Connaissance vs. Synchronie pour l'Accord Tolérant aux Pannes dans les Réseaux Inconnus

Fabiola Greve, Sebastien Tixeuil

Connaissance vs. Synchronie pour la Tolérance aux Pannes dans les Réseaux Inconnus

Fabíola Greve[1†] – Sébastien Tixeuil[2‡]

[1]*DCC - Computer Science Department, Federal University of Bahia, Bahia, Brasil,* fabiola@dcc.ufba.br
[2]*Univ. Paris Sud, LRI-CNRS 8623, INRIA Grand Large, France,* tixeuil@lri.fr

Dans les réseaux auto-organisés, tels que les réseaux mobiles *ad hoc* et les réseaux pair-à-pair, le consensus est une brique fondamentale pour résoudre les problèmes d'accord. Il permet de coordonner les actions de nœuds répartis de manière *ad hoc* de telle sorte que des décisions cohérentes peuvent être prises. Il est notoire que dans les environnements classiques, où les entités se comportent de manière asynchrone et où les identités de chacun sont connues, le consensus ne peut être résolu dès qu'une panne crash est susceptible de se produire. Les systèmes auto-organisés renforcent ce résultat d'impossibilité car les identifiants des participants ne sont pas connus. Nous définissons des conditions nécessaires et suffisantes pour que le consensus puisse être résolu dans de tels environnements. Ces conditions sont liées aux hypothèses de synchronie sur l'environnement, ainsi qu'à la connectivité du graphe des connaissances induit par les nœuds qui souhaitent communiquer avec leurs pairs.

Keywords: Tolérance aux pannes, accord réparti, réseaux inconnus

1 Introduction

Les réseaux de capteurs sans fils et *ad hoc* permettent aux entités qui les constituent d'accéder à des services et informations indépendemment de leur position ou de leur vitesse. Un tel objectif peut être atteint en supprimant la nécessité d'une infrastructure conçue statiquement ou d'une autorité administrative centralisée. Il est dans la nature de ces systèmes d'être auto-organisés, d'autant que les entités peuvent rejoindre ou quitter le réseau de manière arbitraire, impliquant une forte dynamicité du système.

Les problèmes d'accord sont des briques de base fondamentales des systèmes distribués robustes, et la conception de solutions dans les systèmes dynamiques et auto-organisés est un pan très actif de la recherche récente. Le problème fondamental parmi les problèmes d'accord est celui du *consensus*. Informellement, un groupe de processus effectue un consensus de la manière suivante : chaque processus propose initialement une valeur, et tous les processus corrects (*i.e.* ceux qui n'ont pas crashé) doivent décider d'une valeur commune qui doit être l'une des valeurs initialement proposées. Plus précisément, le problème du consensus est défini par les propriétés suivante [FLP85] : *(i) terminaison* (tous les processus corrects finissent par décider), *(ii) validité* (si un processus décide v, alors v a été proposé) et *(iii) accord* (deux processus corrects ne peuvent décider différement). La version *uniforme* du consensus stipule que la propriété d'accord est remplacée par la propriété d'*accord uniforme* (deux processus, corrects ou pas, ne peuvent décider différement).

A la différence des réseaux traditionnels (*i.e.* cablés), où les processus ont connaissance de la topologie du réseau et de tous les autres participants, dans un environnement auto-organisé sans autorité centralisée, le nombre et l'identité des processus participants ne sont *pas* connus initialement. Cependant, même dans un environnement classique, quand les entités fonctionnent de manière asynchrone, le consensus ne peut être résolu si l'un des participants peut crasher [FLP85]. Dès lors, résoudre le consensus sans connaître

[†]Ce travail a bénéficié du support du programme Capes-Cofecub, de Fapesb-Bahia/Brazil et de CNPQ/Brazil.
[‡]Ce travail a bénéficié du support de l'ARC FRACAS de l'INRIA et du projet FRAGILE de l'ACI Sécurité et Informatique.

les participants est encore plus difficile. Du fait du role essentiel de ce problème, nous étudions dans cet article quelles sont les conditions qui permettent de résoudre le consensus dans des réseaux inconnus et asynchrones malgré des crashes potentiels des participants.

