Deterministic Scheduling of Periodic Messages for Cloud RAN

***Abstract*—A recent trend in mobile networks is to centralize in distant data-centers the processing units which were attached to antennas. The main challenge is to guarantee that the latency of the periodic messages sent from the antennas to their processing units and back fulfills protocol time constraints. We show that traditional statistical multiplexing does not allow such a low latency, due to collisions and buffering at nodes. Hence, we propose in this article to use a deterministic scheme for sending periodic messages without collisions in the network thus without buffering and the resulting additional latency. We give several algorithms to compute such schemes for a common topology where one link is shared by all antennas. We show that there is always a solution when the routes are short or the load is small. When the parameters are unconstrained, and some buffering is allowed in the processing units, we propose an algorithm (PMLS) adapted from a classical scheduling method. The experimental results show that even under full load, most of the time PMLS finds a deterministic sending scheme with no latency.**

I. Introduction

Next generations of mobile network architectures evolve toward centralized radio network architectures called C-RAN for Cloud Radio Access Network, to reduce energy consumption costs [1] and more generally the total cost of ownership. The main challenge for this type of architecture is to reach a latency compatible with transport protocols. The latency is measured between the sending of a message by a Remote Radio Head (RRH) and the receptions of the answer, computed by real-time virtualized network functions of a BaseBand Unit (BBU) in the cloud . For example, LTE standards requires to process functions like HARQ (Hybrid Automatic Repeat reQuest) in 3ms [2]. In 5G, some services need end to end latency as low as 1ms [3]. The specificity of the C-RAN context is not only the latency constraint, but also the periodicity of the data transfer in the *frontaul* network between RRHs and BBUs: frames need to be emitted and received each millisecond [2]. Our aim is to operate a C-RAN on a low-cost shared switched network. The question we address is the following: is it possible to schedule messages such that there is no collisions to avoid latency caused by queuing delays? Eliminating this source of latency leaves us with more time budget for latency coming from the physical length of the routes in the network, and thus allows for wider deployment areas.

Let us expose briefly our model: the network topology is modeled by a directed weighted graph and a set of paths (routes) from source nodes (RRHs) to target nodes (BBUs). Time is discretized and a unit of time or slot corresponds to the transmission of a minimal unit of data over the network. The process must be predictable (i.e. the date of arrival of packets is known beforehand). Following LTE standard [2], in this article we assume that the process is periodic. That is, if we look at the network at times t and $t + P$ where P is the period, all messages are at the same position in the network. As we want to avoid any buffering in internal nodes of the graph. Hence we need to design a periodic process with no collision between messages. To ensure that this property is always true we propose a deterministic process. We have two sets of values that we can set when building a periodic sending process, called a *periodic assignment* in this article: the time at which each packet is sent by a RRH in each period and the waiting time in the BBU before the answer is sent back to the RRH. When building a periodic assignment, we must take into account the periodicity which makes many scheduling methods unusable. Not only a message 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. The latency, that is the time between the emission of a message and the complete return of its answer must be minimized. This means that the only buffering we are allowed – the waiting time before sending back the answer– must be small, in particular when the route is long. Note that the model is technology agnostic, i.e. it is compatible with an optical network with a fixed packet size.

Our main contributions are the following. In Sec. II we propose a model of the network and the periodic sending of messages along its routes We then formalize our two problems of finding a periodic assignment for sending messages without collisions: PRA (sending of the message only) and PALL (sending of the message and of the answer). In Sec. III, we prove that the problem PRA and PALL are NP-hard and cannot be approximated even for very restricted classes of graphs. Therefore in the next two sections, we study a simple but very common topology, that we call the star routed network, where all messages can collide on a single arc. In Sec.IV, we study a variant of PALL called PAZL where the waiting times must all be zero. We provide several algorithms, some of their theoretical properties and give experimental evidences that they find periodic assignments when the network is not too loaded. Finally in Sec.V, we propose algorithms for the general PALL problem and exhibit one which works extremely well in our experiments even in loaded networks. In particular we show that the deterministic communication schemes we de-

Others terminologies exist in the literature. The results of this work are fully compatible with any C-RAN architecture definition.

sign largely outperform the traditional stochastic multiplexing with regard to latency.

Related works

Statistical multiplexing even with a large bandwidth does not comply with the latency requirements of C-RAN. Therefore, the current solution [4], [5] is to use dedicated circuits for the fronthaul. Each end-point (RRH on one side, BBU on the other side) is 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. The deterministic approach we propose has gained some traction recently: Deterministic Networking is under standardization in IEEE 802.1 TSN group [6], as well at IETF DetNet working group [7]. Several patents on concepts and mechanisms for DetNet have been already published, see for example [8], [9].

The algorithmic problem we focus on may look like wormhole problems [10], but we want to minimize the time lost in buffers and not just to avoid deadlocks. Several graph colorings have been introduced to model similar problems such as the allocation of frequencies [11], bandwidths [12] or routes [10] in a network or train schedules [13]. Unfortunately, they do not take into account the periodicity of the scheduling and the associated problems are already NP-complete. The only coloring with periodicity is the circular coloring [14] but it is not expressive enough to capture our problem.

II. Model and Problems

A. Network modeling

The network is modeled as a directed graph $G = (V,A)$. Each arc (u,v) in A is labeled by an integer weight $\omega(u,v)$ which represents the time taken by a message to go from u to v using this arc. A **route** r in G is a path, that is, a sequence of adjacent vertices $u_0,\dots,u_k$, with $(u_i,u_{i+1}) \in A$. The **latency** of a vertex u_i in a path r is defined by $\lambda(u_i,r) = \sum_{0\le j<i} \omega(u_j,u_{j+1})$. We also define $\lambda(u_0,r) = 0$. The length of the route r is defined by $\lambda(r) = \lambda(u_k,r)$. We denote by $\mathcal{R}$ a set of routes, the pair $(G,\mathcal{R})$ is called a **routed network** and represents our telecommunication network. The first vertex of a route models an antenna (RRH) and the last one a data-center (BBU) which processes the messages sent by the antenna.

B. Messages dynamic

In the process we study, a message is sent on each route at each period, denoted by P. Let r be a route, if a message is sent at time m from s the first vertex of r then it will arrive at vertex v in r at time $m+\lambda(v,r)$. Since the process is periodic, if the message from r goes through an arc at time $t \in [0,P-1]$, then it goes through the same arc at time $t+kP$ for all positive integers k. Therefore, every time value can be computed modulo P and we say that the first time slot at which a message sent at time m on r reaches a vertex v in r is $t(v,r) = m + \lambda(v,r) \mod P$.

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 $[t(v,r)]_{P,\tau}$ the set of time slots used by route r at vertex v in a period P, that is $[t(v,r)]_{P,\tau} = \{t(v,r)+k \mod P \mid 0 \le k < \tau\}$. Usually P and τ will be clear from the context and we will denote $[t(v,r)]_{P,\tau}$ by $[t(v,r)]$. Let r_1 and r_2 be two routes, on which messages are sent at time m_1 and m_2 in their first vertex. We say that the two routes have a **collision** if they share an arc of first vertex v and $[t(v,r_1)] \cap [t(v,r_2)] \neq \emptyset$.

A (P,τ)**-periodic assignment** of a routed network $(G,\mathcal{R})$ is a function that associates to each route $r \in \mathcal{R}$ its **offset** m_r that is the time at which a message is emitted at the first vertex of the route r. *No pair of routes must have a collision* in a (P,τ)-periodic assignment. An assignment m represents the following process: first a message is sent at u, through the forward route r, at time m_r.

