Détecteur de défaillances minimal pour le consensus adapté aux réseaux inconnus

Thibault Rieutord, Luciana Arantes, Pierre Sens

Détecteur de défaillances minimal pour le consensus adapté aux réseaux inconnus

Thibault Rieutord[1,2], Luciana Arantes[2], Pierre Sens[2]

[1]*ENS Rennes, France*
[2]*Sorbonne Universités, UPMC Univ Paris 06, CNRS, Inria, LIP6 - France*

Cet article présente une implémentation asynchrone d'un nouveau détecteur de défaillances minimal pour la résolution du *consensus* dans le cas de réseaux inconnus, i.e., sans connaissance préalable sur le système. Ce détecteur ne repose pas sur des minuteurs. Nous donnons une propriété comportementale raisonnable, suffisante pour garantir son implémentation.

Keywords: Systèmes asynchrones, consensus, détecteurs de défaillances, élection de leader, quorum, systèmes distribués.

Le consensus permet de résoudre de nombreux problèmes d'accord dans les systèmes répartis en présence de fautes. C'est une brique fondamentale qui a été largement étudiée particulièrement dans les systèmes dont la topologie est connue (i.e., le nombre et la liste des participants est préalablement connue). Les détecteurs de défaillances, que l'on notera $\mathcal{FD}$, permettent de capturer les informations nécessaires et suffisantes sur les fautes pour résoudre le consensus. Par exemple, dans un système asynchrone (i.e., sans borne sur les délais de transmission) initialement connu avec une majorité de processus corrects, il a été montré que le détecteur Ω, qui retourne ultimement un chef correct, permet de résoudre le consensus de façon déterministe [CHT96]. Avec un nombre de fautes quelconque, Ω associé au détecteur quorum Σ est le détecteur minimum pour résoudre le consensus [DGFG10].

Dans des systèmes plus dynamiques tels que les nuages, les réseaux de capteurs sans fils ou les réseaux ad hoc, les hypothèses classiques d'une connaissance préalable de la composition du système, ou même seulement du nombre de participants, ne sont plus possibles.

Nous proposons donc une adaptation du détecteur de défaillances minimal pour le *consensus* adapté aux réseaux inconnus. Après avoir montré que des modèles usuels de synchronie partielle ne permettent pas d'implémenter ce détecteur, nous proposons une implémentation totalement asynchrone du détecteur reposant sur une propriété comportementale du système.

1 Modélisation du problème

Modèle de système. On considère un système distribué asynchrone à passage de messages. L'ensemble des processus, dénoté Π, est de taille finie mais sa taille et sa composition sont inconnues des processus. Le modèle de panne franche est utilisé, les processus se comportent selon les spécifications jusqu'au moment où éventuellement ils s'arrêtent. Les canaux de communication sont fiables, les messages ne peuvent être ni altérés, ni perdus, ni dupliqués. Le réseau de communication n'est pas nécessairement complet mais permet la communication entre chaque couple de processus quelles que soient les pannes qui se sont produites. Un processus est dit *correct* si il ne tombe pas en panne lors d'une exécution donnée, sinon il est dit fautif. L'ensemble des processus *corrects* est noté *correct(F)* où F correspond au motif des fautes d'une exécution particulière. $\mathcal{T}$ correspond à un domaine temporel pouvant être discret, tel que $\mathbb{N}$, ou continu, tel que $\mathbb{R}^+$.

Consensus. Dans le problème du *consensus*, chaque processus propose une valeur v_p et les processus décident d'une certaine valeur en respectant les trois propriétés suivantes :

— *Terminaison* : Tout processus correct décide une valeur.
— *Validité* : Toute valeur décidée doit avoir été proposée.
— *Accord* : Une seule et unique valeur est décidée.

La *propriété d'accord* reflète la correction du protocole, la *propriété de terminaison* assure sa vivacité et la *propriété de validité* empêche l'utilisation de solutions triviales basées sur un accord au préalable.

Ω : Election de Chef. Le $\mathcal{FD}$ Ω a été introduit dans [CHT96] indirectement pour montrer la minimalité d'autres détecteurs. Il s'avère être aussi le $\mathcal{FD}$ minimal pour la résolution du *consensus* lorsqu'il y a une majorité de processus corrects et que le nombre de processus dans le système est connu des processus. Ω retourne à chaque processus p une variable $leader_p$ respectant la propriété suivante :

$$\exists \tau \in \mathcal{T}, \exists q \in correct(F), \forall p \in correct(F), \forall t \in \mathcal{T}, t \geq \tau : leader_p = q$$

