

Plan

- Introduction aux algorithmes distribués
- Algorithme de Naimi-Trehel
- Extensions tolérantes aux pannes
- Implémentation & démonstration
- Conclusion

INTRODUCTION AUX ALGORITHMES DISTRIBUÉS

Types d'algorithmes

- Élection
- Exclusion mutuelle
- Terminaison

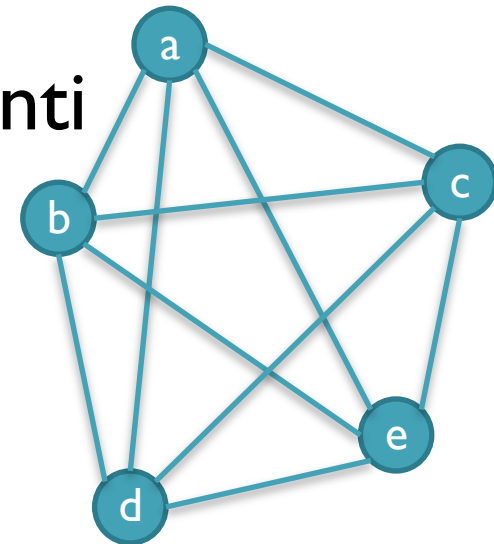
Critères d'efficacité

- Vivacité
- Sûreté
- Respect de l'ordre des demandes
- Tolérance aux pannes
- Complexité
 - Nombre de messages
 - Temps d'exécution

ALGORITHME DE NAIMI-TREHEL [1]

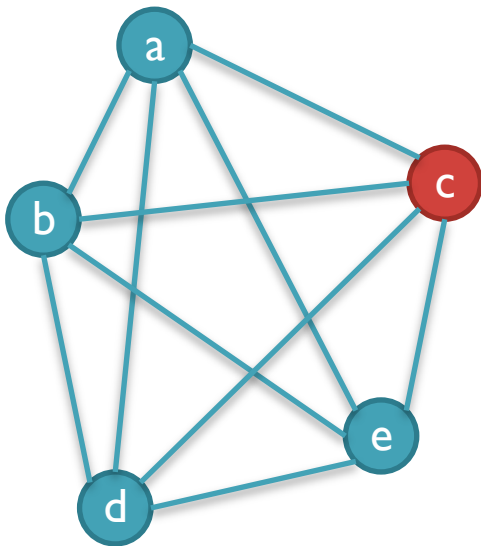
Modèle du réseau

- réseau distribué
- graphe complet à n nœuds
- canaux de communication fiables
- ordre des messages non garanti
- réseau synchrone
- temps de transmission borné

