République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/MIRA,Béjaia
Faculté des Sciences Exactes
Département d'informatique
Soutenance de Master recherche en informatique
Option : réseaux et systèmes distribués
Thème

An optimal indulgent consensus protocol with efficient commmunication complexity

Ahmed MAZARI, d'après M^r Hamouma MOUMEN

22 juin 2015

Plan de l'exposé I

Sommaire I

Sommaire II

- Résoudre le problème du consensus dans les systèmes distribués asynchrones.
- Proposer un protocole de consensus indulgent optimal.

La démarche :

- Planifier pour le pire des cas et espérer le meilleur des cas.
- Réduire la taille et le nombre de messages échangés.
- Minimiser le nombre de tours nécessaires pour atteindre une décision globale.

Sommaire I

Sommaire II

Exemples d'application du consensus I

- Synchronisation des horloges locales pour les systèmes redondants [Lamport,19800].
- Bases de données distribuées et systèmes transactionnels [Bernestein et al,1987].
- Communication de groupes (atomic multicast) [Guerraoui et Schiper,1997].
- Le problème d'élection d'un leader (token ring) [Larrea et al,2000].
- Diffusion fiable.
- Partage de données réparties et coordination répartie :Partager l'accès à un groupe de pages mémoire.

Sommaire I

Sommaire II

- **Physique** : Conception et simulation d'un accélérateur de particules distribué.Université de Berkley,USA.
- **Biologie moléculaire** : La recherche en reconnaissance moléculaire.Université de Sherbrooke,CANADA.
- **Climatologie** : Analyse de méthodes pour améliorer les modèles de prévision du climat.Université d'Oxford,UK.
- **Sismologie** :Utilisation des accéléromètres connectés à un ordinateur pour détecter des séismes.Université de Stanford,USA.

Sommaire I

Sommaire II

1. **Système distribué synchrone** :
   - La vitesse relative ϕ de chaque processus est connue.
   - Le délai de communication Δ est connu.
   - La borne σ entre deux processus p et q est :
   $\sigma = \Delta + s.\gamma + \gamma.\Delta$
2. **Système distribué asynchrone** :
   -L'absence de borne portant sur la vitesse relative des processus et le délai de délivrance des messages.
   -L'absence de l'ordre dans la délivrance des messages.
   - Les deux paramètres ϕ et Δ existent mais demeurent inconnus.
3. **Système partiellement synchrone** :
   -Synchronicité partielle.
   -Δ et ϕ existent mais connus qu'après un GST (general stabilization time) inconnu.
4. **Types de défaillances** :
   -Pannes franches :un processus s'arrête prématurément et ne peut reprendre son exécution,c'est-à-dire :
   $\forall\ t \in T$:F(t) $\subseteq$ F($t+1$) et $correct(F) \neq \oslash$.
   -Pannes par omission .
   -Pannes de temporisation.
   -Pannes byzantines.

5. **Le Problème de consensus** :Un protocole de consensus vérifie les propriétés suivantes :
   (1) **Terminaison** : tous les processus corrects décident finalement.
   (2) **Validité** : si un processus décide v alors v a été proposée par au moins un processus.
   (3) **Intégrité** : Aucun processus ne décide deux fois.
   (4) **Accord** : si p_i décide v et p_j décide v' alors $v = v'$.
   Pease,M.,Shostak,R.,Lamport,L.,1980 Reaching agreement in the presence of faults.
6. **L'algorithme indulgent** : est un algorithme distribué qui tolère les fausses suspicions commises par un détecteur de défaillances.Il tolère des périodes d'asynchronisme.
   Guerraoui,R.,école polytechnique de lausanne,SUISSE,2000 Indulgent algorithms(preliminary version).
7. Algorithme **early-deciding** :un algorithme est dit $early - deciding$ si dans chaque exécution il minimise le nombre de tours jusqu'à ce que tous les processus décident.
8. Algorithme **early-stopping** : un algorithme est $early - stopping$ s'il est $earl - deciding$ et termine après une étape au plus.