Σ : $\mathcal{FD}$ à Quorum. Le détecteur de défaillances Σ a été introduit dans [DGFG10] pour trouver le détecteur minimal pour la résolution du *consensus* dans le cas général. Il est prouvé minimal pour l'émulation de registres atomiques dans un système asynchrone à passage de messages. Grâce à ce résultat, il est montré que lorsque l'on ajoute Ω à Σ (on utilisera la notation $\langle \Omega, \Sigma \rangle$), on obtient le $\mathcal{FD}$ minimal pour la résolution du *consensus* quel que soit le nombre de pannes possibles dans un système connu. Σ retourne à chaque processus p un sous-ensemble du système composé de processus de confiance, Σ_p (i.e., les processus considérés comme *corrects*). Cet ensemble vérifie les deux propriétés suivantes :

— *Intersection* : deux listes de processus contiennent toujours au moins un processus commun [†] :

$$\forall t_1, t_2 \in \mathcal{T}, \forall p_1, p_2 \in \Pi : \Sigma_{p_1}(t_1) \cap \Sigma_{p_2}(t_2) \neq \emptyset$$

— *Complétude* : de façon ultime, les listes retournées ne contiennent que des processus corrects :

$$\exists \tau \in \mathcal{T}, \forall p \in correct(F), \forall t \in \mathcal{T}, t \geq \tau : \Sigma_p(t) \subseteq correct(F)$$

2 Un nouveau détecteur de défaillances pour le *Consensus*

$\langle \Omega, \Sigma \rangle$ a été prouvé minimal pour la résolution du *consensus* dans un système asynchrone classique. Mais Σ nécessite une certaine connaissance de la composition du système pour pouvoir dès l'initialisation assurer la propriété d'intersection.

$\Sigma_\perp$: *Quorum* $\mathcal{FD}$ pour les réseaux inconnus. Nous proposons ici de modifier Σ pour qu'il conserve ses propriétés tout en s'abstrayant des contraintes relatives à l'absence de connaissance sur le système. L'idée est que ce nouveau $\mathcal{FD}$, dénoté $\Sigma_\perp$, doit se comporter comme Σ au bout d'un certain temps, mais peut se permettre d'indiquer que ce n'est pas encore le cas en retournant la valeur $\perp$.

$\Sigma_\perp$ renvoie donc à chaque processus p une valeur noté $\Sigma_{\perp p}$ pouvant être, soit une liste de processus de confiance, soit la valeur $\perp$. Les deux conditions suivantes doivent être respectées :

— *Intersection limitée par* $\perp$: deux processus retournent des listes contenant au moins un processus en commun, si aucun des deux ne retourne $\perp$ [‡] :

$$\forall t_1, t_2 \in \mathcal{T}, \forall p_1, p_2 \in \Pi, \Sigma_{\perp p_1}(t_1) \neq \perp, \Sigma_{\perp p_2}(t_2) \neq \perp : \Sigma_{\perp p_1}(t_1) \cap \Sigma_{\perp p_2}(t_2) \neq \emptyset$$

— *Complétude étendue à* $\perp$: de façon ultime et pour tous les processus *corrects*, $\perp$ ne peut être renvoyé et les listes de processus de confiance retournées ne contiennent que des processus corrects :

$$\exists \tau \in \mathcal{T}, \forall p \in correct(F), \forall t \in \mathcal{T}, t \geq \tau : \Sigma_{\perp p}(t) \neq \perp \text{ et } \Sigma_{\perp p}(t) \subseteq correct(F)$$

L'étude du *consensus* sur réseaux inconnus a déjà été conduite dans certains cas comme dans [ADGF08b]. Les auteurs proposent une implémentation de Σ en considérant une initialisation de Σ à un domaine d'identifiants englobant le système. Ils supposent en outre qu'il existe une estimation du nombre de processus comprise entre la stricte majorité et le total de processus. Il est à noter que cet article s'intéresse au cas sans pannes. Il montre que la résolution du *consensus* est impossible mais le devient avec Σ.

Théorème 1. *$\langle \Omega, \Sigma_\perp \rangle$ est le plus faible $\mathcal{FD}$ permettant la résolution du* consensus *dans un système asynchrone à passage de messages, avec un nombre quelconque de pannes franches.*

Pour prouver le théorème 1, deux aspects doivent être étudiés. Le premier point, le plus direct, est de voir si il est possible d'implémenter $\langle \Omega, \Sigma_\perp \rangle$ dans un système permettant de résoudre le *consensus*, cela se montre

†. Par convention et pour simplifier la formule il est considéré qu'un processus en panne retourne toujours Π.
‡. Par convention il est considéré qu'un processus en panne retourne toujours $\perp$.

