

Contention Management for 5G

DB,CC,MG,OM,YS

March 27, 2017

Abstract

This article treats about Contention Management for 5G.

The evolutions proposed in next generations of mobile network architectures aim at evolving toward centralized radio network architectures (C-RAN, for Cloud Radio Access Network) to reduce consumption costs and power at the base stations [?]. This type of architecture faces the challenge of mastering the latency in the transfer process between the Remote Radio Heads (RRHs) on the field and baseband units (BBUs) in the cloud. Low latency is already critical for the deployment of C-RAN approach. The standard requires meeting time constraints for functions like HARQ (Hybrid Automatic Repeat reQuest) that needs to be processed in less than from 1 to 10ms [?] (depending on targeted services). One specificity in the C-RAN context is not only the latency constraint, but also the periodicity of the data transfer between RRH and BBU (this HARQ constraints must be enforced for each frame emitted every millisecond).

Thus in this article, we work on latency and periodicity constraints in fronthaul. The best current solution is to rely on an almost full optical approach, where each end-points (RRH on one side, BBU on the other side) are connected through direct fiber or full optical switches [?]. This architecture is very expensive and hardly scales in the case of a mobile network composed of about 10,000 base stations. It is then needed to find a solution to offer low latency over commoditized packet based networks. Indeed, dynamical optical bypass and dynamical management of the emission should be considered to guarantee latency constraints, as it is considered in the french ANR project N-GREEN. This project proposes a new type of switching/routing node and a specific network architecture exploiting WDM packets thanks to a new generation of optical add/drop multiplexers (WSADM: WDM slotted add/drop multiplexer). These packets having a fixed duration close to $1\mu\text{s}$ are transported in a transparent way, to better exploit the switching matrix of the node; their headers will be transported over one dedicated wavelength at a lower bit rate, to reduce the physical constraints of the electronic processing and scheduler.

Thus, new scheduling and routing paradigms and new technologies have to be considered to guarantee delay constrained periodic data transfers. One of the most promising approaches relies on the concept of Deterministic Networking (DN) such that one get rid of statistical multiplexing. The traditional queue managements are replaced by time based forwarding. Solutions for Deterministic Networking are under standardization in IEEE 802.1 TSN group [?], as well

at IETF DetNet working group [?]. Several patents on concepts and mechanisms for DetNet have been already published, see for example [?, ?]. To make DN working over a network composed of several nodes, it is required to manage the time at which the packets of deterministic paths are crossing each nodes. The major difficulty of this problem is the periodicity of the process. Indeed, a deterministic sending for the messages between each pair BBU/RRH must not collide with the other messages sent by the others BBU/RRH in the same period, but also in the previous and following periods.

Considering a graph, modeling the network topology, and a set of routes from source nodes (modeling connections to the BBU) and destination nodes (modeling the RRH) in this graph, the purpose is to select, for each destination node a route from one source node to it and a periodic routing scheme allowing to periodically sent a packet to each base station without congestion conflicts between all such packets and to insure a minimum latency. In a slotted time model, the aim is here to minimize the duration of the period, with a constraint of the maximum length of routes to be selected. Even if the selected set of routes is given this optimization problem has been shown to be NP-hard. From an algorithmic point of view, the purpose of this project is first, to study the complexity and the approximability of this problem when the length of the routes is small (which corresponds to realistic cases), and secondly, to propose and implement some heuristics to solve this problem on realistic topologies.

This problem may look like wormhole problem [?], very popular few years ago, but here, we want to minimize the time lost in buffers and not just avoid the deadlock, and the wormhole does not treat about the periodicity. Several graph colorings have been introduced to model similar problems such as the allocation of frequencies [?], bandwidths [?][17] or routes [?][18] in a network or train schedules [?][19]. Unfortunately, they do not take into account the periodicity and the associated problems are also NP-complete. The only model which incorporates some periodicity is the circular coloring [?, ?, ?] but is not expressive enough to capture our problem.

Paper organisation

In the next section, ...

1 Model

1.1 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 **routage graph**, it contains all the informations needed in the forthcoming problems (assignment, routes and delays of the arcs).

TODO: Si on s'en sert, ajouter ici que le routage est cohérent.

