribution

## 2 Model

### 2.1 Definitions

We consider a directed graph  $G = (V, A)$  modelling a network. Each arc  $(u, v)$  in  $A$  is labeled by an integer  $dl(u, v) \geq 1$  that we call the delay and which represents the number of time slots taken by a signal to go from  $u$  to  $v$  using this arc.

A **route**  $r$  in  $G$  is a sequence of consecutive arcs  $a_0, \dots, a_{k-1}$ , with  $a_i = (u_i, u_{i+1}) \in A$ . We will often refer to the first element of the route as a source and the last as a target.

The **latency** of a vertex  $u_i$  in  $r$ , with  $i \geq 1$ , is defined by

$$\lambda(u_i, r) = \sum_{0 \leq j < i} dl(a_j)$$

We also define  $\lambda(u_0, r) = 0$ . The latency of the route  $r$  is defined by  $\lambda(r) = \lambda(u_k, r)$ .

A **routing function**  $\mathcal{R}$  in  $G$  associates to each pair of vertices  $(u, v)$  a route from  $u$  to  $v$ . Let  $\mathcal{C}$  be an **assignment** in  $G$ , i.e., a set of couples of different vertices of  $G$ . We denote by  $\mathcal{R}_{\mathcal{C}}$  the set of routes  $\mathcal{R}(u, v)$  for any  $(u, v)$  in  $\mathcal{C}$ . We call  $\mathcal{R}_{\mathcal{C}}$  a **routing graph**, it contains all the informations needed in the forthcoming problems (assignment, routes and delays of the arcs).

**TODO:** Dire après la définition des problèmes qu'on pourrait demander de trouver l'assignement et même le routage pour optimiser, mais pas dans cet

article et qu'on travail avec des réseaux déjà constitués TODO: Si on s'en sert, ajouter ici que le routage est cohérent.

## 2.2 Slotted time Model

Consider now a positive integer  $P$  called the **period**. In our problem, we send in the network messages with period  $P$ . The time will thus be cut into slices of  $P$  discrete slots. Assume we send a message at the source of the route  $r$ , at the time slot  $m$  in the first period, then a message will be sent at time slot  $m$  at each new period. We define the first time slot at which the message reaches a vertex  $v$  in this route by  $t(v, r) = m + \lambda(v, r) \bmod P$ .

A message usually cannot be transported in a single time slot. We denote by  $\tau$  the number of consecutive slots necessary to transmit a message. Let us call  $[t(v, r)]_{P, \tau}$  the set of time slots used by  $r$  at a vertex  $v$  in a period  $P$ , that is  $[t(v, r)]_{P, \tau} = \{t(v, r) + i \bmod P \mid 0 \leq i < \tau\}$ . Usually  $P$  and  $\tau$  will be clear from the context and we will denote  $[t(v, r)]_{P, \tau}$  by  $[t(v, r)]$ .

A  $(P, \tau)$ -**periodic affectation** of a routage graph  $\mathcal{R}$  is a sequence  $\mathcal{M} = (m_0, \dots, m_{c-1})$  of  $c$  integers that we call **offsets**, with  $c$  the number of routes in  $\mathcal{R}$ . The number  $m_i$  represents the index of the first slot used in a period by the route  $r_i \in \mathcal{R}$  at its source. A  $P$ -periodic affectation must have no **collision** between two routes in  $\mathcal{R}$ , that is  $\forall (r_i, r_j) \in \mathcal{R}^2, i \neq j$ , we have

$$[t(u, r_i)] \cap [t(u, r_j)] = \emptyset.$$

As an exemple of a  $(2, 1)$ -periodic affectation, let consider a routage graph with routes  $\{r_i\}_{i=1, \dots, c}$ , such that all pairs of routes intersect at a different edge. We set  $\tau = 1$  and the delays are chosen so that if  $r_i$  and  $r_j$  have  $v$  as first common vertex then we have  $\lambda(v, r_i) - \lambda(v, r_j) = 1$ . There is a  $(2, 1)$ -periodic affectation by setting all  $m_i$  to 0.

