Université de Strasbourg
UFR Mathématique et Informatique

M1 Informatique
2021/2022

Algorithmes Distribués

TD n° 2 : Exclusion Mutuelle

CORRECTION

I Algorithme de la boulangerie (Lamport 74)

On considère quatre processus P_0, P_1, P_2, P_3 qui exécutent cet algorithme pour accéder en section critique.

Q 1. Montrez pour le scénario ci-dessous :

1. Une évolution possible des variables $num[i]$ et $choix[i]$ pour tout i,
2. L'ordre dans lequel les différents processus accèdent à la section critique.
3. Y a-t-il différents ordres possibles d'accès des processus à la section critique ?
4. Quel est le rôle du tableau choix ?

Scénario :
t1 : P1 est entrant ;
t2 : P2 est entrant ;
t3 : P2 est dedans ;
t4 : P3 est entrant ;
t5 : P3 est dedans ;
t6 : P0 est entrant ;
t7 : P1 est dedans ;
t8 : P0 est dedans ;

	t1	t2	t3	t4	t5	t6	t7	t8
$num[0] = -1$ $choix[0] = F$						- V		3 F
$num[1] = -1$ $choix[1] = F$	- V						1 F	
$num[2] = -1$ $choix[2] = F$		- V	1 F					
$num[3] = -1$ $choix[3] = F$				- V	2 F			

1. cf. tableau ci-dessus ;
2. ici $P_1 - P_2 - P_3 - P_0$;
3. mais toute séquence telle que $P_2 - P_3 - P_0$ (avec P_1 n'importe où donc) est valide ;
4. le tableau choix sert à protéger en particulier le cas où deux processus P_i et P_j tirent le même numéro x (avec $i < j$), car l'identifiant des deux sites ne suffira pas à les départager correctement dans ce cas. Le site avec le plus grand identifiant P_j peut rentrer (être dedans) en premier et donc utiliser à tort la section critique avant P_i si celui-ci est plus lent à choisir son numéro x. Avec la variable choix ($choix[i]$ en l'occurence), P_j attendra P_i même s'ils tirent le même x.

Q 2. Preuve de l'algorithme :

1. Montrez que ce protocole est exempt d'interblocage, et qu'il assure l'équité des processus.
2. Montrez que exclusion mutuelle est assurée.

1. Progrès et équité : pas d'interblocage car ordre total (au pire avec l'identifiant du proc.) et donc il existe un unique proc qui aura l'accès en SC ;
2. Pas deux procs. en section critique pour garantir l'EM. Si un proc. P_i est en SC, c'est qu'il a le plus petit numéro (dans la file) parmi ceux qui sont dedans lorsqu'il y est. S'il existe un proc. P_j non dans cette liste avec un plus petit numéro que P_i et qui peut rentrer en SC alors que P_i y est alors seulement deux cas sont possibles : P_j est devenu entrant alors que P_i était dedans, ou alors que P_i était entrant. Dans le premier cas, l'ordre suffit à garantir l'EM et dans le second $choix[j] = faux$ et l'ordre est à nouveau contradictoire $(num[j], j) > (num[i], i)$ (et cette fois avec l'identifiant du site pour départager).

II Un jeton et un algorithme (Ricart/Agrawala)

Supposons plusieurs processus répartis sur des sites distants. Ces processus partagent une même ressource et cette dernière ne peut être utilisée que par un processus à la fois. Pour garantir le bon fonctionnement d'un tel système, il faut mettre en place un algorithme d'exclusion mutuelle.

Une section critique est une zone de code concernant la ressource partagée : un seul processus au plus doit être en section critique afin de garantir une utilisation correcte de la ressource. On distingue généralement trois approches pour gérer l'exclusion mutuelle dans un système réparti :

— approche centralisée (utilisation d'un coordinateur) ;
— approche par jeton ;
— approche par liste d'attente répartie.

Les protocoles d'exclusion mutuelle partagent les propriétés suivantes :

— **Définition** : à tout instant un processus au plus en section critique ;
— **Atteignabilité** : si plusieurs processus sont bloqués en attente de la section critique alors qu'aucun processus n'est en section critique, alors l'un d'eux doit y accéder en un temps fini ;
— **Progression** : un processus en attente accède à la section critique en un temps fini ;
— **Indépendance des parties conflictuelles et non-conflictuelles** : un processus hors de la section critique ou du protocole d'entrée ne doit pas influer sur le protocole d'exclusion mutuelle ;
— **Banalisation de la solution** : aucun processus ne joue de rôle privilégié.