De manière à modéliser la non-connaissance de la topologie et de l'ensemble des participants dans les systèmes auto-organisés, Cavin *et al.* [CSS04] ont défini un nouveau problème CUP (*Consensus with unknown participants*). Ce nouveau problème garde la définition classique du consensus, excepté concernant la connaissance à propos de l'ensemble des participants du système. Plus précisément, les participants ne connaissent pas Π, l'ensemble des processus du système. Pour permettre des solutions non triviales, il est nécessaire que les participants aient une connaissance partielle des autres processus, pour qu'une coopération soit seulement possible. L'abstraction du *détecteur de participation* a été introduite pour modéliser ce sous-ensemble des processus connus [CSS04]. Ils peuvent être vus comme des oracles distribués qui proposent des indices sur les processus participants. Par exemple, une manière d'implanter des détecteurs de participation dans les réseaux sans fil serait d'utiliser la diffusion locale pour construire une vue locale des voisins à un saut. Suivant les connaissances initiales proposées par les détecteurs de participation, Cavin *et al.* définissent des conditions de connectivité nécessaires et suffisantes sur le graphe induit par les connaissances pour résoudre le problème du consensus dans un environnement asynchrone, mais sans crash.

De manière duale, les *détecteurs de défaillances* constituent une abstraction élégante pour modéliser le degré de synchronie nécessaire à circonvenir au résultat d'impossibilité du consensus tolérant aux pannes dans les réseaux traditionnels [CT96]. Les détecteurs de défaillances sont habituellement classifiés suivant les propriétés qu'ils guarantissent. Le détecteur *parfait* $\mathcal{P}$ satisfait les propriétés d'*exactitude forte* (aucun processus n'est suspecté avant de crasher) et de *complétude forte* (inéluctablement, tout processus crashé est suspecté continuellement par tout processus correct). Un autre détecteur, le plus faible pour résoudre le problème du consensus dans les réseaux connus [CHT96], est $\diamond\mathcal{S}$. Celui-ci satisfait les propriétés d'*exactitude faible inéluctable* (il existe un point de l'exécution après lequel un processus correct n'est jamais suspecté par quiconque), et de *complétude forte*. Il peut commettre un nombre arbitrairement grand d'erreurs, mais en dépit de son inexactitude, il ne compromet jamais la sureté de l'algorithme de consensus qui l'utilise. Un tel protocole de consensus est considéré comme *indulgent* vis à vis de son détecteur, c'est à dire qu'il est conçu pour tolérer des informations non fiables pendant des périodes d'asynchronie et d'instabilité du système, et est capable de résoudre la version uniforme du consensus.

Dans le contexte des réseaux inconnus, le problème FT-CUP (*Faut-tolerant CUP*) a été étudié par Cavin *et al.* [CSS05]. En considérant les prérequis minimaux de connectivité sur le graphe initial des connaissances pour résoudre CUP, ils montrent qu'un détecteur de défaillances parfait $\mathcal{P}$ est nécessaire pour résoudre FT-CUP. Un détecteur de défaillances parfait ne fait jamais d'erreurs et peut seulement être implanté dans un système synchrone. Par conséquent, résoudre FT-CUP dans un scénario où les connaissances sur les participants sont minimales demande l'hypothèse de synchronie la plus forte. Pourtant, la synchronie forte s'oppose à la forte dynamicité et à la décentralisation complète des systèmes auto-organisés. De plus, même avec un détecteur de défaillances parfait, quand on considère des connaissances minimales, la version uniforme du consensus ne peut pas être résolue dans les réseaux inconnus.

Dans cet article, nous montrons qu'il existe un compromis entre la connaissance des participants et la synchronie du système pour le problème du consensus dans les réseaux inconnus. En particulier, nous étudions le problème FT-CUP avec des hypothèses de synchronie minimales (*i.e.* en utilisant le détecteur de défaillances $\diamond\mathcal{S}$), et présentons des conditions nécessaires et suffisantes sur la connectivité du graphe des connaissances. Si le système vérifie nos hypothèses de connectivité des connaissances, n'importe quel algorithme de consensus indulgent conçu pour les réseaux traditionnels peut être utilisé pour résoudre FT-CUP (y compris dans sa version uniforme).

2 Conditions pour le consensus dans les réseaux inconnus

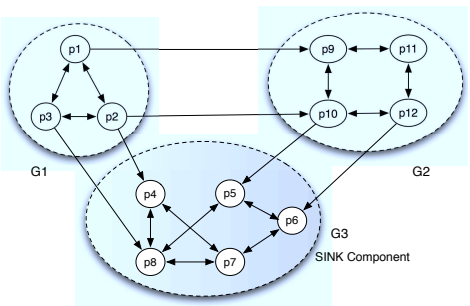
Détecteurs de participation Nous considérons les détecteurs de participation comme des oracles répartis, qui peuvent être questionnés par les processus du système. Le détecteur de participation d'un processus particulier p_i est noté $i.\mathsf{DP}$. Quand p_i fait une requète à $i.\mathsf{DP}$, un sous-ensemble de Π lui est retourné.