Sommaire I

Sommaire II

Un protocole de consensus est dit correct si :

- Il est **partiellement correct**.
- Toutes les exécutions **admissibiles** sont **décisionnelles**.

Fisher,Lynch et Paterson ont démontré que tous les protocoles de consensus partiellement corrects possèdent au moins une exécution **non décisionnelle** et au moins une configuration atteignable **bi-valente**.
Fischer,Lynch et Paterson,1980 Impossibility of distributed consensus with one faulty process.

Sommaire I

Sommaire II

1. **Oracle probabiliste**
2. **Oracle leader**
3. **Détecteurs de défaillances** : ont pour objectif de circonvenir l'impossibilité de FLP grâce aux registres d'informations.
   - **La complétude forte** : Tous les processus défaillants sont finalement et définitivement suspectés par tous les processus corrects.
     $\forall F, \forall H \in D(F), \exists\ t \in \tau, \forall p \in \mathit{crashed}(F), \forall q \in \mathit{correct}(F), \forall t' \geq t : p \in H(q, t')$.
   - **La complétude faible** :Pour tout processus défaillant, il existe au moins un processus correct qui le suspecte.
     $\forall F,\ \forall H \in D(F),\ \exists t \in \tau,\ \forall p \in \mathit{crashed}(F),\ \exists q \in \mathit{correct}(F),\ \forall t' \geq t : p \in H(q, t')$.
   - **La précision forte** : Aucun processus n'est suspecté avant qu'il devienne défaillant.
     $\forall F,\ \forall H \in D(F),\ \forall t \in \tau,\ \forall p,\ q \in \Pi - F(t) : p \notin H(q, t)$.
   - **La précision faible** : Il existe au moins un processus correct qui n'est jamais suspecté.
     $\forall F,\ \forall H \in D(F),\ \exists p \in \mathit{correct}(F),\ \forall t \in \tau,\ \forall q \in \Pi - F(t) : p \notin H(q, t)$.
   - **La précision ultime forte** : Il existe un instant après lequel tous les processus corrects ne sont pas suspectés par un autre processus correct.
     $\forall F,\ \forall H \in D(F), \exists t \in \tau,\ \forall t' \geq t\ \forall p, q \in \mathit{correct}(F) : p \notin H(q, t')$.
   - **La précision ultime faible** : Il existe un instant après lequel il existe des processus corrects qui ne sont pas suspectés par un autre processus correct.
     $\forall F,\ \forall H\ D(F),\ \exists t \in \tau,\ \exists p \in \mathit{correct}(F),\ \forall t' \geq t,\ \forall q \in \mathit{correct}(F) : p \notin H(q, t')$
4. **Détecteur de défaillances fiables** :
   -Il ne commet pas d'erreurs.
   -Il vérifie les deux propriétés : complétude forte et précision forte.

5. **Limites de la programmation avec timeout** :
   $timeout := clock + \rho$.
   Comment déterminer ρ ?
   Solution : déterminer un timeout dynamique et adaptatif à base des techniques de machine learning.
   Critique de la proposition :
   -Couteuse en temps et en espace mémoire.
   -Elle nécessite l'échange de messages instantanément.
6. **Détecteurs de défaillances non fiables** :
   -Ils fonctionnent sur un système partiellement synchrone.
   -La propriété d'accord et de validité sont maintenues.
   -La propriété de terminaison est vérifiée une fois que le détecteur de défaillances s'arrête de commettre des erreurs.
   -Le détecteur de défaillances non fiable le plus faible résolvant le consensus est $\diamond S$
7. **Propriétés des détecteurs de défaillances non fiables**
   1. **Non fiabilité complète** $\square U$:si pour chaque paire de modèles de défaillances F et F', $D(F)=D(F')$.
   2. **Non fiabilité forte** ∇U :
      $H' \in D(F')$ tel que $\forall\ t \leq t_{k'}\ \forall p_i \in \Pi, H'(p_i, t)=H(p_i, t)$.
   3. **Non fiabilité faible** ΔU :
      $H' \in D(F')$ tel que $\forall\ t \leq t_{k'}\ \forall p_i \in \Pi, H'(p_i, t)=H(p_i, t)$.