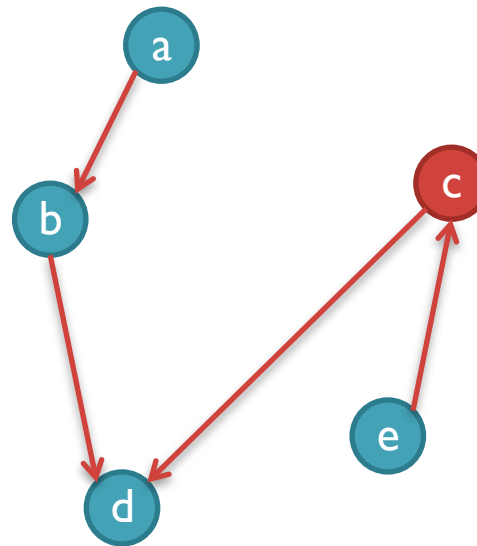


Structures de données

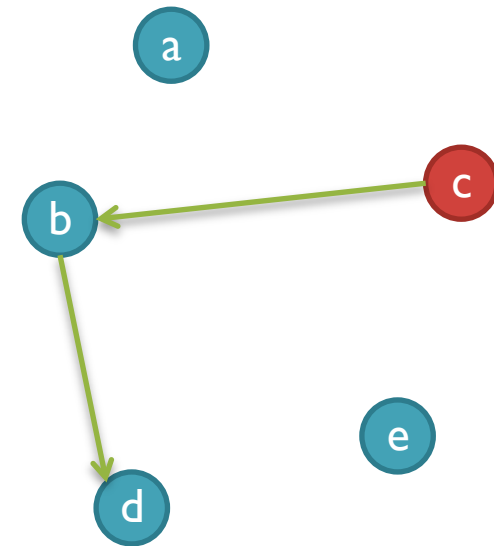
réseau physique



arbre logique

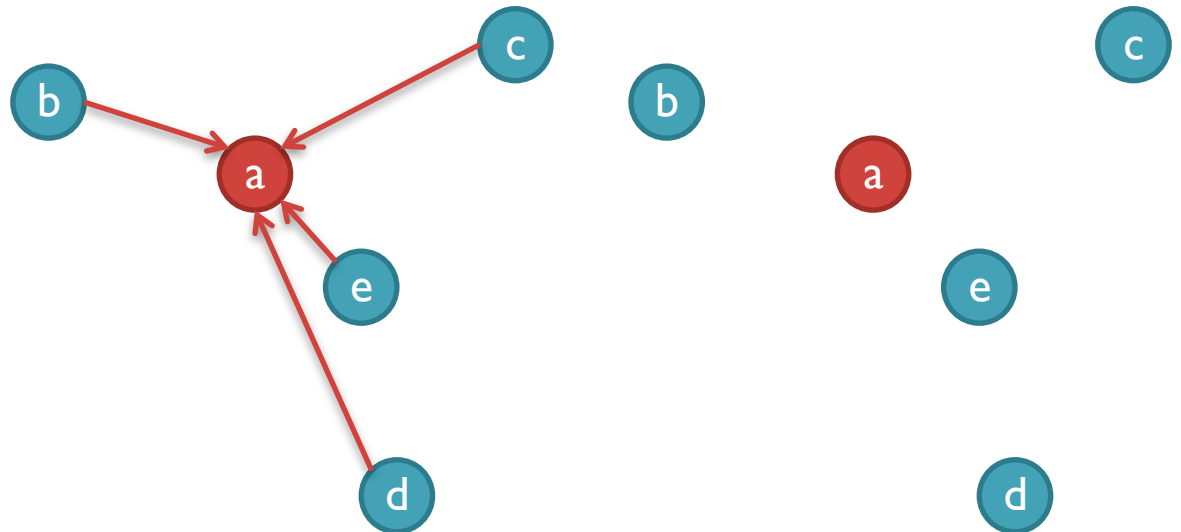
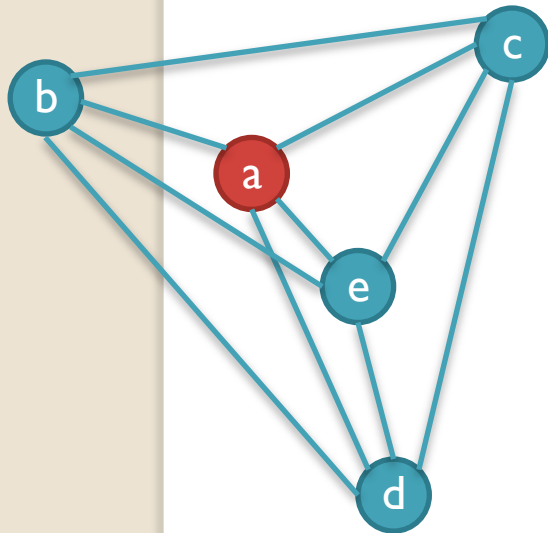


