Le 17/2/2014

KEMALI Ismail
SALAMAT Mohamed
BEN EL HADJ Ali

Algorithmes distribués
2013-2014
Devoir 1

<u>Question 1:</u>

Le réseau est représenté par un graphe orienté G =< Π, V > fortement connexe dans la Figure 1 avec:

Π = {1, 2, 3, 4, 5}
V = {(1, 2), (2, 1), (2, 3), (3, 2), (3, 4), (4, 3), (4, 5), (5, 4)}

Où Π est l'ensemble de nœuds (processus) et V l'ensemble d'arêtes (messages).
Ici le diamètre du graphe est 4 car D = max{d(p, q)|p, q ∈ Π} = d(1,5) = d(5,1) = 4

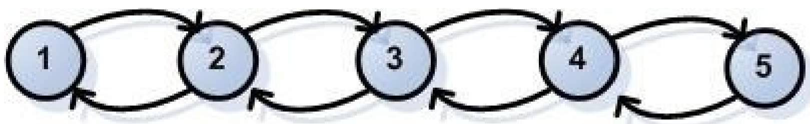
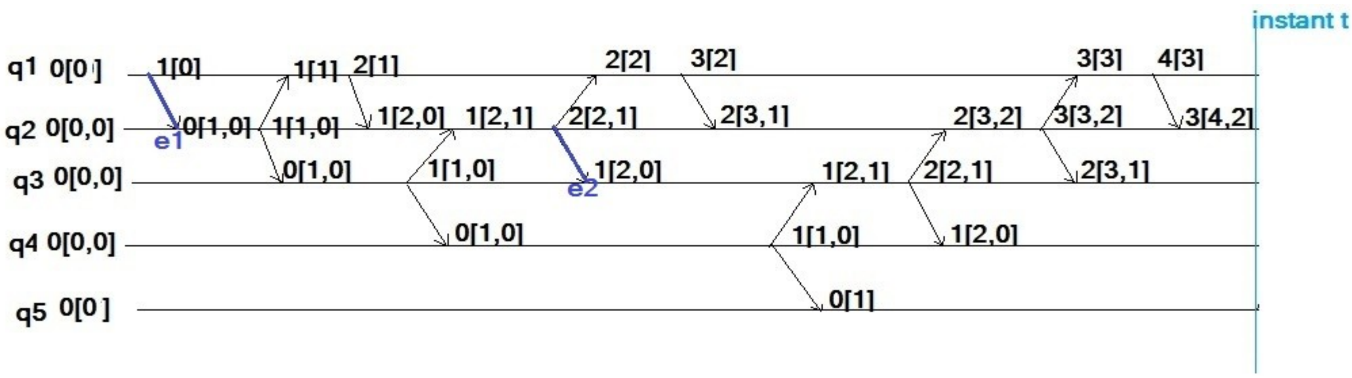
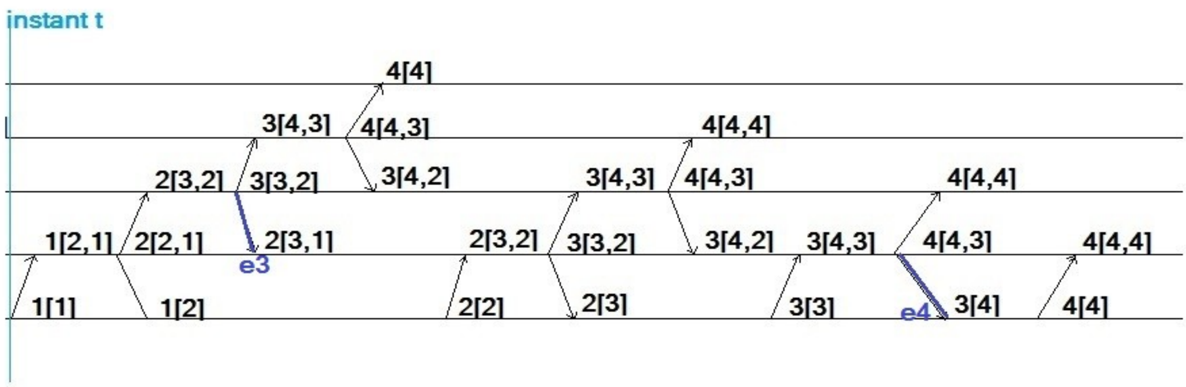


Figure 1

Voici un exemple d'exécution de l'algorithme de phase avec le graphe G:





Dans notre exemple d'exécution, l'ordre des événements est le suivent:

- l'émission du premier message envoyé par le processus 1 au processus 2 (e1)
- l'émission du 2ème message envoyé par le processus 2 au processus 3 (e2)
- l'émission du 3ème message envoyé par leprocessus 3 au processus 4 (e3)
- l'émission du 4ème message envoyé par le processus 4 au processus 5 (e4)

Pour qu'un processus P puisse émettre un message, il faut que le nombre de messages envoyés par chaque processus pouvant lui envoyer un message, soit plus grand ou égale au nombre de message déjà émis pas P.
C'est pour cela, nous sommes sures que l'ordre de ces 4 événements sera toujours le même que celui invoqué ci-dessus.

