Scheduling algorithm to avoid contention in meshed networks

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

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

May 15, 2020

Introduction

1 Model & Problems

1.1 Model

We use the notation $[n]$ to denote the interval of n integers $\{0, \ldots, n-1\}$.

1.1.1 Discrete time model

In the model presented here, the time is discrete. The unit of time is called a **tic**. This is the time needed to transmit an atomic data over a link of the network. We consider that the speed of the links is the same over all the links of the network. In this paper, the size of an atomic data is 64B, the speed of the links is considered to be 10Gbps, TODO: rajouter durée d'un tic.

1.1.2 Network modeling

We study a communication network in which some pairs of source-destination nodes between which some messages are sent. Also, the routing between each such a pair of nodes is given. The network is modeled as a directed acyclic multigraph $G = (V, A)$. The set of vertices is composed of three non intersecting subsets: a subset $\mathcal{S}$ of vertices modeling the sources of the messages, a subset $\mathcal{D}$ modeling the destination of the messages, and a subset $\mathcal{C}$ representing the contention points. Indeed, some links of the network are shared between several pairs of nodes source-destination, we represent the begining of this link (i.e. the pyhsical node which send the messages into this link) as a contention point. An arc in G can represent several physical links or nodes, which do not induce

contention points. Each arc (u, v) in A is labeled by an integer weight $\Omega(u, v)$ which represents the number of tics elapsed between the sending tic of the message in u and the reception tic of this message in v using this arc.
A **route** r in G is a directed path, that is, a sequence of adjacent vertices $u_1, \ldots, u_l$, with $(u_i, u_{i+1}) \in A$. The **weight of a vertex** u_i in a route $r = (u_1, \ldots, u_l)$ is defined by $\lambda(u_i, r) = \sum_{1 \leq j < i} \Omega(u_j, u_{j+1})$. It is the number of tics needed between the sending tic of a message in the first vertex of the route and the reception tic of this message at u_i. We also define $\lambda(u_1, r) = 0$. The weight of the route r is defined by $\lambda(r) = \lambda(u_l, r)$. 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 **contention level** of a node $u_i \in \mathcal{C}$ on a route r is defined by $c(u_i, r) = i-1$. This is the id of contention point on the route. The contention level of a node u is $cl(u) = \max_{r \in \mathcal{R}} cl(u, r)$, with n the number of routes using the node u. TODO: petit dessin avec un reseau et sa représentation en graph The **contention depth** of a routed network $(G, \mathcal{R})$ is denoted CD and is equal to the maximum of the contention level of its vertices. It correspond to the number of contention point on the longest route of the network.

1.2 Messages dynamic

In the process we study, a **datagrams** is sent on each route from the source node. The size of a datagram is an integer, and correspond to the number of tics needed by a node u to emit the datagram through a link. 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 datagram sent by a source node into the network is the same, regardless of the route. Once a datagram has been emitted, it cannot be fragmented during its travel in the network. We consider that the datagrams sent on each routes are all available to be sent at the same tic, which can be simplified, w.l.o.g. by 0.

Let $r = (u_0, \ldots, u_l)$ be a route. In order to avoid the contention, it is possible to buffer the datagrams in the contention points. The time a datagram on the route r waits in buffer of a vertex u_i is defined by $A(r, u_i), i \in \{0, \ldots, l\}$. The function A, called **assingment** , associate an integer value, lower or equal to 0, to each couple route-vertex of the routed network $(G, \mathcal{R})$. Those integers represent the buffering time of the datagram in each nodes of the route.

The **reception date** of a datagram in vertex u_i in r, is the first tic at which the datagram sent on r reaches u_i, and is defined by $t(r, u_i) = \lambda(u_i, r) + \sum_{k=0}^{i-1} A(r, u_k)$. In other words, the date at which a datagram reach a vertex u_i correspond to the physical delay of the links, plus the buffers determined by the logical solution proposed.
sending date of a datagram by vertex u_i in r, is the first tic at which the datagram is sent by u_i. It is defined by $s(r, u_i) = t(r, u_i) + A(r, u_i)$. This is the reception date of the datgram plus the buffer defined by the assignment for the

datagram.