As an example of a $(2,1)$-periodic assignment, let us consider a routed network where all pairs of routes intersect at a different arc. It is easy to design such a network and an example is given in Fig. 1. We set $\tau = 1$ and the weights 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 assignment by setting all m_i to 0.

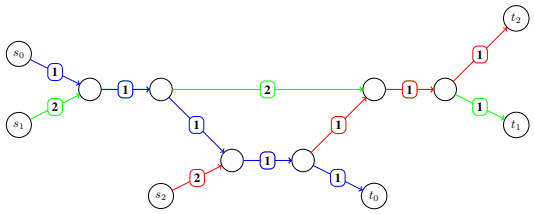


Fig. 1. A routed network with $(0,0,0)$ as a $(2,1)$-periodic assignment

C. Periodic route assignment

We want to find an assignment which allows to send periodic messages from sources to targets without collisions. We introduce the following associated decision problem, useful for hardness proofs.

Periodic Routes Assignment (PRA)

Input: a routed network $(G,\mathcal{R})$, an integer τ and an integer P.

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

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

An unusual property of assignment is that given a routed network, we may have a (P,τ)-periodic assignment but no (P',τ)-periodic assignment with $P' > P$: the existence of an assignment is not monotone with regard to P.

Lemma 1. *For any odd P, there is a routed network such that there is $(2,1)$-periodic assignment but no $(P,1)$-periodic assignment.*

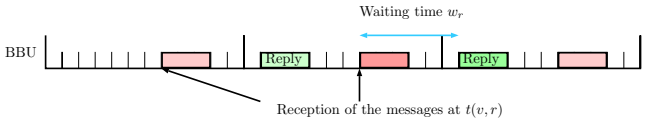
Proof. Consider the routed network $(G,\mathcal{R})$ given in the previous subsection. We change the weights so that for v, the first vertex which belongs to r_i and r_j, we have $\lambda(v,r_i)-\lambda(v,r_j) = P$, where P is an odd number smaller than n, the number of routes in $\mathcal{R}_\mathcal{C}$. In such a graph, there is no (P,τ)-periodic assignment, since the problem reduces to finding a P-coloring in a complete graph with $n > P$ vertices, the colors being the offsets of the routes.
If we consider a period of 2, for all $i \neq j$, $\lambda(v,r_i) - \lambda(v,r_j) \mod 2 = 1$. Therefore $(0,\ldots,0)$ is a $(2,1)$-periodic assignment of $\mathcal{R}$. □

D. Periodic assignment for low latency

In the context of cloud-RAN applications, we need to send message from a RRH u to a BBU v and then we must send the answer from v back to u. We say that a routed network $(G,\mathcal{R})$ is **symmetric** if the set of routes is partitioned into the sets F of forward routes and B of backward routes. Moreover there is a bijection ρ between F and B such that for any forward route $r \in F$ with first vertex u and last vertex v, the backward route $\rho(r) \in B$ has first vertex v and last vertex u. In all practical cases the routes r and $\rho(r)$ will be the same with the orientation of the arcs reversed, which corresponds to bidirectional links in the networks, but we need not to enforce this property.

We now give a new interpretation of a (P,τ)-periodic assignment of a $(G,\mathcal{R})$ symmetric routed network, so that it represents the sending of a message and of its answer. This assignment represents the following process: first a message is sent at u, through the route $r \in F$, at time m_r.

This message is received by v the last vertex of r at time $t(v,r)$. It is then sent back to u on the route $\rho(r)$ in the same period at time $m_{\rho(r)}$ if $m_{\rho(r)} > t(v,r)$, otherwise at time $m_{\rho(r)}$ 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 $w_r = m_{\rho(r)} - t(v,r)$ if $m_{\rho(r)} > t(v,r)$ and $w_r = m_{\rho(r)} + P - t(v,r)$ otherwise.



Thus, the whole process time for a message sent on the route r is equal to $PT(r) = \lambda(r) + w_r + \lambda(r)$. In the process time, we count the time between the first time at which a message is emitted and the first time at which the message comes back. Alternatively we could consider the time between the emission of the first slot and the reception of the last slot of the message, which would add τ to the process time. However, since all messages are of size τ, it will not change the problems we consider in the rest of the article. Finally, we could add to the process time the computation time a BBU needs to deal with one message, but it can be encoded in the weight of the last arc leading to the BBU and thus we need not to consider it explicitly in our model.

The **maximum process time** of the assignment m of $(G,\mathcal{R})$ is defined by $MPT(m) = \max_{r\in\mathcal{R}} PT(r)$. We consider the following decision problem.

Periodic Assignment for Low Latency (PALL)

Input: A symmetric routed network $(G,\mathcal{R})$, the integers P, τ and T_{max}.

Question: does there exist a (P,τ)-periodic assignment m of $(G,\mathcal{R})$ such that $MPT(m) \leq T_{max}$?

As a consequence of the NP-hardness of PRA, we show in the next subsection that this problem is NP-hard. In Sec. V we will study heuristics which try to solve the search version of PALL (computing an assignment), also denoted by PALL for simplicity.

The following table summarize the main notations used in the paper.

$(G,\mathcal{R})$	Routed network
$\omega(u,v)$	Weight of the arc $(u,v) \in A$
$\lambda(u_i,r)$	Latency of the vertex u_i in r
$\lambda(r)$	Length of the route r
P	Period
τ	Size of a message
$[t(v,r)]$	Set of time slots used by route r at vertex v in a period P
$m = (m_0,\ldots,m_{n-1})$	Assignment: an offset for each route
w_r	Waiting time of the route r
$PT(r)$	Process time of the route r
$MPT(m)$	Maximum process time of the assignment

III. Hardness of PRA

In this section we always assume that the size of a message τ is equal to one. We will prove the hardness of PRA and

PALL for $\tau = 1$ which implies the hardness of the general problems. Consider an instance of the problem PRA, i.e., a routed network $(G,\mathcal{R})$ and a period P. The **conflict depth** of a route is the number of arcs of the route which also belong to other routes. The conflict depth of a routed network $(G,\mathcal{R})$ is the maximum of the conflict depth of its routes. The **conflict width** of a routed network is the maximal number of routes sharing the same arc. Remark that a $(P,1)$-periodic assignment must satisfy that P is larger or equal to the load.

We give two alternate proofs that PRA is NP-complete. The first proof works already for conflict depth two. Remark that for conflict depth one, the graph can be seen as a set of disjoint pair of routes, on which PRA and PALL can be solved in linear time. The second proof reduces the problem to graph coloring and implies inapproximability when one tries to find the smallest possible P.
Finally, it is easy to see that PRA is easy on trees and it may be interesting to study its complexity on bounded treewidth networks, since it is a common property of real networks [15].

Theorem 2. *Problem PRA is* NP*-complete on the class of routed networks with conflict depth two.*

Proof. Problem PRA is in NP since given an offset for each route in an assignment, it is easy to check in linear time in the number of arcs whether there are collisions.

Let $H = (V,E)$ be an undirected graph and let d be its maximal degree. We consider the problem to determine whether H is arc-colorable with d or $d+1$ colors. The arc coloring problem is NP-hard [16] and we reduce it to PRA to prove its NP-hardness. To do that, we define from H a routed network $(G,\mathcal{R})$ as follows. The vertices of G are v_1, v_2 for each v in V and $s_{u,v}, t_{u,v}$ for each $(u,v) \in E$. For each arc $(u,v) \in E$, there is a route $s_{u,v}$,u_1,u_2,v_1,v_2,$t_{u,v}$ in $\mathcal{R}$.