Sommaire I

Sommaire II

	Précision-forte	Précision-faible	Précision-ultime-forte	Précision-ultime-faible
Complétude-forte	parfait P	fort S	eventuellement parfait $\Diamond P$	eventuellement fort $\Diamond S$
Complétude-faible	quasi-parfait Q	faible W	eventuellment quasi-parfait $\Diamond Q$	eventuellement faible $\Diamond W$

TABLE 4.1: Les classes des détecteurs de défaillances de *Chandra* et *Toueg*

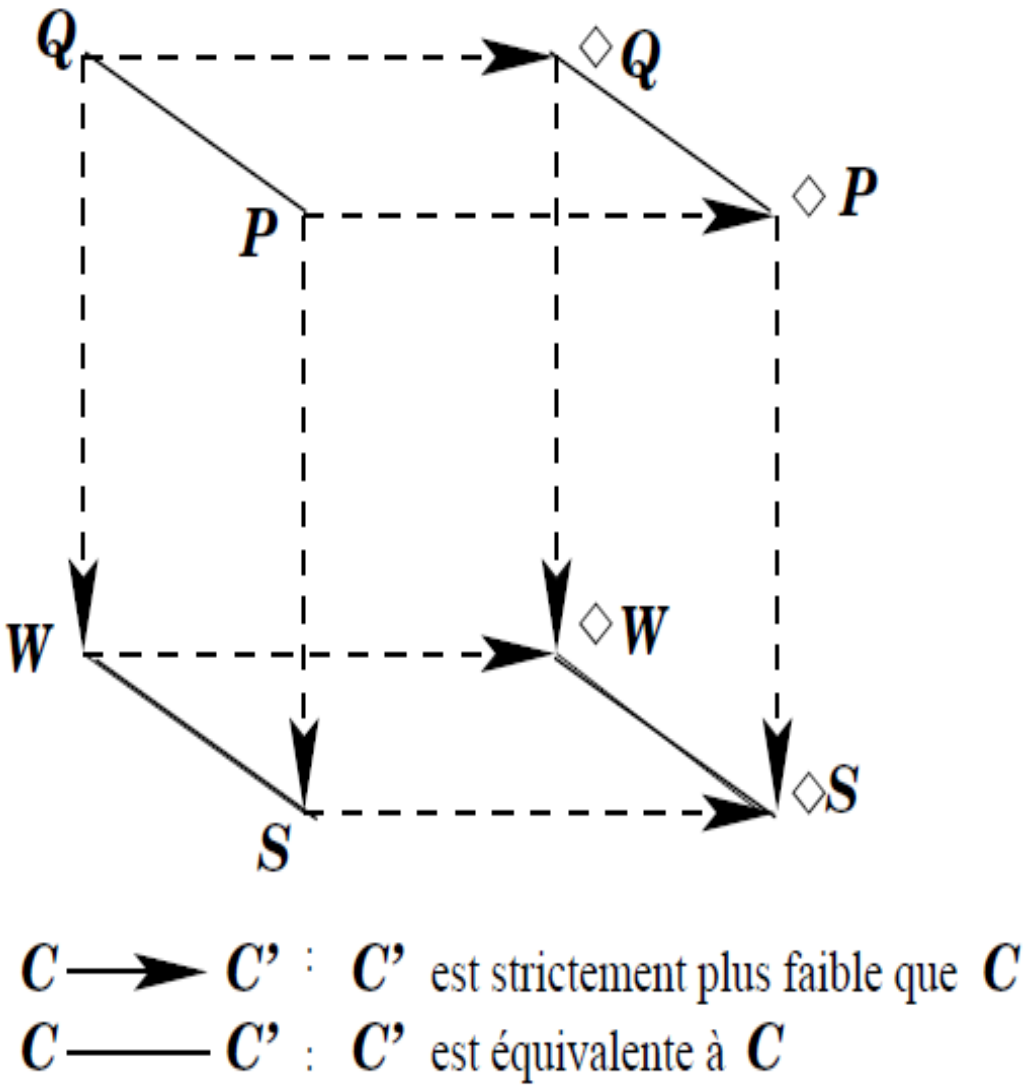

Figure: Réductibilité des classes de détecteurs de défaillances

Sommaire I

Sommaire II

at process p_i

1: **procedure** propose(v_i)
2: $est_i \leftarrow v_i$; $k_i \leftarrow 1$; $msgSet_i \leftarrow \emptyset$; $Halt_i \leftarrow \emptyset$; $nE_i \leftarrow v_i$; $vc_i \leftarrow v_i$; $decided_i \leftarrow false$
3: **Phase 1**
4: **while** $k_i \leq t+1$ **do** {*rounds* $1, ..., t+1$}
5: send(ESTIMATE, k_i, est_i, $Halt_i$) to all
6: **wait until** received messages of round k_i
7: compute()
8: $k_i \leftarrow k_i + 1$
9: **Phase 2**
10: **if** $|Halt_i| > t$ **then** {*round* $t+2$}
11: $nE_i \leftarrow \perp$
12: **else**
13: $nE_i \leftarrow est_i$
14: send(NEWESTIMATE, $t+2$, nE_i) to all
15: **wait until** received messages of round $t+2$
16: **if** every received(NEWESTIMATE, $t+2$, nE) has $nE \neq \perp$ **then**
17: $vc_i \leftarrow$ any one of the nE values received
18: decide(vc_i); $decided_i \leftarrow true$ {*Decision*}
19: **else if** received any (NEWESTIMATE, $t+2$, nE') message *s.t.* $nE' \neq \perp$ **then**
