Ordonnancement multicast pour optimiser la consommation d'énergie avec des contraintes de délai

Ismaïl Bagayoko[$BAGI14069802$]

Université du Québec à Montréal
bagayoko.ismail@courrier.uqam.ca

Abstract. Ces dernières années, le trafic mobile a considérablement augmenté. Il y a une similitude dans les demandes de données au niveau station de base. Les requêtes pour les mêmes données sont à des intervalles de juste quelques microsecondes, la station de base se trouve à faire des opérations très répétitives.
L'ancienne méthode où les requêtes sont servis une à la fois ne fonctionne plus, il n'est pas rentable en matière de ressources (ex: énergie). C'est là qu'intervient la multi-diffusion (multicast), elle consiste à servir plusieurs requêtes en un seul envoi (réponse). Il faut attendre de recevoir plusieurs requêtes pour les servir en un seul envoi. On s'aperçoit tout de suite que l'idéal c'est d'attendre infiniment et servir. Les premières requêtes vont donc beaucoup attendre pour être servis.
Nous présentons trois approches pour ordonnancer les requêtes, afin de réduire la consommation des ressources et le délai.
La multi-diffusion bien qu'existe depuis longtemps, au cours des dernières années elle a suscité beaucoup d'intérêt dû aux nombreux avantages qu'elle apporte: le coût en énergie, l'utilisation efficace du canal et bien d'autres.

Keywords: Multicast · Algorithme génétique · Algorithme glouton.

1 Introduction

Avec l'essor des smartphones et objets connectés en réseau, la demande en données est massive et croissante. La cinquième génération (5G) de systèmes de communication sans fil est la réponse pour satisfaire ces nouvelles exigences. Pour répondre aux nouvelles exigences, de nouvelles technologies doivent être introduites. D'autres technologies, qui autre fois non réalisable en raison de la puissance de calcul, doivent êtres revisité (multi-diffusion).

La multidiffusion permet de servir plusieurs utilisateurs le même contenu dans la même transmission. L'ancienne méthode (mono-diffusion) où les requêtes sont servis une à la fois ne fonctionne plus, elle n'est pas rentable en matière de ressources (ex: énergie). C'est là qu'intervient la multi-diffusion (multicast). Il faut attendre de recevoir plusieurs requêtes pour les servir en un seul envoi. On

s'aperçoit tout de suite que l'idéal c'est d'attendre le plus longtemps possible puis servir. Les premières requêtes vont beaucoup attendre pour être servis.

L'ordonnancement multidiffusion a reçu toute l'attention au cours des dernières années en raison de sa capacité à réduire l'utilisation des ressources dans la communication cellulaire.

Les auteurs de [3] présentent un schéma basé sur l'état des canaux basé sur la qualité pour une planification gloutonne de multidiffusions.

Les auteurs de [2] et [4] proposent un compromis entre l'utilisation des ressources et l'équité en introduisant de multiple seuils condition de canal par groupe.

Le reste de ce projet est organisé comme suit: D'abord, nous présenterons le modèle de réseau, suivit de la formulation du problème d'ordonnancement. Enfin, nous fournirons des résultats numériques.

2 Modèle de système

On considère un système avec une station de base (BS), ayant en mémoire un certain nombre de contenus (message m) fréquemment demandés, avec comme moyen de transmission utilisé est un système de transmission multicast.

Tous les messages disponibles au niveau de la station de base sont associés à un groupe constitué de stations mobiles (MS) demandant le message pendant un slot de temps. On considère alors que la taille de chaque message est d'une unité et qu'il faut une unité de temps pour le transmettre.

Nous utilisons une pénalité de retard, uniformément répartie et d_i est le délai associé à la requête i. Si la requête n'est pas répondu dans le délai la pénalité de retard est appliquée.

Notre ordonnanceur multicast doit prendre la décision d'envoyer ou non un message pendant un créneau horaire, afin de minimiser la consommation d'énergie sur le long terme tout en respectant un délai raisonnable sur toutes les demandes.

Chaque requête est définit par quatre composants:

- a_i est l'instant d'arriver de la requête i
- d_i est le dernier instant que peut tolérer la requête i
- m_i est l'identifiant du contenu (message) demandé par la requête i
- e_i l'énergie nécessaire pour répondre à la requête i

L'énergie pour un groupe émetteur à t est la même que l'énergie de la requête qui a la mauvaise condition de canal (Plus grande énergie) à t. La consommation d'énergie est définie comme dans [1]:

$$E_r(t) = \frac{C}{|h^2|} \tag{1}$$

où C est la capacité de notre station de base et h l'état du canal.