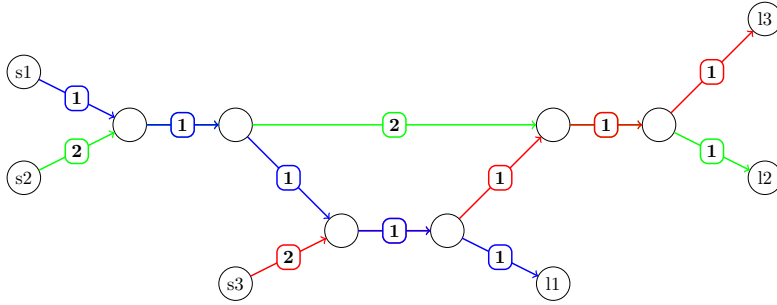


Figure 1: A routage graph with  $(0, \dots, 0)$  as a  $(2, 1)$ -periodic affectation

## 2.3 Problems

We want to ensure that there is an affectation which allows to send periodic messages from sources to target without collisions. The problem we need to

solve is thus the following:

**Periodic Routes Assignment (PRA)**

**Input:** a routage graph  $\mathcal{R}$ , an integer  $\tau$  and an integer  $P$ .

**Question:** does there exist a  $(P, \tau)$ -periodic affectation of  $\mathcal{R}$  ?

We will prove in Sec. 3 that the problem PRA is NP-complete, even in restricted settings. Even approximating the smallest value of  $P$  for which there is a  $(P, \tau)$ -periodic affectation is hard.

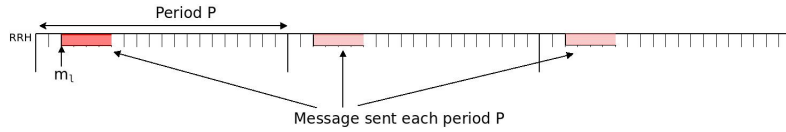
An unusual property of affectation is that given a routage graph, we may have a  $(P, \tau)$ -periodic affectation but no  $(P', \tau)$ -periodic affectation with  $P' > P$ : the existence of an affectation is not monotone with regards to  $P$ .

**Lemma 1.** *For any odd  $P$ , there is a routage graphe such that there is  $(2, 1)$ -periodic affectation but no  $(P, 1)$ -periodic affectation.*

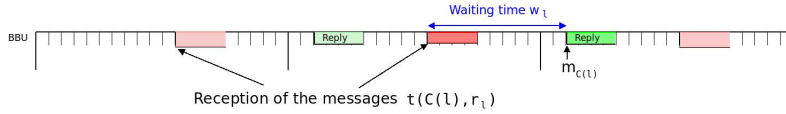
*Proof.* Consider the routage graph  $\mathcal{R}$  given in the previous subsection. We change the delays so that for  $v$ , the first vertex which belongs to  $r_i$  and  $r_j$ , we have  $\lambda(v, r_i) - \lambda(v, r_j) = P$ , where  $P$  is an odd number smaller than  $c$  the number of routes in  $\mathcal{R}$ . If we consider a period of 2, for all  $i \neq j$ ,  $\lambda(v, r_i) - \lambda(v, r_j) = 1$  modulo 2. Therefore  $(0, \dots, 0)$  is a  $(2, 1)$ -periodic affectation of  $\mathcal{R}$ . □

In the context of cloud-RAN applications, we consider here the digraph  $G = (V, A)$  modeling the target network and two disjoint subsets of vertices  $S$  and  $T$  of equal cardinality, where  $S$  is the set of RRHs and  $T$  is the set of BBUs. A **symmetric** assignment  $\mathcal{C}$ , is an involutive function from  $S$  to  $T$ , which maps each element  $s \in S$  to  $\mathcal{C}(s)$  and  $\mathcal{C}(s)$  to  $s$ . It can also be seen as the set of pairs  $(s, \mathcal{C}(s))$  and  $(\mathcal{C}(s), s)$ .

We are given a period  $P$ , a routing function  $\mathcal{R}$ , and we consider a  $(P, \tau)$ -periodic affectation of  $\mathcal{R}_{\mathcal{C}}$  which associates  $m_s$  to the route  $r_s$  which begins by  $s$ . This affectation represents the following process: first a message is sent at  $s$ , through the route  $r_s$ , at time  $m_s$ .



This message is received by  $\mathcal{C}(s)$  at time  $t(\mathcal{C}(s), r_s)$ . It is then sent back to  $s$  in the same period at time  $m_{\mathcal{C}(s)}$  if  $m_{\mathcal{C}(s)} > t(\mathcal{C}(s), r_s)$ , otherwise at time  $m_{\mathcal{C}(s)}$  in the next period. The time between the arrival of the message and the time it is sent back is called the **waiting time** and is defined by  $w_s = m_{\mathcal{C}(s)} - t(\mathcal{C}(s), r_s)$  if  $m_{\mathcal{C}(s)} > t(\mathcal{C}(s), r_s)$  and  $w_s = m_{\mathcal{C}(s)} + P - t(\mathcal{C}(s), r_s)$  otherwise.