<u>Question 2:</u>

a)

sur chaque lien de communication, on peut envoyer D messages (où D représente le diamètre du graphe). Un processus envoie au maximum un message durant une phase et comme le nombre de phase est égale au diamètre au moment de la décision du processus alors le nombre de messages est égale au diamètre D=4.
Le nombre maximal de messages qui peuvent circuler est égale à D*|V| = 4 x 8 = 32 qui représente le nombre maximal de message qui peut être envoyé sur un lien multiplié fois le nombre de liens.

b)

i. Vérifier que dans γ à partir de l'instant t0 , on a:

- **Phasep > 0 pour tout processus p:**
  A l'instant t0 , on sait que plus aucun processus ne peut envoyer de messages et que tous les messages ont été reçus. Or, dans l'algorithme, pour qu'un processus puisse envoyer un message, il faut que Recp[q] >= Phasep et Phasep < D. Comme Recp[q] = D (i.e. tous les messages ont été reçus), et comme Phasep a été strictement inférieur à D avant l'instant t0, on a que Phasep a été incrémenté de 1 plusieurs fois avant t0, d'où Phasep > 0 à l'instant t0.
- **Recp [q] = Phaseq pour tout processus p, q tel que q ∈ Inp:**
  Au départ, nous avons phasep=Recp[q]=0. Et à t0 q a envoyé D messages, alors son voisin p a reçu D messages de la part de q car à chaque envoie d'un message de q à p phaseq s'incrémente de 1, et à la réception du message par p Recp[q] s'incrémente aussi de 1.
  Donc à t0, où tous les messages ont été reçus et plus aucun message ne sera envoyé, on aura phaseq=Recp[q]=D.
- **Phasep = D ou ∃q ∈ Inp : Recp [q] < Phasep:**
  Démonstration par l'absurde:
  On suppose qu'à t0, tous les messages ont été reçus et plus aucun message ne sera envoyé et que Phasep != D et ∃q ∈ Inp : Recp[q] ≥ Phasep, mais comme Phasep ne peut être supérieur à D, alors, Phasep < D et ∃q ∈ Inp : Recp[q] ≥ Phasep
  Avec cette dernière condition, l'algorithme permet au processus d'envoyer encore des messages ce qui est contradictoire avec ce que nous avons supposé au départ.
  Donc Phasep = D ou ∃q ∈ Inp : Recp [q] < Phasep

ii. En déduire que tous les processus décident. Pour cela on pourra considérer la valeur minimale
des Phasep après t0 , et montrer en utilisant les propriétés précédentes que cette valeur est au moins égale à D.

D'après ce que nous avons vérifié dans (i), à t0 et pour tout processus p et q:

-Phasep=D

-Recp[q] = Phaseq

Donc Phaseq=D aussi, et par conséquence Recp[q]=D ce qui satisfait la condition de la dernière ligne de l'algorithme. Donc Tous les processus décident.

<u>Question 3:</u>

a)

Démonstration par induction:

Cas l=1:

P0,P1 est un chemin de P0 à P1 et $l<D$, le chemin commence par P0, alors $E(P0)<=F(1,P0,P1)$ et l'énoncé nous dit que $F(1,P0,P1)<=G(1,P0,P1)$, alors $E(P0)<=G(1,P0,P1)$ OK

Cas l=D-2:

P0,P1,...,Pl est un chemin avec $l<D$, l'hypothèse d'induction est $E(p0)<=G(l,Pl-1,Pl)$. Soit Pl+1 un autre processus tel que pl ∈ Inpl+1 alors $F(l+1,Pl,Pl+1)<=G(l+1,Pl,Pl+1)$ et $G(l,Pl-1,Pl)<=F(l+1,Pl,Pl+1)$ Alors $E(p0)<=G(l+1,Pl,Pl+1)$.

On a donc prouvé par induction que si p0,...,pl une chemin de longueur l et $l<D$ alors, $E(p0) <= G(l,pl-1,pl)$.

Donc Aucun processus ne peut décider tant que tout les autres processus n'ont pas encore démarré. Autrement dit, Pour tout processus p ∈ Π, alors pour tout s ∈ Π, es<=dp .

b)

On peut donc déduire que l'algorithme est bien un algorithme de vague car:

-L'algorithme s'arrête à Phase=D

-Tous les processus décident

-Le début de chaque processus est causalement avant la fin de chaque processus

-Tous les processus sont initiateurs

c)

Un message doit être envoyer pour qu'on puisse le recevoir. Donc même si la communication n'est plus FIFO, cette règle reste valable. La seule différence sera donc que les messages ne seront pas reçus dans le même ordre de l'émission.

Nous n'avons pas utilisé la propriété FIFO dans la question précédente pour prouver que l'algorithme est de vague, donc avec le même raisonnement on peut prouver que même sans FIFO l'algorithme est de vague.

<u>Question 4:</u>

Initialisation:

1. ∀ q ∈ Inp: Recp[q] := 0
2. Phasep := 0

CODE POUR LE PROCESSUS P:

3. {Vq de q} → recevoir <Vq> ;
4. Recp[q] := Recp[q] + 1
5. {Vq < Vp} → Vp :=Vq
6. {∀ q ∈ Inp: Recp[q] ≥ Phasep ∧ Phasep < D} → send<Vp> à tous les q ∈ Outp
7. Phasep:= Phasep + 1
8. { ∀ q ∈ Inp: Recp[q] ≥ D ∧ Phasep ≥ D} → decider

Question 5:

Initialisation:

1. $\forall q \in$ Inp: Recp[q]:=0
2. Phasep:=0

CODE POUR LE PROCESSUS P:

3. Idp := <p>
4. {<Idq> de q}$\rightarrow$ recevoir <Idq>;
5.   Recp[q]:=Recp[q]+1
6. Idp:= Idp U Idq
7. {$\forall q \in$ Inp: Recp[q] $\geq$ Phasep $\wedge$ Phasep $<$ D}$\rightarrow$ send <Idp> à tout les q $\in$ Outp
8.   Phasep := Phasep + 1
9. {$\forall q \in$ Inp: Recp[q] $\geq$ D $\wedge$ Phasep $\geq$ D} $\rightarrow$ decider