Les objets dégradables

Boubacar KANE, Tuteur : Pierre SUTRA

Université de Versailles Saint Quentin

12 Septembre 2019

Plan

1 Introduction

2 Définition d'un objet dégradable

3 Différents objets partagés

4 Exemple d'objet dégradable : le compteur

5 Conclusion

Contexte

- Algorithmie distribuée
  - Implémentation sans-attente
- Modèle à mémoire partagée

1 Introduction

2 Définition d'un objet dégradable

3 Différents objets partagés

4 Exemple d'objet dégradable : le compteur

5 Conclusion

Définition d'un objet dégradable

- Structure de données **variable**
- Modifier les *pré-conditions* et/ou les *post-conditions*
- Niveaux de dégradation
- Exemple : calcul de la vitesse d'une voiture

Problème de consensus

- Cohérence forte ou faible
- Exemple : calcul de la vitesse d'une voiture
- Pouvoir de consensus :
  - Nombre de processeurs pour lequel un objet partagé peut résoudre le problème de consensus
- Exemple d'un protocole de consensus :

Problème de consensus

Exemple d'un protocole de consensus pour un système à n processeurs

Algorithm 1 $Propose(obj : object, v : value)$**returns**$(value)$

Require: $v \in \mathbb{N}$

```
if obj.val = ⊥ then
    obj.val ← v
end if
return obj.val
```

1 Introduction

2 Définition d'un objet dégradable

3 Différents objets partagés

4 Exemple d'objet dégradable : le compteur

5 Conclusion

Terminologie

- **État** d'un système
- Un état 0-**valent** ou 1-**valent**
- Un état **bivalent**
- Un état **décisionnel**

L'objet registre atomique

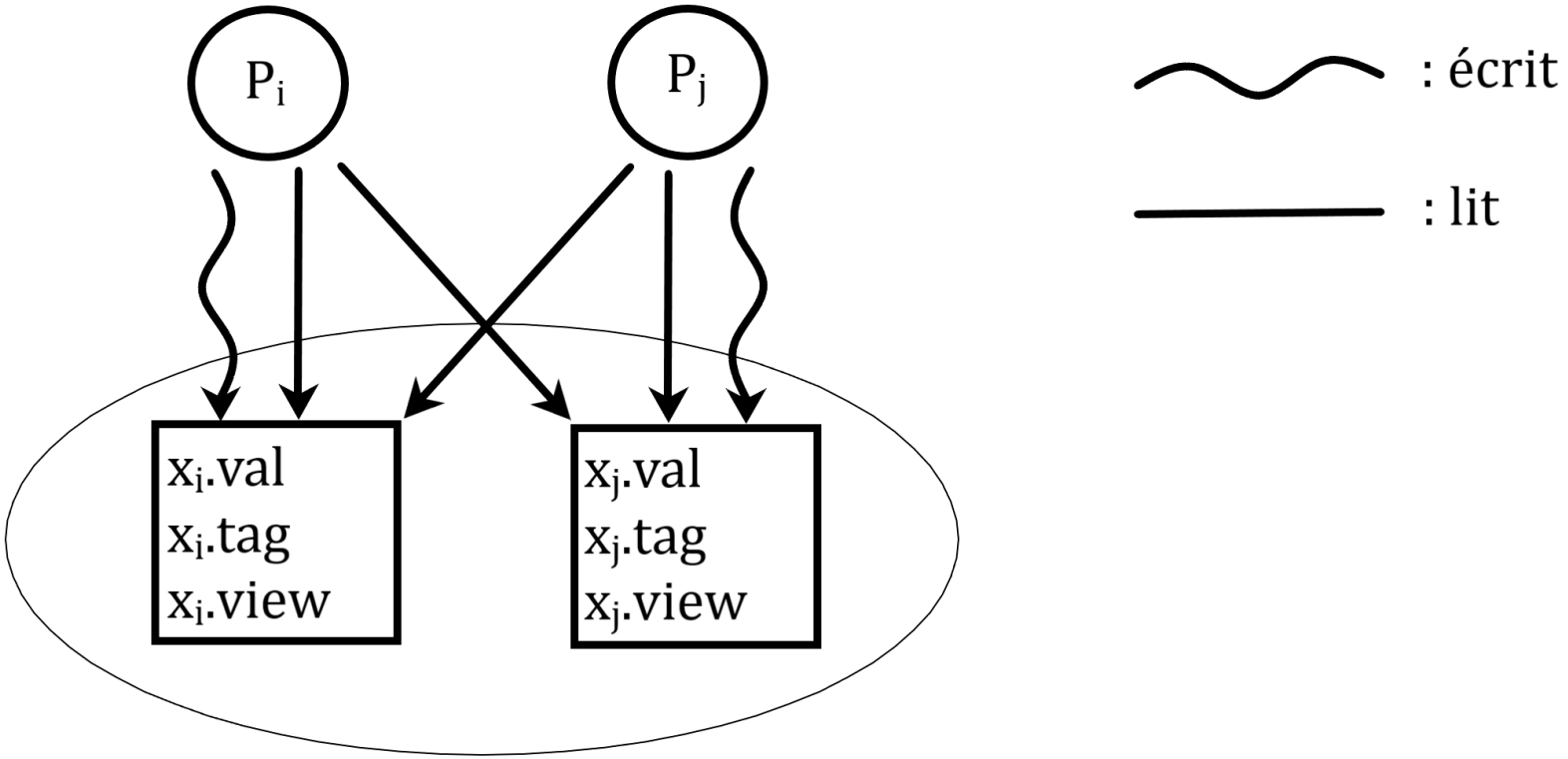
- Deux opérations : *read*() et *write*(v)
- Pouvoir de consensus de 1

