Deterministic Scheduling of Periodic Messages for Cloud RAN

Dominique Barth[1], Maël Guiraud[1], Brice Leclerc[2], Olivier Marcé[2], and Yann Strozecki[1]

[1]David Laboratory, UVSQ
[2]Nokia Bell Labs France

***Abstract*—A recent trend in mobile networks is to centralize in distant data-centers the processing units which were attached to antennas until now. 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 saving the latency incurred by buffering.**

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)[1] in the cloud. For example, LTE standards require 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 are no collisions to avoid latency caused by queuing delays? Eliminating this source of latency leaves us with more time budget for latency due to 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 time needed to transmit a minimal unit of data over the network. Since statistical multiplexing does not ensure a good latency we want to avoid any buffering in internal nodes of the graph. We take advantage of the deterministic nature of the messages we must manage i.e. the dates of arrival of messages are known beforehand. In fact, following LTE standard [2], we assume that the arrivals of all the packets are periodic with the same period. We propose to design a *periodic* process to send the messages through the network without collisions. By periodic process we mean that the network at times t and $t+P$ where P is the period, is in the exact same state.

We assume that the routes of the messages are already fixed, and there are no buffering allowed inside the network. Hence we only have two sets of values that we can set when building a periodic sending process, called a *periodic assignment*: 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 and we introduce the star routed network that we study in this article. We formalize the problem of finding a periodic assignment for sending messages without buffering (PAZL) or with buffering in the processing unit only (PALL). In Sec. III, we propose two algorithms solving PAZL

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

when the routes between RRHs and BBUs are small or when the load is light. Finally in Sec. IV, we introduce a family of algorithms to solve PALL and our experimentations allow to select one which works well, even in loaded networks. In particular we show that the deterministic communication schemes we design vastly outperform the traditional statistical 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. The problem PALL on a star routed network is very close to a two flow shop scheduling problem [15] with the additional constraint of periodicity. To our knowledge, all periodic scheduling problems are quite different from PALL. Either the aim is to minimize the number of processors on which the periodic tasks are scheduled [16], [17] while our problem correspond to a single processor and a constraint similar to makespan minimization. Or, in cyclic scheduling [18], the aim is to minimize the period of a scheduling to maximize the throughput, while our period is fixed.

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 directed path, that is, a sequence of adjacent vertices $u_0, \ldots, u_k$, with $(u_i,u_{i+1}) \in A$. The **latency** of a vertex u_i in a path r is defined by $\lambda(u_i,r) = \sum_{0 \leq j < i} \omega(u_j, u_{j+1})$. 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.

In the context of cloud-RAN applications, we need to send a 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.

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. In this paper, we assume that τ is the same for all routes. Indeed, the data flow sent by an RRH to its BBU is the same, regardless of the route. 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 $[t(v,r)]_{P,\tau} = \{t(v,r) + k \mod P \mid 0 \leq k < \tau\}$. 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 (v,w) 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. 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, i.e., 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

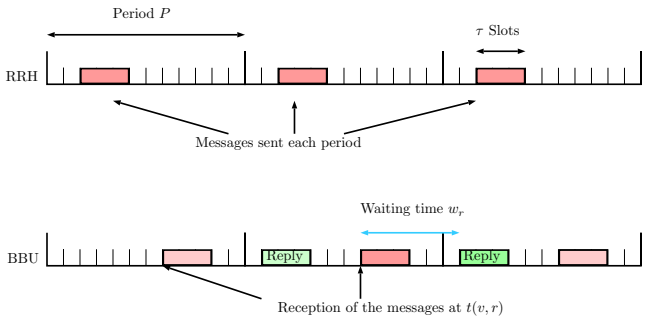

Fig. 1. Periodic process

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.

Note that, in the process we decribe, we do not take into account time the computation time a BBU needs to deal with one message. It can be encoded in the weight of the last arc leading to the BBU and thus we do not need to consider it explicitly in our model. The whole process time for a message sent on the route r is equal to $PT(r) = \lambda(r) + w_r + \lambda(r)$. The **maximum process time** of the assignment m of $(G,\mathcal{R})$ is defined by $MPT(m) = \max_{r\in\mathcal{R}} PT(r)$. In the process time, we count the time between the time the first slot of the message is emitted and the first time at which the first slot of 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 adds τ to the process time. We chose the first definition which is slightly simpler since the choice of definition does not change the problems we consider in the rest of the article, assuming all messages are of size τ. Each route must respect some time limit that we call a *deadline*. To represent these deadlines, we use a deadline function d, which maps to each route r an integer such that $PT(r)$ must be less than $d(r)$.

C. Periodic assignment for low latency

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 a deadline function d. **Question:** does there exist a (P,τ)-periodic assignment m of $(G,\mathcal{R})$ such that for all $r \in \mathcal{R}$, $PT(r) \leq d(r)$?