When a BBU receives a message, it must compute the answer before sending it back to the RRH. This time can be encoded in the last arc leading to the BBU and thus we need not to consider it explicitly in our model.

Thus, the whole process time for a message sent at vertex  $s$  is equal to

$$PT(s) = \lambda(r_s) + w_s + \lambda(r_{\mathcal{C}(s)}).$$

The **maximum process time** of the  $(P, \tau)$ -periodic affectation  $\mathcal{M}$  is defined by  $MT(\mathcal{M}) = \max_{s \in S} PT(s)$ . The problem we want to solve is the following.

### Periodic Assignment for Low Latency(PALL)

**Input:** A routage graph  $\mathcal{R}_{\mathcal{C}}$  with  $\mathcal{C}$  a symmetric assignment, a period  $P$ , an integer  $\tau$ , an integer  $T_{max}$ .

**Question:** does there exist a  $(P, \tau)$ -periodic affectation  $\mathcal{M}$  of  $\mathcal{R}_{\mathcal{C}}$  such that  $MT(\mathcal{M}) \leq T_{max}$ ?

**TODO:** Si on veut, on peut parler du temps moyen aussi ici, seulement si on fait quelque chose dessus dans la suite

## 3 Solving PRA

### 3.1 NP-Hardness

In this section we assume that the size of a message  $\tau$  is equal to one. We will prove the hardness of PRA and PALL for this parameter, which implies the hardness of the general problems. Consider an instance of the problem PRA, i.e., a routage graph  $\mathcal{R}$ , a message size  $\tau$  and a period  $P$ . The **conflict depth** of a route is the number of other routes which share an edge with it. The conflict depth of a routage graph  $\mathcal{R}$  is the maximum of the conflict depth of the routes in  $\mathcal{R}$ . The **load** of a routage graph is the maximal number of routes sharing the same arc. It is clear that a  $(P, \tau)$ -periodic affectation must satisfy that  $P$  is larger or equal to the load times  $\tau$ .

We give two alternate proofs that PRA is NP-complete. The first one works for conflict depth 2 and is minimal in this regards since we later prove that for conflict depth one, it is easy to solve PRA. The second one reduces the problem to graph coloring and implies inapproximability when one tries to minimize the parameter  $P$ .

**Proposition 1.** *Problem PRA is NP-complete, when the routing is of conflict depth two.*

*Proof.* The problem PRA is in NP since given an offset for each route in an affectation, it is easy to check in linear time in the number of edges whether there are collisions.

Let  $H = (V, E)$  be a graph and let  $d$  be its maximal degree. We consider the problem to determine whether  $H$  is edge-colorable with  $d$  or  $d + 1$  colors. The edge coloring problem is NP-hard [1] and we reduce it to PRA to prove its

NP-hardness. We define from  $H$  an instance of PRA as follows. The graph  $G$  has for vertices  $v_1, v_2$  for each  $v$  in  $V$  and  $s_{u,v}, t_{u,v}$  for each  $(u, v) \in E$ . The set  $A$  of arcs of  $G$  is:

$$\{(v_1, v_2) \mid v \in V\} \cup \{(u_2, v_1) \mid u \neq v \in V^2\} \cup \{(s_{u,v}, u_1), (v_2, t_{u,v}) \mid (u, v) \in E\}.$$

All these arcs are of weight 0. For each edge  $(u, v) \in E$ , there is a route  $s_{u,v}, u_1, u_2, v_1, v_2, t_{u,v}$  in  $\mathcal{R}$ . The affectation  $\mathcal{C}$  is the set of pair of vertices  $(s_{u,v}, t_{u,v})$ .

Observe that the existence of a  $d$ -coloring of  $H$  is equivalent to the existence of a  $(d, 1)$ -periodic affectation of  $\mathcal{R}_{\mathcal{C}}$ . Indeed, a  $d$ -coloring of  $H$  can be seen as a labeling of its edges by the integers in  $\{0, \dots, d-1\}$  and we have a bijection between  $d$ -colorings of  $H$  and offsets of the routes of  $\mathcal{R}_{\mathcal{C}}$ . By construction, the constraint of having no collision between the routes is equivalent to the fact that no two adjacent edges have the same color. Therefore we have reduced edge coloring to PRA which concludes the proof.  $\square$

**TODO: Faire un dessin d'illustration ?**

Remark that we have used zero weight in the proof. If we ask the weights to be strictly positive, which makes sense in our model since they represent the latency of physical links, it is easy to adapt the proof. We just have to set them so that in any route the delay at  $u_1$  is equal to  $d$  and thus equal to 0 modulo  $d$ . We now lift this hardness result to the problem PALL.