L'objet *snapshot*

- Capture instantanée de l'état globale d'un système distribué
- Opérations exécutées de manière **atomique**
- Deux opérations : $update(i, w)$ et $snap()$

L'objet *snapshot*

Illustration de l'objet *snapshot* dans un système à deux processeurs



L'objet *snapshot*

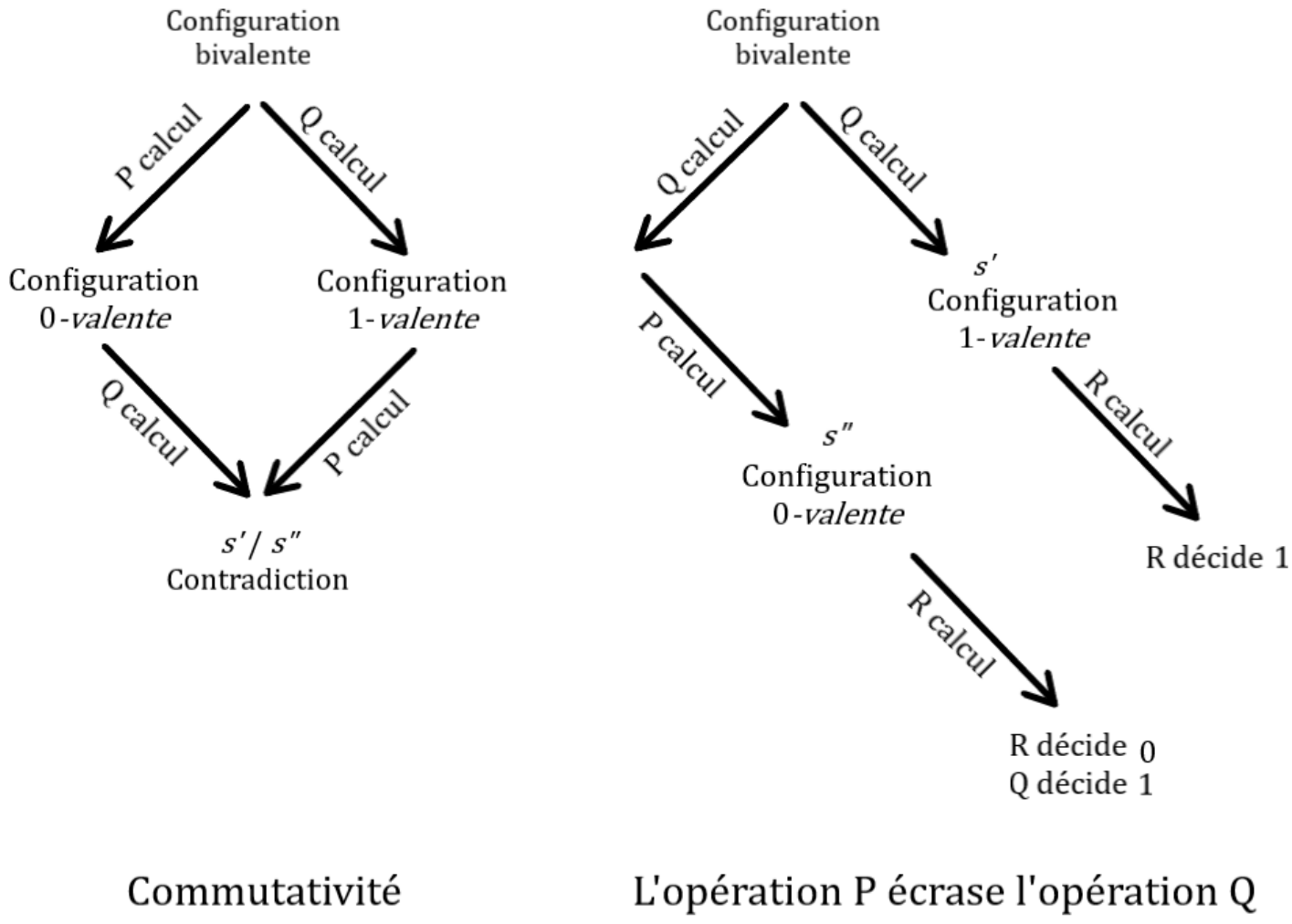
- Capture instantanée de l'état globale d'un système distribué
- Opérations exécutées de manière **atomique**
- Deux opérations : $update(i, w)$ et $snap()$
- Pouvoir de consensus de 1

