Scheduling periodic messages and their answers on a single link

Maël Guiraud[1,2] 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 processing units which, until now, were attached to antennas. The main challenge is to guarantee a low latency for the periodic messages sent from the antennas to their processing units and back to fulfill protocol time constraints. The problem we adress is to find a sending scheme from the antennas to their processing units and back without contention nor buffering.**

We focus on a simple but common star shaped topology, where all contentions are on a single arc shared by all antennas. For messages of arbitrary size, we show that there is always a solution as soon as the load of the network is less than 40%. Moreover, we explain how we can restrict our study to packet of size 1 without increasing too much the global latency.

For message of size 1, we prove that it is always possible to schedule them on a star shaped topology, when the load is less than $2/3$ using a polynomial time algorithm. Moreover, using a simple random greedy algorithm, we show that almost all instances of a given load admit a solution, explaining why most greedy algorithms work so well in practice.

I. Introduction

Lister les usages possibles: sans retour mais de profondeur 2 ou plus. Sonar ? Train ? Réseau de capteurs communicant (industrie) Logistique dans une usine (chaine avec différent produits)

Comparer à la litérature. Aller voir ce qu'on nous a cité sur l'ordonnancement périodique et les trucs de train périodique qu'on a trouvé.

Mélanger l'introduction et le modèle. Voir comment c'est présenté dans two flow shop

In this article, we model a very simple network in which periodic messages flow through a single link. The answer to each message is then sent back through the same bidirectional link. The model and problem can easily be generalized to any network, that is any directed acyclic graph with any number of contention points, see [1]. We choose here to present the simplest non trivial such network, for which we can still obtain some theoritical results.

The time is discretized and the process we consider is periodic of fixed integer period P. We use the notation $[P]$ for the set $\{0,\dots,P-1\}$. In our application, all messages are of the same nature, hence they are all of the same size denoted by τ. This size corresponds to the time it requires to send it through some contention point of the network. We denote by n the number of messages, which are numeroted from 0 to $n-1$. A message i is caracterized by its delay d_i. It means that if the message number i goes in the link at time t, then it returns to it in the other direction at a time $t+d_i$.

Ici mettre un dessin avec le réseau avec les deux points de contention, les n messages et les délais.

Since the process we describe is periodic, we chose any interval of P unit of time. Describing the messages going through the two contention points during such an interval completely define the periodic process. We call the representation of the interval of time in the first contention point the *first period* and the *second period* for the second contention point.

An *offset* of a message is a choice of time at which it arrives at the first contention point. Let us consider a message i of offset o_i, it uses the interval of time $[i]_1 = \{(o_i+t) \mod P \mid 0 \leq t < \tau\}$ in the first period and $[i]_2 = \{(d_i + o_i + t) \mod P \mid 0 \leq t < \tau\}$ in the second period. We say that two messages i and j collide if either $[i]_1 \cap [j]_1 \neq \emptyset$ or $[i]_2 \cap [j]_2 \neq \emptyset$.

We want to send all messages, so that they are no collision in the common link. In other word, we look for a way to send the messages without using buffering and hence limiting the latency to the physical length of the link. An assignment is a choice of an offset for each message such that *no pair of message collide*. Formally, an *assignment* is a function from the messages, represented by their indices in $[n]$ to their offsets in $[P]$.

We call Periodic Message Assignment or PMA the problem we solve in this article, which asks, given an instance of n messages, a period P and a size τ to find an assignment or to decide there is none.

Ici donner un exemple d'instance et de solution, représenté graphiquement.

The complexity of PMA is not known. However, we have proven that when parametrized by n the number of messages, the problem is FPT [2]. A slight generalization, with many contention points but each message only goes through two as in PMA is NP-hard [2] and we conjecture that PMA is NP-hard.

To overcome the hardness, we study PMA in this article when the load of the system is small enough. The *load* is

defined as the number of unit of time used in a period by messages in a contention point divided by the period. Hence the load is equal to $n\tau/P$. Our aim is to prove that for small load, there is *always* an assignment and that it can be found by a polynomial time algorithm.