Pour qu'une solution soit valide, il faut démontrer que ces propriétés sont bien respectées.
Soit 4 processus P_0, P_1, P_2, et P_3 qui se trouvent dans la situation suivante :

— P_3 est dans la section critique S
— P_1 demande d'entrer dans la section critique S à T_0 avec une horloge locale (estampille) de 1.
— P_0 demande d'entrer dans la section critique S à T_1 avec une horloge locale (estampille) de 5.
— P_2 demande d'entrer dans la section critique S à T_2 avec une horloge locale (estampille) de 3.
— P_3 sort de la section critique à T_3

On suppose que $T_0 < T_1 < T_2 < T_3$.

Algorithme : pour un processus P_i

```
Demande_Entree_SC()
{
    estampille ++;
    requete.emetteur = i;
    requete.date = estampille;
    demande[i] = estampille;
    diffuser(requete);

    si ( ! jeton_present) :
        attendre((jeton, new_tampon));

    tampon = new_tampon;
    dedansSC = VRAI;
    jeton_present = VRAI;
}
```

```
Reception_Requete()
{
    k = requete.emetteur;
    demande[k] = max(demande[k], requete.date);

    si (jeton_present) && ( ! dedansSC) :
        Sortie_SC()
    sinon rien (pas de jeton)
}

Sortie_SC()
{
    tampon[i] = estampille;
    dedansSC = FAUX;

    pour j=(i+1)% N; j==i; j=(j+1)% N :
        si (demande[j] > tampon [j]) && jeton_present) :
            jeton_present = FAUX;
            envoyer((jeton, tampon), j);
}
```

Q 1. Dessinez le chronogramme des différents événements.

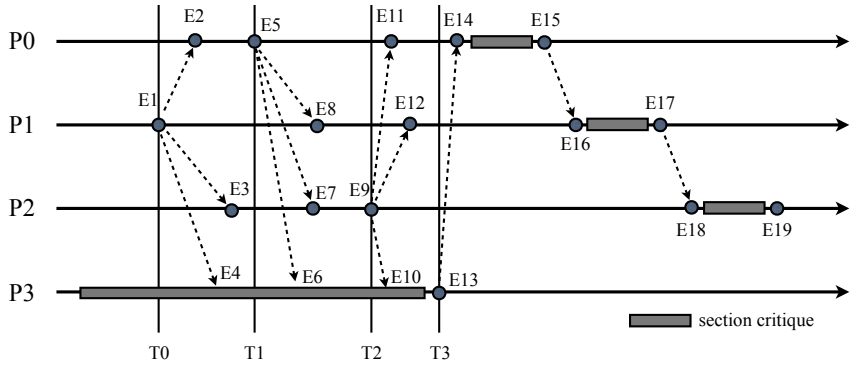

FIGURE 1 – Déroulement des différents événements

Q 2. Preuve de l'algorithme :

1. Montrez que l'exclusion mutuelle est assurée ;
2. Montrez l'absence d'inter-blocage ;
3. Montrez que l'algorithme est équitable (les messages sont délivrés en un temps fini).

1. un seul proc. a la valeur $jeton\_present$ à vrai. Initialement vraie, cette propriété l'est toujours car seul le postlude permet d'envoyer le jeton à un autre proc. et pour y arriver il faut d'abord passer par le prélude après réception du jeton ;
2. si des procs. sont bloqués en attente alors, dès lors que le jeton sera en transit (soit après relâchement par un proc. sortant de SC, soit parce qu'il était déjà en circulation) le site le plus proche de lui - dans l'anneau - ne sera plus bloqué. L'anneau définit un ordre intrinsèque pour assurer la progression ;
3. cette progression est dite équitable (chacun son tour dans l'anneau) : comme aucun message n'est perdu, tout proc. finira par entrer en SC en un temps fini. En effet, un proc. P_i ne pourra ré-accéder en SC qu'une fois que tous les proc. qui étaient en attente lorsqu'il a relâché le jeton auront accédé eux-aussi à la SC. Il attendra son tour qui viendra car ces mêmes procs. qui étaient en attente seront informés de sa nouvelle requete (en un temps fini).