20: $vc_i \leftarrow nE'$
21: $k_i \leftarrow k_i + 1$ {*round* $t+3$}
22: **if** $decided_i$ **then**
23: send (DECIDE, $t+3$, vc_i) to $\Pi \backslash p_i$
24: **else**
25: propose$_C$(vc_i)
26: **return** {*return from propose(*)*}

27: **upon** receiving (DECIDE, k', x) from p_j **do**
28: **wait until** $k_i \geq k'$
29: **if** $\neg decided_i$ **then**
30: **stop** propose$_C$(); decide(x); send (DECIDE, k', x) to $\Pi \backslash p_i$ {*Decision*}
31: **return** {*return from propose(*)*}

32: **procedure** compute()
33: $Halt_i \leftarrow Halt_i \cup \{p_j | (p_i$ received(ESTIMATE, k_i, $*$, $Halt_j)$ from p_j *s.t.* $p_i \in Halt_j)$ **or** (p_i did not receive round k_i message from p_j)$\}$
34: $msgSet_i \leftarrow \{$(ESTIMATE, k_i, $*$, $Halt_j$) | p_i received(ESTIMATE, k_i, $*$, $Halt_j$) from $p_j \notin Halt_i\}$
35: $est_i \leftarrow$ **Min**$\{est |$ (ESTIMATE, k_i, est, $*) \in msgSet_i\}$

Figure: Protocole de consensus indulgent de Dutta et Guerraoui 2005

	Message Complexity	Round Complexity
Alg. 1 (Section 4):	$O(n)$	$O(n^{1+\varepsilon})$
Alg. 2 (Section 5):	$O(n \log^6 n)$	$O(f)$

Fig. 1. Message and round complexity of the two algorithms presented in this paper. Both refer to synchronous executions in which there are no more than $f \leq t$ failures.

Figure: Protocoles de consensus indulgents de Gilbert et al 2007

Sommaire I

Sommaire II

Function Consensus(v_i)

Init : $r_i \leftarrow 1$; $S_i[1..(t+1)] \leftarrow 0$;