Les objets *Read-Modify-Write*

- Objets bien connus : *test & set*, *compare & swap*, *fetch & add* ...
- Une opération : $RMW(r, f)$
- f est dans un ensemble **interférent**

Les o

Illustration de la preuve du nombre de consensus des objets *Read-Modify-Write*



Les objets *Read-Modify-Write*

- Objets bien connus : *test & set*, *compare & swap*, *fetch & add* ...
- Une opération : $RMW(r, f)$
- f est dans un ensemble **interférent**
- Pouvoir de consensus de 2

1 Introduction

2 Définition d'un objet dégradable

3 Différents objets partagés

4 Exemple d'objet dégradable : le compteur

5 Conclusion

L'objet compteur

- Un objet que l'on peut incrémenter et consulter
- Trois niveaux de dégradabilités
  - Pouvoir de consensus de 1
  - Pouvoir de consensus de 2
  - Pouvoir de consensus de n

L'objet compteur (Pouvoir de consensus de 1)

- Utilisation de l'objet *snapshot*
- Spécification :
  - $increment() : \perp$
  - $get() : \mathbb{N}$

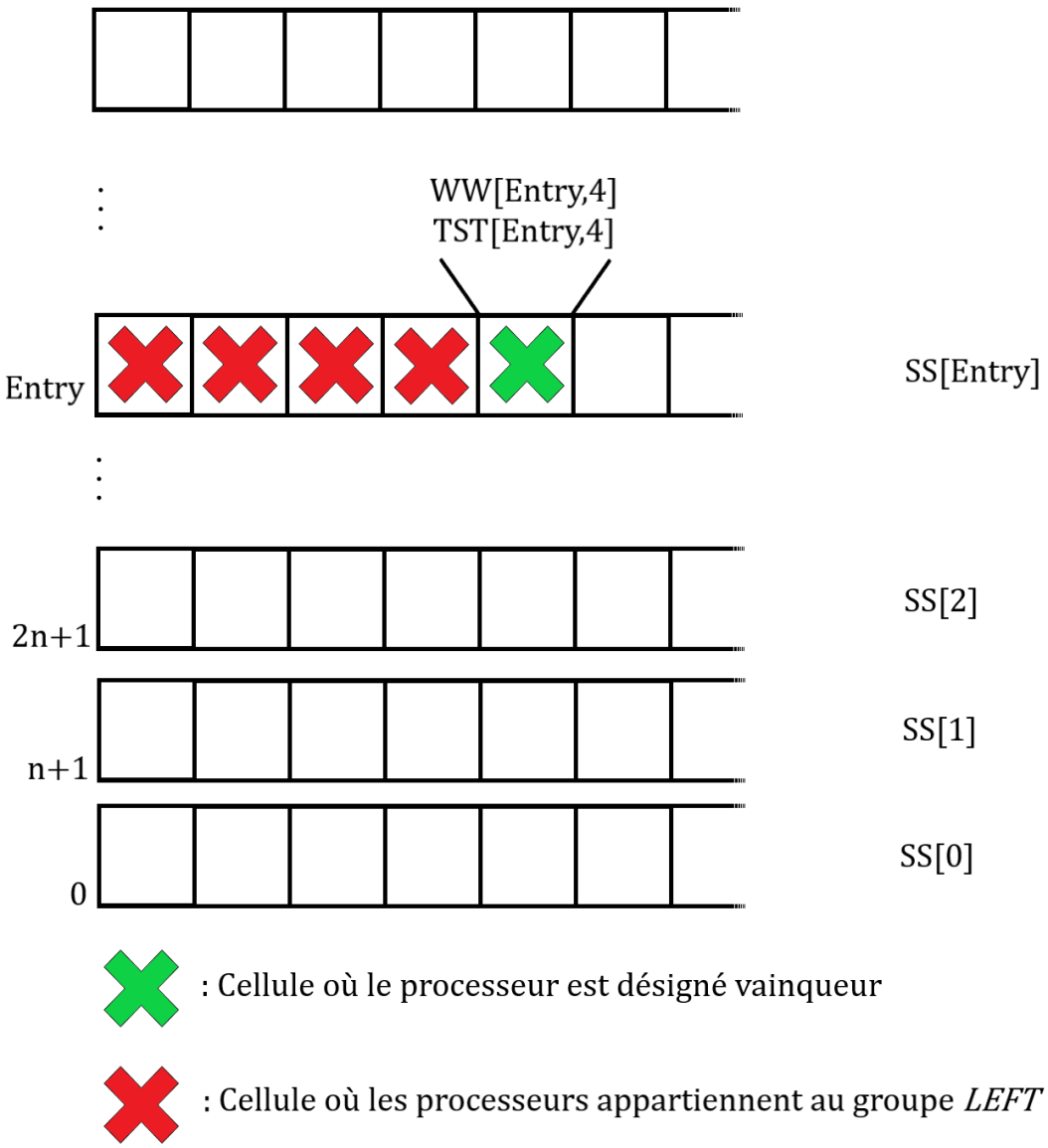
L'objet compteur (Pouvoir de consensus de 2)

- Utilisation de l'objet *fetch&add*
- Spécification :
  - $increment() : \mathbb{N}$
  - $get() : \mathbb{N}$

L'objet compteur (Pouvoir de consensus de 2)

- Utilisation des objets *test&set* et *wigwag* pour implémenter l'objet *fetch&add*
- L'objet *test&set* :
  - Une opération : $T\&S()$
- L'objet *wigwag* :