file d'attente



Exemple

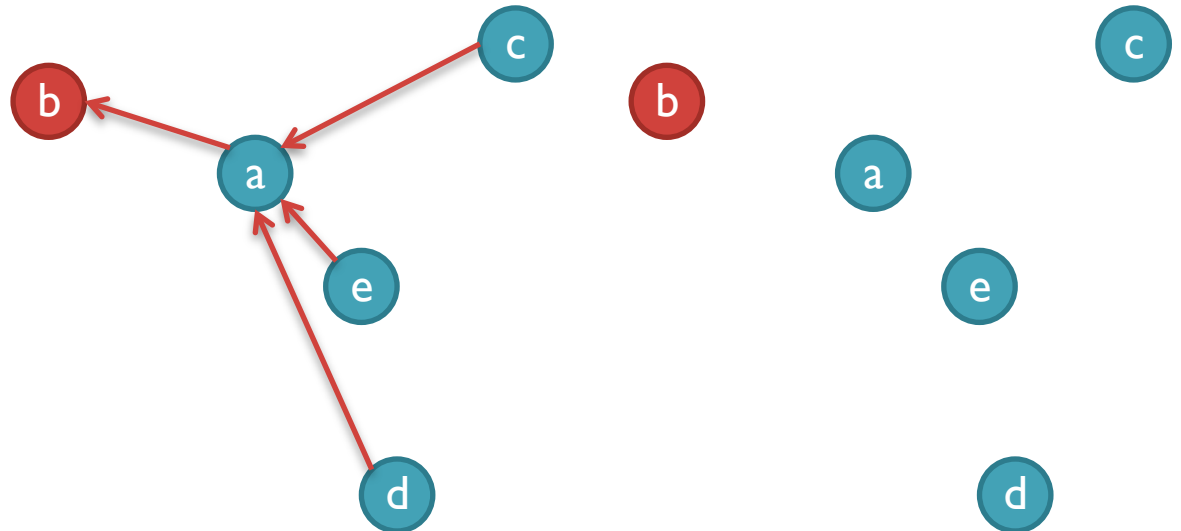
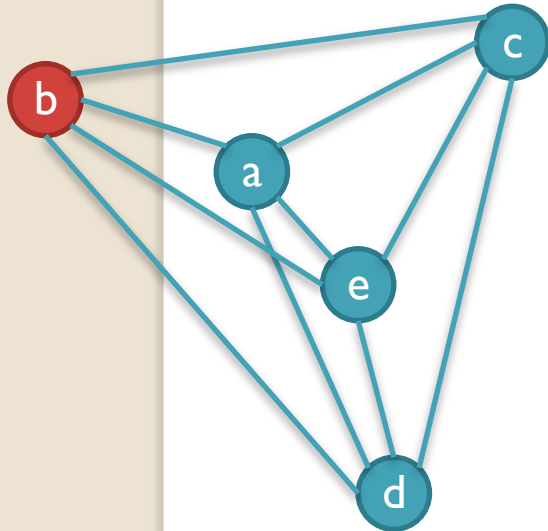
- Étape initiale : *a* possède le jeton



- *b* fait une demande de section critique

Exemple

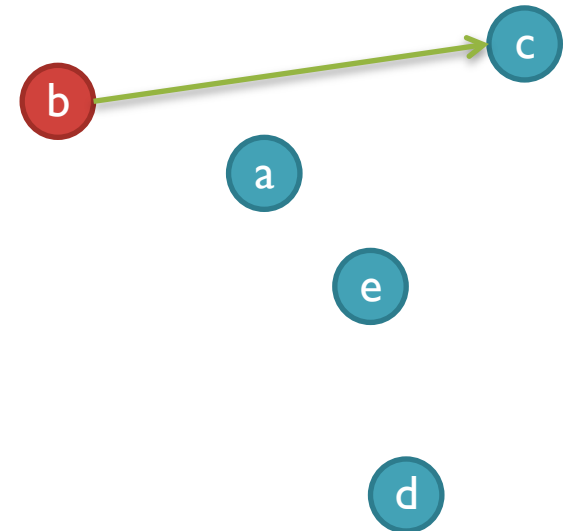
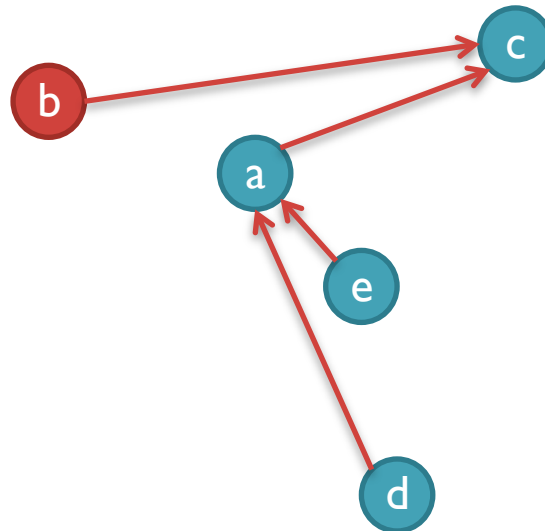
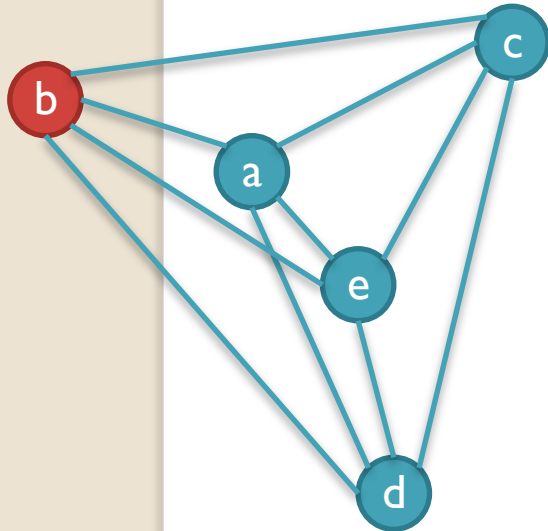
- *b* fait une demande de section critique