1.2 Slotted time Model

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

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

A (P, τ) -**periodic affectation** of a routage graph $\mathcal{R}_{\mathcal{C}}$ is a sequence $\mathcal{M} = (m_{s_0}, \dots, m_{s_{n-1}})$ of n integers that we call **offsets**, with n the number of routes in $\mathcal{R}_{\mathcal{C}}$. The number m_{s_i} represents the index of the first slot used in a period by the route $r_{s_i} \in \mathcal{R}_{\mathcal{C}}$ at its source s_i . A (P, τ) -periodic affectation must have no **collision** between two routes in $\mathcal{R}_{\mathcal{C}}$, that is $\forall (r_{s_i}, r_{s_j}) \in \mathcal{R}_{\mathcal{C}}^2, i \neq j$, we have

$$[t(u, r_{s_i})] \cap [t(u, r_{s_j})] = \emptyset.$$

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

1.3 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:



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

Periodic Routes Assignment (PRA)

Input: a routage graph \mathcal{R}_C , an integer τ and an integer P .

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

We will prove in Sec. 2 that the problem PRA is NP-complete, even in restricted settings. Even approximating the smallest value of P for which there is a (P, τ) -periodic affectation is hard.

An unusual property of affectation is that given a routage graph, we may have a (P, τ) -periodic affectation but no (P', τ) -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 graph such that there is $(2, 1)$ -periodic affectation but no $(P, 1)$ -periodic affectation.*

Proof. Consider the routage graph \mathcal{R}_C 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 n , the number of routes in \mathcal{R}_C . In such a graph, there is no (P, τ) -periodic affectation, because of $P < n$.

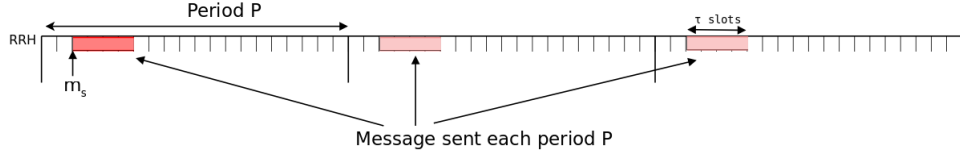
If we consider a period of 2, for all $i \neq j$, $\lambda(v, r_i) - \lambda(v, r_j) \bmod 2 = 1$. Therefore $(0, \dots, 0)$ is a $(2, 1)$ -periodic affectation of \mathcal{R}_C .

□

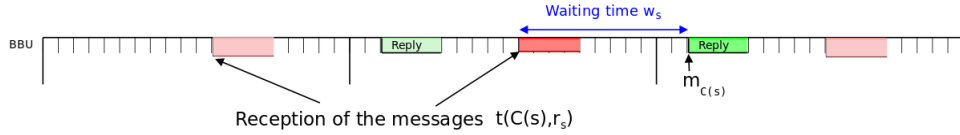
In the context of cloud-RAN applications, we consider here the digraph $G = (V, A)$ modeling the target network and two disjoint subsets of vertices S and T of equal cardinality, where S is the set of RRHs and T is the set of BBUs. A **symmetric** assignment \mathcal{C} , is an involutive function from 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 $(\mathcal{C}(s), s)$.

The routing function \mathcal{R} associates to each couple $(s, \mathcal{C}(s))$ a route called r_s and to each couples $(\mathcal{C}(s), s)$ a route called $r_{\mathcal{C}(s)}$.

We are given a period P , a routing function \mathcal{R} , and we consider a (P, τ) -periodic affectation of \mathcal{R}_C which associates m_s to the route r_s which begins by s , and $m_{C(s)}$ to the routes $r_{C(s)}$ which begins by $C(s)$. 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 $C(s)$ at time $t(C(s), r_s)$. It is then sent back to s in the same period at time $m_{C(s)}$ if $m_{C(s)} > t(C(s), r_s)$, otherwise at time $m_{C(s)}$ 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 $w_s = m_{C(s)} - t(C(s), r_s)$ if $m_{C(s)} > t(C(s), r_s)$ and $w_s = m_{C(s)} + P - t(C(s), r_s)$ 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

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

The message size τ in this process time, if we want to consider the time between the emission of the first slot and the reception of the last slot of the message. Since τ is the same for every routes, it does not change the calculations.

The **maximum process time** of the (P, τ) -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}_C with \mathcal{C} a symmetric assignment, a period P , an integer τ , an integer T_{max} .

Question: does there exist a (P, τ) -periodic affectation \mathcal{M} of \mathcal{R}_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

In this article, we works on given routages, but we could imagine a problem, in which, considering a given graph $G = (V, A)$ and a period P , the question is to find, if it exists, a routage graph \mathcal{R}_C , in which there is (P, τ) -periodic affectation.

