Greedy algorithms for scheduling periodic message

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 is then to propose a sending scheme from the antennas to their processing units and back without contention and buffer.**

We study both general networks and a simple but common star shaped topology, where all contentions are on a single arc shared by all antennas. For packet of arbitrary size, we show that there is always a solution as soon as the load of the network is less than $1/2$. Moreover, we explain how we can restrict our study to packet of size 1 without increasing the global latency.

For packet 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.

I. Introduction

Lister les usages possibles: sans retour mais de profondeur 2 ou plus. Sonar ? Train ? Réseau de capteurs communicant (industrie)

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.

II. Model

Pas la peine de parler d'aller et de retour dans le modèle. Les sommets représentent les point de contention et les arêtes le temps pour aller au prochain point de contention.

We model the network by a directed acyclic graph $G = (V,A)$. The nodes of indegree zero are the sources of the data (the antenna) and the nodes of outdegree zero are the destination of the data (the data centers). There is a labeling Ω from V to $\mathbb{N}$, such that for an arc $e = (u,v)$, $\Omega(e)$ represents the time a unit of data needs to go from u to v.

Routes, routed network.

The period is an integer denoted by P. We use the notation $[P]$ for the set $\{0,\dots,P-1\}$.

Message, message size. Use of the ressources at each contention point.

Route assignment.

Collision.

(Conflict free) Periodic assignment.

Definition of the problem: Periodic routes assignment PRA. Dire que les arcs des sommets sources vers le premier sommet de contention ne changent rien par rapport à PRA et donc on peut les supposer à 0.

Notion of depth: depth 1 is trivial, complexity of depth 2 already NP hard.

+ the simplified version for the star with a picture + a set of number $d_1,\dots,d_n$ and two values P,τ. Notion of load. [1]

III. Greedy Algorithms for large τ

Notion of partial solution: a subset of the routes with their offsets fixed

All algorithms build the solution incrementally, by fixing

are greedy and most can work online. Once a route is placed, it is not changed, hence in a partial solution we are only interested in the set of positions used in the backward and forward period. If n tasks have been placed, we denote them by $x_1,\dots,x_n$ and $y_1,\dots,y_n$.

A. Star shaped network

Présenter la structure, version la plus simple d'un graphe de profondeur 2. Donnée comme un ensemble d'entiers $d_1,\dots,d_n$ qui correspondent au poids pour chaque route de l'arc entre les deux points de contention. Étudié dans notre papier avec plein d'autre problèmes similaires ... Dire que c'est une variante périodique du 2 flow shop problem.

We present three greedy algorithms which try to find assignments for a star shaped network. We show that they *always* find an assignment when the load is small enough.

The first one deal with the route in any order, and for each route it tests all offsets from 0 to P until one do not create a collision. 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), that is the packet of a route 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* PRA *positively on a star shaped network of load less than* $1/3$.

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

The second method is described in ..... 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 Fit, since it works as first fit, but using meta-intervals in the first contention node.

Theorem 2 (Citer)**.** *Meta Fit always solves* PRA *positively on a star shaped network of load less than* $1/3$*.*

Finally, a refinment of the previous method allows to obtain solutions a soon as the load is less than $1/2$. It is as good as the First Fit algorithm in the simpler case of $\tau = 1$. The routes are sorted according to $d_i \mod \tau$. The we use Meta Fit on the instance, we call this variant Ordered Meta Fit.

Theorem 3. *Ordered Meta Fit always solves* PRA *positively on a star shaped network of load less than* $1/2$*.*

Proof. □

B. General graph

Algo first fit en général. Si on utilise la coherent routing property. :$1/4$ for any τ and P, and any DAG of depth k.

C. 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.

IV. 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.

V. 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 4. *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 5. *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 6. *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 5, 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 4 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 7. *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.

VI. 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 8. *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 $n_t h$ steps assuming it has not failed before.

Theorem 9. *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.

VII. 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.

VIII. 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] D. Barth, M. Guiraud, B. Leclerc, O. Marce, and Y. Strozecki, "Deterministic scheduling of periodic messages for cloud RAN," in *2018 25th International Conference on Telecommunications (ICT) (ICT 2018)*, (Saint Malo, France), June 2018.