To define $\mathcal{R}$ an arbitrary orientation of each arc (u,v) is chosen. Then for each arc (u,v) 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 previously defined routes.

Observe that the existence of a d-coloring of H is equivalent to the existence of a $(d,1)$-periodic assignment of $(G,\mathcal{R})$. Indeed, a d-coloring of H can be seen as a labeling of its arcs by the integers in $\{0,\ldots,d-1\}$ and we have a bijection between d-colorings of H and offsets of the routes of $\mathcal{R}$. By construction, the constraint of having no collision between the routes is equivalent to the fact that no two adjacent arcs have the same color. Therefore we have reduced arc coloring to PRA by a polynomial time transformation which concludes the proof. □

Remark that we have used zero weights in the proof. If we ask the weights to be strictly positive, which makes sense in our model since they represent the delay of physical links, it is easy to adapt the proof. We just have to set them so that in any route the weight 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 on the class of routed networks of conflict depth two.*

Proof. We consider $((G,\mathcal{R}),P,\tau)$ an instance of PRA. We assume that no vertex is the first of some route and the last of another one. Remark that this condition is satisfied in the previous proof, which makes the problem PRA restricted to this kind of instance NP-complete. Let us define $T_{max} = 2\times\max_{r\in\mathcal{R}} \lambda(r)+P$. We define $(G',\mathcal{R}')$ a symmetric routed network from $(G,\mathcal{R})$ where for every route we add a symmetric route with new arcs of opposite orientation and the same weights. We prove that the instance $((G',\mathcal{R}'),P,\tau,T_{max})$ is in PALL if and only if $((G,\mathcal{R}),P,\tau)$ is in PRA. If $((G',\mathcal{R}'),P,\tau,T_{max})$ is in PALL, then a (P,τ)-assignment of $(G',\mathcal{R}')$ restricted to R is a (P,τ)-assignment of $((G,\mathcal{R})$ since they cannot be any collision between routes of $\mathcal{R}$.

Assume now that $((G,\mathcal{R}),P,\tau)$ is in PRA. First remark that the waiting time of each route is by definition less than P and thus the maximal process time is always less than T_{max}, therefore the constraint on T_{max} is satisfied. Moreover a (P,τ)-assignment of $(G,\mathcal{R})$ can be extended into a (P,τ)-assignment of $(G',\mathcal{R}')$ in the following way. For each route $r \in \mathcal{R}$ of last vertex v, the time at which the message arrives is $t(v,r)$, then we choose as offset for $\rho(r) \in \mathcal{R}'$ $-t(v,r) \mod P$. The symmetry ensures that each new route $\rho(r)$ in $\mathcal{R}'$ uses exactly the same times slot as r one each of its node and thus avoid collisions. □

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

Theorem 3. *If* $\mathsf{P} \neq \mathsf{NP}$*, problem MIN-PRA on the classes of routed networks of conflict width two cannot be approximated in polynomial time within a factor* $n^{1-o(1)}$ *where* n *is the number of routes.*

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 arc in H; this arc is the only one shared by these two routes. All arcs are of weight 0. Note that it is easy to build a graph with such routes as in Fig. 2.

Observe that the existence of a k-coloring of H is equivalent to the existence of a $(k,1)$-periodic assignment 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 [17]. □

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

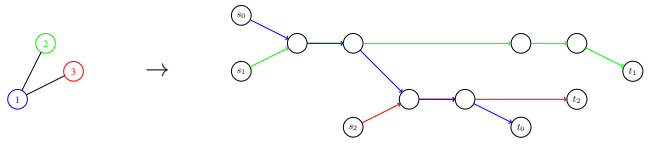
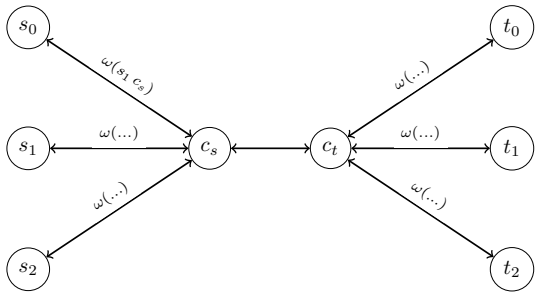


Fig. 2. Reduction from k-coloring to MIN-PRA

IV. The Star Routed Network: No Waiting Time

A. The star routed network

Let us define a family of simple symmetric routed networks. The graph G has two sets of vertices, $S = \{s_0,...,s_{n-1}\}$ and $T = \{t_0,...,t_{n-1}\}$ of cardinality n and two special nodes: the central source node c_s and the central target node c_t. There is an arc between c_s and c_t and for all i, there is an arc between s_i and c_s and between t_i and c_t. All the symmetric arcs are also in the graph with the same weights. The forward routes are the directed paths $r_i = (s_i,c_s,c_t,t_i)$ and $\rho(r_i) = (t_i,c_t,c_s,s_i)$ which defines a symmetric routed network. The symmetric routed networks $(G, \{r_i,\rho(r_i)\}_{i<n})$ is called a **star routed network**.



Since the central arc appears in every route, its value does not matter when considering the process time of an assignment. Moreover an assignment without collisions is also an assignment without collisions of the same star routed network with the weight of the central arc set to 0. Hence, we assume from now on that the weight of the central arc is 0.

Collisions between messages can only appear on the arc (c_s,c_t) between forward routes or on the arc (c_t,c_s) between backward routes. The flow of messages in a star routed network is completely described by their repartition in two time windows of size P, the **forward period** which contains all $[t(c_s,r)]$ with r a forward route and the **backward period** which contains all $[t(c_t,r)]$ with r a backward route.

B. PALL with no waiting time: PAZL

In this subsection, we deal with a simpler version of the problem PALL. We ask for a (P,τ)-periodic assignment **with all waiting times equal to** 0 and we call this restricted problem **Periodic Assignment for Zero Latency** or PAZL. We consider PAZL since it is simpler to get theoretical results and good algorithms than for PALL. Moreover, as we show in our experimentations of Sec.IV-F, this problem can often be solved positively (albeit less often than the general problem). Finally, a solution to PAZL is simpler to implement in real telecommunication networks, since we do not need to implement any buffering at all.

When the waiting times are zero, $MPT(m)$ is equal to twice the length of the longest route, thus T_{max} is not relevant anymore. For a route r, choosing the offset m_r also sets the offset of the route $\rho(r)$ to $m_{\rho(r)} = m_r + \lambda(r) \mod P$. Consider an assignment $(m_0,\dots,m_{n-1})$ of a star routed network and let $m'_i = m_i - t(c_s,r_i) \mod P$. Then $\{m'_0,\dots,m'_{n-1}\}$ is an assignment of the same star routed network where the weights of the arcs from s_i to c_s are 0 for all i. Therefore we assume from now on that the *weight of the arcs from s_i to c_s are all equal to* 0.

One may consider a variant of PAZL where the central link is unidirectional, that is collisions can happen between messages going from c_s to c_t and messages going from c_t to c_s. This problem can be shown to be NP-complete by a reduction from the subset sum problem as it is done for a similar problem of scheduling pair of tasks [18]. We conjecture that the problem PAZL is also NP-complete, but we have yet not been able to prove it. On the other hand we show positive results: when the period is large or when the routes are short there is always a solution to PAZL and it can be found in polynomial time. We then give an exponential time exhaustive search algorithm which always finds a solution if there is one.