2 Solving PRA

2.1 NP-Hardness

In this section we assume that the size of a message τ 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}_C , a message size τ 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}_C is the maximum of the conflict depth of the routes in \mathcal{R}_C .

The **load** of a routage graph is the maximal number of routes sharing the same arc. It is clear that a (P, τ) -periodic affectation must satisfy that P is larger or equal to the load times τ .

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 [?] and we reduce it to PRA to prove its NP-hardness. We define from H an instance of PRA as follows. For each v in V , the graph G has two vertices v_1, v_2 , and for each $(u, v) \in E$, the graph G has two vertices $s_{u,v}, t_{u,v}$. Set an arbitrary orientation of the edge (u, v) , such that the following route is directed from u to v .

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} . All these arcs are of weight 0. The set of arcs of G is the union between all the arcs of the previous routes. 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}_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}_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}_C, P, τ) 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}_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}_C, P, τ) 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, τ) -affectation of \mathcal{R}_C can be extended into a (P, τ) -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

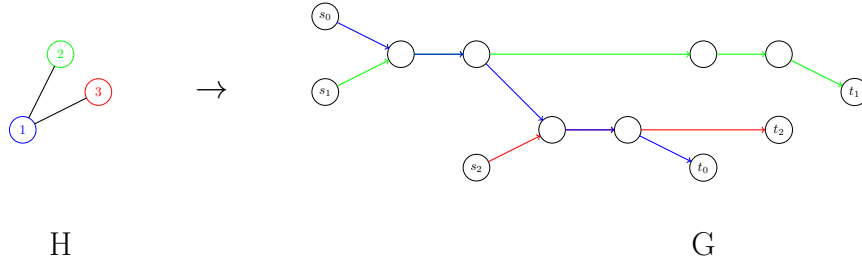
Let **MIN-PRA** be the problem, given a routage graph and an assignment, to find the minimal period P such that there is a P -periodic affectation.

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 [?]. \square

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



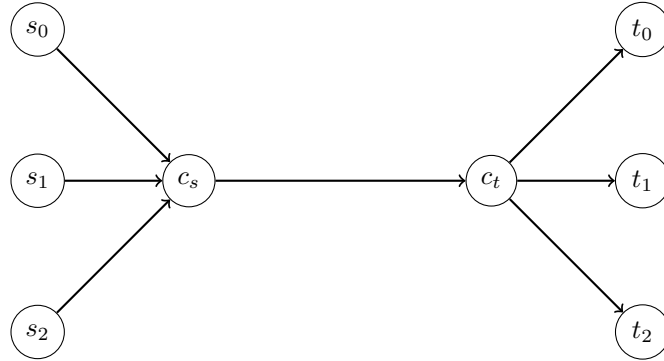
2.2 Tractable cases of PRA

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.

3 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 central source node, denoted by c_s , then the central arc to c_t , the central target node, and an arc to its target. In addition to those central nodes, there are two sets of vertices, $S = s_0, \dots, s_{n-1}$ and $T = t_0, \dots, t_{n-1}$, of cardinality n and \mathcal{C} a symmetric assignment from S to T . The routes are the directed paths of the form s_i, c_s, c_t, t_i and t_i, c_t, c_s, 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 c_s on the way forward and at node c_t on the way back. Thus, we only have to check if there is no collisions in a period at those two points. We consider two periods P at these vertices, the **forward period** at c_s , and the **backward period** at c_t . A message issued by the source s_1 at time m_{s_1} will reach c_s at time $m_{s_1} + t(c_s, r_{s_1})$ in the forward period, and c_t at time $m_{t_1} + t(c_t, r_{t_1})$ in the backward period. A (P, τ) -periodic affectation consist in choosing $\forall i \in n, m_i$, and $m_{\mathcal{C}(s_i)}$ such that, for any couples of routes $(r_{s_j}, r_{s_k}), j, k \in n$, we have $[t(c_s, r_j)] \cap [t(c_s, r_k)] = \emptyset$. and for any couples of routes $(r_{t_j}, r_{t_k}), j, k \in n$, we have $[t(c_t, r_{t_j})] \cap [t(c_t, r_{t_k})] = \emptyset$

3.1 Solving PALL without waiting times

In this subsection, we deal with a simpler version of the problem PALL. We ask for a (P, τ) -periodic affectation with all waiting times equal to 0. In that

case $MT(\mathcal{M})$ is equal to twice the delay of the longer route, thus, 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 $m_i + \lambda(r_i)$. Moreover, in this context the weight of the arcs from s_i to c_s have no impact, therefore we assume they are equal to 0.