Let v_r be the last vertex of the route r. We define the **full transmission time** of an assingment A as $TR(A) = \max_{r \in \mathcal{R}} t(r, v_r)$. This is the time ellapsed before the reception of the first tic of the last datagram. In the application we study the full transmission time is bounded . We define $Tmax$ such that $TR(A) \leq Tmax$.

1.3 Cloud-RAN context

TODO: completement ré ecrire cette section en disant qu'on peut ou non décomposer la BBU, et que le graph est symetrique car c'est full duplex mais que ce n'est pas obligé

1.4 Periodic sending of the datagrams

The process we describe here is **periodic**: during each period of P slots, a datagram is sent from each source node in the network at the same tic in the period. There is an infinite number of datagram sent during each period. Thus, we chose to study the behavior of the datagrams on each nodes of the network during one period, and to apply the pattern for every subsequent periods. Let us call $[t(r,u)]_{P,\tau}$ the set of time slots used a datagram on the by route r at vertex u in a period P, that is $[t(r,u)]_{P,\tau} = \{t(r,u) + i \mod P \mid 0 \leq i < \tau\}$.

Let us consider two routes numbered 1 and 2. We say that the two routes have a **collision** at the contention point u if $[t(1,u)]_{P,\tau} \cap [t(2,u)]_{P,\tau} \neq \emptyset$.

An assignment A of a routed network $(G, \mathcal{R})$ is said **valid** if *no pair of routes has a collision*. This notion of valid assignment depends of the period P and the size of the messages τ but to simplify the reading, we do not imply those paremeters in the notations.

1.5 Problems

Synchronized Periodic Assignment for Low Latency (SPALL)
Input: A symmetric routed network $(G, \mathcal{R})$, the period P, $Tmax$.
Question: does there exist a valid assignment A of $(G, \mathcal{R})$ such that $TR(A) \leq Tmax$?
Optimisation problem: Minimizing $TR(A)$.

TODO: np completude, meme sur l'etoile, parler du papier avec PALL

2 Compact representation of an assignment

We explain in this section how to represent an assignment for the problem SPALL in a compact way. Not all assigments can be represented in a compact way, but there is at least one optimal valid assignment A for SPALL that is,

that minimize $TR(A)$. It allows to design FPT algorithm by going through all compact representations. We obtain good heuristic algorithms using taboo search or simulated annealing, since one can easily define the neighborood of a compact representation.

Definition 1. *A compact representation of an assignment A over a contention point u routes is a pair (O_u, S_u), where O_u is an order on the routes and S_u is a subset of the routes.*
A compact representation of an assignment A on routed network is thus $CR(A) = \{(O_u, S_u)\}, u \in V$.

We now explain how to decompress a compact representation into an assignment. We only consider one contention point u, thus, we simplify the notation of the functions $r(r, u)$ and, $s(r, u)$ by respectively $b(r)$, $r(r)$ and $s(r)$. As a reminder those functions gives respectively the reception date and the sendinc date of the datagram of the route r in the contention point u. Also, we simplify the notation of the compact representation on the contention point u by (O, S), and the buffering time of route r on the contention point given by $A(r, u)$ is denoted here $A(r)$.

Say w.l.o.g. that the datagrams are indexed in the order given by O. We fix the sending date of the first datagram $s(1) = r(1)$, that is $A(1) = 0$, the datagram does not wait in a buffer. Then, in each period beginning by the first datagram, the datagrams will be in order. To simplify, we assume that $s(1) = r(1) = 0$, which can be obtained by removing $r(1)$ to all arrival times. We fix the sending date of the datagram in order, when the first i datagrams have their sending date computed, we fix $s(i + 1)$ in the following way.

