M1 Informatique 2025/2026 - Algorithmes distribués
Devoir à la maison

Détection de la terminaison d'un calcul distribué

Dans cet exercice vous allez étudier l'algorithme dû à D. Kumar [1] pour la détection de la terminaison d'un calcul distribué.

N nœuds (appelés *agents*) d'un système distribué effectuent un calcul : chaque agent peut être actif ou inactif. Un agent actif peut devenir inactif à tout moment ou alors il peut envoyer un message `'m'` à un autre agent (que ce dernier soit actif ou pas). À la réception d'un message `'m'` un agent redevient actif (ou reste actif s'il l'est déjà). Le système a terminé globalement si tous les agents sont inactifs et s'il n'y a plus de message en transit entre deux nœuds. (Clairement, si c'est le cas le système d'agents reste terminé à toujours.) L'objectif de l'algorithme de Kumar est de détecter la terminaison globale (on ne suppose pas que la terminaison va forcément intervenir dans toutes les exécutions du système).

Un nœud supplémentaire appelé *moniteur* est chargé de détecter la terminaison globale. Chaque agent p maintient des compteurs $aSent[q]$ et $aRcvd[q]$ (pour chaque agent q) qui reflètent les nombres de messages envoyés à (voire reçus de) l'agent q. Quand p termine localement, il envoie un message `'t'`, accompagné de ses compteurs *aSent* et *aRcvd*, au moniteur. Le moniteur maintient des tableaux $mSent[p][q]$ et $mRcvd[p][q]$ des nombres de messages qu'il sait envoyés de p à q (et reçus par p de q). Quand le moniteur a reçu au moins un message `'t'` de chaque agent p et que pour tous agents p, q, $mSent[p][q] = mRcvd[q][p]$, il déclare que la terminaison globale a été atteinte.

Une description en pseudo-code est donnée à la figure 1. On suppose que les canaux entre chaque agent et le moniteur respectent la politique FIFO, c'est à dire que les messages arrivent dans l'ordre de leur envoi.

Question 1. En partant de la version incomplète de l'algorithme que vous trouverez sur Arche et *après avoir lu les conseils plus loin*, implantez l'algorithme de Kumar en DistAlgo. Faites exécuter votre programme plusieurs fois et assurez-vous que le programme ne déclare la terminaison du calcul que si tous les processus sont inactifs et tous les messages ont été reçus.

NB : Selon le choix aléatoire d'exécuter des transitions spontanées ou non, la terminaison peut intervenir plus ou moins tard, et elle n'est même pas garantie.

Question 2. Dans ce qui suit on notera $p.var$ la valeur de la variable locale *var* de l'agent p. Aussi on définit

$$ds[p][q] = \begin{cases} s[q] & \text{pour le dernier message (`'t'`, } s, r) \text{ en cours de transmission} \\ & \text{de } p \text{ au moniteur si un tel message existe} \\ mSent[p][q] & \text{sinon} \end{cases}$$

$$dr[p][q] = \begin{cases} r[q] & \text{pour le dernier message (`'t'`, } s, r) \text{ en cours de transmission} \\ & \text{de } p \text{ au moniteur si un tel message existe} \\ mRcvd[p][q] & \text{sinon} \end{cases}$$

et on écrit *eseen* pour l'union de l'ensemble *seen* et les agents pour lesquels un message de la forme (`'t'`, s, r) est en transit.

1. D. Kumar : A class of termination detection algorithms for distributed computations. 5th Conf. Foundations of Software Technology and Theoretical Computer Science, 1985.