For those reasons, in this case, we only have to consider the forward and backward periods. Indeed, setting the offset m_{s_i} in the forward period automatically set the offsets $m_{c(s_i)}$ in the backward period. The goal is now to put messages in the forward period such that there is no collision in both periods.

Here, weight of arcs (s_i, c_s) can be ignored, thus, the offsets m_{s_i} are calculated like if all these weight are equal to 0. To find the real values, it is easy to delay the real sending in sources by the weight of the corresponding arc.

3.1.1 Shortest-longest policy

Algorithm We present a simple policy, which works when the period is large with regards to the delay of the routes. Send the messages in order from the shortest route to the longest route, one after each others in the forward period, without any slots between two messages, that is $m_{s_i} = \tau \cdot i$, if the routes $r_i, i \in 0, \dots, n-1$ are ordered by $\lambda(r_i)$ from the shortest to the longest one.

The message using the shortest route is sent before the others, so it comes back to the central node before the following ones. Then, the second message is sent before the others, and also will be back before the third etc... Because the message i is sent τ slots after the message $i-1$ ($m_{s_i} = m_{s_{i-1}} + \tau$), it comes back after the end of the message $i-1$: $m_{s_{i-1}} + \tau + 2\lambda(r_{s_{i-1}}) \leq m_{s_i} + 2\lambda(r_{s_i})$ because of $\lambda(r_{s_i}) \geq \lambda(r_{s_{i-1}})$.

Period Because of $m_{s_i} = \tau \cdot i$ in the forward period, there is no collisions. To ensure that there is no collisions in the backward period, we must ensure that the last message ends before the end of the period P . We have to find the minimum period P such that the last message have totally came back before P slots after the first message.

The this policy ensure us a period of

$$P = \tau * n + 2(\lambda(r_{s_{n-1}}) - \lambda(r_{s_0}))$$

if routes $\{r_0, \dots, r_{s_{n-1}}\}$ are ordered from the shortest to the longest.

We have to calculate the time taken by the last message to reach c_t . $2\lambda(r_{s_0})$ is the first slot in which the first message is coming back in the backward period, and $2\lambda(r_{s_{n-1}}) + n\tau$ is the time at which the last message has totally passed the switch. Therefore, the period only depends of the difference between the longest and the shortest route: the larger this value is, the larger the period is.

3.1.2 Greedy Algorithm

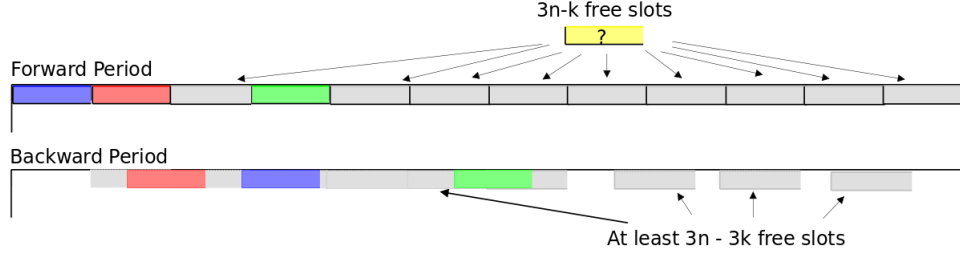
We introduce a greedy algorithm to build a (P, τ) -periodic affectation, which provably works when the period is large enough with regards to the number of

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, τ) -periodic affectation in time $O(n^2)$ when $P \geq 3n\tau$.*

Proof. We consider the forward period and cut it into consecutive macro slots of time of size τ . The algorithm works in the following way: select the first route $k + 1$ for which the offset has not been chosen. Try every of the $3N - k$ offsets which put the message in one free macro slot in the forward period. It also fixes the position of a message in the backward period, and we choose the first offset which does not create a collision in the backward period. 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 backward period. All these positions are separated by at least τ slots. There are already k messages of size τ in the backward period. 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, τ) -periodic affectation. \square

This algorithm works in a complexity n^2 , since we have to try every free macro-slots for every routes.



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 $P1[P/\tau]$ macro slots of size τ in forward period. $P2[P]$ slots backward period.**for all** source s_i in S **do** **for all** macro slot j in $P1$ **do** **if** $P1[j]$ is free AND the corresponding slots in $P2$ are free **then** $m_{s_i} \leftarrow j \cdot \tau$ **end if** **if** No intervals are found for s_i **then**

return FAILURE

end if **end for****end for**

We can try to 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, and continue from there. Once a leaf is obtained, the algorithm stops and return the $(P - \tau)$ -periodic affectation.