II. Greedy Algorithms for Large Messages

In this section, we study the case of large messages. When modelizing real problems, it is relevant to have $\tau > 1$ when the transmission time is large with regard to the delay.

A partial assignment is a function defined from a subset S of $[n]$ to $[P]$. We say that a message in S is *scheduled* (by A), and a message not in S is *unscheduled*. We will only consider partial assignments such that no pair of messages of S collide. We denote assignment and partial assignment by A. If A has domain S, and $i \notin S$, we define the extension of A to i by v, denoted by $A(i \to v)$, the function defined as A on S and such that $A(i) = v$.

All presented algorithms build the assignment incrementally, by growing the size of the domain of a partial assignment. Moreover, the algorithm of this section are greedy since once an offset is chosen for a message, it is never changed.

A. First fit

Assume that for some partial assignment A, the message i has offset t: it uses all times from t to $t+\tau-1$ in the first period. If a message j is scheduled with some offset t' before t, then the last time it uses in the first period is $t'+\tau-1$ and it should be less than t, which implies that $t' \leq t-\tau+1$. If t' is sent after t, to avoid collision, it should be larger or equal to $t+\tau$. Hence the message i forbid $2\tau-1$ positions for messages still not scheduled becaus eof tis use of time in the first period. The same reasonning can be done for the second period, wich again forbid $2\tau-1$ offsets. Hence, if $|S|$ messages are already scheduled, then $|S|(4\tau-2)$ offsets are forbidden for the remaining messages. Note that this is an upper bound on the number of forbidden offsets, the same offset can be forbidden because of a message on the first and also on the second period.

We can thus define $Fo(A)$, the maximum number of forbidden offsets when extending A. Formally, assume A is defined over S and $j \notin S$, it is the maximum over all values of d_j, of $|\{ v \in [P] \mid A(i \to v) \text{ has no collision } \}|$. The previous paragraph shows that $Fo(A)$ is always bounded by $(4\tau-2)|S|$.

The first algorithm deals with the route in the order they are given: for each unscheduled route it tests all offsets from 0 to $P-1$ until one do not create a collision with the current assignment. We call this algorithm *First Fit*. Remark that if $Fo(A) < P$, then whatever the delay of the route we want to extend A with, it is possible to find an offset. Since $Fo(A) \leq (4\tau-2)|S|$ and $|S| < n$, First Fit (or any greedy algorithm) will always succed when $(4\tau-2)n \leq P$, that is when $n\tau/P \leq 1/4$. It turns out that First Fit always create compact assignments (as defined in [2]), that is a message is always next to another one in one of the two period. Hence, we can prove a better bound on $Fo(A)$, when A is built by First Fit, which implies the following therem.

Theorem 1. *First Fit always solves* PMA *positively on instances of load less than* $1/3$.

Proof. To prove the theorem, it is enough to prove that $Fo(A) \leq |S|(3\tau-1)+\tau-1$. We prove that by induction on the size of S. For $S=1$, it is clear since a single message forbid at most $(3\tau-1)+\tau-1 = 4\tau-2$ offsets. Now, assume $Fo(A) \leq |S|(3\tau-1)+\tau-1$ and consider a route $i \notin S$ such that First Fit builds $A(i \to v)$. By definition, choosing $v-1$ as offset creates a collision, w.l.o.g. say it is a collision on the first period. It means that there is a message between $v-\tau$ and $v-1$ (modulo P), hence all these offsets which are forbidden the the choice of offset of i were already forbidden. It implies that only $3\tau-1$ new offsets are forbidden, that is $Fo(A(i \to v)) \leq Fo(A) + (3\tau-1)$, which proves the induction and the theorem. □

B. Meta-intervals

The second method is described in [2] and achieves the same bound on the load using a different method which will also be used in more involved algorithm. The idea is to restrict the possible offsets at which messages can be scheduled. It seems counter-intuitive, since it decreases artificially the number of offsets to schedule new messages. However, it allows to reduce the number of forbidden offsets, when the algorithm is designed to this aim. A *meta-offset* is an offset of value $i\tau$, with i an integer from 0 to P/τ. We call Meta-Offset the greedy algorithm which works as first fit, but consider only meta-offsets when scheduling new messages.