If $i+1 \notin S$, then $s(i+1) >= r(i+1)$ otherwise the datagram $i+1$ should go in the period after the one it is available in, that is $s(i+i) > (r(i+1)/P+1)P$. The value of $s(i+1)$ is the smallest value which satisfies the previous constraint, ensures that there are no collision with the first i datagrams and satisfies the order, that is $s(i + 1) \mod P > s(i) \mod P$. It is possible that the process fails to find a correct value for $s(i)$ at some point, in that case there are no assignment associated to this compact representation.

We denote this transformation by Sol, that is $Sol(O, S)$ is the solution previously defined (the routed network is implicit) or a special value to denote there is no assignment compatible with this compact representation.

We can also define an inverse function which from most assignment A computes a compact representation, that we already defined in def. 1 by $CR(A)$. The function is defined only for assignments A, such that the first datagram does not wait in a buffer, that is $s(1) = r(1)$. Assume w.l.o.g that $r(1) = s(1) = 0$, by considering the equivalent problem where $r(i) = r(i)-r(1)$ and $s(i) = s(i)-r(1)$. Compute the values $(s(i)) \mod P$ and call their order O. Let S be the set of i such that $(s(i) \geq (r(i)/P + 1)P$. We let $CR(A) = (O, S)$.

TODO: donner un petit exemple en dessin, + parler de la complexité du changement de forme A compact representation of a solution for an instance of depth larger than k is a list of compact representations, one for each contention

arc. The following theorem explains why it is enough to explore the compact representations to solve SPALL.

Theorem 1. *Among all assignments A for a routed network $G, \mathcal{R})$, there is a compact representation which minimizes $TR(A)$.*

Proof. Consider an assingment A and a vertex u. If $A(r,u) > 0, \forall r \in \mathcal{R}$, it is possible to compute the assignment A' such that $A'(r,u) = A(r,u) - \min_{i \in \mathcal{R}} A(i,u)$.
Consider a vertex v such that $cl(v) = cl(u) + 1$, or such that $v \in \mathcal{D}$ that is, v comes after u in any routes shared by both vertices.
For all routes r passing through v, by definition, and since $A'(r,u) \leq A(r,u)$:

$$t'(r,v) = \lambda(v,r) + \sum_{u \in r, cl(u,r) < cl(v,r)} A'(r,u) \leq t(r,v)$$

Note that by reducing the buffers in a node u_i, we allow a datagram to reach the node u_{i+1} earlier. Nevertheless, we do not change the order of the datagrams in u_{i+1}, even if it is possible. By induction, if v is the last vertex of the route r, then :

$$t'(r,v) \leq t(r,v) \rightarrow TR(A') \leq TR(A)$$

Thus, for all existing assignments A solving SPALL, it is possible to find an equivalent assignment A' which have compact representation $CR(A')$ and such that $TR(A') \leq TR(A)$.

□

We want to find the assignment for which $TR(A)$ is minimal. To do so, as explained in theorem 1, we search between all the compact representations the one which minimize $TR(A)$.

Lemma 2. *The number of compact representation (O,S) for a contention point with k routes is $k!2^k$.*

From lemma 2, one can determine the numbre of compact representation of an entire router network. We define $k_1, \ldots, k_l$ the number of routes on the l contention points of a routed network.

Lemma 3. *The number of compact representation for a routed network $(G, \mathcal{R})$ is $\prod_{i=1}^{l} k_i!2^{k_i}$.*

We consider a routed network $(G, \mathcal{R})$. As a reminder, CD is the contention depth of the routed network, and there is n routes in $\mathcal{R}$.

Theorem 4. *Problem* SPALL *is FPT when parametrized by the number of routes and can be solved in time $O((n!2^n)^{CD})$.*

Proof. TODO: idée : faire Contention level par contention level, sur un level on a qu'une seule fois la route et $\prod 2^k < 2^n, \prod k! < n!$. De plus, il faut rajouter le facteur multiplicatif pour passer d'un compact assignment a l'autre et decompser le compact en sol □

3 Find the best assignment

3.1 Greedy

Greedy sans buffers dans le graph -> greedy avec buffers dans le graph.