It turns out that the problem PALL is hard to solve for *general* routed networks. The long version of this article contains complete proofs of the NP-hardness or hardness of approximation of PALL and of several of its variants.

We also consider a simpler version of PALL, that we call **Periodic Assignment for Zero Latency** or PAZL: we ask for a (P,τ)-periodic assignment **with all waiting times equal to** 0. We introduce PAZL because it is simpler to study and we are able to prove theoretical results and find better algorithms than for PALL. As we experimentally show in Sec. III-C, 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 buffering at all.

D. 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 define a symmetric routed network. The symmetric routed networks $(G,\{r_i,\rho(r_i)\}_{i<n})$ is called a **star routed network**. This graph looks very simple, but every network in which all routes share an arc can be reduced to a star routed network. This topology is realistic, since often all the BBUs are located in the same data-center. In such a situation, we can see the weights of the arcs (c_t,t_i) either as all equals (in that case, the problem is trivial, see Sec.IV) or different due to the structure of the network inside the data-center and the various hardwares used for different BBUs.

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.

III. The Star Routed Network: No Waiting Time

A. PALL *with no waiting time:* PAZL

In this section, we deal with the problem PAZL on star routed networks. We want an assignment with waiting times zero, that is, $d(r) = 2\lambda(r)$. For a route r, choosing the offset m_r also sets the offset of the route $\rho(r)$ to $m_{\rho(r)} = m_r + \lambda(r) \mod P$.

B. Three algorithms to solve PAZL

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 collisions 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. On the full version of this paper, one can found the proof that this algorithm always gives a solution if $n\tau+2(\lambda(r_{n-1})-\lambda(r_0)) \leq P$.

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. One can find in the full paper version 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 $1/3$.

This algorithm, contrarily to the previous one, may work well, even for loads higher than $1/3$. In fact, experimental data in Subsection III-C suggest 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.

In the full paper version, 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 [23]). This is justified since n is small in practice (from 10 to 20) and the other parameters such as P, τ or the weights are large. We call this algorithm **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.

C. Experimental evaluation

We now want to compare the three presented algorithms. Both Greedy algorithm and Shortest-Longest are polynomial time algorithms but can fail, while the exhaustive search finds a solution if it exists, but works in exponential time. From the 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 slots of 64 bits, messages of approximately 1Mb and links of bandwidth 10Gbps when P is one millisecond.

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 a BBU and an RRH.

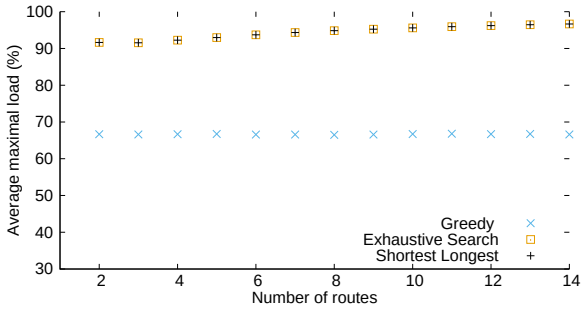

Fig. 2. Maximal load averaged over 1,000 random instances

For short routes, there is much of the time a solution to PAZL found by Shortest Longest or the exhaustive search, even with high loads. The Greedy algorithm seems to perform better than its theoretical bound.

We now look at the performance of the algorithms on longer routes. The weights of the arcs are drawn between 0 and 20,000.

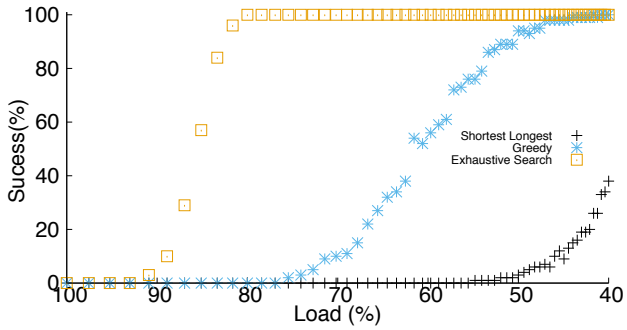

Fig. 3. Success rate for 8 routes in regard of the load, over 1,000 random instances

On long routes, the performances of Shortest-Longest are abysmal since the difference between the length of the routes is large. 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 also investigate the computation time of the exhaustive search and notice that for more than 20 routes, this algorithm needs more optimisations to run in a reasonable time.

IV. Solving PALL on Star Routed Networks

In this section, we consider the more general PALL problem on star routed networks. The messages are allowed to wait in the target vertices (BBUs) to find assignments more easily. Hence, 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}. There are easy to solve special cases either if all the weights on the arcs (c_t,t_i) are the same or if all the weights on the arcs (s_i,c_s) are the same.

In this section, we first choose the offsets of the forward routes randomly. One can found the study of this choice in the full paper version. The purpose of the following algorithm is then to set the offsets of the backward routes.

A. Greedy scheduling of backward routes

Consider a forward route r_i, whose offset is m_{r_i} and its backward route is $\rho(r_i)$. 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 $PT(r_i) \leq T_{max}$.