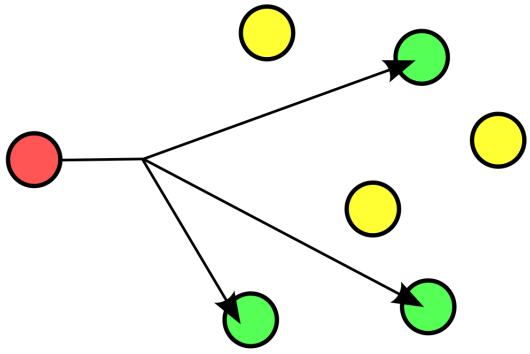

Fig. 1. Source: Wikipedia https://en.wikipedia.org/wiki/Multicast [visité en avril, 2020]

3 Formulation du problème

Nous considérons un ensemble R de requêtes défini comme $R = \{(a, d, m, e)_i\}$

On définit les variables de décision:

- x_i est l'instant d'envoi de la requête i à déterminer
- y_m est l'instant d'envoi du message m à déterminer
- E_t représente l'énergie consommée a un instant t
- E_t^m représente l'énergie consommée a un instant t pour le message m

Nous considérons les deux cas suivant lorsqu'un seul contenu où plusieurs sont disponibles pour la multidiffusion est disponible au niveau de la station de base.

Pour un message la decision d'ordonnancement est simple, il consiste a répondre a une seul question: Quand transmettre ?

$$\begin{aligned} \textbf{minimiser} \quad & \sum_t E_t \\ \textbf{Sous contraintes} \quad & \\ & E_t = \max_{x_i = t} e_i \\ & a_i \leq x_i \leq d_i \end{aligned}$$

Plus d'n message On a plusieurs questions cette fois-ci : Quel message doit être transmis? Quand le transmettre ? Il faut ajouter une condition pour que deux messages différents ne puissent pas être envoyés au même moment.

L'idée c'est de dire qu'un seul E_t^m peut être différent de zéro a la fois.

$$\textbf{minimiser} \quad \sum_m \sum_t E_t^m$$

$$\textbf{Sous contraintes}$$

$$a_i \leq x_i \leq d_i$$

$$E_t^m = \max_{x_i=t \textbf{ et } m_i=m} e_i$$

$$\sum_m 1_{E_t^m \geq 0} \leq 1 \textbf{ pour l'ensemble des messages}$$

4 Solutions proposés

Nous présentons trois approches pour ordonnancer les requêtes, afin de réduire la consommation ressources et le délai.

4.1 Une approches naïve

Elle consiste à vérifier toutes les solutions possibles et d'en choisir une(la meilleure). Avec l'ensemble des requêtes, nous générons tous les regroupements possibles. Ceci est équivalent à générer toutes les partitions d'un ensemble. le nombre de partition d'un ensemble S est $2^{|S|}$. Le pseudo-code suivant représente les principales étapes de l'algorithme.

Algorithm 1 Naïve

```
1: P ← Toutes les partitions de R
2: for p ∈ P do                                   ▷ Pour chaque partition
3:     Verifier si la partition respecte les contrainte
4:     Calculer la consommaion d'energie et le delai
5: Choisir la partition qui consomme le moins d'énergie
```

4.2 Une méthode gloutonne

Pour l'algorithme glouton la prise de décisions est locale. Il n'ordonnance pas avec anticipation des répercussions d'une décision.

Notre algorithme glouton récupère la requête qui coûte le plus en transmission. À cette requête il essaie d'ajouter toutes les requêtes qu'il peut en respectant les différentes contraintes. l'algorithme fait cela jusqu'à ce qu'il n'y ait plus de requête à laquelle répondre. *La demande sélectionnée ayant la plus grande consommation d'énergie, la consommation du groupe sera la même.*

Nous observons une complexité de $O(|R|^2)$

Algorithm 2 Glouton

```
1: while R ≠ ∅ do
2:     G ← ∅
3:     S ← requête qui demande la plus grande énergie
4:     R ← R − {S}
5:     G ← G ∪ {S}
6:     for r ∈ R do
7:         if r peut être ajouté à G then
8:             R ← R − {r}
9:             G ← G ∪ {r}
       Planifier G pour temps de S
```

4.3 Un algorithme génétique

Pour mettre en place l'approche génétique nous avons plusieurs prérequis:

Un individu de la population : Comment représenter un individu de la population? Nous utilisons un système d'indices (ou intervalles) pour représenter une partition. Un individu est une représentation d'une partition.