Task $T1$: % basic task %

repeat for $r_i \leq t+1$

——————————- first $t+1$ rounds ——————————

01 let A_i = a set of processes with $A_i = \{p_1, p_2, .., p_t, p_{t+1}\}$;
02 if $p_i \in A_i$ then *send* propose(r_i, v_i, S_i) to all ;
03 wait until received propose$(r_i, *, *)$ messages of round r_i ;
04 if ((p_i did not receive propose(r_i, v_j, S_j) message from $p_j \in A_i$)and$(S_i[j] = 0)$) then $S_i[j] \leftarrow 1$;
05 if p_i receives propose (r_i, v_j, S_j) message from $p_j \in A_i$ then for each k, such that $1 \leq k \leq t+1$ do
$S_i[k] \leftarrow max\{S_i[k], S_j[k]\}$;
07 let rec_i = a set of propose$(r_i, *, *)$ messages received from p_j such that $s_i[j] = 0$
08 if $|rec_i| = 0$ then $aux_i \leftarrow \perp$ else $aux_i \leftarrow$ **min**$\{v_k | propose(r_i, v_k, s_k) \in rec_i\}$;
06 $r_i \leftarrow r_i + 1$;

——————————————————————————————————————

end repeat

——————————- round $t+2$ ——————————-

09 *send* filt(r_i, aux_i) to all processes in A_i ;
10 if $p_i \in A_i$ then wait until received filt(r, aux) of round $t+2$ store values in V_i ;
11 case $(V_i = \{v\})$ then decide(v) ;
12 $(V_i = \{v, \perp\})$ then $est_i \leftarrow v$;
13 endcase ;
14 $r_i \leftarrow r_i + 1$;

——————————- round $t+3$ ——————————-

15 if p_i has decided then *send* dec(r_i, v) to all except of itself else esc(est_i) ;
16 if p_i has not decided then
17 upon *receipt* of dec(r, v) : wait until $r_i \geq r$;
18 stop esc(est_i) ; decide(v) *send* dec(r_i, v) to all except of itself ;

Sommaire I

Sommaire II

Function Consensus(v_i)

Init : $r_i \leftarrow 1$;

——-——————————————— First round ———————————————

01 **if** $p_i = p_1$ **then** *send* propose$(1, v_i)$ to all ;
02 **wait until** received propose$(1, *)$ message from p_1 ; $r_i \leftarrow r_i + 1$;

———————————————————————- Second round ———————————————————————-

03 **if** p_i did not receive propose$(1, *)$ message from p_1 **then** $aux_i \leftarrow \perp$;
04 **if** p_i receives propose $(1, v)$ message from p_1 **then** $aux_i \leftarrow v$;
05 *send* filt$(2, aux_i)$ to p_1 ;
06 **if** $p_i = p_1$ **then wait until** received filt(r, aux) from all processes ; **store values in** V_i ;
07 **case** $(V_i = \{v\})$ **then** decide(v) ;
08 $(V_i = \{v, \perp\})$ **then** $est_i \leftarrow v$;
09 **endcase** ;
10 $r_i \leftarrow r_i + 1$;

———————————————————————- Third round ———————————————————————-

11 **if** p_i has decided **then** *send* dec(r_i, v) to all except of itself **else** esc(est_i) ;
12 **if** p_i has not decided **then**
upon *receipt* of dec(r, v) : **wait until** $r_i \geq r$;
stop esc(est_i) ; decide(v) *send* dec(r_i, v) to all except of itself ;

———-

Figure: Protocole de consensus indulgent en absence de défaillances

Sommaire I

Sommaire II

Lemme 1 : accord
La propriété d'accord assure qu'il n'y ait pas deux processus qui décident différement.
Preuves :
-Si aucun processus ne décide, alors la propriété d'accord n'est pas violée.
-Si $k > t+2$, alors aucun processus ne décide dans les premiers $t+2$ tours d'où aucun processus n'envoie un message *DECIDE* dans ligne 15.Par conséquent,la propriété d'accord résulte de la propriété d'accord de ESC.
-$k = t+2$ est le tour le plus bas dont un processus peut décider.
-Etant donné que p_i décide v au tour $t+2$,chaque message reçu par p_i a la valeur $aux_i \neq \perp$ et p_i reçoit au moins un message avec $aux = v$.
-D'après *ESC*, il y a au moins $n - t > (n/2)$ messages reçus par p_i dans le tour $t+2$.
-Par conséquent,au moins une minorité de processus envoie le message $FILT(r, aux)$ avec $aux \neq \perp$.
-De la propriété d'élimination de la ligne $4, 5, 6$ et pour chaque *FILT* message $aux \in (v, \perp)$,n'importe quel processus $\in A$ reçoit au moins un message avec $(aux = v)$.
-Par conséquent,si un processus décide v dans le tour $t+2$,alors ce dernier décide v et envoie un message *DECISION* avec la valeur décidée v dans le tour $t+3$.
-Si le processus invoque $ESC()$ alors la valeur invoquée est v.
-Par la validité de la propriété de l'algorithme *ESC*,aucun processus ne peut décider une valeur autre que v.