Q 3. Donnez le nombre de messages nécessaires pour une entrée en SC.

Soit N le nombre de sites. Le nombre de message est soit $N-1+1$ (jeton non présent) soit $N-1$ (dans cette version) ou 0 (dans le cours de Stella) selon la version quand je jeton est déjà présent.

Q 4. Déroulez l'algorithme d'exclusion mutuelle basée sur un jeton pour déterminer les événements de chaque processus (valeur du jeton, section critique, etc.).

Evénements	0	1	2,3,4	5	6,7,8	9
Dedans	Faux		Faux	Faux		
Estampille	4		4	5		
Jeton	Faux		Faux	Faux		
Demande	(4,0,2,X+1)		(4,1,2,X+1)	(5,1,2,X+1)		
Tampon	(4,0,2,X)		(4,0,2,X)	(4,0,2,X)		
Action	/		/	R(0,5)		
Dedans	Faux	Faux			Faux	
Estampille	0	1			1	
Jeton	Faux	Faux			Faux	
Demande	(4,0,2,X+1)	(4,1,2,X+1)			(5,1,2,X+1)	
Tampon	(4,0,2,X)	(4,0,2,X)			(4,0,2,X)	
Action	/	R(1,1)			/	
Dedans	Faux		Faux		Faux	Faux
Estampille	2		2		2	3
Jeton	Faux		Faux		Faux	Faux
Demande	(4,0,2,X+1)		(4,1,2,X+1)		(5,1,2,X+1)	(5,1,3,X+1)
Tampon	(4,0,2,X)		(4,0,2,X)		(4,0,2,X)	(4,0,2,X)
Action	/		/		/	R(2,3)
Dedans	Vrai		Vrai		Vrai	
Estampille	X+1		X+1		X+1	
Jeton	Vrai		Vrai		Vrai	
Demande	(4,0,2,X+1)		(4,1,2,X+1)		(5,1,2,X+1)	
Tampon	(4,0,2,X)		(4,0,2,X)		(4,0,2,X)	
Action	SC		/		/	

Evénements	10,11,12	13, 14	15,16	17,18	19
Dedans	Faux	Vrai	Faux		
Estampille	5	5	5		
Jeton	Faux	Vrai	Faux		
Demande	(5,1,3,X+1)	(5,1,3,X+1)	(5,1,3,X+1)		
Tampon	(4,0,2,X)	(4,0,2,X+1)	(5,0,2,X+1)		
Action	/	SC	T -> P_1		
Dedans	Faux		Vrai	Faux	
Estampille	1		1	1	
Jeton	Faux		Vrai	Faux	
Demande	(5,1,3,X+1)		(5,1,3,X+1)	(5,1,3,X+1)	
Tampon	(4,0,2,X)		(5,0,2,X+1)	(5,1,2,X+1)	
Action	/		SC	T -> P_2	
Dedans				Vrai	Faux
Estampille				3	3
Jeton				Vrai	Vrai
Demande				(5,1,3,X+1)	(5,1,3,X+1)
Tampon				(5,1,2,X+1)	(5,1,3,X+1)
Action				SC	/
Dedans	Vrai	Faux			
Estampille	X+1	X+1			
Jeton	Vrai	Faux			
Demande	(5,1,3,X+1)	(5,1,3,X+1)			
Tampon	(4,0,2,X)	(4,0,2,X+1)			
Action	/	T -> P_0			

III Liste d'attente répartie (Lamport)

Avec une exclusion mutuelle basée sur une liste d'attente répartie, chaque processus gère une copie de la file d'attente. Cette dernière est mise à jour par des messages de requêtes et de libération de la section critique.

Chaque processus doit :
- recevoir tous les messages de requêtes et de libération de tous les autres processus ;
- savoir ordonner ces messages en fonction de leur date ⇒ utilisation d'horloges logiques.