- *c* fait une demande de section critique

Exemple

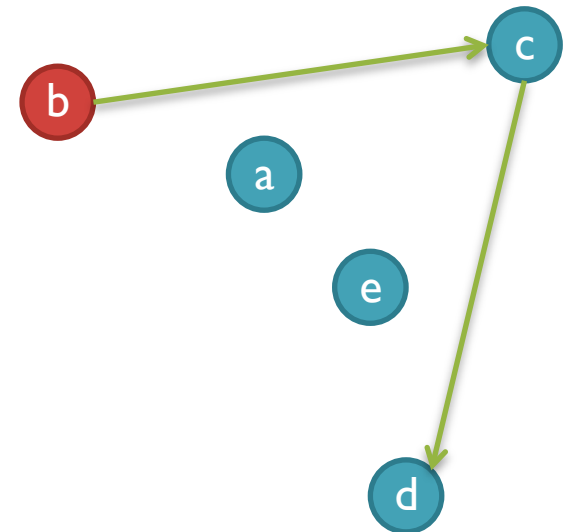
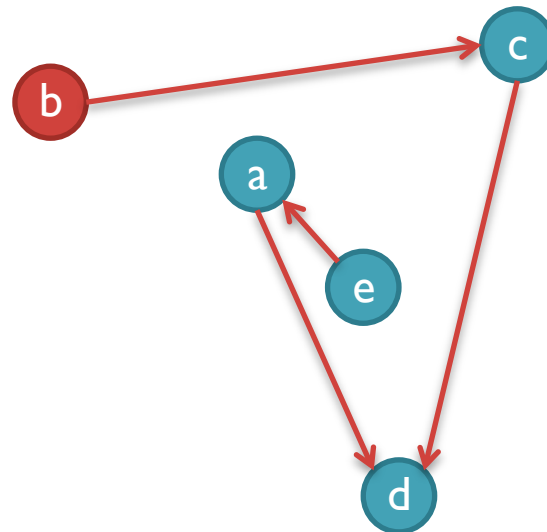
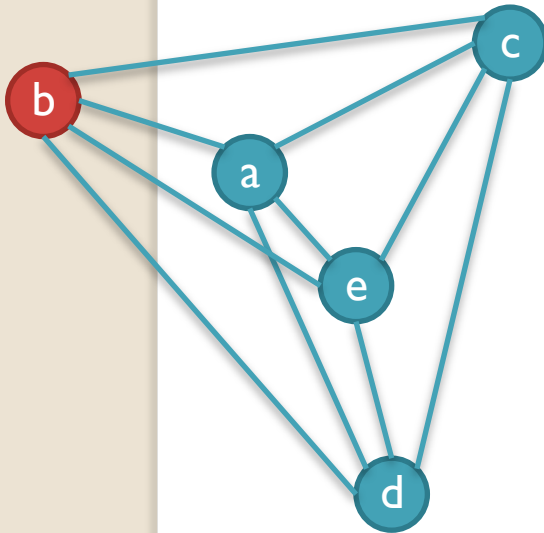
- c fait une demande de section critique