Such an algorithm is an **Exhaustive Generation** and has an exponential complexity. Thus, once we search for $(P - \tau)$ -periodic affectations on graphs with a huge number of routes and an huge period P , if there is no solutions available, this algorithm takes too many time. To reduce the number of useless calculation, we can imagine cuts in the tree :

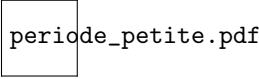
1. In the forward windows, each message is placed after the previous one, that is $m_{s_i} > m_{s_j}$ (here, s_i is the source placed before s_j by the algorithm). Each time a new message is scheduled, there is some possible lost slots between $m_{s_i} + \tau$ and m_{s_j} . We hold a counter of remaining slots and decrease it from the number of lost slots each time a new message is scheduled. Thus, if the number of the remaining slots in the forward period is lower than the number of slots needed to put the remaining messages, we can consider that the actual partial solution is not good.
2. Each time we add a new message in the backward period, we cut a free interval in two new smaller free intervals, plus the scheduled message. Each free intervals has a number of macro slots (ex: 2,6). If the sum of all the integer parts of the macro slots is lower than the number of remaining messages to schedule, the actual partial solution is not good.
3. ...

3.1.3 Results

The following performance evaluation results compares the previous algorithms. Since we try to find a linear time algorithm for the problem PALL, Exhaustive Generation is not a good solution. However, Greedy 3NT and Shortest-Longest are some greedy heuristics and gives us a linear complexity. We tried to determine which algorithm gives a solution with the minimal period. In a first time, we will have a look at the results with the parameters corresponding to Cloud-RAN context.

Short routes To correspond with the C-RAN applications, we made some simulations on graphs in which $\lambda(r_{s_i}) < 2100$ slots, that is, at the most, 5 km between the RRH and its BBU. The messages size τ is to 2500 slots, that is, approximately 1.228 Gbps of data for each route. The size of a slot is given by the time taken to send 64 bytes on the network, in which the links has 10Gbps throughput.

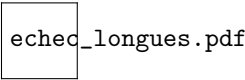
For each algorithm, we made a linear research of the minimal period for which, there is a solution on the given instance. The following figure shows us the average periods on 1000 graphs, for a growing number of routes. The results of the Exhaustive generation can be considered as a lower bound, indeed, for a given period, this algorithm finds the solution, if it exists. As an information, we drew the $\tau.n$ (i.e. the size taken by the messages in the period) and the $2.(\tau.n)$ lines.



periode_petite.pdf

With those parameters, we can observe that the average period of the Shortest-Longest algorithm, which is determined by the difference between the longest and the shortest route, is always equal to the Exhaustive generation, that is, the lower bound. The next results is to theoretically prove that, with those parameters, the Shortest-Longest algorithm gives optimal solutions, like this experimental study suggests it does. About the Greedy 3NT Algorithm, with those parameters, it always gives us bigger periods than the Shortest longest. We can also observe that, for any algorithm, the average minimal periods increases linearly with the number of routes. Thus, the following experiences are made on a fixed number of routes : 8.

Then, we looked at the evolution of the minimal period required with the different algorithm if we significantly increase the size of the route. The following figure shows us the success rate of the algorithms, for 8 routes of size between 0 and 60000 slots, according to the period.



echeq_longues.pdf

As we can see, the Shortest longest has a very low success rate, because of the distribution of the size of the routes is now broader, so is the required

period. Whereas, the Greedy 3NT gives us solutions with much lower period than Shortest Longest. Because of its polynomial complexity, the Exhaustive generation can freeze the simulation when the solution is not easy to find. Thus, we let it stop after a defined number of calculation. The Exhaustive Generation curve gives us the solution when the algorithm has found a solution, and the Upper bound gives us the number of found solutions plus the number of time the algorithm stopped. The real Upper bound is probably between those two curves.

So, those experiences shows us that for real topologies with small routes, the Shortest-Longest algorithm finds some affectations with small Periods, but, the longer the routes are, the better it is to use the Greedy policy to find a solution without waiting times.

Nevertheless, if we want to consider networks, the period is fixed. As we just saw, if the load increases, the needed period increases too, and it is not possible anymore to find a solution without waiting times with a small period. The solution is to allow some messages to wait on target nodes, as described in the next section.