We define $Fom(A)$ as the maximal number of meta-offsets forbidden by A. By definition, two messages with a different meta-offset cannot collide in the first period. Hence, $Fom(A)$ can be bounded by $3|S|$ and we obtain the following theorem.

Theorem 2 (Proposition 3 of [2])**.** *Meta-Offset always solves* PMA *positively on instances of load less than* $1/3$.

C. Tuples and meta-intervals

Finally, we propose more complicated greedy algorithms which always solves PMA positively for larger load. The idea is to schedule several routes at once, to maximize the compacity of the obtained solution. We first describe the algorithm which places pairs of routes and then explain quickly how we can extend it by placing tuples of routes at the same time.

We first prove a lemma which allows to assume that the period P is a multiple of τ. It makes easier the division of both periods into meta intervals of size τ and simplify the presentation of the algorithm.

Lemma 3. *Let us consider an instance with n messages of size τ and period P, then there is an instance with n messages of size τ' and period $P' = m\tau'$ such that an assignment of the second instance can be transformed into an assignment of the first.*

Proof. Let $P = m\tau + r$ with $r \leq \tau$. We define $P' = mP$, $d'_i = md_i$ and $\tau' = m\tau + r$. With this choice, we have $P' = m(m\tau + r) = m\tau'$. Consider an assignment A' of the instance P',τ',$(d'_0, \ldots, d'_{n-1})$. If we consider $\tau'' = m\tau$, then A' is also an assignment for P',τ'',$(d'_0, \ldots, d'_{n-1})$ since we only reduce the number of used position in each period. Then we use a compactifcation procedure as in [2] to obtain A from A' so that all positions of the messages in the first and second period are multiple of m. Hence, if we define A as $A(i) = A'(i)$, we obtain a solution of the original instance. □

Pas sur que ça soit nécessaire mais ça aide pour certaines valeur de contrainte.

We are interested in the remainder modulo τ of the delays of each message. We write $d_i = d'_i\tau + r_i$ and assume the messages are sorted by increasing r_i. A *compact pair* is a pair of messages (i,j) such that we can put them next to each other in the second period using only meta-intervals in the second. Say that $i < j$, hence $r_i \leq r_j$ we denote by g the gap between the two message. We have $d'_i = g+1+d'_j \mod P/\tau$ hence, we require that $v \neq 0$ so that there are no collision in the first period.

Lemma 4. *Given 3 messages, two of them always form a compact pair.*

Proof. If the first two messages or the first and the third message form a compact pair, then we are done. If not, then $d'_2 = d'_3$ and the messages 2 and 3 form a compact pair. □

We call compact pairs the following greedy algorithm. From the n routes in order, we build a sequence of at least $n/3$compact pairs using Lemma 4. Then they are scheduled in order using meta offsets. When there a no free meta offset for some pair, the remaining routes are scheduled as in Meta Offset.

Donner le lemme général + définition de compact tuple, c'est plus simple. Faire un dessin avec un compact k uple + la zone interdite pour un k' uple (selon qu'il soit ou pas de reste plus grand) en tant que preuve du lemme.

Lemma 5. *A compact pair forbid in the second period four meta offsets for another compact pair to be scheduled.*

Theorem 6. *Compact pairs always solves* PMA *positively on instances of load less than* $3/8$.