Algorithme : pour un processus P_i

$Init\_Site()$
{
  $\forall$ j $F_i[j]$ = (Hors, 0, J) ;
  $H_i = 0$;
}

$Entree\_SC()$
{
  Diffuser(requete, H_i, i) ;
  $F_i[i]$ = (requete, H_i, i) ;
  H_i ++ ;
  Attendre que $\forall$ j $\neq$ i :
    estampille de $F_i[i]$ < estampille $F_i[j]$;
}

$Sortie\_SC()$
{
  Diffuser(Hors_SC, H_i, i) ;
  $F_i[i]$ = (Hors_SC, H_i, i) ;
  H_i ++ ;
}

$Reception\_message((type, H, j))$
{
  H_i = max(H,H_i) + 1 ;

  switch(type) :
  {
    case requete :
      $F_i[j]$ = (requette,H,j) ;
      if type($F_i[i] \neq$ requete) :
        envoi(Ack, H_i, i) $\rightarrow P_j$;

    case Hors_SC :
      $F_i[j]$ = (Hors_SC, H, j) ;

    case Ack :
      $F_i[j]$ = (Ack, H, j) ;
  }

Q 1. Reprenez le scénario précédent en l'ajustant pour dérouler l'algorithme d'exclusion mutuelle basée sur une liste d'attente répartie afin de déterminer les événements de chaque processus (entrée en section critique, etc.). Petite différence, P_2 demandera à entrer en section critique à T_2 avec une horloge locale de 6.

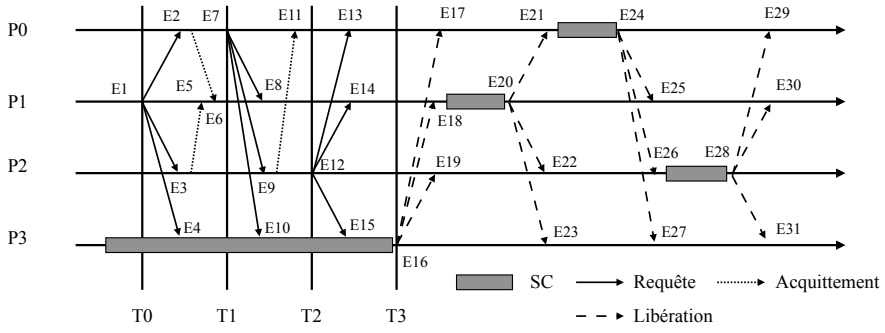

FIGURE 2 – Déroulement des différents événements

Q 2. Preuve de l'algorithme :

1. Montrez que les demandes d'entrée en section critique sont traitées selon l'ordre total imposé par le mécanisme d'estampillage ;
2. Montrez que le protocole assure l'exclusion mutuelle.

1. En l'absence de déséquencement, l'émission des messages req, le recalage qu'il induit et les acks ainsi obtenus garantissent cette propriété. La progression fonctionne même sans certains acks car les messages

bloquants ne sont que ralentissants au pire et finissent par arriver et être départagés. En effet, en particulier dans cette version, si deux req sont concurrentes, elles seront départagées seulement après reception de leurs estampilles respectives (et donc chaque proc. aura alors accès à la req de l'autre pour comparaison – sinon bloquant sur une ancienne estampille plus faible en attendant ou ack libérateur si pas de requête concurrente) ;

2. Seul le message de relâchement peut supprimer le message req qui empêche l'accès concurrent en SC. Ce message arrive après les requêtes concurrentes et potentiellement antérieures (estampilles plus petites) car absence de déséquencement.

Q 3. Quel est le nombre de messages émis pour mettre en œuvre un accès à la section critique ?

Soit N le nombre de sites. Le nombre de message pour assurer UN accès en SC est $\approx 3 \times (N-1)$: diffusion requêtes, retour des acks et relâchement SC. Il s'agit d'un cas au pire en nb de messages pour UN accès (dans ce cas on peut considérer $3N-4$ pour être plus précis, c'est à dire pas d'acquittement pour le proc. déjà en SC qui devra la relâcher).

