Arbre couvrant en largeur avec détection de terminaison

Alain Cournier et Stéphane Devismes

Hypothèses

- Processus et canaux asynchrones
- Canaux FIFO
- Pas de fautes
- Topologie connexe quelconque d’au moins deux noeuds
- Mono-initiateur

Idée générale

- On modifie ***ACL*** pour y ajouter la détection de terminaison

Principes

1. Accusés de réception

    1. Deux types de messages : *Bâtir(Val)* et *Répond(Val)*

        1. *Bâtir(Val)* : Val = estimation de la distance à l'initiateur de l'émetteur au moment de l'envoi

        2. *Répond(Val) :* identifié le message *Bâtir* dont le noeud accuse réception (*Répond(Val)* ignoré si *Val* ≠ *Dist*)

Principles

2. Pour la détection de terminaison, les non-initiateurs accusent réception à leur père en dernier (comme dans *ACDT*)

    1. Compteur *NbRep* : *Répond(Dist-1)* au père si *NbRep* = |*Voisins*| - 1

    2. Attention au interblocage : lorsqu'un processus non-initiateur reçoit une valeur n'améliorant pas son estimation de distance, il accuse quand même la réception de la valeur

        (l'initiateur ignore les messages *Bâtir*)

    3. Quand l'initiateur a reçu des accusés réceptions de tous ses voisins : FIN

Principles

3. Changement d'estimation de distance (non-initiateurs)
    1. Réception de *Bâtir(Val)* avec *Val+1 < Dist*
        1. Réinitialiser *NbRep*
        2. MAJ *Père*
        3. MAJ *Dist* et transmission aux voisins (sauf *Père*)
            1. Attention : en cas de degré 1 => accusé réception au père
        4. En cas de changement de père, il faut accuser réception à l'ancien père de la valeur qu'il a envoyé si cela n'a pas encore été fait

Algorithme 17 Algorithme $ACLDT$, code pour l'initiateur

Messages

1: $Bâtir(Valeur$: entier naturel$)$
2: $Répond(Valeur$: entier naturel$)$

Constantes

3: $Voisins$: ensemble de canaux
4: $Dist$: entier naturel égal à 0

Variables

5: $NbRep$: entier naturel initialisé à 0
6: Val : entier naturel
7: C : canal

Spontanément

8: `Envoyer` $Bâtir(Dist)$ `à` $Voisins$

Réception de $Bâtir(Val)$ **de** C

9: Rien

Réception de $Répond(Val)$ **de** C

10: $NbRep \leftarrow NbRep + 1$
11: **Si** $NbRep = |Voisins|$ **alors**
12: C'est fini !
13: **Fin Si**

Algorithme 18 Algorithme $ACLDT$, code pour les suiveurs

Messages

1: $Bâtir(Valeur$: entier naturel)
2: $Répond(Valeur$: entier naturel)

Constantes

3: $Voisins$: ensemble de canaux

Variables

4: C, $Père$: canaux
5: $Dist$: entier naturel initialisé à ∞
6: Val : entier naturel
7: $NbRep$: entier naturel initialisé à 0

