Auto-Stabilisation et Confinement de Fautes Malicieuses : Optimalité du Protocole $min+1$

Swan Dubois[1], Toshimitsu Masuzawa[2] et Sébastien Tixeuil[3]

[1] *UPMC Sorbonne Universités & INRIA (France), swan.dubois@lip6.fr*
[2] *Université d'Osaka (Japon), masuzawa@ist.osaka-u.ac.jp*
[3] *UPMC Sorbonne Universités & IUF (France), sebastien.tixeuil@lip6.fr*

Un protocole *auto-stabilisant* est par nature tolérant aux fautes *transitoires* (*i.e.* de durée finie). Ces dernières années ont vu apparaître une nouvelle classe de protocoles qui, en plus d'être auto-stabilisants, tolèrent un nombre limité de fautes *permanentes*. Dans cet article, nous nous intéressons aux protocoles auto-stabilisants tolérant des fautes permanentes très sévères : les fautes *Byzantines*. Nous introduisons deux nouveaux concepts de confinement de fautes Byzantines dans les systèmes auto-stabilisants. Nous montrons que, pour le problème de la construction d'arbre couvrant en largeur, le protocole auto-stabilisant connu sous le nom de $min+1$ offre sans aucune modification fondamentale le meilleur confinement de fautes Byzantines possible au sens de ces nouveaux concepts.

Keywords: Arbre couvrant en largeur, auto-stabilisation, stabilisation stricte, stabilisation forte, tolérance Byzantine

1 Motivations et Définitions

Le développement des systèmes distribués à large échelle a démontré que la tolérance aux différents types de fautes doit être incluse dans les premières étapes du développement d'un tel système. L'*auto-stabilisation* permet de tolérer des fautes *transitoires* tandis que la tolérance aux fautes traditionnelle permet de masquer l'effet de fautes *permanentes*. Il est alors naturel de s'intéresser à des systèmes qui regrouperaient ces deux formes de tolérance. Cet article s'inscrit dans cette voie de recherche.

Auto-stabilisation Dans cet article, nous considérons un système distribué asynchrone anonyme, *i.e.* un graphe non orienté connexe G où les sommets représentent les processus et les arêtes représentent les liens de communication. Deux processus u et v sont *voisins* si l'arête (u,v) existe dans G. Les variables d'un processus définissent son *état*. L'ensemble des états des processus du système à un instant donné forme la *configuration* du système. Nous souhaitons résoudre une classe particulière de problèmes sur ce système : les problèmes *statiques* (*i.e.* les problèmes où le système doit atteindre un état donné et y rester). Par exemple, la construction d'arbre couvrant est un problème statique. De plus, nous considérons des problèmes pouvant être spécifiés de manière locale (*i.e.* il existe, pour chaque processus v, un prédicat $spec(v)$ qui est vrai si et seulement si la configuration est conforme au problème). Les variables apparaissant dans $spec(v)$ sont appelées *variables de sortie* ou *S-variables*.

Un système auto-stabilisant [Dij74] est un système atteignant en un temps fini une configuration légitime (*i.e.* $spec(v)$ est vraie pour tout v) indépendamment de la configuration initiale (propriété de *convergence*). Une fois cette configuration légitime atteinte, tout processus v vérifie $spec(v)$ pour le restant de l'exécution et, dans le cas d'un problème statique, le système ne modifie plus ses S-variables (propriété de *clôture*). Par définition, un tel système peut tolérer un nombre arbitraire de fautes *transitoires*, *i.e.* de fautes de durée finie (la configuration initiale arbitraire modélisant le résultat de ces fautes). Cependant, la stabilisation du système n'est en général garantie que si tous les processus exécutent correctement leur protocole.