Lemme 2 : Décision rapide
Dans chaque exécution synchrone du protocole,n'importe quel processus décidant,décide au plus au tour $t+2$ ou au plus au tour $t+3$
Preuves :
-On suppose ,par **contradiction**, qu'il y a une exécution synchrone dans laquelle certains processus de A complètent le tour $t+2$ sans pouvoir décider.
-Suivant la ligne 9 ,certains processus ont envoyé $FILT$ avec $aux = \perp$.Par conséquent ,l'exécution synchrone $|recu_i| = 0$ (ligne 7).
-**On déduit** que plus de t processus sont défaillants avant d'avoir complété le tour $t+1$,alors contradiction.
Théoréme :
Il existe un protocole de consenus indulgent s'exécutant avec une complexité de $O(n.t)$ en terme de nombre de messages échangés et de $O(t)$ tours dans les exécutions synchrones.
La preuve découle directement du **Lemme 1**,**Lemme 2** et du fait qu'il y ait dans chaque étape de communication du protocole au plus $O(n.t)$ messages échangés.

Sommaire I

Sommaire II

Tableau comparatif des résultats

	complexité en messages	complexité en tours
Alg Dutta et Guerraoui	$\Omega(n^2)$	$O(t)$
Alg 1 Gilbert et al	$O(n)$	$O(n^{1+\epsilon})$
Alg 2 Gilbert et al	$O(n.log^6.n)$	$O(f)$
Notre Algorithme	$O(n.t)$	$O(t)$

Table: Tableau comparatif des résultats

Notre algorithme réduit le nombre de messages échangés et la taille du message (**taille=t+1 bits**) avec communication optimale $O(t)$.

Sommaire I

Sommaire II

1. Peut-on trouver un algorithme de consensus optimal en nombre de tours de $O(t)$ et $O(n.t)$ en nombre de messages échangés avec les contraintes *early – deciding* et *early – stopping* dans le cas de présence de défaillances franches ?
2. *Early – stopping* est-il plus coûteux en terme de communication ? [18]
3. La borne minimale en nombre de messages échangés est de $\Omega(n^2 f)$ tolérant jusqu'à t défaillances byzantines.Nous posons la question suivante : peut-on réduire sa complexité à $O(f)$ en nombre de tours et $O(n^2 f)$ en terme de messages échangés avec les contraintes *early – deciding* et *early – deciding* ?
4. Peut-on trouver un protocole de consensus indulgent pour le cas de défaillances byzantines ?
5. Peut-on trouver un protocle d'indulgence résolvant le k-accord ? si oui, peut-il être *early – deciding* et *early – stopping*

Sommaire I

Sommaire II

1) Peut-on trouver un algorithme de consensus optimal en nombre de tours de$O(t)$ et $O(n.t)$ en nombre de messages échangés avec les contraintes $early - deciding$ et $early - stopping$ dans le cas de présence de défaillances franches ?

2) La recherche du détecteur de défaillance le plus faible pour résoudre le probléme du k-accord dans les systémes distribués asynchrones.
Notre piste consiste à chercher le synchronisme minimal en se basant sur le modèle $M^{sink(x)}$.
Biely,M.,Robinson,P.,Schmid,U.,2014 The generalized loneliness detector and weak system models for k-set agreement.IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.VOL.25,NO.4.
Nous avons procédé à la comparaison des détecteurs de défaillances présentés par Raynal,M.2011 failure detectors to solve asynchronous k-set agreement.