**Corollary 1.** *Problem PALL is NP-complete for routing of conflict depth two.*

*Proof.* We consider  $(\mathcal{R}_{\mathcal{C}}, P, \tau)$  an instance of PRA. We assume that no vertex appears both in the first and second position in a pair of  $\mathcal{C}$ . Remark that this condition is satisfied in the previous proof, which makes the problem PRA restricted to this kind of instances NP-complete. Let us define  $T_{max} = 2 \times \max_{r \in \mathcal{R}} \lambda(r) + P$ . We consider  $\mathcal{C}'$  and  $\mathcal{R}'_{\mathcal{C}'}$ , symmetrized versions of  $\mathcal{C}$  and  $\mathcal{R}_{\mathcal{C}}$  where for every route there is a symmetric route with new arcs and the same weights. The instance  $(\mathcal{R}'_{\mathcal{C}'}, P, \tau, T_{max})$  is in PALL if and only if  $(\mathcal{R}_{\mathcal{C}}, P, \tau)$  is in PRA. Indeed the waiting time of each route is by definition less than  $P$  and thus the maximal process time is always less than  $T_{max}$ . Moreover a  $(P, \tau)$ -affectation of  $\mathcal{R}_{\mathcal{C}}$  can be extended into a  $(P, \tau)$ -affectation of  $\mathcal{R}'_{\mathcal{C}'}$  in the following way. For each route  $r_{u,v}$ , the time at which the message arrives is  $t(v, r_{u,v})$ , then we choose as offset for  $r_{v,u}$ ,  $-t(v, r_{u,v}) \bmod P$ . The symmetry ensures that each new route  $r_{v,u}$  in  $\mathcal{R}'_{\mathcal{C}'}$  uses the same times slot as  $r_{u,v}$  and thus avoid collisions.  $\square$

Let MIN-PRA be the problem, given a routage graph and an assignment, to find the minimal period  $P$  such that there is a  $P$ -periodic affectation.

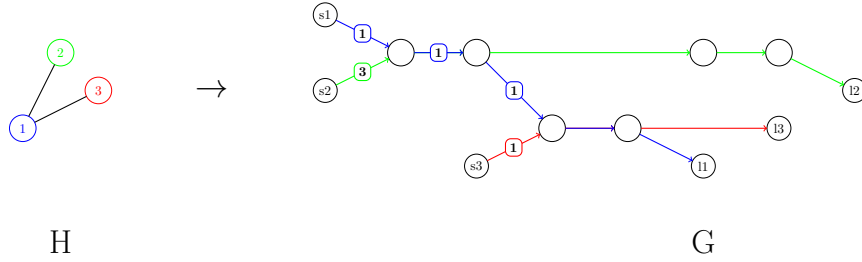
**Theorem 2.** *The problem MIN-PRA cannot be approximated in polynomial time within a factor  $n^{1-o(1)}$ , with  $n$  the number of routes, unless  $P = NP$  even when the load is two.*

*Proof.* We reduce graph coloring to PRA. Let  $H$  be a graph instance of the  $k$ -coloring problem. We define  $\mathcal{R}$  in the following way: for each vertex  $v$  in  $H$ ,

there is a route  $r_v$  in  $\mathcal{R}$ . Two routes  $r_v$  and  $r_u$  share an arc if and only if  $(u, v)$  is an edge in  $H$ ; this arc is the only one shared by these two routes. All arcs are of delay 0.

Observe that the existence of a  $k$ -coloring of  $H$  is equivalent to the existence of a  $(k, 1)$ -periodic affectation in  $G$ , by converting an offset of a route into a color of a vertex and reciprocally. Therefore if we can approximate the minimum value of  $P$  within some factor, we could approximate the minimal number of colors needed to color a graph within the same factor. The proof follows from the hardness of approximability of finding a minimal coloring [2].  $\square$

In particular, this reduction shows that even with small maximal load, the minimal period can be large.



