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 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. The problem we adress is to propose a sending scheme from the antennas to their processing units and back without contention and buffer.**

We focus on a simple but common star shaped topology, where all contentions are on a single arc shared used 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 ??. Moreover, we explain how we can restrict our study to packet of size 1 without increasing 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 on average, 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 chose to present here the simplest non trivial such network, for which we can still obtain some theoritical results.

The process we consider is periodic of fixed integer period P. We use the notation $[P]$ for the set $\{0, \ldots, P-1\}$. In our application, all messages are of the same nature, hence they are all of the same size denoted by τ. 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 through the link at time t, then it returns to it in the other direction at a time $t + d_i$.

An offset of a message is a choice of time at which it arrives in the first contention point in the first period. 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 contention point and $[i]_2 = \{(d_i + o_i + t) \mod P \mid 0 \leq t < \tau\}$ in the second contention point. 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 conflict 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*.

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 message τ to find an assignment or to decide there is none.

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

Complexity observation: not known, FPT in n, but a slight generalization is NP-hard and we conjecture it is NP-hard. We define a natural parameter of the problem, *the load*, which is equal to $n\tau/P$. Even if the general problem is certainly NP-hard, we study PMA when the load is small enough. We are able to prove that for some value of the load (depending whether τ is one or not), there is always an assignment and it can be found in polynomial time.

II. Greedy Algorithms for Large Messages

An assignment is a function from the messages, represented by their indices in $[n]$ to their offsets in $[P]$. A partial assignment is a function defined from a subset S of $[n]$ to $[P]$. We denote assignment and partial assignment by A. Moreover, if A has domain S, and $i \notin S$, we write $A(i \to v)$ for the function defined as A on S and such that $A(i) = v$.

All presented algorithms build the solution incrementally, by growing the size of the domain of an 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

Bien expliquer que les messages de taille τ genent de $2\tau-1$ sur le deuxième intervalle. Ce qui explique que si on les place n'importe comment on a presque $1/4$ comme borne alors que si $\tau = 1$ on a $1/2$. Dans cette partie, on essaye de passer de $1/4$ à $1/2$.

The first algorithms deals with the route in the order they are given: for each route it tests all offsets in $[P]$ until one do not create a collision with the current assignment. We call this algorithm *First Fit*. A naïve analysis of the algorithm, shows it always succeed for a load of $1/4$, since each packet placed forbid at most $2\tau-1$ offsets on each contention node. It turns out that the algorithm always create compact assignment (as defined in [2]), that is a message is always next to another one in one of the two contention nodes. Hence, the constraints are less stringent when placing a new route, as shown in the next theorem.

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

Proof. Analyzed through compacity $1/3$. □

B. Meta-intervals

The second method is described in ..... and achieves the same bound using a different method which will be used later. The period is divided into meta-intervals of size τ and we only consider offsets for a route, such that its message uses exactly a meta-interval We call it Meta Interval, since it works as first fit, but using meta-intervals in the first contention node.

Theorem 2 (Citer)**.** *Meta Interval 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 for larger load. They rely on placing several routes at once, to maximize the compacity of the solution. We first describe the algorithm which places pairs of routes and then explain 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 c'est nécessaire.

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 2-uple 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 a set of k^2 messages, one can always build a compact k-uple from messages of this set.*

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 τ can be assumed to be one if we allow buffering

Rank the d_i by value, and compute $d_i + d_n \mod \tau$. We allow buffering, but the worst time should not increase ! Bufferize each route during $\tau - (d_i + d_n \mod \tau)$. All routes have the same remainder mod τ, can assume they are of size one. A bit more complicated on the general graphs, proofs on the depth of the graph. Should take into account the length of the graph.

IV. Above a load of 1/2

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 5. *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 6. *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 7. *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 6, 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 5 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 8. *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 9. *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 10. *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.

VI. Non greedy algorithm

Greedy + swap one element if necessary. It can be used as the second step of the algorithm using potential. One can show that we use only the free routes and we swap if necessary. We show that, if the potential is at least $2\epsilon Pn$, then at least one of any $P-n$ elements has positive potential and can be moved. Such an operation lose at most $2n$ of potential. As a consequence, placing the last elements lose $2n\epsilon P$ in potential. Hence if the potential is more than $4\epsilon Pn$, the algorithms terminates.

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.