Distributed System

Master ALMA 2eme année

Causal Order Message Delivery : From Logical to Physical Time

Encadrant : M.Achour

A.Marguerite
R.Rincé

Université de Nantes
2 rue de la Houssinière, BP92208, F-44322 Nantes cedex 03, FRANCE

Table des matières

1 Logical Time

1.1 Question 1 Design a logical-time base algorithm

Posons les variables du processus suivantes :

- `ListMessages` : représente la liste des messages qui ont été reçu mais pas délivrés.
- `m.i` indique l'identifiant du processus à l'origine du message.
- `m.sn` indique le prédécesseur du message envoyé par le processus émetteur.
- `m.cb` Liste des prédécesseurs du message.

Nous choisissions délibérément un langage pseudo-code verbeux pour justifier l'algorithme.

Algorithm 1 CO_Receive(Message m

```
1:  if ∀ p∈ m.cb | p.sn ≤ del(p.i) then
2:     del(m.i) ++
3:     cb +={m.sn, m.i}
4:     CO_Deliver(m)
5:     for all m´ ∈ ListMessages do
6:        aa
7:     end for
8:  else
9:     ListMessages.add(m)
10: end if
```

La fonction `CO_Receive2(Message m)` est la fonction suivante :

Algorithm 2 CO_Receive2(Message m)

```
1:  if ∀ p∈ m.cb | p.sn ≤ del(p.i) then
2:     del(m.i) ++
3:     cb +={m.sn, m.i}
4:     CO_Deliver(m)
5:     ListMessage.delete(m)
6:     for all m´ ∈ ListMessages do
7:        CO_Receive2(m´ )
8:     end for
9:  end if
```

1.2 Question 2 Analyse of the algorithm

Les valeurs que peuvent prendre cb sont les suivantes :

Minimun ??

Maximun nombre de messages **??**

Validité Les canaux étant sûr, aucun messages de ne peut être « perdu ». En effet il n'y a pas de parasites dans les canaux.

Intégrité Dans l'algorithme **??** si un message est reçu et qu'il est délivré immédiatement alors il n'est lu qu'une fois. Au contraire s'il n'est pas délivré il est mis en attente et sera supprimé dès qu'il sera délivré (*cf.* ligne de l'algoritme 5.

CO_Delivry Cette propriété est vérifiée grâce à la condition à la ligne de l'algoritme **??**

Terminaison On ne peut pas perdre de messages car les canaux sont sur.

Regardons si les propriétés blah blah

Validité Les canaux étant sûr, aucun messages de ne peut être « perdu ». En effet il n'y a pas de parasites dans les canaux.

Intégrité Dans l'algorithme **??** si un message est reçu et qu'il est délivré immédiatement alors il n'est lu qu'une fois. Au contraire s'il n'est pas délivré il est mis en attente et sera supprimé dès qu'il sera délivré (*cf.* ligne de l'algoritme 5.

CO_Delivry Cette propriété est vérifiée grâce à la condition à la ligne de l'algoritme **??**

Terminaison On ne peut pas perdre de messages car les canaux sont sur.

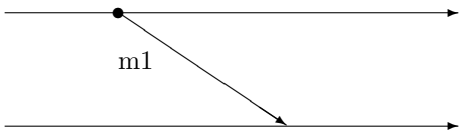

FIGURE 1.1 – blah

2 From Logical Time to Physical Time

2.1 Question 3 Modify your algorithm from logical time to physical time

Posons les variables globales du system
- `GlobalClock` : Horloge globale du système.

Redéfinissions la procedure `CO_Brodcast` : 3.

Algorithm 3 CO_Broadcast()

```
/* Avec horloge globale */
1: Broadcast(m,cb)
2: S_n ← GC.clock
3: cb ← {(sn, i}
```

Algorithm 4 CO_Receive(Message m)

```
/* Avec horloge globale */
1: if ∀ p∈ m.cb | p.sn ≤ del(p.i) then
2:    del(m.i) ← m.sn
3:    CO_Deliver(m)
4:    for all m´ ∈ ListMessages do
5:       aa
6:    end for
7: else
8:    ListMessages.add(m)
9: end if
```

Algorithm 5 CO_Receive2(Message m)

```
/* Avec horloge globale */
1: if ∀ p∈ m.cb | p.sn ≤ del(p.i) then
2:    del(m.i) ← m.sn
3:    CO_Deliver(m)
4:    ListMessages.delete(m)
5:    for all m´ ∈ ListMessages do
6:       aaa
7:    end for
8: end if
```

3 Messages with Bounded Lifetime

3.1 System model : messages with a bounded lifetime

3.1.1 Question 4 Extended the previous algorithm

Algorithm 6 Send_Time()

```
/* Avec horloge globale */
1: if ∃ m ∈ ListeMessages | { m.send_time + Δ = GC.clock() } then
2:    CO_Deliver(m)
3:    for all p ∈ m.pred do
4:       if del(p.i) ≤ p.sm then
5:          del(p.i) ← p.sn
6:       end if
7:    end for
8: end if
```

Algorithm 7 CO_Recceive_With_Delay(Message m)

```
/* Au moment où on reçoit un message */
1: if m.send_time + Δ ≤ GC.Clock then
2:    Delete(m)
3: else {∀ p∈ pred | p.sn ≤ del(p.i)}
4:    del(e.i) ← e.sn
5:    cb +={e.sn, e.i}
6:    CO_Deliver(m)
7:    for all m´ ∈ ListMessages do
8:       aaa
9:    end for
10: else
11:    ListMessages.add(m)
12: end if
```

3.1.2 Question 5 Analize the algorithm

blah blah blah blah blah blah blah blah

Table des figures