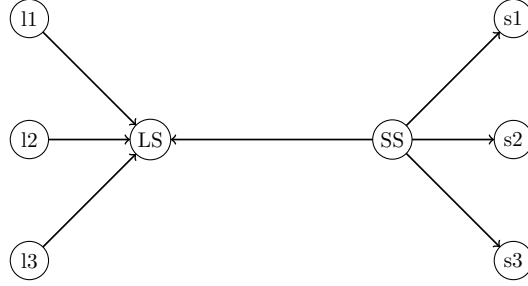
### 3.2 Tractable cases of PRA

In this section, we present a few polynomial cases, in particular when the conflict depth is one.

The problem PALL seems to be hard to solve, even for very simple classes of graphs, as seen in the next section.

## 4 The Star Topology

In this section, we consider graphs with a very simple topology that we call the **star topology**. First, for each arc  $(u, v)$ , there is also an arc  $(v, u)$  with the same weight. Moreover, there is a special arc, the central arc, which is shared by all routes. All routes consist from an arc from its source to the source node, denoted by **SN**, then the central arc to **TN**, the target node, and an arc to its target. In addition to SN and TN, there are two sets  $S$  and  $T$  of cardinality  $n$  and  $\mathcal{C}$  a symmetric assignment from  $S$  to  $T$ . The routes are the directed paths of the form  $s_i, SN, TN, t_i$  and  $t_i, TN, SN, s_i$ .



We assume the weight of the central arc to be 0 since it appears in every route, its value does not matter when considering affectations. Collisions between messages can only appear at node SN on the way forward and at node TN on the way back. **TODO: Définir la forward period et la backward period et dire à quoi correspond une affectation en terme de forward et backward period (placement de messages dans la première période)**

## 4.1 Solving PALL without waiting times

In this subsection, we deal with a simpler version of the problem PALL. We ask for a  $(P, \tau)$ -periodic affectation with all waiting times equal to 0. In that case  $T_{max}$  is not relevant anymore. Since  $w_i = 0$ , choosing  $m_i$ , the offset of the route from  $s_i$  to  $t_i$ , also sets the offset of the route from  $t_i$  to  $s_i$  to  $m_i + t(r_i)$ . Moreover, in this context the weight of the arcs from  $s_i$  to SN have no impact, therefore we assume they are equal to 0.

### 4.1.1 Shortest-longest policy

We present a simple policy, which works when the period is large with regards to the delay of the routes. We send the messages in order from the shortest route to the longest route. **TODO: Preuve en deux lignes que ça ne collisionne pas à condition que la période soit assez grande**

**Algo**

**Period**

### 4.1.2 Greedy Algorithm

We introduce a greedy algorithm to build a  $(P, \tau)$ -periodic affectation, which provably works when the period is large enough with regards to the number of routes and size of messages. In other words, we can always find a solution when the network is far from saturated.

**Proposition 2.** *There is an algorithm, which finds a  $(P, \tau)$ -periodic affectation in time  $O(n^2)$  when  $P \geq 3n\tau$ .*

*Proof.* We consider the time period at the vertex SN and cut it into consecutive macro slots of time of size  $\tau$ . The algorithm works in the following way: select the first route for which the offset as not been chosen. Try every offset which put the message in a free macro slot in the period at SN. It also fixes the position of a message a TN, and we choose the first offset which does not create a collision at TN. Assume we choose the offset of the route  $r_{k+1}$ , we have at least  $3n - k$  free macro-slots, since  $P \geq 3n\tau$ . Each of these  $3n - k$  possible offset values translates into  $3n - k$  positions of messages in the period at TN. All these positions are separated by at least  $\tau$  slots. There are already  $k$  messages of size  $\tau$  in the period at TN. One such message can intersect at most  $2k$  potential positions, therefore amongst the possible  $3n - k$  positions, there are  $3n - k - 2k$  which are without collision. Since  $k < n$ , one of the choice is without collision, which proves that the algorithm terminates and find a  $(P, \tau)$ -periodic affectation. To ensure a complexity linear in  $n$ , it is enough to maintain a list of the free slices at SN and a list of the free intervals at TN.  $\square$

TODO: Avec une structure de données plus maligne on doit pouvoir faire n, non ?

---

**Algorithm 1** Greedy affectation

---

**Input:**  $\mathcal{R}_c$ , period  $P$

**Output:** A P-periodic affectation in  $p \leq P$ , or FAILURE

$P1[P]$  slots of size  $\tau$  in first way period.

$P2[P]$  slots of size  $\tau$  in back way period.