Proof. The number of forbiden offsets when n_2 pairs have been scheduled is:

$$4n_2 + 4n_2 = 8n_2.$$

Hence, the first phase can continue while the number of forbidden offset is less than m, hence n_2 is at least $m/8$. In the second phase, a compact pair forbid 3 meta offsets in the second period and two in the first. If we denote by n_1 the number of messages scheduled in the second phase, we have the following number of forbiden offsets:

$$n_2 * 5 + n_1 * 3.$$

The algorithm can always schedule messages when this number is less than m, hence

$$n_2 * 5 + n_1 * 3 \geq m$$
$$n_1 \geq \frac{8m - 5m}{24}$$
$$n_1 \geq \frac{1}{8}$$

The number of routes scheduled is thus at least $2n_2 + n_1$, which is $\frac{3}{8}$. □

Description de compact uples -¿ on choisit un k. On places des k-uples compact tant qu'on peut, puis des $k-1$ uples et ce jusqu'à 1.

Theorem 7. *Compact tuples always solves* PMA *positively on instances of load less than* $4/10$.

Proof. Un lemme comme 4 pour dire qu'on peut constituer des pairs. Un lemme comme 5 pour dire combien un k-uple gêne un l-uple qu'on veut placer: $l + k + 1$ et même $l + k$ si le l uple est après le k uple. Il suffit de commencer à 10. □

D. Experimental results

Results better in practice, the worst case is different from the average case. Two additional heuristics to test:

- heuristic to build super compact assignment (among the compact assignments possible, chose the one which maximize the gain on the second bloc)
- heuristic to maximize the free position of the remaining elements, c'est l'idée de la partie d'après

Quality of the results explained by the average analysis done later.

III. Why we can restrict ourself to message of size one

A. Reduction from large message

Methode avec perte de load d'un facteur $1/2$: considère des messages deux fois plus gros et on se sert d'un trick de décalage pour pour que tous les d_i soient bien alignés. Utile pour les arguments qui montrent qu'en moyenne on réussit pour une charge 1, se transfert pour charge 1.

B. Reduction from large message to small messages using buffering

Possibility of buffering the route during a time b: change d_i to $d_i + b$. Form of tradeoff between latency (more latency implies smaller message size implies solution with higher load).

All messages are buffered enough time so that their d_i have the same remainder modulo τ. It costs at most τ of buffering (which is not good in our application but not horrible either). We can choose the longest route as reference, hence that route cost zero. If all distances are of d_i a multiple of τ, we have an easy reduction to the case of $\tau = 1$, by dividing all values by τ.

We can do the same kind of transformation by buffering all messages, so that the d_i are multiples of τ/k. The cost in term of latency would be τ/k but the reduction yields message of size k. For small size of messages, it is easy to get better algorithm (in particular for $\tau = 1$ as we show in the next section).

Theorem 8. *Compact pairs on instances with $\tau = 2$ always solves* PMA *positively on instances of load less than $4/9$.*

Proof. We assume w.l.o.g that there are less message with even d_i than odd d_i. We schedule compact pairs of messages with even d_i, then we schedule single message with even d_i. The worst case is when there is the same number of the two type of messages. The number of forbidden offsets for n scheduled messages is bounded by

$$n/2(1 + 3/2) + n/2(1 + 1).$$

Hence, we can always schedule messages when $n \leq (4/9)m$.

□

If we want to optimise for the average latency, we win a factor of two as we now explain.

Define the average buffer time for a choice of reference: B(t). If we sum the $B(t)$ for $t = 0$ to $\tau - 1$, the contribution of each message will be $\sum_{i=0}^{\tau-1} i$. Hence, the sum of the $B(t)$ is $nP(P-1)/2$. There is at least one term of the sum less than the average, hence there is a t_0 such that $B(t_0) \leq n(\tau-1)/2$. In other word, the average delay for the message is less than $\tau/2$.

Finally, solution with no buffering but which doubles the load.

IV. Going Above Load of 1/2 for messages of size one

Expliquer que le genre de méthode utilisé dépend de toutes les routes, impossible en streaming, impossible de rajouter une route avec les mêmes garanties. Peut-être possible de faire un swap avec une autre route ?

We give a method which always finds a solution for load $1/2 + \epsilon$.