Exemple : Soit l'ensemble $\{1, 2, 3, 4\}$ Les indices de partitions possibles sont : $[[0, \textbf{Fin}], [0, 1, \textbf{Fin}], [0, 1, 2, \textbf{Fin}], [0, 1, 2, 3, \textbf{Fin}], [0, 1, 3, \textbf{Fin}], [0, 2, \textbf{Fin}], [0, 2, 3, \textbf{Fin}], [0, 3, \textbf{Fin}]]$
L'individu [0, 1, 2, **Fin**] permets de générer la partition : $\{\{0\}, \{1\}, \{2, 3\}\}$

- de l'indice 0 à 1 forme un sous ensemble : $\{0\}$
- de l'indice 1 à 2 forme un sous ensemble : $\{1\}$
- de l'indice 2 à **Fin** forme un sous ensemble : $\{3, 4\}$

Table 1. D'autres examples de partition de indices

Indices (Individu)	Partition
$[0, \textbf{Fin}]$	$\{\{0, 1, 2, 3\}\}$
$[0, 1, 2, 3, \textbf{Fin}]$	$\{\{0\}, \{1\}, \{2\}, \{3\}\}$
$[0, 1, \textbf{Fin}]$	{{0}, {1, 2, 3}}

Le passage à la nouvelle génération: Ici nous avons considéré que la mutation comme moyen d'évolution de la population. En le principe un sous-ensemble de la partition est choisit afin d'être agrandit ou réduit.

Les répercussions que peut avoir cette operation est la fusion de deux sous ensemble ou la division d'un sous ensemble en deux.

Algorithm 3 Génétique

```
P ← générer une population
S ← meilleur individu de notre population P
while i < Nombre maximum d'iterations do
    for Individu dans P do
        Fils ←NouvelleGeneration(Individu)
        Individu ← Fils
        if Fils.Energie ¡ S.score then
            S ← Fils
return S
```

Algorithm 4 NouvelleGeneration

```
sélectionner au hasard un sous-ensemble
Fils1 ← le sous-ensemble agrendit
Fils2 ← le sous-ensemble réduit
Fils3 ← augmenté le nombre de sous ensemble
Calculer les scores
return Le fils avec le meilleur score
```

5 Résultats numériques

Pour évaluer les performances de nos algorithmes, nous avons estimé la consommation totale d'énergie en fonction du trafic au niveau de notre station de base dans un premier temps. Le délai des utilisateurs est aussi mesuré. Dans notre simulation, nous considérons que le gain du canal est constant et suit une distribution uniforme de [1]. Nous considérons également les valeurs d'initialisation de [1] pour le bruit τ comme une unité et la puissance ρ comme $1watt$. Pour le débit de données, nous utilisons $1Kb/ms/1Mhz$. La simulation est faite sur une durée de T =20. Le délai d d'une requête est généré entre [1, 5].

5.1 Consommation d'énergie vs Charge du système

Fig. 2 et 3 représentent la consommation d'énergie sur la charge du système (nombre de requêtes par unité de temps) avec 1 et 4 messages. Au tout début, nos méthodes ont des consommations similaire parce que les trois font de la mono diffusion (unicast). Le nombre de requêtes n'est pas assez nombreux pour du multicast. L'algorithme génétique n'est pas stable, ceci est dû au caractère aléatoire dans sa pratique.