- d fait une demande de section critique

Exemple

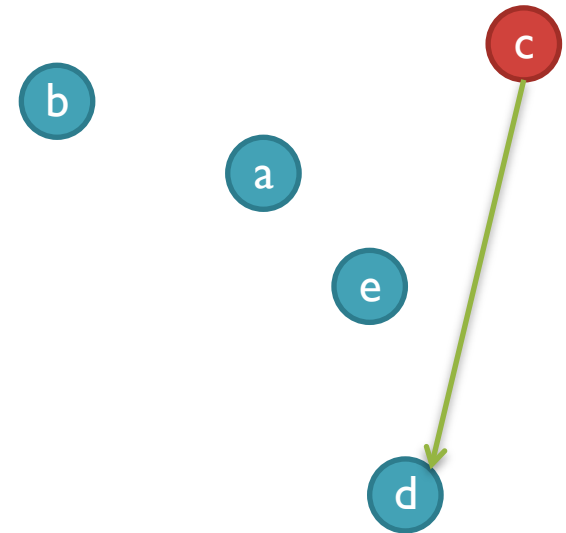
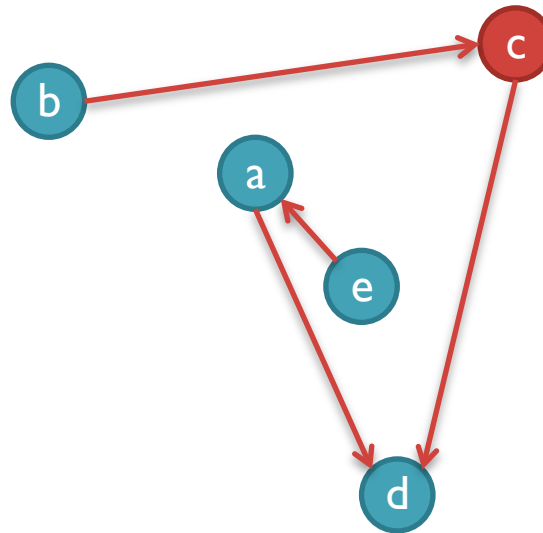
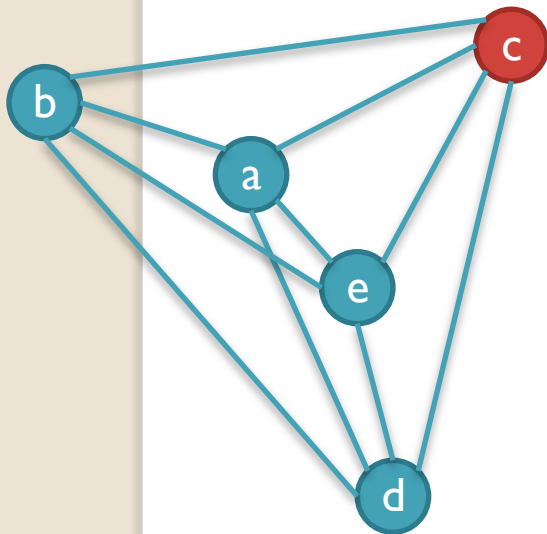
- *d* fait une demande de section critique



- *b* relâche la section critique

Exemple

- *b* relâche la section critique et envoie le jeton à *c*



Sureté

- au plus un processus en SC simultanément
- garantie par le jeton
 - unicité initiale
 - un site qui envoie le jeton le perd pour lui-même

Vivacité

- une demande abouti en temps fini
- garantie par les structures de données
 - demande retransmise dans l'arbre
 - ajout dans la file
 - exécution de la SC en temps fini : demande aboutie

Complexité

- nombre de messages pour que la demande atteigne la racine de l'arbre : M_n
- $M_n \cong H_{n-1} \cong \log(n-1) + 0.577$
- Nombre moyen de messages : $O(\log(n))$
- Temps : $(N - 1) \times (T_{\text{msg}}/T_{\text{cs}})$

EXTENSIONS TOLÉRANTES AUX PANNES

Extension de Naimi-Trehel ^[2]

- Détection de panne : demande de SC avec timer
- Vérification et recherche du jeton
- Élection
- Construction de l'arbre et de la file

Efficacité

- Complexité
 - 4 diffusions
- Connaissance requise
 - temps d'exécution de la SC

Extension de Sopena^[3]

- Accusé de réception à chaque demande
- Conservation de la position dans la file
- Conservation des k prédécesseurs
- Vérification périodique du prédécesseur

Algorithme

- Détection de panne
 - vérification régulière
 - accusé de réception non reçu
- Mécanismes
 - M1 : AR reçu et $< k$ pannes consécutives
 - M2 : AR reçu et $> k$ pannes consécutives
 - M3 : AR non reçu