**for all** route  $i$  in  $\mathcal{R}_c$  **do**

**for all** slot  $j$  in  $P1$  **do**

**if**  $P1[j]$  is free AND  $P2[j + \lambda(r_i)]$  is free **then**

$o_i \leftarrow j$

**end if**

**if** No intervals are found for  $i$  **then**

        return FAILURE

**end if**

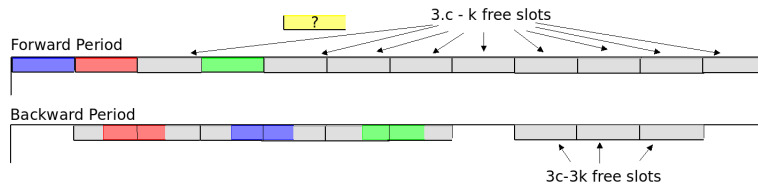
**end for**

**end for**

---

**Period** This algorithm gives us a solution without waiting times in a maximum period  $3\tau.c$ , if we have  $c$  routes.

Suppose that we have a period of  $3\tau.c$  slots, divided in  $\tau$  macro-slots. Put a message in the first slot of size  $\tau$  in the forward period, such that the corresponding area in the backward period is free. This message takes at most 2 slots of time  $\tau$  in the backward period.





When  $k < c$  messages are put in the forward period, and we want to add another message, there is  $3.c - k$  free slots of size  $\tau$  in the forward period. Those  $3.c - k$  gives us  $3.c - k$  possible slots in the backward periods. The  $k$  messages uses at most  $2k$  slots of size  $\tau$  used in the backward period. Since  $k < c$ ,  $2k < 3.c - k$ , thus using the pigeonhole principle, there is at least one free slot in the backward period for the new message.

#### 4.1.3 Exhaustive generation

---

**Algorithm 2** Exhaustive Generation

---

**Input:** A routage graph  $\mathcal{R}_C$ , period  $P$ , packet size  $\tau$

**Output:**  $(P, \tau)$ -periodic affectation of  $\mathcal{R}_C$

Forward-budget  $\leftarrow P - c * \tau$

Backward-budget  $\leftarrow P - c * \tau$

Free-Intervals  $\leftarrow$  list of free intervals, init to  $[0; P[$

**for all** route  $i$  in  $\mathcal{R}_C$  **do**

**for all**  $j$  in Free-Intervals **do**

**if** Message of the route  $i$  does not collides with scheduled routes **then**

$m_i \leftarrow$  the first slot of Free-Intervals[ $j$ ]

            Split the Free-Intervals considering the new packet

            Forward-budget  $\leftarrow$  Forward-budget - *lost size*

            Backward-budget  $\leftarrow$  Backward-budget - *lost size*

            call Exhaustive Generation on remaining routes

**end if**

**end for**

**end for**

---

**Lost size**, is the size lost when an interval is too few to put another packet.  
Ex : if the interval  $[250; 3250[$  receives a packet of size 2500, there is 500 slots lost, because there is no way to uses the slots for another packet.

This algorithm enumerates all the solutions by traversing a tree. The leaves of the tree are the  $(P\tau)$ -periodic affectations, and the nodes are partial solutions. A partial solution is a choice of a starting time for a subset of the routes. For each route, the algorithm tries every possible starting offset until it finds one available (that allows the message to come back without collisions). When a route have been scheduled, it take another route that have not been scheduled yet. This creates a new sub-tree in which the algorithm will try every possible starting offset too, and create a sub-tree on the following route too. If no offset are available for a route, this means that the partial solution is not good, so the algorithm backtracks over the tree that is, it goes back to the father of the current sub-tree. Once a leaf is obtained, the algorithm stops and return the scheduling.

To reduce the number of useless calculation, in each time we add another packet, we consider the lost size in both periods, thus, we know when a period has no many great enough free area to put a packet.

#### 4.1.4 Results

Resultats des simulations : Shortest-longest optimal pour ces parametres.

## 4.2 Allowing waiting times

### 4.2.1 Intro

Importance des waiting times quand la période est donnée (Résultats D'expérience et preuve avec l'exemple)

### 4.2.2 LSG

#### Algorithm

**Analysis** Parler de LSO et expliquer pourquoi LSG mieux avec nos params

### 4.2.3 Results

#### Random

#### Distributions

## 5 Conclusion

Open questions. Can we improve the results if:

- The routes are smaller than the size of a message.
- The routes are smaller than the period.
- The largest difference between two routes is smallest than one of these parameters

On a general graph:

- The routing is coherent
- The graph is symmetric

## References

- [1] Ian Holyer. The np-completeness of edge-coloring. *SIAM Journal on computing*, 10(4):718–720, 1981.

- [2] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2006.