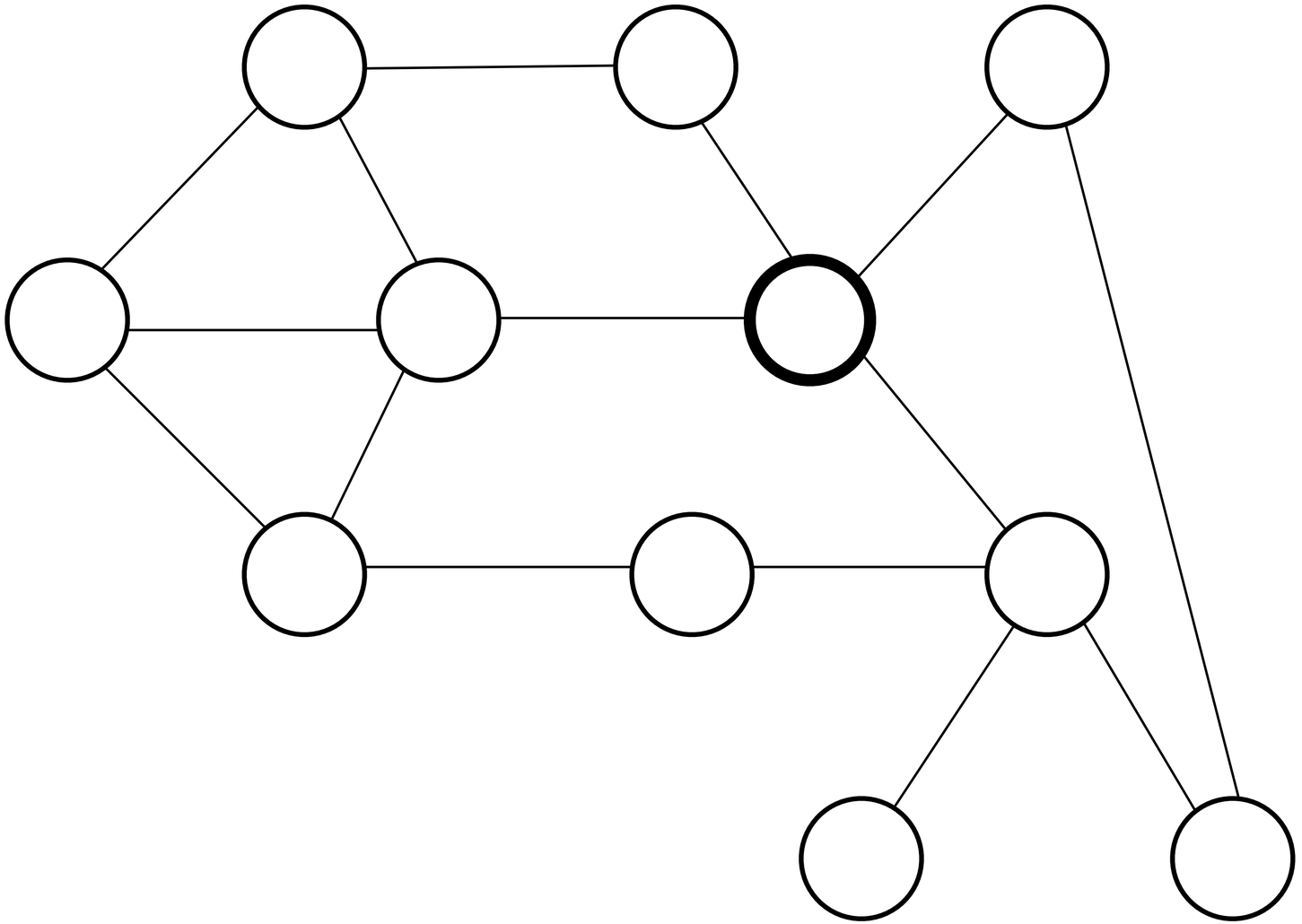
Réception de $Bâtir(Val)$ **de** C

8: **Si** $Dist > Val + 1$ **alors**
9: **Si** $Dist \neq \infty \wedge C \neq Père \wedge NbRep \neq |Voisins| - 1$ **alors**
10: `Envoyer` $Répond(Dist - 1)$ `à` $Père$
11: **Fin Si**
12: $Père \leftarrow C$
13: $Dist \leftarrow Val + 1$
14: $NbRep \leftarrow 0$
15: **Si** $Voisins \setminus \{C\} \neq \emptyset$ **alors**
16: `Envoyer` $Bâtir(Dist)$ `à` $Voisins \setminus \{C\}$
17: **Sinon**
18: `Envoyer` $Répond(Val)$ `à` $Père$
19: **Fin Si**
20: **Sinon**
21: `Envoyer` $Répond(Val)$ `à` C
22: **Fin Si**

Réception de $Répond(Val)$ **de** C

23: **Si** $Val = Dist$ **alors**
24: $NbRep \leftarrow NbRep + 1$
25: **Si** $NbRep = |Voisins| - 1$ **alors**
26: `Envoyer` $Répond(Dist - 1)$ `à` $Père$
27: **Fin Si**
28: **Fin Si**

Exemple d’exécution



Complexité en messages

- #*Bâtir* = #messages *ACL* $\leq (n-1) \times 2 \times m$
- Pire des cas : 1 message *Répond* par message *Bâtir* : $\leq (n-1) \times 2 \times m$
- Total : $\leq 4 \times (n-1) \times m = O(n \,.\, m)$

Complexité en temps

- Similaire à *ACDT* ($\leq \mathscr{D} + H + 1$) mais $H \leq \mathscr{D}$
- D'où $\leq 2\mathscr{D} + 1 = O(\mathscr{D})$

Complexité en mémoire

- $O(\log \Delta + \log \mathscr{D})$ bits par processus

FIN