C. Shortest-longest policy

We first present a simple policy, which works when the period is large with regard to the lengths of the routes. The messages are sent in order from the shortest route to the longest route, without any gap between two messages in the forward period. In other words, we assume that the route r_i are sorted by increasing $\lambda(r_i)$ and we set m_{r_i} the offset of r_i to $i\tau$. We call this algorithm **Shortest-Longest**.

By definition, there are no collision in the forward period and if the period is long enough, it is easy to see that in the backward period the order of the messages are the same as in the forward period and that no collision can occur.

Proposition 1. *Let $(G,\mathcal{R})$ be a star routed network, and let $n\tau + 2(\lambda(r_{n-1}) - \lambda(r_0)) \leq P$. There is a (P,τ)-periodic assignment of $(G,\mathcal{R})$ with waiting times 0 given by Shortest-Longest in time $O(n\log(n))$.*

Proof. Since $m_{r_i} = i\tau$, $[t(c_s,r_i)] = \{i\tau,\dots,(i+1)\tau-1\}$ and there are no collision on the forward period.

We may assume that $\lambda(r_0) = 0$, since removing $\lambda(r_0)$ from every arc (c_t,t_i) does not change the order on the length of the routes nor the collisions between messages. Since $\lambda(r_0) = 0$, by hypothesis we have $n\tau + 2\lambda(r_n) \leq P$ which implies that

$[t(c_t,r_i)] = \{2\lambda(r_i) + i\tau, \ldots, 2\lambda(r_i) + (i+1)\tau - 1\}$. Since $\lambda(r_i) \leq \lambda(r_{i+1})$ by construction, we have $2\lambda(r_i) + i\tau - 1 < 2\lambda(r_{i+1}) + (i+1)\tau$ which proves that there are no collision on the backward period. The complexity of the algorithm is dominated by the sorting of the routes in $O(n\log(n))$. □

If the period is slightly smaller that the bound of Proposition 1, a collision will occur on the first route in the backward period. Hence, this policy is not useful even as a heuristic for longer routes as confirmed by the experimental results of Subsection IV-F.

D. Greedy algorithm

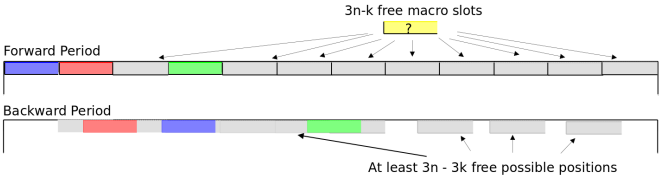
We define the **load** of a star routed network as $\frac{n\tau}{P}$, it is the proportion of time slots used by messages on the central arc in a period. Therefore if the load is larger than 1 there cannot be an assignment. We propose a greedy algorithm to build a (P,τ)-periodic assignment, which always finds an assignment when the load is less than $1/3$. Therefore in the rest of the article we will be only concerned with load larger than $1/3$.

Proposition 2. *There is a (P,τ)-periodic assignment of a star routed network with waiting times 0 if the load is less than $1/3$ and it can be found in time $O(n^2)$.*

Proof. We consider the forward period and cut it into consecutive intervals of size τ that we call macro-slots. The algorithm works by choosing an offset for each route in the following way: try all offsets which put the message in a yet not used macro-slot in the forward period. Since the choice of an offset also fixes the position of the message in the backward period, chose the first one which does not create a collision. We now prove that this algorithm always finds a (P,τ)-periodic assignment without waiting time when $P \geq 3n\tau$ that is the load is less than $1/3$.

Assume we are choosing the offset of the route r_{k+1}, we have at least $P - k \geq 3n - k$ free macro-slots in the forward period, 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 2 potential positions since they are disjoint intervals. Therefore amongst the possible $3n - k$ positions, there are at least $3n - k - 2k$ which are without collision. Since $k < n$, $3n - k - 2k \geq 1$, which proves that the algorithm terminates and find a (P,τ)-periodic assignment.

This algorithm works with a complexity $O(n^2)$, since for the k^{th} route we have to try at most $2k$ offsets before finding a correct one. We can test the $2k$ offsets of the backward period in time $O(k)$ by maintaining an ordered list of the intervals used by already set routes. □



This algorithm, contrarily to the previous one, may work well, even when the condition on the load is not true. In fact, experimental data in Subsection IV-F suggests that the algorithm finds a solution when the load is less than $1/2$. Note that we have experimented with other greedy algorithms which do not use macro-slots, they work even better in practice but their theoretical upper bound is worse.

E. An FPT algorithm

In this section we show how every assignment without waiting time can be put into a canonical form. We use that to provide an algorithm which finds an assignment when it exists, in fixed parameter tractable time (FPT) with parameter n the number of routes (for more on parametrized complexity see [19]). This is justified since n is small in practice and the other parameters such as P, τ or the weights are large.

Let $(G,\mathcal{R})$ be a star routed network and m a (P,τ)-periodic assignment. A set of routes S is **coherent** if for all $r \in \mathcal{R}$, $r \in S$ if and only if $\rho(r) \in S$. We say that a coherent set $S \subseteq \mathcal{R}$ is **compact** for the assignment m if there is a route $r_0 \in S$ such that the following holds: for all coherent subsets $S' \subset S$ with $r_0 \notin S'$, if we remove 1 from all offsets of routes in S' then there is a collision with a route of $S \setminus S'$. We say that m is compact if $\mathcal{R}$ is compact for m.

Proposition 3. *Let $(G,\mathcal{R})$ be a star routed network. If there is a (P,τ)-periodic assignment of $(G,\mathcal{R})$, then there is a compact (P,τ)-periodic assignment of $(G,\mathcal{R})$.*

Proof. Consider m a (P,τ)-periodic assignment of $(G,\mathcal{R})$. Let r_0 be an arbitrary route of $\mathcal{R}$, and let $COMP = \{r_0\}$. Now we apply the following algorithm to m and $COMP$ while $COMP$ is not equal to $\mathcal{R}$. While there is no collision, remove 1 (modulo P) from all offsets of routes in $\mathcal{R} \setminus COMP$. Then choose a route r in $\mathcal{R} \setminus COMP$ which would have a collision with a route r' of $COMP$ if one is subtracted from its offset. If r' is a forward route, let $COMP = COMP \cup \{r, \rho(r)\}$ otherwise $COMP = COMP \cup \{r, \rho^{-1}(r)\}$.

We prove by induction that $COMP$ is compact for m at every step of the algorithm. At the beginning $|COMP| = 1$ and the property is trivially satisfied. Then we assume that $COMP$ is compact and that we add to it $\{r, \rho(r)\}$ at some step of the algorithm. W.l.o.g we assume that it is the offset of r which cannot be decremented without collision. Consider $S \subseteq COMP$, if S contains an element different from r and $\rho(r)$ by induction hypothesis we cannot decrement the offsets of S without collision. If $S = \{r, \rho(r)\}$ by construction, we cannot decrement the offset of r.

Finally, there is no collisions between routes at the beginning and since we modify m only if it creates no collision, the assignment we obtain at the end has no collisions between routes. □

We now present an algorithm to find a (P,τ)-periodic assignment by trying all compact assignments.