en prouvant ce résultat pour $\langle \Omega, \Sigma \rangle$ et en remarquant qu'une implémentation de Σ en est une pour $\Sigma_\perp$ [§]. Le deuxième point est la question de la résolution du *consensus* dans un réseau inconnu avec $\langle \Omega, \Sigma_\perp \rangle$. Pour cela considérons un algorithme le résolvant avec $\langle \Omega, \Sigma \rangle$ comme dans [ADGF08a]. Σ permet de fournir une liste de processus dont les messages pourront être attendus à différentes étapes de l'algorithme de *consensus*. Avec $\Sigma_\perp$ on introduit une attente lorsque $\perp$ est retourné. Si $\perp$ n'est pas observé, le comportement et les garanties sont identiques à Σ et le *consensus* peut être résolu. Si $\perp$ est retourné à un moment donné, l'attente sera finie par la propriété de *complétude étendue à* $\perp$. Pour l'algorithme de *consensus*, une telle attente est gérée exactement comme si l'exécution avait été retardée à cause d'un message lent émis par un des processus de la liste fournie.

$\Sigma_\perp$ et la nécessité de synchronie. Dans un réseau inconnu le nombre de processus *corrects* ne peut être utiliser pour attendre un certain nombre de messages et avoir un algorithme non-bloquant. De ce fait on peut montrer que l'implémentation de $\Sigma_\perp$ et donc la résolution du *consensus* est impossible sans des canaux de communications synchrones, i.e., ayant un délai de transmission borné et de borne connue des processus.

Théorème 2. *Un chemin composé de canaux de communications synchrones est nécessaire entre chaque couple de processus, dans une des deux directions, pour implémenter* $\Sigma_\perp$ *et donc résoudre le* consensus *lorsque le réseau est inconnu.*

Ceci se montre grâce à l'impossibilité de déterminer en temps limité, dans le cas purement asynchrone, si un processus est juste lent ou si il est tombé en panne. Un processus isolé se doit de décider sans attendre indéfiniment un message afin de respecter la propriété de *terminaison* du *consensus*. Mais un processus ne peut savoir si il est isolé ou si les autres sont juste trop lents à communiquer sans garanties sur le nombre de processus correct. Les temps de calculs et de communication doivent alors être bornés et de bornes connues pour pouvoir résoudre le *consensus*. En s'intéressant à tous les couples de processus possibles on arrive à montrer cette nécessité dans une des deux directions pour au moins un chemin les reliant.

3 Implémentation de $\Sigma_\perp$: Le *quorum* gagnant

Le seul moyen d'éviter la nécessité de liens synchrones dans le système,cf. théorème 2, est de considérer qu'une borne minimale de processus *correct*s existe et est connue des processus, nous appelons α cette borne. L'implémentation de Ω peut se faire avec des hypothèses similaires [AGSS13].

Modèle à motif de messages. Proposée pour la première fois dans [MMR03], une méthode basée sur un déséquilibre dans la vitesse de réception des messages permet l'implémentation d'algorithmes non-bloquants et totalement asynchrones. Ce modèle se base sur un mécanisme de rondes de *requête-réponses* successives, où les processus envoient une *requête* à tous les processus et attendent un certain nombre de *réponses*, ici α. L'article définit aussi une propriété s'appliquant sur les canaux de communications :

Définition 1. α-canal-gagnant *: p est relié à q par un* α-canal-gagnant *si la* réponse *de p est toujours reçue parmi les* α *premières réponses à chaque requête de q.*

Il est montré dans [MMR03] que si tout processus, p, est connecté à un même processus *correct*, p_l, par un α-*canal-gagnant* de p_l vers p, alors on peut implémenter le $\mathcal{FD}\ \mathcal{S}$, strictement plus fort que $\langle \Omega, \Sigma \rangle$.

Cette propriété est imposée sur des échanges directs dans [MMR03]. Elle peut être relaxée sur des chemins. Ces chemins doivent cependant être contraints pour exclure au final les processus en panne par exemple en limitant la longueur de ces chemins à au plus un certain β [¶].

Définition 2. (α, β)-chemin-gagnant *: p est relié à q par un* (α, β)-chemin-gagnant *si pour chaque requête de q la réponse de p est reçue après une chaîne d'au plus* β *relais de* α-canal-gagnant.

Le *quorum* gagnant. Pour implémenter notre nouveau $\mathcal{FD}\ \Sigma_\perp$ il n'est pas nécessaire d'avoir un processus dont la réponse est toujours reçue à temps, l'objectif étant de fournir un ensemble de processus respectant la propriété d'intersection. On peut donc relaxer la condition d'implémentation de $\mathcal{S}$ d'un processus vers un ensemble, que l'on appellera le *quorum*, et dont les réponses d'une majorité stricte des processus du *quorum*

§. Il est à noter que cela implique que tout résultat sur des conditions suffisantes pour implémenter Σ est aussi valable pour $\Sigma_\perp$.