Evénements	0	1	2, 3, 4	5 puis 6	7	8, 9, 10	11	12
H_0	4		5		6		7	
$F_0[0]$	hs,0,0		hs,0,0		rq,5,0		rq,5,0	
$F_0[1]$	hs,0,1		rq,1,1		rq,1,1		rq,1,1	
$F_0[2]$	hs,0,2		hs,0,2		hs,0,2		ack,6,2	
$F_0[3]$	rq,0,3		rq,0,3		rq,0,3		rq,0,3	
Action	/		E(ack,5,0)		E(rq,5,0), Att		Att	
H_1	1	2		6		7		
$F_1[0]$	hs,0,0	hs,0,0		ack,5,0		rq,5,0		
$F_1[1]$	hs,0,1	rq,1,1		rq,1,1		rq,1,1		
$F_1[2]$	hs,0,2	hs,0,2		ack,2,2		ack,2,2		
$F_1[3]$	rq,0,3	rq,0,3		rq,0,3		rq,0,3		
Action	/	E(rq,1,1), Att.		Att.		/		
H_2	1		2			6		7
$F_2[0]$	hs,0,0		hs,0,0			rq,5,0		rq,5,0
$F_2[1]$	hs,0,1		rq,1,1			rq,1,1		rq,1,1
$F_2[2]$	hs,0,2		hs,0,2			hs,0,2		rq,6,2
$F_2[3]$	rq,0,3		rq,0,3			rq,0,3		rq,0,3
Action	/		E(ack,2,2)			E(ack,6,2)		E(rq,6,2)
H_3	1		2			6		
$F_3[0]$	ack		hs,0,0			rq,5,0		
$F_3[1]$	ack		rq,1,1			rq,1,1		
$F_3[2]$	ack		hs,0,2			hs,0,2		
$F_3[3]$	rq,0,3		rq,0,3			rq,0,3		
Action	SC		/			/		

Evénements	13, 14, 15	16	17, 18, 19	20	21, 22, 23	24	25, 26, 27	28
H_0	8		9		10	11		
$F_0[0]$	rq,5,0		rq,5,0		rq,5,0	hs,10,0		
$F_0[1]$	rq,1,1		rq,1,1		hs,9,1	hs,9,1		
$F_0[2]$	rq,6,2		rq,6,2		rq,6,2	rq,6,2		
$F_0[3]$	rq,0,3		hs,7,3		hs,7,3	hs,7,3		
Action	Att		Att		SC	E(hs,10,0)		
H_1	8		9	10			11	
$F_1[0]$	rq,5,0		rq,5,0	rq,5,0			hs,10,0	
$F_1[1]$	rq,1,1		rq,1,1	hs,9,1			hs,9,1	
$F_1[2]$	rq,6,2		rq,6,2	rq,6,2			rq,6,2	
$F_1[3]$	rq,0,3		hs,7,3	hs,7,3			hs,7,3	
Action	Att		SC	E(hs,9,1)			/	
H_2			8		10		11	12
$F_2[0]$			rq,5,0		rq,5,0		hs,10,0	hs,10,0
$F_2[1]$			rq,1,1		hs,9,1		hs,9,1	hs,9,1
$F_2[2]$			rq,6,2		rq,6,2		rq,6,2	hs,11,2
$F_2[3]$			hs,7,3		hs,7,3		hs,7,3	hs,7,3
Action			Att		Att		SC	E(hs,11,2)
H_3	7	8			10		11	
$F_3[0]$	rq,5,0	rq,5,0			rq,5,0		hs,10,0	
$F_3[1]$	rq,1,1	rq,1,1			hs,9,1		hs,9,1	
$F_3[2]$	rq,6,2	rq,6,2			rq,6,2		rq,6,2	
$F_3[3]$	rq,0,3	hs,7,3			hs,7,3		hs,7,3	
Action	/	E(hs,7,3)			/		/	

Evénements	29, 30, 31
H_0	12
$F_0[0]$	hs,10,0
$F_0[1]$	hs,9,1
$F_0[2]$	hs,11,2
$F_0[3]$	hs,7,3
Action	/
H_1	12
$F_1[0]$	hs,10,0
$F_1[1]$	hs,9,1
$F_1[2]$	hs,11,2
$F_1[3]$	hs,7,3
Action	/
H_2	
$F_2[0]$	
$F_2[1]$	
$F_2[2]$	
$F_2[3]$	
Action	
H_3	12
$F_3[0]$	hs,10,0
$F_3[1]$	hs,9,1
$F_3[2]$	hs,11,2
$F_3[3]$	hs,7,3
Action	/