Theorem 4. PALL $\in$ FPT *when parametrized by the number of routes.*

Proof. Let $(G,\mathcal{R})$ be a star routed network and let m be a (P,τ)-periodic assignment of $(G,\mathcal{R})$. First remark that for a given assignment and a route r_0 with offset m, by removing m to all offsets, we can always assume that its offset is zero. Therefore we need only to consider all *compact assignments* with an *offset* 0 for the route r_0. We now evaluate the number of compact assignments and prove that it only depends on n the number of routes which proves the theorem. We give a way to build a compact assignment m by determining its offsets one after the other. We fix an arbitrary total order on $\mathcal{R}$. First a route r_0 is chosen arbitrarily and its offset set to 0. Then at each step, if the offsets of $S \subseteq \mathcal{R}$ have been chosen, we select the smallest route r in S for the order. Then we select a route in $r' \in \mathcal{R} \setminus S$ and set its offset such that if we remove 1 then r' collides with r. Note that if r is a forward route (resp. a backward route) then r' is also a forward route (resp. a backward route). We can also decide to definitly skip r. At a given step of the algorithm, if $|S| = 2i$, we have $n - i$ choices of routes to select. The value of the offset of the selected route is entirely determined by the values of the offsets of routes in S. Therefore there are at most $n!$ different compact assignments with offset r_0 fixed to 0.

The algorithm to solve PALL builds every possible compact assignment as described here, and tests at each step whether there is a collision which can be done in time linear in the size of $(G,\mathcal{R})$. Therefore PALL $\in$ FPT. □

We call the algorithm described in Theorem 4 **Exhaustive Search of Compact Assignments**. To make it more efficient in practice, we make cuts in the search tree used to explore all compact assignments. Consider a set of k forward routes whose offsets has been fixed at some point in the search tree. We consider the times at which the messages of these routes cross the central arc. It divides the period into $[(a_0,b_0),\dots,(a_{k-1},b_{k-1})]$ such that the central arc is free only during the intervals (a_i,b_i). Therefore at most $\sum_{i=0}^{k-1} \lfloor (b_i - a_i)/\tau \rfloor$ forward routes can still use the central arc. If this value is less than $n - k$, it is not possible to create a compact assignment by extending the one on S and we backtrack in the search tree. The same cut is used for the backward routes.

F. Experimental evaluation

The following experimental results compare the three presented algorithms. Notice that both Greedy algorithm and Shortest-Longest are polynomial time algorithms but are not always able to find a solution, depending on the load or the size of the routes. On the other hand, the exhaustive search finds an optimal solution if it exists, but works in exponential time. We compare the performance of the algorithms in two different regimes: routes are either short with regard to τ, or unrestricted. From our C-RAN context we choose the following parameters: the number of routes is at most $n = 20$, τ is equal to 2,500. It corresponds to messages of approximately 1Mb for links of bandwidth 10Gbps. The code in C will be available on the webpage of the authors under a copyleft license.

In our experiments we try to understand how our algorithms work with regards to the load. To change the load, we fix the parameters τ and n and modify the period P, which allows for a smoother control of the load and does not change the run time of our algorithms.

a) Short routes: First we consider routes which are shorter than τ: a message cannot be contained completely in a single arc which is very common in our applications. We generate star routed networks in which the weights of the arcs (c_t,t_i) are drawn uniformly between 0 and 700 which corresponds to links of less than 5km between the BBU and the RRH.

Our aim is to understand how well the algorithms are working under high load. To do that we evaluate the highest load under which a (P,τ)-periodic assignment can be found by each algorithm when we change the number of routes. In our experiment, we generate 1,000 random instances of PAZL for 1 to 14 routes. We represent in Fig. 3 the average maximal possible load, for which each algorithm finds a solution. Remind that the exhaustive search always finds the highest load for a given instance.

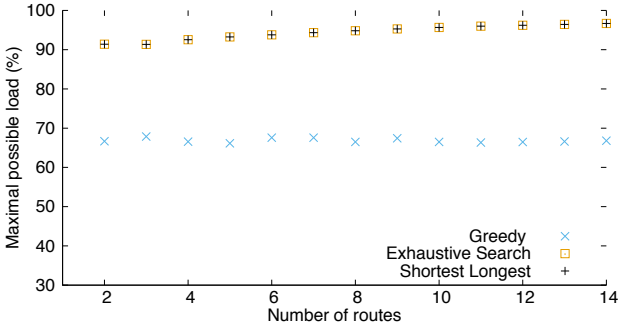


Fig. 3. Minimum period averaged over 1,000 random instances

First, we remark that the exhaustive search finds a solution even when the load is high, especially when there are more routes. It justifies the idea to look for an assignment without

waiting time, in this short routes regime. Second, remark that the Shortest-Longest algorithm is as good as the exhaustive search. While it was expected to be good with short routes, it turns out to be optimal for all the the random star routed networks we have tried. Therefore, we should use it in practical applications with short routes, instead of the exhaustive search which is much more computationally expensive. Finally, note that the greedy algorithm works on average when the load is less than $2/3$ which is twice better than the theoretical bound. This algorithm seems to depends on the load only and not on the number of routes.

b) Long routes: We now want to understand the performance of these algorithms when the size of the routes is unbounded. In this experiment we fix the number of routes to 8 and the weights of the arcs (c_t,t_i) are drawn following an uniform distribution between 0 and $20{,}000$ slots (in the same range as the period). We represent in Fig. 4 the percentage of success of each algorithm, for load from 100% down to 40%.

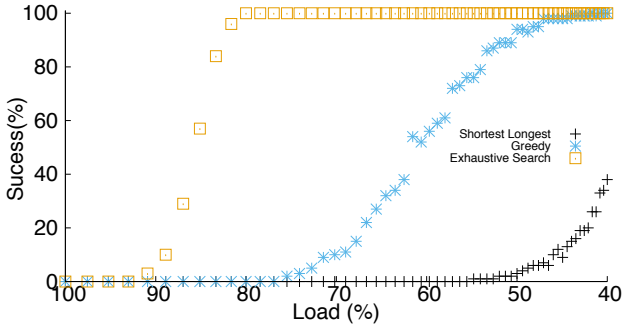


Fig. 4. Success rate for 8 routes over $1{,}000$ random instances

In this regime, the performances of Shortest-Longest are abysmal since it depends on the difference between the longest and the smallest route which is large here. On the other hand, the greedy algorithm has a performance not so far from the case of short routes, which is expected since it does not directly depend on the size of the route. In fact, if we do the previous experiment for short routes but with long routes, we find that, on average, the greedy algorithm finds a solution when the load is less than 59%.

When the load is larger than 50%, the exhaustive search finds more solutions than the greedy algorithms which justifies its use. However, for load larger than 80% there are many instances for which there are no solutions to PAZL. It means that with long routes and high load, looking for an assignment without waiting time is far too restrictive. That is why we present algorithms for the general PALL problem in our next section. We will test them on 8 long routes and a load between 100% and 80% for which, as shown here, there are often no assignment without waiting times.

The computation time of the exhaustive search is in $O(n!)$ as shown int Th. 4, hence we need to study its scalability when n grows. In the following experiment, the weights of the arcs (c_t,t_i) are drawn following an uniform distribution between 0 and $20{,}000$ slots. We chose 95% of load. The following table shows the time before the exhaustive search ends, for 8 to 16 routes, averaged on 100 random star routed networks. This shows that for less than 15 routes, which corresponds to all current topologies, the algorithm is efficient enough, but we should improve it further to work on more routes.

n	8	10	12	14	16
Time (s)	6.10^{-5}	8.10^{-4}	2.10^{-2}	0.4	11

V. Solving PALL on Star Routed Networks

In this section, we consider the more general PALL problem on star routed networks. It means that the messages can wait in the target vertices (BBUs) in order to find assignments which is not possible for PAZL. We allow the process time of the routes to be greater than twice the weights of the routes, but it should be less than T_{max}.