To go above $1/2$ of load, we use a two-pass algorithm. In the second pass, a greedy algorithm is used to place a subsets of the routes, selected because they have less conflicts with the already placed routes. The first pass is not greedy, since we allow to change fixed route, but we could us a greedy first pass or even a single pass greedy algorithm to go over $1/2$. However, the proofs are much harder and the ϵ for which they hold is much smaller.

Definition 1. *The potential of a route of shift s in a partial solution (whether fixed or not in the partial solution), is the number of integers $i \in [P]$ such that i is used in the forward window and $i + s \mod P$ is used in the backward window. We denote by Pot(S) the sum of potentials of the routes in the partial solution S.*

Definition 2. *The potential of a position i of the forward window, for a partial solution, is the number of routes of shift s such that $i + s$ is used in the partial solution.*

The potentials of the positions satisfy the following simple invariant.

Lemma 9. *The sum of potentials of all positions of the forward window in a partial solution of size k is nk.*

We then link Pot(S) to the potential of the positions in the forward window.

Lemma 10. *The sum of potentials of all used positions in the forward window in a partial solution S is equal to Pot(S).*

We now describe the algorithm to solve our problem with load $1/2+\epsilon$. The first pass assign routes in any greedy manner, until it cannot assign some route anymore. Then, it applies some procedure described later which remove a route from the solution and add anther one. If at some point this procedure fails, it stops. When this algorithm stops, it can be shown that the potential of the obtained partial solution is larger than some value. Then, we select R the set of the ϵP routes of largest potential. The routes in R and in the partial solution are removed, then the free routes not in R are added to the partial solution and finally the route in R are added, using any greedy algorithm.

A. Swap and potential improvement

Let S be some partial solution of size k and let r be a free route of shift s. Assume that r cannot be used to extend S. The swap operation is the following: select a free position p, remove the route of position $p + s$ in the forward window of S and add r at position p in the forward window. We denote this operation by $Swap(r,p,S)$.

Lemma 11. *Let S be some partial solution of size k and let r be a free route of shift s. If r cannot be used to extend S, then either $Pot(Swap(r,p,S)) > Pot(S)$ or $Pot(S) \geq kn/2$.*

Proof. The positions in the forward window can be partitionned into two part: P_u the positions used in the forward windows and P_f the positions unused in the forward windows. Let us denote by V_f the value of the positions in P_f and by V_u the potential of the positions of P_u. By Lemma 10, since P_f and P_u partitions the positions, we have $V_f + V_u = kn$.

By hypothesis, since r cannot be placed, for all $p \in P_f$, $p+s$ is used in the backward window. We now define a function F which associates to $p \in P_f$ the position p' such that there is a route r' in S placed at p' in the forward window and at $p+s$ in the backward window. The function F is an injection from P_f to P_u. Rmeark now that if we compare $Swap(r,p,S)$ to S, on the backward window nothing changes. Hence the potential of each position in the forward window is the same. Hence, doing the operation $Swap(r,p,S)$ add to $Pot(S)$ the potential of the position p and removes the potential of position $F(p)$. Assume now, to prove our lemma, that for all p, $Pot(Swap(r,p,S)) \leq Pot(S)$. It implies that $V_f \leq V'_u \leq V_u$ and by Lemma 9 we have $V_f \leq Pot(S)$. Since $V_f + Pot(S) = kn$, we have that $Pot(S) \geq kn/2$. □

B. Analysis of the Algorithm

We give an analysis of the algorithm, showing that it works for some value of ϵ. We will later show that some refinments of this algorithm: a better selection of the values added in the second step, the possibility to repeat the first step to guarantee a higher potential yields a better ϵ.

Theorem 12. *The two-pass algorithm solves positively our problem with load* $1/2 + 1/16$.

Proof. The first pass of the algorithm guarantes that we obtain a partial solution S of size k such that $Pot(S) \geq kn/2$ by Lemma **??**. Moreover, $k \geq P/2$ since one can always place $P/2$ routes with a greedy algorithm.