Stabilisation stricte et forte Si certains processus exhibent un comportement Byzantin (*i.e.* ont un comportement arbitraire, et donc potentiellement malicieux), ils peuvent perturber le système au point que certains processus corrects ne vérifient jamais $spec(v)$. Pour gérer ce type de fautes, [NA02] définit un

protocole *strictement stabilisant* comme un protocole auto-stabilisant tolérant des fautes Byzantines permanentes. Plus précisèment, étant donné c (appelé *rayon de confinement*), [NA02] définit une configuration *c-confinée* comme une configuration dans laquelle tout processus v à une distance supérieure à c de tout processus Byzantin vérifie $spec(v)$. Un protocole strictement stabilisant est alors défini comme un protocole satisfaisant les propriétés de convergence et de clôture par rapport à l'ensemble des configurations c-confinées (et non plus l'ensemble des configurations légitimes comme en auto-stabilisation). Cela permet d'assurer que seuls les processus dans le c-voisinage (*i.e.* à distance inférieure ou égale à c) d'un processus Byzantin peuvent ne pas vérifier infiniment souvent la spécification. Cependant, [NA02] fournit une série de résultats d'impossibilité. Intuitivement, il n'existe pas de solution strictement stabilisante (pour tout rayon de confinement inférieur au diamètre) pour tout problème global (constructions d'arbre couvrant,...).

Pour contourner de tels résultats d'impossibilité, [MT06] définit un modèle de tolérance plus faible : la *stabilisation forte*. Intuitivement, il s'agit d'affaiblir les contraintes relatives au rayon de confinement. En effet, certains processus à l'extérieur de ce rayon sont autorisés à ne pas respecter la spécification en raison des Byzantins. Cependant, ces *perturbations* (*i.e.* périodes d'exécution durant laquelle des processus en dehors du rayon de confinement ne vérifient plus la spécification) sont limitées dans le temps : les processus ne peuvent être perturbés par les Byzantins qu'un nombre fini de fois et toujours pendant un temps limité même si les Byzantins agissent infiniment longtemps. La mesure d'efficacité principale de ce type de protocole est leur *nombre de pertubations*, *i.e.* le nombre maximal de perturbations possibles dans une exécution. Dans ce modèle de tolérance, nous avons montré [DMT10a] qu'il est possible de résoudre le problème de construction d'arbre couvrant, ce qui illustre le fait que la stabilisation forte permet de résoudre plus de problèmes que la stabilisation stricte (en contrepartie, les propriétés assurées sont plus faibles).

Construction d'arbre couvrant en largeur Dans cet article, nous nous intéressons au problème de la *construction d'arbre couvrant en largeur*. Un processus du réseau est désigné a priori comme étant la *racine* (notée r) du système. Nous supposons que cette racine n'est jamais Byzantine. Une configuration satisfait la spécification du problème lorsqu'il existe un arbre couvrant le système enraciné en r vérifiant la propriété suivante : pour tout processus, la distance entre v et r dans l'arbre couvrant est égale à celle dans le système initial. Il s'agit d'un problème fondamental car il permet de mettre en œuvre de nombreux protocoles de communication (par exemple, diffusion, routage par les plus courts chemins, etc.). Ce problème a été largement étudié dans le domaine de l'auto-stabilisation (voir par exemple [Gär03, DT01]).

Lorsque le système contient des processus Byzantins, notons B l'ensemble de ces Byzantins. Dans ces conditions, il est impossible, pour un processus correct, de distinguer la racine réelle r d'un Byzantin se comportant comme une racine. Nous devons donc autoriser le système à construire une forêt couvrante du système (donc un ensemble d'arbres couvrant le système) dans laquelle chaque racine est soit r soit un Byzantin. Plus précisèment, chaque processus v a deux S-variables : un pointeur sur son parent P_v et une hauteur H_v. Un chemin $(v_0,\ldots,v_k)$ $(k \geq 1)$ est un chemin *correct* s'il vérifie (i) $P_{v_0} = \perp$, $H_{v_0} = 0$ et $v_0 \in B \cup \{r\}$, (ii) $\forall i \in \{1,\ldots,k\}, P_{v_i} = v_{i-1}$ et $H_{v_i} = H_{v_{i-1}} + 1$ et (iii) $\forall i \in \{1,\ldots,k\}, H_{v_{i-1}} = min\{H_u | u \in N_{v_i}\}$. Nous pouvons à présent donner la spécification locale de notre problème.