  - Deux opérations : $wig - Pass(w, i, val)$ et $wig - Block(w, i, vec)$

Schéma de l'objet *fetch&add*



L'objet compteur (Pouvoir de consensus de n)

- Utilisation de la construction universelle décrite par Herlihy [2]
- Spécification :
  - $increment() : \perp$
  - $seal(); \perp$
  - $get() : \mathbb{N}$

1 Introduction

2 Définition d'un objet dégradable

3 Différents objets partagés

4 Exemple d'objet dégradable : le compteur

5 Conclusion

Conclusion

- Construire un objet dégradable avec un pouvoir de consensus de k
- Faire la même recherche dans un modèle à passage de messages

Sources

[1] Y. Afek, E. Weisberger, and H. Weisman. A completeness theorem for a class of synchronization objects. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Distributed Computing*, PODC '93, pages 159–170, New York, NY, USA, 1993. ACM. ISBN 0-89791-613-1. doi : 10.1145/164051.164071. URL http://doi.acm.org/10.1145/164051.164071.

[2] M. Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1) :124–149, Jan. 1991. ISSN 0164-0925. doi : 10.1145/114005.102808. URL http://doi.acm.org/10.1145/114005.102808.

[3] L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans. Comput.*, 28(9) :690–691, Sept. 1979. ISSN 0018-9340. doi : 10.1109/TC.1979.1675439. URL https://doi.org/10.1109/TC.1979.1675439.

[4] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996. ISBN 9780080504704.