FD class	Introduced in	Presented in Sec.	Property
Ω	[10]	4.3	Weakest for Consensus in SM
Ω_k	[32]	4.5	Solves k-set agreement in SM
Υ	[21]	3.2	Sufficient for $(n-1)$-set agreement in SM
$\overline{\Omega}_{n-1}$	[41]	3.2	Weakest for $(n-1)$-set agreement in SM
$\overline{\Omega}_k$	[33]	3.2	Weakest for k-set agreement in SM
Σ	[14]	4.2	Weakest for Register in MP
(Σ, Ω)	[15]	4.3	Weakest for consensus in MP
Σ_k	[5]	4.5	Necessary for k-set agreement in MP
$\mathcal{L}$	[16]	4.3	Weakest for $(n-1)$-set agreement in MP
$\mathcal{L}_k$	[4]	4.4	Solves k-set agreement in MP
Π_k	[5]	4.5	Same power as to (Σ_k, Ω_k)

Table 1: Global picture: failure detector classes related to k-set agreement

les résultats de la comparaison sont comme suit :

1. ℓ n'est pas le détecteur de défaillances eventuel mais $anti-\Omega$ l'est.
2. $anti-\Omega$ est plus faible que ℓ.
3. aucun détecteur de défaillances éventuel n'est plus fort que ℓ .
4. pour n=2 ℓ et $\sum$ sont équivalents.
5. ℓ est strictement faible à $\sum$.
6. ℓ n'est pas plus robuste que $\sum$ si n>2.
7. La famille π_K-$(\sum_k,\Omega_K)$ coïncide avec $(\sum,\Omega)$ et avec ℓ_n-1 .
8. $\sum$ est le plus faible pour le consensus.
9. pour n=2 ℓ et $\sum$ sont équivalents.
10. Ω et $anti-\Omega$ sont les plus faibles dans les systèmes asynchrones communiquant par passage de messages lorsque t<(n÷2) alors : Ω et $anti-\Omega$ sont équivalents .
11. $(\Omega * \sum)$ est le détecteur de défaillances le plus faible pour résoudre le consensus lorsque t< n.
12. $(\Omega^{n-1} * \sum_{n-1})$ est équivalent à ℓ.
13. La famille π_K-$(\sum_k,\Omega_K)$ coïncide avec $(\sum,\Omega)$ pour k=1.
14. $(anti-\Omega_x,\sum_x)$ est optimal pour n trés grand.

Sommaire I

Sommaire II

Références bibliographiques I

[1] Guerraoui, R., Raynal, M. : The information structure of indulgent consensus. IEEE Transactions on Computers 53(4) 2004).

[2] Gafni, E. : Round-by-round fault detectors (extended abstract) : Unifying synchrony and asynchrony. In : Proceedings of the 17th Symposium on Principles of Distributed Computing 1998

[3] Guerraoui, R. : Indulgent algorithms (preliminary version). In : Proceedings of the 19th Symposium on Principles of Distributed Computing (PODC) 2000.

[4] Lynch, N. : Distributed Algorithms.Morgan Kaufman publisher1996.

[5] Dutta, P., Guerraoui, R. : The inherent price of indulgence. Distributed Computing 2005

[6] SAMPAIO, L.,RASILEIRO, F. :Adaptive indulgent consensus. In Proceedings of the International Conference on Dependable 2005 Systems and Networks (DSN).

[7] Ben-Or M. : Another Advantage of Free Choice : Completely Asynchronous Agreement Protocols. *Proc. 2nd ACM Symposium on Principles of Distributed Computing (PODC'83)*, acm press, pp. 27-30, 1983.

[8] P. Dutta, R. Guerraoui :Fast Indulgent Consensus with Zero Degradation, EDCC ?2, in : LNCS, vol. 2485, 2002.

[9] Wu, W., Cao, J., Yang, J. and Raynal, M. : Using asynchrony and zero degradation to speed up indulgent consensus protocols. Journal of Parallel and Distributed Computing,984-996,2008.