Mécanisme MI

- Panne du prédécesseur
- Envoi de ARE_YOU_ALIVE aux préd.
- Réception de I_AM_ALIVE depuis j
- j devient le nouveau prédécesseur

Mécanisme M2

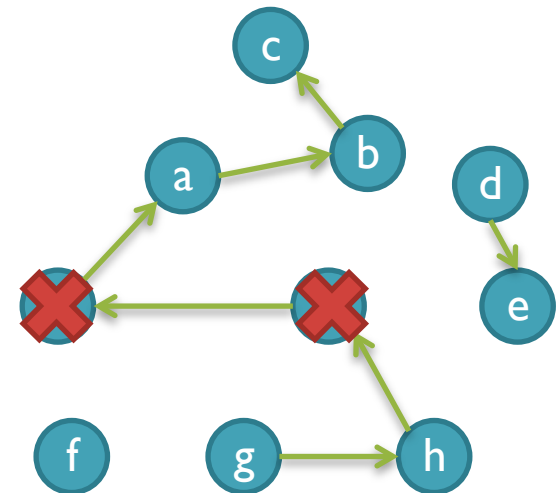
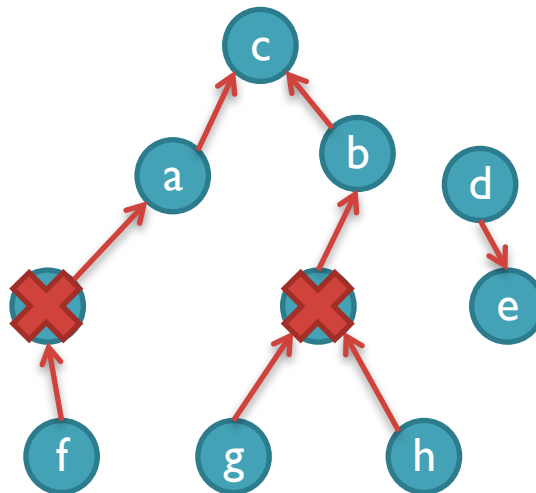
- Panne du prédécesseur
- Envoi de ARE_YOU_ALIVE aux préd.
- Pas de réponse
- Diffusion de SEARCH_PREV
 - réponse : envoi de CONNECTION
 - pas de réponse : régénération du jeton

Mécanisme M3

- Message de COMMIT non reçu
- Détection de la panne
 - a) par un seul site
 - diffusion de SEARCH_QUEUE
 - réponse par ACK_SEARCH_QUEUE par j
 - re-connection à j
 - b) par plusieurs sites
 - élection puis a)
- Reconstruction de l'arbre

Exemple

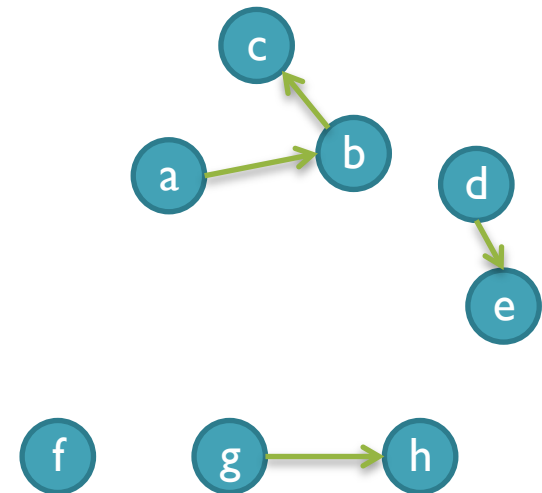
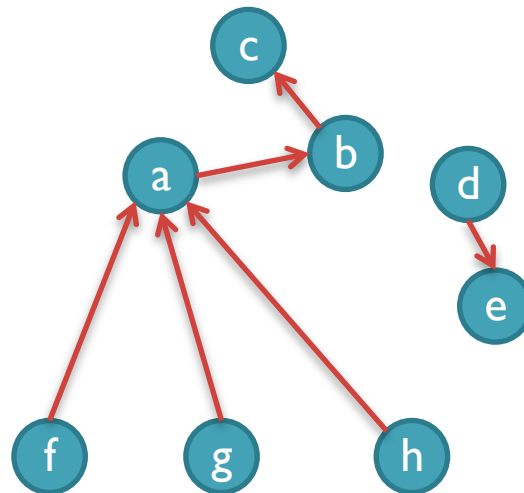
- Situation initiale : deux sites en panne