$$spec(v) : \begin{cases} P_v = \perp \text{ et } H_v = 0 \text{ si } v = r \\ \text{il existe un chemin correct } (v_0,\ldots,v_k) \text{ tel que } v_k = v \text{ sinon} \end{cases}$$

Il est possible de remarquer que, dans le cas où aucun processus n'est Byzantin et où tout processus v vérifie $spec(v)$, il existe un arbre couvrant du système au sens "classique". Cette spécification implique le théorème d'impossibilité suivant :

Théorème 1 *Il n'existe pas de solution strictement stabilisante ou fortement stabilisante pour la construction d'arbre en largeur pour tout rayon de confinement et tout nombre de perturbations.*

L'impossibilité de la stabilisation stricte est dûe à [NA02] tandis que celle de la stabilisation forte provient du fait que tout processus Byzantin peut perturber infiniment souvent la moitié des processus sur le chemin qui le sépare de la racine (rendant donc impossible la majoration de cette distance par une constante).

Stabilisation stricte et forte topologiquement dépendante Afin de contourner les résultats d'impossibilité du Théorème 1, nous introduisons ici une nouvelle idée dans le domaine de l'auto-stabilisation confinant les fautes permanentes. Nous relâchons la contrainte sur le rayon de confinement. Pour cela, nous assurons que l'ensemble des processus infiniment souvent perturbés par les Byzantins n'est plus l'union des

c-voisinages des processus Byzantins mais un ensemble de processus dépendant du graphe de communication et de la position des processus Byzantins. C'est pourquoi nous avons nommé ce concept stabilisation stricte (respectivement forte) *topologiquement dépendante* (abrégé TD dans la suite). Pour en donner une définition formelle (dans le cas des problèmes statiques), nous devons introduire quelques définitions.

Nous prenons comme modèle de calcul le *modèle à états* : Les variables des processus sont partagées : chaque processus a un accès direct en lecture aux variables de ses voisins. En une *étape* atomique, chaque processus peut lire son état et ceux de ses voisins et modifier son propre état. Un *protocole* est constitué d'un ensemble de règles de la forme $< garde > \longrightarrow < action >$. La *garde* est un prédicat sur l'état du processus et de ses voisins tandis que l'*action* est une séquence d'instructions modifiant l'état du processus. A chaque étape, chaque processus évalue ses gardes. Il est dit *activable* si l'une d'elles est vraie. Il est alors autorisé à exécuter son *action* correspondante (en cas d'exécution simultannée, tous les processeurs activés prennent en compte l'état du système du début de l'étape). Les *exécutions* du système (séquences d'étapes) sont gérées par un *ordonnanceur* : à chaque étape, il sélectionne au moins un processus activable pour que celui-ci exécute sa règle. Cet ordonnanceur permet de modéliser l'asynchronisme du système. La seule hypothèse que nous faisons sur l'ordonnancement est qu'il est *fortement équitable*, *i.e.* qu'aucun processus ne peut être infiniment souvent activable sans être choisi par l'ordonnanceur (cette hypothèse est nécessaire pour borner le nombre de perturbations de notre protocole). Nous considérons un ensemble S_B de processus corrects déterminé par l'ensemble B des processus Byzantins et la topologie du système. Intuitivement, S_B regroupe les processus qui peuvent être infiniment souvent pertubés par les processus Byzantins. Il est appelé *zone de confinement*. Nous introduisons ici quelques notations. Un processus correct est *S_B-correct* s'il n'appartient pas à S_B. Une configuration est *S_B-légitime* pour *spec* si tout processus S_B-correct v vérifie $spec(v)$. Une configuration est *S_B-stable* si tout processus S_B-correct ne modifie pas ses S-variables tant que les Byzantins n'effectuent aucune action. A présent, nous définissons la stabilisation stricte (respectivement forte) topologiquement dépendante (Définition 2, respectivement Définition 5).