Références bibliographiques II

[10] Chandra T.D. and Toueg S., Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2) :225-267, 1996.

[11] Dwork C., Lynch N.A. and Stockmeyer L. :Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2) :288-323, 1988.

[12] Fischer M.J., Lynch N. and Paterson M.S., Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2) :374-382, 1985.

[13] Mostefaoui, A. and Raynal, M. :Leader-Based Consensus Parallel Processing Letters, 11(1) :95-107. 2001.

[14] SAMPAIO, L. AND BRASILEIRO, F. . Adaptive indulgent consensus. In Proceedings of the International Conference on Dependable Systems and Networks (DSN05)2005.

[15] Pease L., Shostak R. and Lamport L., Reaching Agreement in Presence of Faults. *Journal of the ACM*, 27(2) :228-234, 1980.

[16] Rabin M., Randomized Byzantine Generals. *Proc. 24th IEEE Symposium on Foundations of Computer Science (FOCS'83)*, pp. 403-409, 1983.

[17] Gilbert, S., Guerraoui, R.,Kowalski, Dariusz R. : On the message complexity of indulgent consensus. A pele(Ed.) : DISC 2007, LNCS 4731, pp.283-297,2007

[18] Dolev,D.,Lenzen,C. : Early-deciding consensus is Expensive.Proceeding of the 2013 ACM symposium on principles of distributed computing page 270-279

Références bibliographiques III


[19] Fitzi,M.,Hirt, M. : Optimally Efficient Multi-valued Byzantine agreement.Proceedings of the twenty-fifth annual ACM symposium on principle of distributed computing 2006 ,page 163-168.


[20] Hurfin,M.,Raynal,M. : simple and fast synchronous consensus protocol based upon a weak failure detector.Springer verlag 1999


[21] Alistrath,D.,Gilbert, S.,Guerraoui,R.,Travers,C. : generating fast indulgent algorithms. Proceeding ICDCN'11 Proceedings of the 12th international conference on Distributed computing and networking page 41-52


[22] Aguilera, Marcos K.,Chen, W.,Toueg, S. : heartbeat a timeout-free failure detector for quiescent reliable communication.ACM 1997


[23] Chandra,T.,Hadzilacos,V.,Toueg,S. : The weakest failure detectors for solving consensus.ACM july 1996


[24] Raynal,M. : failure detectors to solve asynchronous k-set agreement : a glimpse of recent results.Bulletin of the EATCS no 103,pp.74 – 95,European Association for Theoretical Computer Science,February 2011.


[25] Freiling,F.,Guerraoui,R.,Kuznetsov,P. : the failure detector abstraction.ACM Computing Surveys,Vol.43,No,Article 9.January 2011


[26] Alistarh,D.,Gilbert,S.,Guerraoui,R.,Travers,C. :of choices failures and asynchrony the many faces of set agreement.Algorithmica 2012


[27] Chaudhuri,S. : more choices allow more faults set consensus problem in totally asynchronous system.Information and computation 1993


[28] Keidar,L.,Rajsbaum :On the cost of fault-tolerant consensus when there are no faults-a tutorial.ACM SIGACT News,2001


[29] Hurfin,M.,Mostefaoui,A.,Raynal,M. : aversatile family of consensus protocoles based on Chandra-Toueg's unreliable failure detectors.IEEE transactions 2002


[30] Mostefaoui,A.,Raynal,M. :solving consensu using Chandra-Toueg's unreliable failure detectors : a general quorum based approach.Springer 1999


[31] Schiper,A. :early consensus in asynchronous system with a weak failure detector.Distributed computing,Sptringer 1997


[32] Mostefaoui,A.,Rajsbaum,S.,Raynal,M. :Conditions on input vectors for consensus solvability in asynchronous distributed systems.ACM 2003


[33] Biely,M.,Robinson,P.,Schmid,U. : The generalized loneliness detector and weak system models for k-set agreement.IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.VOL.25,NO.4, April 2014


[34] Cristian,F.,Fetzer C. : The timed asynchronous distributed system model.IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS 1999