3.2 Local Search

There is a large number of compact representation and to find the one which minimize $TR(A)$, we try serveral local search technique like ill climbing, Simulated annealing or Tabu search.

As a reminder, on a contention point u of the graph, the pair (O_u, S_u) of an order O_u an a subset S_u of the routes of the contention point represent the compact representation of an assignment A for u. Also, remember that the compact representation of A on routed network is detoned $CR(A)$. We define $O(r)$ the position of the route r in an order O, and the inverse fuction $O^{-1}(r)$. We consider a compact represtation (O_u, S_u) on a contention point u. A *neighbor order* of O over a route r, is an order O' such that $O'(r) = O(r) + 1$. This means that $O'(j) = O(j), \forall j \notin \{r; O^{-1}(r) + 1\}$. Indeed, by definition, $O'(O^{-1}(O(r) + 1)) = O(r)$. If r is the last route in the order O, then $O'(r) = 1$ and $O'(O^{-1}(O(1)) = O(r)$ A *neigbhor subset* of S over a route r is defined by $S' = S - \{r\}$ if $r \in S$ or $S' = S \cup \{r\}$ if $r \notin S$. The **neighborhood of the pair** (O_u, S_u) ouver a route r is defined by the set $N_u = \{(O_u, S_u); (O_u, S'_u); (O'_u, S_u); (O'_u, S'_u)\}$.

Consider a compact representation $CR(A)$ over a route r. $CR'(A)$ si a **neighbor** of $CR(A)$ if for all pair $(O_u, S_u)'$ of $CR'(A)$:

- If $u \notin r$, $(O_u, S_u)' = (O_u, S_u)$
- If $u \in r$, $(O_u, S_u)' \in N_u$

The **neighborhood of a compact representation** CR **over a route** r is thus the set of compact representations: $\prod_{u \in r} N_u$.

The **neighborhood of a compact representation** CR is then the set of the compact representations over all routes, that is:

$$\bigcup_{r \in \mathcal{R}} (\prod_{u \in r} N_u)$$

.

Consider that the routes are numeroted $1, \ldots, n$ and belongs to respectively $k_1, \ldots, k_n$ contention arcs, a compact representation has $\sum_{i=1}^{n} 2^{2k_i}$ neighbors.

3.3 FPT

We define a **partial compact represetation** Pcr of a routed network (G, R) as a choice of pairs (O, S) on a subset of the arcs of the graph. The function $Bound(Pcr)$ gives a lower bound of $TR(G, \mathcal{R})$, considering Pcr.

As explained on theroem 1 there is a compact representation which minimizes $TR(G, \mathcal{R})$. We browse the entire set of compact representation to find the one which minimizes $TR(G, \mathcal{R})$.
To do so, we design the following branch and bound algorithm. First, we determine an upper bound for our branch and bound algorithm by computing $TR(G, \mathcal{R})$ with a greed algorithm. Then, for all arcs (sorted by contention level), we enumerate all the possible compact representations. For each of those compact representation, we compute $Bound(Pcr)$. If the result given by this function is higher than the upper bound already found, we do no consider this compact representation. Otherwise, we go to the next arc. Once we have chose an pair (O, S) for all arcs, we compute $Sol(O, S)$ for all arcs, and if the given $TR(G, \mathcal{R})$ is lower than the previous upper bound we update this value for the rest of the search.

In order to speed up the search, we made the following cuts:
définition des différentes coupes.

4 Results

4.1 Génération des instances

Dans le cas d'application C-RAN, on prends un nombre de routes limitées ($<$20) et des tailles de routes petites. Néanmoins, on regarde des instances pour lesquelles le problème est difficile.

Définition du type de graph regardés, expliquer que les instances sont plus dures quand les routes sont de tailles similaires et quand on augmente le nombre de points de contentions.

4.2 Détails d'implémentations des algorithmes de voisinage

Taboo : Avec une hash table pour la mémoire, ce qui rend long la recherche quand on fait beaucoup de pas
Recuit, expliquer comment j'ai réglé la température etc

FPT ?

5 Conclusion