- *a* et *d* détectent une panne simultanément

Exemple

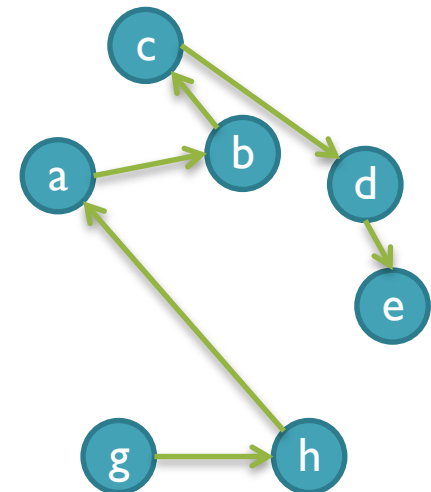
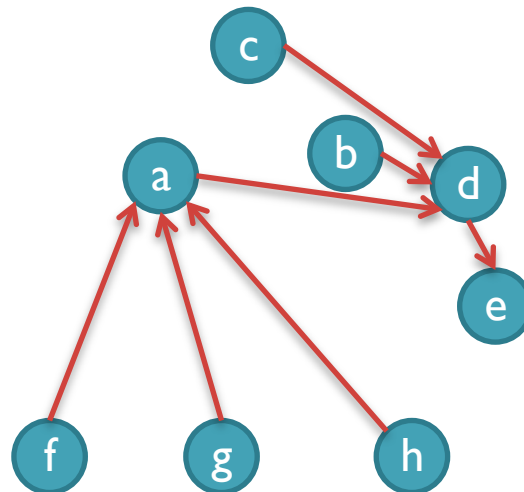
- *a* et *d* détectent une panne simultanément
- ils diffusent SEARCH_QUEUE



- *a* est élu

Exemple

- a reçoit `ACK_SEARCH_QUEUE` depuis h
- a envoie `CONNECTION` à h



- d demande la section critique à a

Propriétés

- Vivacité
 - garantie par la restauration de la file
- Sureté
 - garantie par l'unicité du jeton
- Ordre des messages
 - garantie par la restauration de la file

Complexité

- Nombre de messages
 - un accusé par demande de SC
 - diffusion de SEARCH_PREV en cas de panne
 - renvoie des requêtes perdues
- Temps
 - $((N - 1) + 1) \times T_{\text{msg}}$

IMPLÉMENTATION

Stratégie & mise en œuvre

- Proche des conditions réelles
- Langage C
- Bibliothèque standard pour le réseau
- Protocole UDP

Phase d'initialisation

- Diffusion d'un message de type HELLO
- Réception d'une réponse
 - Récupération du père dans l'arbreou
 - Élection
- Pas de réponse
 - Génération du token

Implémentation de Naimi-Trehel

- Mise en place de deux sockets
- Attente non-bloquante
- Gestion des mécanismes grâce aux protocoles réseau

Implémentation de l'extension

- thread : vérifie si le prédécesseur en vie
 - signal pour la communication en cas de panne
- Simulation de panne avec signaux

Problème concret : partition

- Ressource présente sur un site sup.
- Thread de calcul dans chaque site
- Demande de SC à la fin du calcul
- Recherche de solution approchée

Problème rencontrés

- Mise en place de la phase d'élection avec pertes de messages
- Difficultés de debug sur une architecture distribuée
- Gestion de l'adressage mémoire sur le réseau

DÉMONSTRATION

BIBLIOGRAPHIE

- [1] M. Naimi, M. Trehel, and A. Arnold. A log (n) distributed mutual exclusion algorithm based on path reversal. *Journal of Parallel and Distributed Computing*, 34(1) :1–13, 1996.
- [2] Mohamed Naimi and Michel Trehel. How to detect a failure and regenerate the token in the log (n) distributed algorithm for mutual exclusion. *Distributed Algorithms*, pages 155–166, 1988.
- [3] J. Sopena, L. Arantes, M. Bertier, and P. Sens. A fault-tolerant token-based mutual exclusion algorithm using a dynamic tree. *Euro-Par 2005 Parallel Processing*, pages 644–644, 2005.