Définition 1 *Une configuration* ρ *est* (S_B, f)-TD contenue *pour spec si, étant donné au plus f Byzantins, toute exécution issue de* ρ *ne contient que des configurations S_B-légitimes et que tout processus S_B-correct ne modifie pas ses S-variables.*

Définition 2 *Un protocole est* (S_B, f)-TD strictement stabilisant *pour spec si, étant donné au plus f Byzantins, toute exécution (issue d'une configuration arbitraire) contient une configuration (S_B, f)-TD contenue pour spec.*

Définition 3 *Une portion d'exécution $e = \rho_0, \ldots, \rho_t$ ($t > 1$) est une* S_B-TD perturbation *si : (1) e est finie, (2) e contient au moins une action d'un processus S_B-correct modifiant une S-variable, (3) ρ_0 est S_B-légitime pour spec et S_B-stable, et (4) ρ_t est la première configuration S_B-légitime pour spec et S_B-stable après ρ_0.*

Définition 4 *Une configuration* ρ_0 *est* (t, k, S_B, f)-TD temporellement contenue *pour spec si, étant donné au plus f Byzantins : (1) ρ_0 est S_B-légitime pour spec et S_B-stable, (2) toute exécution issue de ρ_0 contient une configuration S_B-légitime pour spec après laquelle les S-variables de tout processus S_B-correct ne sont pas modifiées (même si les Byzantins exécutent une infinité d'actions), (3) toute exécution issue de ρ_0 contient au plus t S_B-TD perturbations, et (4) toute exécution issue de ρ_0 contient au plus k modifications des S-variables de chaque processus S_B-correct.*

Définition 5 *Un protocole $\mathcal{P}$ est* (t, S_B, f)-TD fortement stabilisant *pour spec si, étant donné au plus f Byzantins, toute exécution (issue d'une configuration arbitraire) contient une configuration (t, k, S_B, f)-TD temporellement contenue pour spec.*

Par définition, un protocole TD strictement stabilisant (respectivement TD fortement stabilisant) est plus faible qu'un protocole strictement stabilisant (respectivement fortement stabilisant). Cependant, il est plus puissant qu'un protocole auto-stabilisant (qui peut ne jamais stabiliser en présence de Byzantins).

2 Construction d'Arbre Couvrant en Largeur

Zones de confinement optimales Nous définissons les zones de confinement suivantes : $S_B = \{v \in V | min\{d(v,b) | b \in B\} \leq d(r,v)\}$ et $S_B^* = \{v \in V | min\{d(v,b) | b \in B\} < d(r,v)\}$. Intuitivement, S_B regroupe l'ensemble des processus situés plus près (ou à égale distance) du plus proche des Byzantins que de la racine tandis que S_B^* regroupe l'ensemble des processus situés strictement plus près du plus proche des Byzantins que de la racine. Il est alors possible de prouver le résultat suivant.

Algorithme 1 $\mathcal{CALFS}$: Construction d'arbre couvrant en largeur pour le processus v.

Constante : N_v l'ensemble des voisins de v doté d'un ordre circulaire
S-variables : $P_v \in N_v \cup \{\perp\}$: parent de v
$H_v \in \mathbb{N}$: hauteur de v
Macro : Pour tout sous ensemble A de N_v, $suivant_v(A)$ retourne le premier élément de A qui est supérieur à P_v
Règles : $(v = r) \wedge ((P_v \neq \perp) \vee (H_v \neq 0)) \longrightarrow H_v := 0;\ P_v := \perp$
$(v \neq r) \wedge ((P_v = \perp) \vee (H_v \neq H_{P_v} + 1) \vee (H_{P_v} \neq min\{H_q | q \in N_v\}))$
$\longrightarrow P_v := suivant_v(\{p \in N_v | H_p = min\{H_q | q \in N_v\}\});\ H_v := H_{P_v} + 1$