The first algorithm we propose to solve PALL 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. This algorithm can fail since it does not take into account the periodicity.

The problem pall is very similar to the following scheduling problem. Given a set of jobs with *release times* and *deadlines*, schedule all jobs on a single processor, that is choose the time at which they are computed, so that no two jobs are scheduled at the same time. Since the running time is the same on all jobs, this problem can be solved in polynomial time [24] . On the other hand if the running times are different the problem is NP-complete [25].

We reduce PALL to this scheduling problem, and propose a variant of the polynomial algorithm with an improvement that allows this algorithm to consider the periodicity, that we call **Periodic Minimal Latency Scheduling (PMLS)**.

There is also some FPT algorithms to solve PALL which are not convenient in practice.

B. Experimental evaluation

We set the number of routes to 8 to make comparisons with the results of Section III-C easier. We draw uniformly the weights of the arcs between 0 and 20,000. When studying many networks with different longest route's size, comparing the smallest process time for which an algorithm finds a solution is not relevant since it depends mostly on the size of the longest route. Hence, we define the **margin** as the difference between T_{max} and twice the longest route. 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 solve the problem without waiting times as shown in Section III-C. As mentioned in sec IV,to set the forward routes offsets, we choose many random orders, and stop when a solution is found with the studied algorithm. We then want to compare the performances of the two different algorithms used to set the backward offsets. We represent the success rate of the three algorithms with regards to the margin.

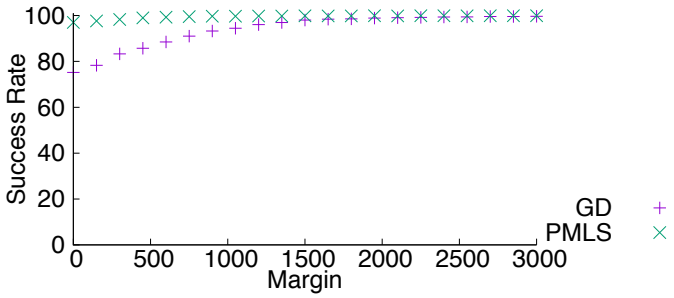

Fig. 4. Success rate of GD and PMLS, over 10,000 instances, 95% load

TODO: ajouter l'algo FPT qui améliore PMLS

Fig. 4 shows that PMLS is far better than GD since it 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.

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 Figure. 5, we represent the probability of success for statistical multiplexing and PMLS for different margin. The distribution is computed from 1000 star routed networks drawn with the same parameters as before. We represent the distribution under high and light load for statistical multiplexing and under high load only for PMLS since under light load the margin is always 0.

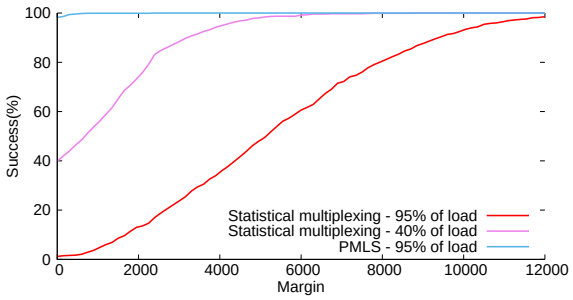

Fig. 5. Probability of success of statistical multiplexing and PMLS for several margins

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 network 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 economical 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.

V. 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 cycles in particular are different since their forward and backward routes are not symmetric. We would like to design an FPT algorithm for PALL which is as efficient as the one for PAZL and prove that both problems are NP-hard.

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. We could allow preemption, that is the messages are allowed to be cut into pieces, which would certainly change the complexity of the problem. Instead of periodic communication we could try to organize communications with pseudo-periodic schemes or even a temporal law. Moreover we could allow a bounded message loss. 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).

Acknowledgments: The authors thank Christian Cadéré and David Auger for the friendly discussions they had on the subject and their insightful remarks. This work is partially supported by the french ANR project N-GREEN.

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] W. Yu, H. Hoogeveen, and J. K. Lenstra, "Minimizing makespan in a two-machine flow shop with delays and unit-time operations is np-hard," *Journal of Scheduling*, vol. 7, no. 5, pp. 333–348, 2004.

[16] J. Korst, E. Aarts, J. K. Lenstra, and J. Wessels, "Periodic multiprocessor scheduling," in *PARLE'91 Parallel Architectures and Languages Europe*, pp. 166–178, Springer, 1991.

[17] C. Hanen and A. Munier, *Cyclic scheduling on parallel processors: an overview*. Université de Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique, 1993.

[18] E. Levner, V. Kats, D. A. L. de Pablo, and T. E. Cheng, "Complexity of cyclic scheduling problems: A state-of-the-art survey," *Computers & Industrial Engineering*, vol. 59, no. 2, pp. 352–361, 2010.

[19] 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.

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

[21] 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.

[22] 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.

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

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

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