Variables d'état de chaque agent :

active	initialement vrai
aSent[*q*]	initialement 0, pour chaque agent q
aRcvd[*q*]	initialement 0, pour chaque agent q

Transitions spontanées

send : si *active* est vrai
envoyer un message `'m'` à un agent q
incrémenter *aSent*[*q*]

terminaison locale : si *active* est vrai
active := faux
envoyer un message (`'t'`, *aSent*, *aRcvd*) au moniteur

Réception d'un message `'m'` depuis l'agent q :

active := vrai
incrémenter *aRcvd*[*q*]

Variables du moniteur

seen	ensemble des agents ayant envoyé un message `'t'`, initialement {}
mSent[*p*][*q*]	initialement 0, pour chaque pair d'agents p, q
mRcvd[*p*][*q*]	initialement 0, pour chaque pair d'agents p, q

Réception d'un message (`'t'`, s, r) depuis l'agent p :

seen := *seen* $\cup \{p\}$
copier les tableaux s et r dans les lignes *mSent*[*p*] et *mRcvd*[*p*]

Détection de la terminaison globale :

si *seen* contient tous les agents et pour tous p, q, *mSent*[*p*][*q*] = *mRcvd*[*q*][*p*],
déclarer la terminaison globale

FIGURE 1 – L'algorithme de Kumar.

1. Montrez que les conditions suivantes sont vraies le long de tout exécution de l'algorithme (pour tous agents p, q) :
   - $mSent[p][q] \leq ds[p][q] \leq p.aSent[q]$,
   - $mRcvd[p][q] \leq dr[p][q] \leq p.aRcvd[q]$,
   - $p.aSent[q]$ est égal à la somme de $q.aRcvd[p]$ et le nombre de messages `'m'` en transit entre p et q,
   - si un message (`'t'`, s, r) précède un autre message (`'t'`, s', r') dans le canal entre un agent et le moniteur, alors $s[p][q] \leq s'[p][q]$ et $r[p][q] \leq r'[p][q]$.
2. Pour un ensemble A d'agents on définit les deux prédicats suivants :
   - *Coherent*(A) ssi $ds[p][q] = dr[q][p]$ pour tous $p, q \in A$
   - *Menace*(A) ssi il existe $p \in A$ tel qu'une des conditions suivantes est vraie :
     - $p.active$ est vrai ou
     - $ds[p][q] \neq p.aSent[q]$ pour un agent q ou
     - $dr[p][q] \neq p.aRcvd[q]$ pour un agent q.

   Montrez que tout au long de l'exécution et pour tout ensemble $A \subseteq eseen$, si les conditions *Coherent*(A) et *Menace*(A) sont vraies alors il existe $p \in A$ et $q \notin A$ tel que $dr[p][q] \neq p.aRcvd[q]$.
3. Concluez que l'algorithme est correct : lorsque le moniteur déclare la terminaison, alors le système a terminé globalement.

Consignes. Vous rendrez une archive contenant le programme DistAlgo correspondant à la question 1 ainsi qu'un fichier texte ou pdf en réponse à la question 2. Le code devra être proprement commenté et contenir suffisamment de messages à imprimer sur la console pour pouvoir suivre l'exécution du programme. Si besoin, vous pouvez ajouter une page de description expliquant votre implémentation.

Vos réponses sont à rendre au plus tard le **3 février 2026** via Arche. En cas de questions, n'hésitez pas à me contacter par `stephan.merz@loria.fr`. Vous pouvez travailler en binôme, dans ce cas n'oubliez pas d'indiquer les deux noms.

Conseils.

1. DistAlgo nécessite Python 3.7 ou une version antérieure, il est donc recommandé d'installer cette version sur votre machine. Si nécessaire, adaptez le script `dar` (et éventuellement `dac`) en remplaçant la première ligne par une ligne comme

   ```
   #!/path/to/python@3.7/bin/python3.7
   ```

2. La bibliothèque `random` contient des méthodes utiles pour effectuer des choix aléatoires. En particulier,
   — `random.uniform(a,b)` tire un nombre réel compris entre `a` et `b`,
   — `random.randint(a,b)` tire un entier compris entre `a` et `b` (inclus),
   — `random.choice(l)` tire un élément arbitraire de la liste (non-vide) `l`.
3. Une fois que le moniteur déclare la terminaison, il peut envoyer un message spécifique au programme principal en utilisant une instruction du genre

   ```
   send('done', to = parent())
   ```

   et le programme principal peut ensuite envoyer un message à tous les agents pour les demander de s'arrêter et ainsi terminer l'exécution du programme DistAlgo.