¶. Une contrainte de taille dynamique peut aussi être formellement décrite pour s'abstraire d'un paramètre supplémentaire β et des contraintes implicites sur le diamètre maximal du réseau lié à son choix, cela pouvant se faire par exemple en adaptant la taille au nombre de processus actifs connus ou à l'ancienneté relative de leur activité connue.

sont reçues à chaque ronde parmi les α premières, α étant la borne minimale de processus *corrects*. Les processus du quorum sont interchangeables d'une requête à l'autre et d'un processus à l'autre, tant qu'une stricte majorité est présente à chaque ronde. La propriété comportementale en résultant est la suivante :

Définition 3. (α,β)-quorum-gagnant *: Un ensemble de processus, Q_w, tel que tout processus, p, est connecté à une stricte majorité de processus de Q_w, de p vers les processus de Q_w, par des* (α,β)-chemins-gagnants. ‖

Algorithme 1: Implémentation de $\Sigma_\perp$ avec un (α,β)-*quorum-gagnant*

init : $counter_i \leftarrow 0$, $rec\_from_i \leftarrow \{(id_i,0),(\perp,0)\}$, $\Sigma_i \leftarrow \perp$;
Task T1 :
Repeat forever
  broadcast QUERY($counter_i$);
  wait until (RESPONSE(c,rec_from) with $c = counter_i$ **received from** α distinct processes);
  let I = the set of processes from which p_i received a RESPONSE message;
  let X = the set of the rec_from sets received from the processes in I;
  $rec\_from_i \leftarrow \{(ls,0) : ls \in I\}$;
  foreach $R \in X$ **do**
    foreach $(ls,age) \in R, age < \beta$ **do**
      if $(ls,\_) \notin rec\_from_i$ **then**
        $rec\_from_i \leftarrow rec\_from_i \bigcup \{(ls,age+1)\}$;
      else if $age < age', (ls,age') \in rec\_from_i$ **then**
        $rec\_from_i \leftarrow (rec\_from_i \setminus \{(ls,age')\}) \bigcup \{(ls,age+1)\}$;
  $counter_i = counter_i + 1$;
  if $(\perp,\_) \notin rec\_from_i$ **then**
    let $\Sigma_i \leftarrow \{ls : (ls,\_) \in rec\_from_i\}$;
End repeat;
task T2 :
upon reception of QUERY(c) **from** p_j **do send** RESPONSE(c,rec_from_i) **to** p_j

Si le système fourni un (α,β)-*quorum-gagnant*, alors l'algorithme 1 implémente $\Sigma_\perp$. L'algorithme envoie des requêtes périodiques et attend α *réponses*. Ce processus permet de transporter les informations d'activité des processus, limités à un age de β retransmissions à l'aide d'un compteur *age*. $\perp$ est utilisé le temps que β phases de *requêtes-réponses* se soient terminées.

Bilan. Les résultats présentés montrent que l'absence de connaissance sur la composition du système ou du nombre de participants corrects empêche l'application de méthodes connues pour la résolution du *consensus*. Alors que de fortes hypothèses de synchronie sont nécessaires en cas d'absence totale de connaissance, l'hypothèse raisonnable qu'une borne minimale de processus *corrects* soit connue permet la résolution du *consensus* à l'aide de modèles tel que celui à *motif de messages*.

Références

[ADGF08a] M. Abboud, C. Delporte-Gallet, and H. Fauconnier. Agreement and consistency without knowing the number of processes. In *NOTERE'08*, page 38. ACM, 2008.

[ADGF08b] M. Abboud, C. Delporte-Gallet, and H. Fauconnier. Agreement without knowing everybody : a first step to dynamicity. In *NOTERE'08*, page 49. ACM, 2008.

[AGSS13] L. Arantes, F. Greve, P. Sens, and V. Simon. Eventual leader election in evolving mobile networks. In *Principles of Distributed Systems*, pages 23–37. Springer, 2013.

[CHT96] T.D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. *JACM*, 43(4) :685–722, 1996.

[DGFG10] C. Delporte-Gallet, H. Fauconnier, and R. Guerraoui. Tight failure detection bounds on atomic object implementations. *JACM*, 57(4) :22, 2010.

[MMR03] A. Mostefaoui, E. Mourgaya, and M. Raynal. Asynchronous implementation of failure detectors. In *DSN'03*, pages 351–351. IEEE, 2003.

‖. De manière implicite des contraintes de tailles sont imposées aux différents paramètres : $\lfloor |Q_w|/2 \rfloor + 1 < \alpha \leq |correct(F)|$.