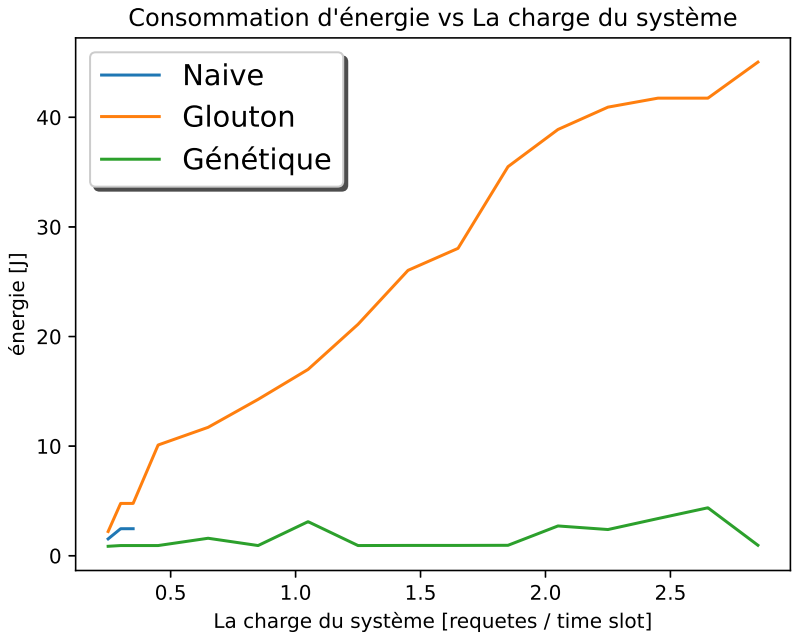

Fig. 2. Consommation d'énergie vs La charge du système, pour le nombre de contenu disponible à la station de base $m = 1$

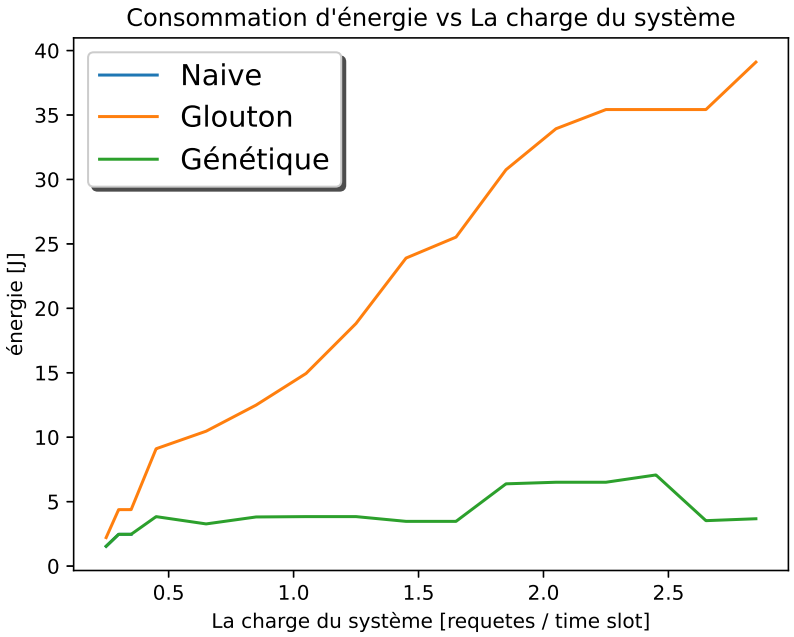

Fig. 3. Consommation d'énergie vs La charge du système, pour le nombre de contenu disponible à la station de base $m = 4$

Nous avons utilisé une différente fonction pour évaluer la consommation d'énergie pour l'algorithme génétique. La contrainte qui dit qu'un seul message doit être envoyé à la fois n'est pas respecté a chaque fois. On accepte cette exception mais avec des pénalités. on peut observer les compromis sur Fig. 4 et 5

5.2 Pénalité délai vs Charge du système

Sur les fig. 4 et 5 on peut observer les pénalités de retard pour la solution génétique. Ce qui offre une qualité de service assez mauvaise aux utilisateurs. Même si le coût d'opération est peu coûteux l'algorithme glouton réduit la consommation des ressource avec toujours le délai satisfait. La satisfaction du délai est incontournable