A. Special cases

Often in real networks, the length of the routes are not arbitrary and we may exploit that to solve PALL easily. For instance if all the weights on the arcs (c_t,t_i) are the same, we can replace them all by 0 and substract this weight to T_{max}. It corresponds to a situation where all the BBUs are in the same datacenter and have the same processing power. The assignment in that case is trivial, just send all messages so that they follow each other without gaps in the central arc. Since the arcs (c_t,t_i) are of weight 0, all messages will go through (c_t,c_s) on their way back in the same order and thus do not collide.

Another possible assumption would be that all weights of the arcs (s_i,c_s) are the same.

Theorem 5. *Let $(G,\mathcal{R})$ be a star routed network with n routes and let $P \geq n\tau$. If there is a constant c such that $\forall i < n$, $\omega(s_i,c_s) = c$, then there is a (P,τ)-periodic assignment with process time $2\max_{r\in\mathcal{R}} \lambda(r)$ and it can be built in time $O(n)$.*

Proof. We can simplify the weights on the arcs (s_i,c_s) by setting them all to 0. We find the route with maximal length, say that it is $\lambda(r_0)$. The idea is to set the waiting times of all routes so they behave exactly as the message in r_0.

The offset of the forward route r_i is set to $i\tau$, which ensures that there are no collision on the arc (c_s,c_t) as soon as $P \geq n\tau$ which is the minimal possible period. We set the waiting time for the route i to $w_i = 2(\lambda(r_0) - \lambda(r_i))$. We can compute the time at which the message of the route r_i arrives at the vertex c_t on its way back: $t(c_t,\rho(r_i)) = w_i + i\tau + 2\lambda(r_i)$ by replacing w_i by its value we obtain $t(c_t,\rho(r_i)) = i\tau + 2\lambda(r_0)$ As a conclusion there are no collision on the arc (c_t,c_s) as soon as the period is larger than $n\tau$ (there are no gaps between the messages). The process time of the route r_i defined by $PT(r_i) = w_i + 2\lambda(r_i)$. We obtain $PT(r_i) = 2\lambda(r_0)$ and

thus the maximum process time of the assignment is also equal to $2\lambda(r_0)$. Finally the complexity is $O(n)$ since we have to find the maximum of the length of the n routes and the computation of each w_i is done by a constant number of arithmetic operations. □

Assume that for any pair of indices (i,j) the difference between the weights of the arcs (s_i,c_s) and (s_j,c_s) is bounded by d and the length of the longest route is l. Using the same idea as in Th. 5, there is an assignment with maximum process time bounded by $2l + d$

B. A two stage approach

We can decompose any algorithm solving the PALL problem in two parts: first set the offsets of the forward routes and then knowing this information set the offset of the backward routes or equivalently the waiting times. The offsets of the forward routes will be chosen so that all messages have no collision on the central arc and such that there are no free slots inbeetween the end of a message on the central arc and the beginning of the next one. It is done to minimize the period needed to send the messages of the forward route. We tried the five following orders.

- Longest-Shortest on Routes (LSR): Decreasing order on the length of the routes.
- Shortest-Longest on Routes (SLR): Increasing order on the length of the routes.
- Longest-Shortest on last Arc (LSA): Decreasing order on the length of the arcs (c_t,t_i).
- Shortest-Longest on last Arc (SLA): Increasing order on the length of the arcs (c_t,t_i). This sending order yields a (P,τ) periodic assignment in which all the $w_i = 0$, if the period is large enough (see proposition 1).
- Random: A random order of the routes.

In the rest of the section we will study different methods to compute the offsets of the backward routes, once the offsets of the forward routes are fixed by one of the previous orders.

C. Greedy scheduling of backward routes

We can decompose any algorithm solving the PALL problem in two stages: first set the offsets of the forward routes and then, knowing this information set the offset of the backward routes or equivalently the waiting times. The offsets of the forward routes are chosen so that they have no collisions and such that there are no free slots in between the end of a message on the central arc and the beginning of the next one. It is done to minimize the period needed to send the messages of the forward routes. For this section and the next, we assume that the offsets of the forward routes are fixed by some order.

Consider a forward route r_i and the corresponding backward route $\rho(r_i)$. We define the **deadline** of $\rho(r_i)$ as $m_{r_i} + T_{max} - \omega(s_i,c_s)$, that is the latest time at which the message can go out of c_t such that $PT(r_i) \leq T_{max}$. We say that a backward route $\rho(r_i)$ is **eligible** at time t if $m_i + \lambda(r_i) + \omega(c_t,t_i) \leq t$, that is the message of the route $\rho(r_i)$ arrives at c_t before time t when $w_i = 0$.

The first algorithm we propose is a greedy algorithm which sets the offset $m_{\rho(r_i)}$ of the backward routes. It prioritizes the routes with the earliest deadline to best satisfy the constraint on the process time. Set $t = 0$ and repeat the following: find $s \geq t$ the first time for which there is an eligible route with its offset not fixed. Then amongst all eligible routes at time s choose the one with the smallest deadline, fix its offset to $s - \omega(c_t,t_i)$ and set $t = s + \tau$.

This algorithm does not take into account the periodicity. Say that $t_0 = t(c_t,r)$ such that r is the first backward route selected by the algorithm. Then if all backward routes r are such that $t(c_t,r)$ is smaller than $t_0 + P - \tau$, by construction, there are no collisions on the central arc. However, if a route r has a larger $t(c_t,r)$, since we should consider everything modulo P, it may collide with another backward route. Therefore we must adapt the greedy algorithm of the previous paragraph by finding $s \geq t$ the first time for which there is an eligible route with its offset not fixed and *such that there is no collision if a message go through the central arc at time s.*

Algorithm 1 Greedy deadline (**GD**)

Input: A routed network $(G,\mathcal{R})$, a period P, packet size τ, T_{max}, the offsets m_i
Output: (P,τ)-periodic assignment of $(G,\mathcal{R})$, or failure

```
H ← empty set //set of eligible routes
free_ intervals ← [0,P] //list of intervals of free slots
for all route r_i do
    deadline[r_i] ← m_i + T_max − ω(s_i,c_s)
    eligible_time[r_i] ← m_i + λ(r_i) + ω(c_t,t_i)
end for
while There is some non-assigned routes do
    if H is empty then
        r_i ← min_non_assigned(eligible_time)
        insert ( H,r_i).
    end if
    r ← extract_min(H)
    t ← next_free_interval(free_intervals, t) //if there is no more free intervals of size τ, the algorithm fails
    w_i ← t - eligible_time[r_i]
    update(t,free_ intervals)
    t ← t + τ
    for all routes r_i with eligible_time[r_i] ≤ t do
        insert ( H,r_i).
    end for
end while
```

The function min_non_assigned(eligible_time) returns the route with lowest time in eligible_time, which is not assigned yet. The function update(t,free_intervals) removes an interval of size τ beginning at t, which correspond to the message, from free_intervals.

The greedy algorithm can be made to work in time

$O(n\log(n))$. To do that the set of eligible routes must be maintained in a binary heap to be able to find the one with smallest deadline in time $O(\log(n))$. To deal with the possible collisions, one can maintain a list of the intervals of time at which a message can be sent on the arc (c_t,c_s). Each time the offset of a route is fixed an interval is split into at most two intervals in constant time. Since the algorithm goes over the elements of the list at most twice when doing an insertion or looking for the next free interval, the time needed to maintain it is $O(n)$.