At the end of the first pass, we have a potential of at least $Pn/4$ and we select the ϵP routes of largest potential. They must be of potential at least $2\epsilon P, 2\epsilon P + 2, \ldots, 4\epsilon P$. Sort all routes by decreasing potential and assume that the previous condition is not met, that is the ith route in order of potential is of potential less than $4\epsilon P - 2i$. The potential of a single route is bounded by $P/2$ since each placed route contribute at most one to its potential. Therefore, the first i routes are of potential at most $P/2$ and the following ones of potential at most $4\epsilon P - 2i$. Therefore the potential is less than $iP/2 + (4\epsilon P - 2i)(n-i)$. This function is decreasing for $i \leq \epsilon P$, hence the potential should be less than $4\epsilon Pn$.

For the algorithm to succeed, we want the potential to be larger than $4\epsilon Pn$ so that the condition on the routes of largest potential is met. Hence we must satisfy the following equation:

$$Pn/4 \geq 4\epsilon Pn.$$

$$\epsilon \leq 1/16.$$

□

Pour améliorer les résultats on peut répéter l'algo de swap une fois qu'on a obtenu au moins $(1/2 + \epsilon)P$ routes placées, pour obtenir $n^2/2$ en potentiel. On doit alors avoir $n^2/2 \geq 4\epsilon Pn$, ce qui donne $\epsilon \leq 1/14$. Si on veut pousser la technique plus loin, il faut améliorer la borne sur le potentiel. Au lieu de prendre les ϵP plus grandes routes, on prend pour i de 1 à ϵP la route de plus petit potentiel supérieur à $2\epsilon P$ et pour qu'elle existe, il suffit que le potentiel soit supérieur à $2\epsilon P(n-i)$. À vérifier, si à un moment tout est de potentiel $2\epsilon P$ au moins, alors c'est facile de conclure. On obtient alors $\epsilon \leq 1/6$, ce qui donne un algo dès que la charge est inférieure à $2/3$.

Autre possibilité, améliorer le potentiel en obtenant plus que la moyenne, en jouant notemment sur la symétrie Backward et Forward.

Expliquer les nombreuses raisons pourquoi ça marche mieux en pratique.

V. Algorithms for random instances

a) $\tau = 1$: We analyze the following process, called **Uniform Greedy** or UG. For each element in order, chose one admissible position uniformly at random. We analyze the probability that Uniform Greedy solve the problem, averaged over all possible instances. It turns out that this probability, for a fixed load strictly less than one goes to zero when m grows.

Définir l'ensemble des solutions de taille n parmi m.

Theorem 13. *Given an instance of size n uniformly at random UG produces a solution uniformly at random or fail.*

Proof. Regarder mes notes partielles pour compléter ça. □

Let us denote by $P(m,n)$ the probability that UG fails at the n_th steps assuming it has not failed before.

Theorem 14. *We have*

$$P(m,n) = \frac{\binom{n}{2n-m}}{\binom{m}{n}}.$$

In particular, $P(m,n) \leq f(\lambda)^m$, where $f(\lambda) < 1$.

Proof. Probability independent of the shift of the n element, can say it is 0. It is the probability that two sets of size n in $[m]$ are of union $[m]$. It is the same as the probability that it contains a given set of size $m - n$. Could find an asymptotic online. □

Can we make the same argument for a deterministic algorithm? The not average version of the argument is the previous proof.

Show that we can find a solution for load $1/2 - \epsilon$, even for large τ if n is large enough, using the trnasformation without buffering.

VI. Experimental results for $\tau = 1$

VII. Lower bounds

Example/family of examples for which some greedy alg fail. Example/family of examples with a given load such that there are no feasible solution.

References

[1] B. Dominique, G. Maël, and S. Yann, "Deterministic scheduling of periodic messages for cloud ran," *arXiv preprint arXiv:1801.07029*, 2018.

[2] D. Barth, M. Guiraud, B. Leclerc, O. Marcé, and Y. Strozecki, "Deterministic scheduling of periodic messages for cloud ran," in *2018 25th International Conference on Telecommunications (ICT)*, pp. 405–410, IEEE, 2018.