Entre deux requètes par un même processus p_i, l'information retournée par $i.\mathsf{DP}$ peut varier, mais les deux propriétés suivantes sont vérifiées : *inclusion* (l'information ne décroît pas avec le temps) et *exactitude* (le détecteur de participation ne fait pas d'erreurs, c'est à dire que l'ensemble retourné est un sous ensemble de Π). Le graphe des connaissances $G = (V,E)$ est le graphe induit par le détecteur de participation, c'est à dire que l'arc (p_i, p_j) est dans E si et seulement si $p_j \in i.\mathsf{DP}$.

Dans les réseaux fiables, [CSS04] montrent que pour résoudre le consensus, il est nécessaire que G soit connexe (classe *CO*), il est suffisant que G soit fortement connexe (classe *SCO*), et il est nécessaire et suffisant que G soit *réductible à un puits* (classe *OSR*). Cette dernière propriété stipule que : *(i)* G est connexe et *(ii)* le graphe orienté acyclique obtenu en réduisant G à les somposantes fortement connexes possède exactement un puits. Dans [CSS05], les mêmes auteurs montrent que ces conditions ne sont pas suffisantes pour résoudre FT-CUP. Dans cet article, nous introduisons trois nouvelles classes de détecteurs de participation : *(i)* *k-CO* (le graphe G est k-connexe), *k-SCO* (le graphe G est k-fortement connexe), et *k-OSR* (le graphe orienté acyclique obtenu en réduisant G à ses composantes k-fortement connexes est k-connexe et possède exactement un puits). Nous montrons que si le nombre effectif de fautes f est strictement inférieur à une constante k, alors les propriétés suivantes sont vérifiées : *(i)* le détecteur *k-CO* est nécessaire pour résoudre FT-CUP, *(ii)* le détecteur *k-SCO* est suffisant pour résoudre le consensus uniforme avec $\diamond\mathcal{S}$, et *(iii)* le détecteur *k-OSR* est nécessaire pour résoudre le consensus et suffisant pour résoudre le consensus uniforme avec $\diamond\mathcal{S}$. Notons que nos hypothèses sur le détecteur de défaillance sont minimales, puisque $\diamond\mathcal{S}$ est minimal pour résoudre le consensus dans les réseaux connus.

Nécessité de *k-CO* L'intuition du résultat est simple : si le graphe des connaissances n'est pas k-connexe (mais par exemple $(k-1)$-connexe), alors le crash initial de $k-1$ processus peut déconnecter le graphe des connaissances avant toute requète à un détecteur de participation. Par suite, les processus corrects situés dans des composantes connexes distinctes ne peuvent communiquer entre eux, et il est possible qu'ils choisissent des valeurs différentes, ce qui viole la propriété d'accord du consensus.

Suffisance de *k-SCO* Notre approche est constructive. Nous proposons un algorithme, COLLECT, qui permet de réutiliser un algorithme de consensus supposant $\diamond\mathcal{S}$. L'algorithme COLLECT permet aux nœuds d'obtenir une vue partielle du système, comprenant l'ensemble de tous les processus dont il est possible de trouver l'identité, y compris *via* des processus intermédiaires. On suppose que $f < k$ processus peuvent crasher pendant l'exécution de l'algorithme. Quand un processus p_i initie le protocole, il effectue une requète à son détecteur de participation pour obtenir $i.\mathsf{DP}$; puis p_i effectue itérativement des requètes et demande aux nouveaux processus dont il a obtenu l'identité de lui fournir les identités des processus qu'il ne connait pas encore, jusqu'à ce qu'aucune connaissance supplémentaire ne puisse être obtenue *via* un processus déjà connu. Ainsi, l'algorithme COLLECT opère par rondes : dans chaque ronde $r > 0$, p_i contacte tous les nœuds qu'il ne connaissait pas lors de l'étape $r-1$ de manière à accroître sa connaissance du réseau. Lors de la ronde 0, p_i ne connaît que lui-même. Dans notre approche, nous supposons que le détecteur de participation de p_i est appelé exactement une fois, ce qui correspond à l'implantation *via* une mémoire cache du contenu du résultat de la première requète au détecteur de participation. Cette propriété guarantit que l'instantané partiel du réseau est consistant pour tous les nœuds, et définit un graphe des connaissances commun $G = (V,E)$.