D. Earliest deadline scheduling

The problem to solve in the second stage of our approach is very similar to the following scheduling problem: we are given a set of jobs and each job has a *release time* and a *deadline*. The problem is to schedule all jobs on a single processor, that is choosing the time at which they are computed, so that no two jobs are scheduled at the same time. A job is always scheduled after its release time and it must be finished before its deadline. Let us call n the number of jobs, the problem can be solved in time $O(n^2\log(n))$ [20] when all jobs have the same running time and it gives a solution with the earliest possible deadline. On the other hand if the running times are different the problem is NP-complete [21]. The polynomial time algorithm which solves this scheduling problem is similar to the Greedy algorithm presented in the previous section. However, when it fails because a job finishes after its deadline, it changes the schedule of the last messages to find a possible schedule for the problematic job. The change in the scheduling is so that the algorithm cannot fail on the same job a second time except if there is no solution, which proves that the algorithm is in polynomial time.

We reduce our problem of setting the offsets of the backwards routes once the order of the forward routes is fixed to this scheduling problem. The backward routes are the jobs, the size of a message is the running time of a job, the deadline of a route is the deadline of the corresponding job and the smallest time at which it is eligible is the release time. Let us call **Minimal Latency Scheduling (MLS)** this algorithm. Remark that we do not deal with the periodicity. When MLS finds an assignment m its always satisfies $MPT(m) < T_{max}$. On the other hand, it is a (P,τ) periodic assignment only if $m_{max} - m_{min} \leq P - \tau$ where m_{max} is the largest offset and m_{min} the smallest one. The algorithm we use minimize this quantity, so it may work in practice (as shown in Section V-E).

We now present a variant of the previous algorithm that we call **Periodic Minimal Latency Scheduling (PMLS)** which takes into account the periodicity. We fix arbitrarily a backward route so that its message is the first to go through the central arc at time t. Then we modify the deadline of each backward route to be the minimum of their previous deadline and $t + P$. We execute this algorithm for every possible first backward route and we return the solution that minimizes $MPT(m)$. Since we run the previous algorithm at most n times, the complexity is in $O(n^3\log(n))$. Remark that it finds an assignment more often than MLS, because if MLS finds a solution with first route r and such that $m_{max}-m_{min} \leq P-\tau$, then this solution will be found by PMLS when it selects r as the first route. Moreover when PMLS finds a solution, it is always a (P,τ) periodic assignment since we guarantee that all messages are scheduled in a period of time of size P and it satisfies $MPT(m) < T_{max}$ by construction of the deadlines. Remark that PMLS may fail while there is an assignment. Indeed, in the algorithm we restrict the assignments we look for by not allowing to choose an offset for a route after $t + P - \tau$.

E. Experimental evaluation

We fix the number of routes to 8 to make comparisons with the results of Section IV-F easier. We draw uniformly the weights of the arcs between 0 and $20{,}000$. For a given routed network, we define the *margin* as the difference between T_{max} and twice the longest route. Hence, for a given routed network choosing the margin or T_{max} is the same, but since we will consider many networks with different longest route's size, it will be more meaningful to set a global margin rather than T_{max}. In fact the margin represents the latency imposed by the communication process without taking into account the physical length of the network which cannot be changed. In our experiments the margin range from 0 to $3{,}000$. We look at two different regimes, a medium load of 80% and a high load of 95%. Considering smaller load is not relevant since then we know how to solve the problem even without waiting times as shown in Section IV-F.

We first try to understand what is the best choice of order for the first stage of the algorithm which is followed by the GD algorithm in this experiment. In Fig. 5 , we represent the success rate of the five kind of orders with regards to the margin. The value is an average computed over $10{,}000$ random star routed networks. Note that for the random order, we compute 1000 random orders and count it as a success as soon as there is a solution for one order.

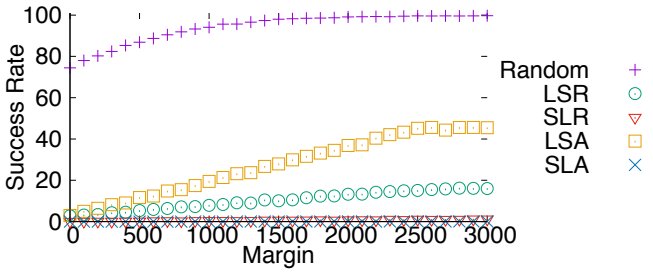


Fig. 5. Success rate of different orders, 95% load.

According to our experiments, sending the messages from the shortest to the longest route or arc does not work well. It corresponds to the policy of Prop. 1 which we already know to be bad for PAZL when the routes are long as in this experiment. Sending from the longest to the shortest route or arc works

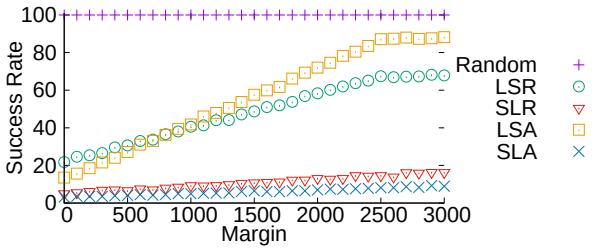


Fig. 6. Success rate of different orders, 80% load.

better and it seems that sorting the routes according to the length of the last arc rather than the route is better, at least in a loaded network.

The random order is the best by far: with a load of 95%, there is solution with margin 0 most of the time. Note that, when disallowing waiting times, there were no instances with an assignment for these parameters (see Section IV-F), which justifies the interest of studying the PALL problem. We now want to compare the performances of the three different algorithms used in the second stage. Since GD already showed excellent results on mild loads, it is more interesting to focus on the behavior of the algorithms on high load. Moreover, we will use 1,000 random orders for the first stage as it gives the best results. In the following experiment, we represent the success rate of the three algorithms with regards to the margin, computed over 10,000 random star routed networks and the same parameters as previously.

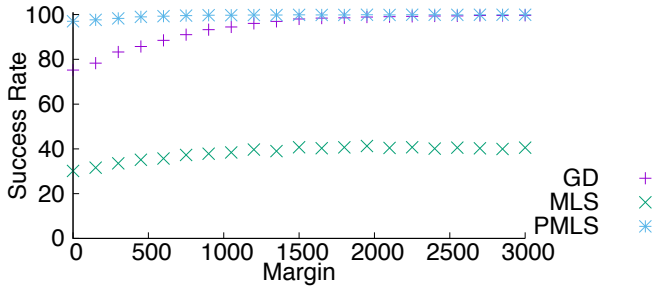


Fig. 7. Success rate of GD, MLS and PMLS, 95% load

The MLS algorithm performs poorly, worst than GD and PMLS, which shows that *taking into account the periodicity is fundamental*. The GD algorithms is close to 100% success rate for margins larger than 1,500 while the PMLS algorithm finds a solution in more than 97% of the experiments, even with a margin 0. Therefore, for the worst possible constraints on load and margin, there are a few instances for which we do not find a solution. However, with a margin of 1,000, which corresponds to about 0.05ms of additional delay with the chosen parameters, we always find a solution.

Recall that the random order corresponds to the best of 1,000 orders drawn uniformly which means that a computation can be up to 1,000 time slower than for a fixed order. The choice of the number of random orders drawn yields a trade-off between the computation time and the success rate. Up to now, we chose 1,000 random orders arbitrarily. We investigate the success rate of our algorithm with regards to the number of random orders drawn, a load of 95% and a margin 0. The following tables presents the average success rate for each number of sending orders, on 10,000 instances, for GD and PMLS.