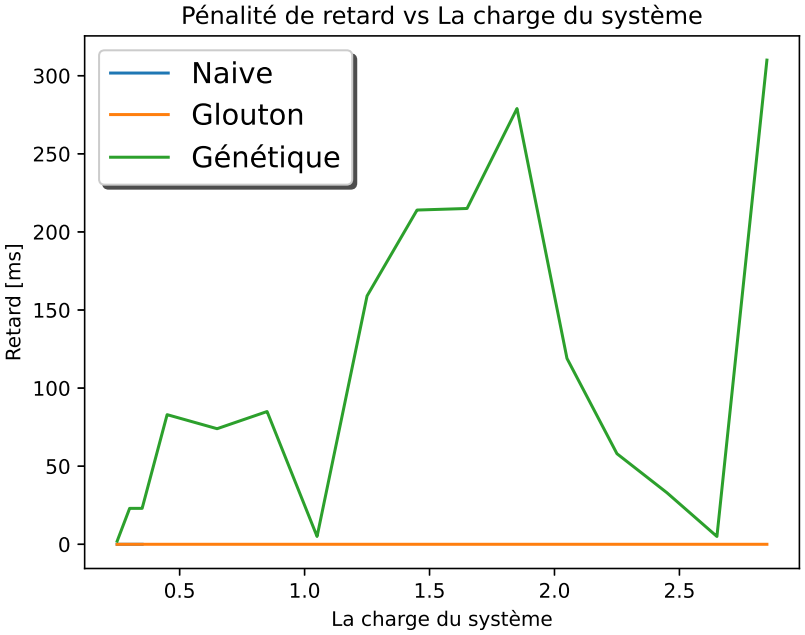

Fig. 4. Pénalité de retard vs La charge du système, pour le nombre de contenu disponible à la station de base $m = 1$

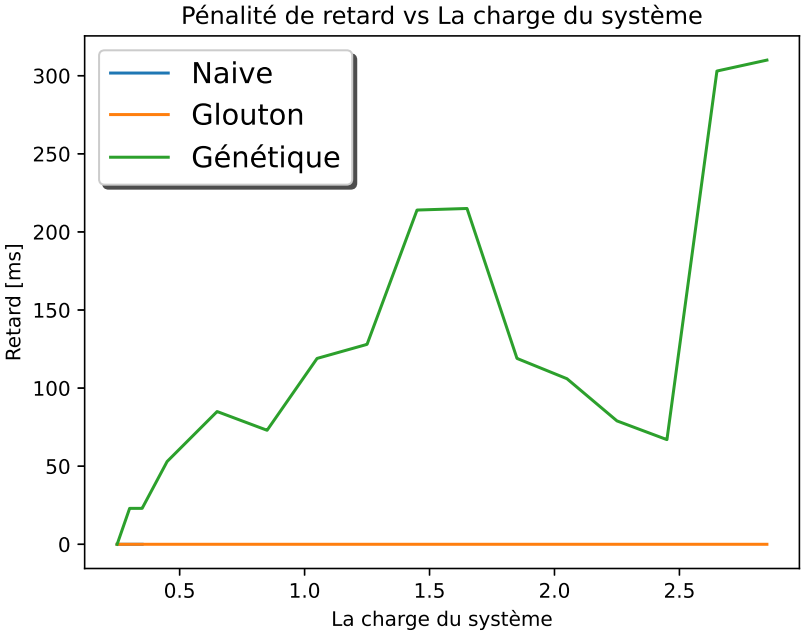

Fig. 5. Pénalité de retard vs La charge du système, pour le nombre de contenu disponible à la station de base $m = 4$

Conclusion

Après avoir proposé une solution naïve nous avons présenté deux solutions de faible complexités pour l'ordonnancement de multicast de plusieurs messages. Les deux solutions ont chacune leur point forts. L'algorithme glouton respecte toujours le délai même si la consommation des ressources n'est pas la meilleure.

L'algorithme génétique réduit à consommation des ressources mais les utilisateurs doivent attendre pour recevoir leurs données.

Nous avons présenté également des résultats numériques pour la consommation d'énergie de la station de base dans différents scénarios. Ils montrent comment la station de base se comporte lorsque la charge de trafic change.

References

1. Huang, C., al.: Delay-energy tradeoff in multicast scheduling for green cellular systems. IEEE Journal on Selected Areas in Communications **34**(5), 1235 – 1249 (2016)
2. Lee, I.H., Jung, H.: Capacity and fairness of quality-based channel state feedback scheme for wireless multicast systems in non-identical fading channels. IEEE COMMUNICATIONS LETTERS **6**(4), 1430 – 1433 (2017)

3. Lee, I.H., Kwon, S.C.: Performance analysis of quality-based channel state feedback scheme for wireless multicast systems with greedy scheduling. IEEE COMMUNICATIONS LETTERS **9**(8), 1430 – 1433 (2015)
4. Li, H., Huang, X.: Multicast systems with fair scheduling in non-identically distributed fading channels. IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY **66**(10), 8835 – 8844 (2017)