Si le graphe des connaissances est dans *k-SCO*, alors COLLECT termine et retourne Π ; il est alors possible réutiliser un algorithme de consensus (uniforme) classique. Sinon, quand le graphe des connaissances est dans *k-OSR*, chaque processus p_i obtient tous les nœuds atteignables dans sa composante k-fortement connexe, plus les nœuds atteignables dans d'autres composantes (ce qui inclus en particulier tous les nœuds de la composante puits). Dans l'exemple de la figure, COLLECT retourne pour $p_i \in G_1$ un ensemble contenant $p_j \in \{G_1 \cup G_2 \cup G_3\}$; pour $p_i \in G_2$ un ensemble formé par $p_j \in \{G_2 \cup G_3\}$; pour $p_i \in G_3$ un ensemble formé par $p_j \in \{G_3\}$.

Proposition 1 *Le détecteur de participation k-SCO est suffisant pour résoudre le FT-CUP uniforme, en dépit de $f < k < n$ crashes, en supposant $\diamond\mathcal{S}$.*

Nécessité et suffisance de *k-OSR* Notre approche est également constructive. L'algorithme CONSENSUS que nous proposons réutilise l'algorithme COLLECT défini précédemment, et un second algorithme, SINK, qui détermine si un nœud se trouve dans l'unique composante puits du graphe des connaissances (en supposant que ce graphe est dans *k-OSR*).

L'algorithme SINK est composé de deux phases. Tout d'abord, chaque nœud exécute COLLECT pour obtenir une vue partielle du système. Lorsque COLLECT termine, chaque processus possède une liste de processus qui sont ensuite interrogés sur leur propre liste de processus (obtenue elle aussi *via* l'algorithme COLLECT). Si deux liste sont égales, cela signifie que les deux processus se trouvent dans la même composante k-fortement connexe du graphe des connaissances ; si elles diffèrent, cela signifie que les deux processus sont dans des composantes k-fortement connexes disjointes. Maintenant, si pour *tous* les processus interrogés (*i.e.* connus) par p_i, les listes sont identiques, cela signifie que p_i est dans la composante puits du graphe des connaissances. S'il existe un processus interrogé tel que les listes sont différentes, c'est que p_i ne se trouve pas dans le puits.

L'algorithme CONSENSUS se déroule alors comme suit : dans la phase initiale, tous les processus utilisent SINK pour déterminer s'ils font partie de la composante puits du graphe des connaissances ; par la suite deux comportements sont possibles suivant que le nœud est dans le puits ou pas. Si le processus est dans le puits, alors tous les processus connus ont la même vue partielle du système, qui comprend tous les processus du puit. Il est alors possible d'exécuter un algorithme de consensus uniforme classique supposant $\diamond\mathcal{S}$ dans cette composante, et de décider d'une valeur, dès que le puit comprend au moins $2k+1$ processus. Les autres nœuds (qui se trouvent dans les autres composantes k-connexes) effectuent une requète à chaque nœud connu (qui comprennent au moins les $2k+1$ nœuds du puit) pour leur demander leur valeur de décision. Comme au moins un processus du puit est correct, au moins une réponse avertira de la valeur de décision. Le détail des algorithmes et des preuves peut être trouvé dans [GT07].

Proposition 2 *Le détecteur de participation k-OSR est suffisant pour résoudre le FT-CUP uniforme et nécessaire pour résoudre FT-CUP, en dépit de $f < k < n$ crashes, et en supposant $\diamond\mathcal{S}$.*

Références

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

[CSS04] D. Cavin, Y. Sasson, and A. Schiper. Consensus with unknown participants or fundamental self-organization. In *Proc. of the 3rd Int. Conf. on AD-NOC Networks & Wireless (ADHOC-NOW'04)*, pages 135–148, Vancouver, July 2004. Springer-Verlag.

[CSS05] D. Cavin, Y. Sasson, and A. Schiper. Reaching agreement with unknown participants in mobile self-organized networks in spite of process crashes. Research Report IC/2005/026, EPFL, 2005.

[CT96] T. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2) :225–267, March 1996.

[FLP85] M. J. Fischer, N. A. Lynch, and M. D. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of ACM*, 32(2) :374–382, April 1985.

[GT07] Fabíola Greve and Sébastien Tixeuil. Knowledge connectivity vs. synchrony requirements for fault-tolerant agreement in unknown networks. Research Report 6099, INRIA, 01 2007.