# orders	1	10	100	1,000	10^4	10^5
GD	0.6	7.1	33.4	76.4	90.0	90.0
PMLS	38.9	83.0	95.3	97.2	97.5	97.7

Fig. 8. Impact of the number of random sending orders

First remark that we can improve our previous results by taking 10,000 random orders, instead of 1,000. The number of different orders is $7! = 5{,}040$ since we have 8 routes and the solutions are invariant up to a circular permutation of the order. Therefore instead of doing 10,000 random draws we could test every possible order in less time. However the computation time of this method would scale badly with n. On the other hand, remark that the success rate of PMLS is already high for 100 random orders, which means it will work even better than the other methods when n is larger and that the number of random orders drawn becomes critical.

Now that we have found the best amongst the algorithms solving PALL, we need to compare its performances against the actual way to manage the messages in a network: statistical multiplexing, where there is a FIFO buffer in each node of the network to resolve collisions. The time at which the messages are sent in the network is not computed as in our approach, thus we fix the offsets of each route to some random value. Even if this policy seems to work in practice when the network is not too loaded, it does not give any guarantee on the latency. Remark that the process is not periodic, therefore we must measure the process time of each route over several periods if we want to compute its maximum. We choose to simulate it for 1,000 periods and we have observed that the process time usually stabilizes after 10 periods. The margin is defined as the maximum process time, computed as explained, minus twice the size of the longest route.

In Fig. 9, we represent the cumulative distribution of margins for statistical multiplexing and PMLS. The distribution is computed from 1000 star routed networks drawn with the same parameters as before. We represent the distribution under high

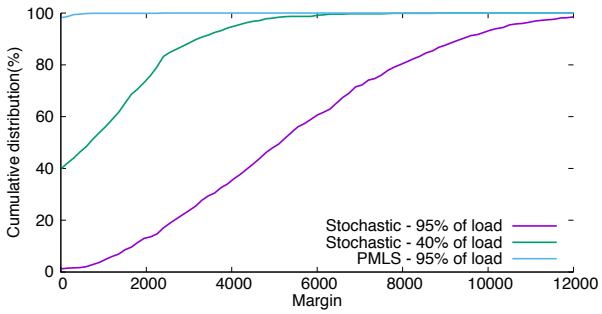


Fig. 9. Cumulative distribution of the margin of statistical multiplexing

and light load for statistical multiplexing and under high load only for PMLS since under light load the margin is always 0.

The experiment clearly shows that statistical multiplexing does not ensure a minimal latency. The latency is extremely high when the load is high, with a margin of about 10,000 for the worst 10% which corresponds to half the period, that is 0.5ms. Even when the network is lightly loaded, 20% of the instances have a margin of more than 2,000. On the other hand, PMLS finds an assignment with margin 0 in a highly loaded 97% of the time! For each 1000 slots of latency we save from the periodic process, we are able to lengthen the routes of 10km, which has a huge economic impact. We feel that it strongly justifies the use of a deterministic sending scheme for latency critical applications such as our C-RAN motivating problem.

VI. Conclusion

In this paper, we proposed two deterministic methods to establish a low latency periodic communication between BBUs and RRHs in a star shaped fronthaul network. The first method uses no buffering and has no latency overhead. It works when the routes are short (Longest-Shortest policy) or when the load is less than 80% (Exhaustive search of compact assignments). When the load is higher, buffering is allowed in the BBUs and we propose the algorithm PMLS which finds a deterministic communication scheme with almost no additional latency.

We plan to generalize our study of the PALL problem to other common fronthaul topologies, such as caterpillar, trees, cycles or bounded treewidth graphs. The cycle in particular are different since their forward and backward routes are not symmetric. We would like to design FPT algorithms in the number of routes for PALL on the star routed network, by generalizing the compact assignment idea used for PAZL.

We could also study variations of our problem. Instead of minimizing the maximum process time, we could try to minimize the average process time, a linear objective which could make linear programming useful. Moreover we could allow preemption, that is the messages are allowed to be cut into pieces, which would certainly change the complexity of the problem. Finally, the routes may not be fixed but chosen in the graph to minimize the maximum process time, which would make the problem even more difficult (maybe Π_2-complete instead of NP-complete).

References

[1] C. Mobile, "C-RAN: the road towards green RAN," *White Paper, ver*, vol. 2, 2011.

[2] Y. Bouguen, E. Hardouin, A. Maloberti, and F.-X. Wolff, *LTE et les réseaux 4G*. Editions Eyrolles, 2012.

[3] 3GPP, *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for the 5G system;*. Stage 1 (Release 16).

[4] A. Pizzinat, P. Chanclou, F. Saliou, and T. Diallo, "Things you should know about fronthaul," *Journal of Lightwave Technology*, vol. 33, no. 5, pp. 1077–1083, 2015.

[5] Z. Tayq, L. A. Neto, B. Le Guyader, A. De Lannoy, M. Chouaref, C. Aupetit-Berthelemot, M. N. Anjanappa, S. Nguyen, K. Chowdhury, and P. Chanclou, "Real time demonstration of the transport of ethernet fronthaul based on vran in optical access networks," in *Optical Fiber Communications Conference and Exhibition (OFC), 2017*, pp. 1–3, IEEE, 2017.

[6] N. Finn and P. Thubert, "Deterministic Networking Architecture," Internet-Draft draft-finn-detnet-architecture-08, Internet Engineering Task Force, 2016. Work in Progress.

[7] "Time-sensitive networking task group." http://www.ieee802.org/1/pages/tsn.html. Accessed: 2016-09-22.

[8] W. Howe, "Time-scheduled and time-reservation packet switching," Mar. 17 2005. US Patent App. 10/947,487.

[9] B. Leclerc and O. Marcé, "Transmission of coherent data flow within packet-switched network," June 15 2016. EP Patent App. EP20,140,307,006.

[10] R. J. Cole, B. M. Maggs, and R. K. Sitaraman, "On the benefit of supporting virtual channels in wormhole routers," in *Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, pp. 131–141, ACM, 1996.

[11] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin, "Frequency assignment in cellular phone networks," *Annals of Operations Research*, vol. 76, pp. 73–93, 1998.

[12] T. Erlebach and K. Jansen, "The complexity of path coloring and call scheduling," *Theoretical Computer Science*, vol. 255, no. 1, pp. 33–50, 2001.

[13] C. Strotmann, *Railway scheduling problems and their decomposition*. PhD thesis, PhD thesis, Universität Osnabrück, 2007.

[14] B. Zhou, "Multiple circular colouring as a model for scheduling," 2013.

[15] F. De Montgolfier, M. Soto, and L. Viennot, "Treewidth and hyperbolicity of the internet," in *Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on*, pp. 25–32, IEEE, 2011.

[16] I. Holyer, "The np-completeness of edge-coloring," *SIAM Journal on computing*, vol. 10, no. 4, pp. 718–720, 1981.

[17] D. 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*, pp. 681–690, ACM, 2006.

[18] A. J. Orman and C. N. Potts, "On the complexity of coupled-task scheduling," *Discrete Applied Mathematics*, vol. 72, no. 1-2, pp. 141–154, 1997.

[19] R. G. Downey and M. R. Fellows, *Parameterized complexity*. Springer Science & Business Media, 2012.

[20] B. Simons, "A fast algorithm for single processor scheduling," in *Foundations of Computer Science, 1978., 19th Annual Symposium on*, pp. 246–252, IEEE, 1978.

[21] J. K. Lenstra, A. R. Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of discrete mathematics*, vol. 1, pp. 343–362, 1977.