3.2 Allowing waiting times

3.2.1 Intro

We now consider waiting times. The weights on links from sources to c_s are now relevant because the goal is to minimize $MT(\mathcal{M})$, with \mathcal{M} the $(P - \tau)$ -periodic affectation of \mathcal{R}_C . To build such a $(P - \tau)$ -periodic affectation of \mathcal{R}_C , we must choose the offsets m_{s_i} and m_{t_i} , such that there is no $PT(s)$ greater than a given T_{max} . As a reminder, $PT(s) = \lambda(r_s) + w_s + \lambda(r_{t_i})$.

3.2.2 LSG

To find a solution with waiting times, the following heuristic is suggested:

On the forward period, the messages are scheduled so that they are following each others, from the one using the longest route, to the one using the shortest route. Thus, we set the m_{s_i} of the routes such that $t(c_s, r_{s_i}) = \tau \cdot i$, if the routes r_i are ordered from the longest to the shortest.

On c_t , the policy is the following one, after setting the clock to 0:

1. To be eligible, a job needs to be able to come back on the switch at the current clock (if clock = 0, take the first message able to come back).
2. Between the eligible jobs, schedule first the one which has the longest route and gives it the offset time $m_{C(j)} = clock$.
3. Update the clock at the time in which the scheduled task is over:

$$clock = clock + message\ length$$

Algorithm 2 LSG

Input: routage graph \mathcal{R}_C , period P , packet size τ

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

```
clock  $\leftarrow$  0
for all route  $i$  in  $r_{s_i}$  from the longest to the shortest do
   $m_{s_i} \leftarrow$  clock
  clock  $\leftarrow$  clock +  $\tau$ 
end for
clock  $\leftarrow$  0
take  $i$  such that  $r_{\mathcal{C}(f_j)}$  is the first route to come back in sources switch
 $m_{\mathcal{C}(f_j)} \leftarrow$  clock;
clock  $\leftarrow t(c_t, r_{\mathcal{C}(f_j)}) + \tau$ 
while there is a route which has not been scheduled do
  take  $i$  such that  $r_{\mathcal{C}(f_j)}$  is the eligible route.
   $m_{\mathcal{C}(f_j)} \leftarrow$  clock -  $t(c_t, r_{\mathcal{C}(f_j)})$ ;
  clock  $\leftarrow$  clock +  $\tau$ 
end while
```

Algorithm

Analysis One of the closest approach to real networks is the case in which all the weight on the arcs going from c_t to targets are the, this algorithm gives an optimal solution. Indeed, all the same weight can be simplified by 0, thus the scheduling in the forward and backward period is the same, that is, all message following each others, so the period is minimal : $n \cdot \tau$, where n is the number of couples $(s, \mathcal{C}(s))$. Furthermore, all the waiting times are set to 0 i.e. $w_i = 0, \forall i \in n$, thus, $MT(\mathcal{M})$ is equal to twice the longer delay of the routes, that is not alterable.

Another case of this topology is the case in which all the weight on the arcs between c_s and the sources are the same. In this case, the algorithm gives optimal solutions too:

We consider only the delay between c_t and the targets, that are $\lambda(r_s)$. Order the routes such that $\lambda(r_{s_i}) > \lambda(r_{s_j}), i, j \in n, i < j$, i.e. the route 1 is the longest one, and the route n is the shortest. The messages are sent in the forward period from 0 to $n - 1$, that is $m_{s_i} = \tau * n$.

The first message has totally reach the backward period at time $2 \cdot \lambda(r_{s_0}) + \tau$. The process time of the first message is so $PT(s_0) = 2 \cdot \lambda(r_{s_0}) + \tau$. The second message is eligible in the backward period at time $\tau + 2 \cdot \lambda(r_{s_1})$. Thus, the waiting time of the second route is $w_{s_1} = 2 \cdot \lambda(r_{s_0}) + \tau - (\tau + 2 \cdot \lambda(r_{s_1})) = 2 \cdot (\lambda(r_{s_0}) - \lambda(r_{s_1}))$. Thus $PT(s_1) = 2 \cdot \lambda(r_{s_1}) + 2 \cdot (\lambda(r_{s_0}) - \lambda(r_{s_1})) + \tau = 2 \cdot \lambda(r_{s_0}) + \tau = PT(s_0)$. By induction, $PT(s_i) = PT(s_0), \forall i \in n$. Consequently, for this case, the heuristic is optimal, because the longest route has no waiting time all the other routes will take exactly the same time, thus, $MT(M) = PT(s_0) = 2 \cdot \lambda(r_{s_0}) + \tau$, which is not alterable

3.2.3 Results

Random

Distributions

4 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