Théorème 2 *Il n'existe pas de protocole $(A_B, 1)$-TD strictement stabilisant ou $(t, A_B^*, 1)$-TD fortement stabilisant pour la construction d'arbre en largeur avec $A_B \subsetneq S_B$ et $A_B^* \subsetneq S_B^*$ (pour tout t).*

Solution optimale Il existe un protocole auto-stabilisant simple pour construire un arbre couvrant en largeur (voir [Gär03, DT01]). Celui-ci est connu sous le nom de $min+1$ en raison de sa règle principale. Tout processus (différent de la racine) teste si la hauteur de son père actuel est minimale parmi celles de ses voisins et si sa hauteur et égale à celle de son père plus un. Si ce n'est pas le cas, le processus est alors activable pour choisir comme père son voisin ayant la hauteur minimale (règle *min*) et calculer sa nouvelle hauteur en fonction (règle $+1$). L'algorithme $\mathcal{CALFS}$ présenté en Algorithme 1 suit ce principe à l'exception suivante près : si plusieurs voisins présentent une hauteur minimale, alors le processus choisi le plus petit qui est supérieur au père actuel (selon un ordre circulaire défini sur les voisins).

Il est alors possible de montrer que l'ensemble des processus strictement plus près de la racine que d'un processus Byzantin agiront exactement comme si aucun Byzantin n'était présent (étant donné qu'ils sont suffisament proches de la racine pour que sa hauteur soit considérée comme minimale). Il est également possible de montrer que les processus à égale distance de la racine et d'un processus Byzantin peuvent être perturbés par les processus Byzantins. En revanche, nous pouvons borner ce nombre de perturbations. En d'autres termes, nous obtenons le résultat suivant :

Théorème 3 *$\mathcal{CALFS}$ est un protocole $(S_B, n-1)$-TD strictement stabilisant et $(n\Delta, S_B^*, n-1)$-TD fortement stabilisant pour spec (avec n nombre de processus et Δ degré maximal du système).*

3 Conclusion

Dans cet article, nous nous sommes intéressés aux protocoles auto-stabilisants confinant de plus l'effet de fautes Byzantines permanentes. Nous avons montré que le protocole $min+1$ connu pour construire un arbre couvrant en largeur de manière auto-stabilisante fournit de plus un confinement Byzantin optimal. L'ensemble de ces résultats ont été publiés dans [DMT10b].

En utilisant les résultats de [DT01] sur les r-opérateurs, nous pouvons facilement étendre les résultats obtenus à d'autres métriques. Il est alors naturel de se demander si ce nouveau concept de confinement Byzantin topologiquement dépendant peut être étendu à d'autres problèmes, statiques ou non.

Références

[Dij74] E. Dijkstra. Self-stabilizing systems in spite of distributed control. *Com. ACM*, 17(11) :643–644, 1974.

[DMT10a] S. Dubois, T. Masuzawa, and S. Tixeuil. Construction auto-stabilisante d'arbre couvrant en dépit d'actions malicieuses. In *AlgoTel*, 2010.

[DMT10b] S. Dubois, T. Masuzawa, and S. Tixeuil. On byzantine containment properties of the min+1 protocol. In *12th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, 2010.

[DT01] B. Ducourthial and S. Tixeuil. Self-stabilization with r-operators. *Distributed Computing*, 14(3) :147–162, July 2001.

[Gär03] F. Gärtner. A survey of self-stabilizing spanning-tree construction algorithms. TR ic/2003/38, EPFL, 2003.

[MT06] T. Masuzawa and S. Tixeuil. Bounding the impact of unbounded attacks in stabilization. In *8th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 440–453, 2006.

[NA02] M. Nesterenko and A. Arora. Tolerance to unbounded byzantine faults. In *21st Symposium on Reliable Distributed Systems*, page 22. IEEE Computer Society, 2002.