

Tema 2

Sincronización en memoria compartida

SCD para GIIM

Asignatura *Sistemas Concurrentes y Distribuidos*

Fecha 4 noviembre 2022



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada

Introducción al problema del "exclusión mutua"



Introducción histórica al problema de la exclusión mutua

1962 Dekker propone el problema de la **exclusion mutua** para multiprocesadores:

"diseñar un protocolo que garantice el acceso mutuamente excluyente, sin que exista interbloqueo, a una sección crítica por parte de un determinado número de procesos que compiten por entrar a dicha sección..." → Para 2 procesos bien, pero para más da problemas

1965 Dijkstra propone una solución **segura, libre de interbloqueo**, pero que puede producir **inanición**

→ Este sí va bien para n procesos

1966 Knuth propone una solución sin **inanición**; garantiza retraso limitado de los procesos, pero no FIFO

→ Es restrictivo y usa herramientas de sincronización no muy legales

1974 Lamport: permite a los procesos "detenerse" en la ejecución del protocolo de adquisición, solapar las operaciones de lectura con la escritura y **retraso FIFO** de los procesos que ya esperan entrar

1981 Peterson propone una solución **equitativa** para 2 y " n " procesos; garantiza el retraso cuadrático de los procesos, es la solución más simple hasta fecha para multiprocesadores

1983 Algoritmos totalmente distribuidos que resuelven el problema para multicomputadores;
Ricart-Aggrawala, Suzuki-Kasami

→ Es el primero que se propone pero para implementarlo hacen falta tantos mensajes que puede haber colapso

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua

Solución al problema con bucles de espera activa

- Los procesos iteran en un bucle vacío hasta que la entrada en Sección Crítica (SC) sea segura
- Aceptable si el sistema/aplicación no tuviera muchos procesos

→ Para que el algoritmo sea válido

Condiciones de Dijkstra

- 1 No hacer ninguna suposición acerca de las instrucciones o número de procesos soportados por el multiprocesador
→ No valen semáforos
- 2 Ni tampoco acerca de la velocidad de ejecución de los procesos, excepto que no es cero (*Progreso Finito*)
- 3 Cuando un proceso se encuentra ejecutando código fuera de la sección crítica no puede impedir a los otros procesos entrar en ésta
- 4 La sección crítica siempre será alcanzada por alguno de los procesos que esperan entrar → Condición de alcanzabilidad



Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua

Método de refinamiento sucesivo



Esquema de "corrutinas": no se cumple la tercera propiedad de Dijkstra!

1A Etapa

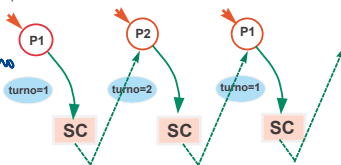
Proceso P1	Proceso P2
<pre>while true do begin <<resto instrucciones>> while turno <> 1 do nothing; enddo; <<seccion critica>> turno:= 2; end enddo;</pre>	<pre>while true do begin <<resto instrucciones>> while turno <> 2 do nothing; enddo; <<seccion critica>> turno:= 1; end enddo;</pre>

Es para 2 procesos

Protocolo de salida (cambio turno al ident. del otro)

Variable global compartida: turno

Protocolo de entrada (Cada proceso comprueba si $\text{turno} == \text{id_proc}$)



Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua

Método de refinamiento sucesivo –II



2A Etapa

Proceso P1

```
while true do
begin
  <<resto instrucciones>>
enddo;
```

La salida de la espera activa y el cambio de la clave no se realizan atómicamente

```
while c2=0 do
  nothing;
enddo;
```

bucle espera activa

```
c1:= 0;
<<seccion critica>>
c1:= 1;
```

```
end
enddo;
```

Proceso P2

```
while true do
begin
  <<resto instrucciones>>
enddo;
```

```
while c1= 0 do
  nothing;
enddo;
```

```
c2:= 0;
<<seccion critica>>
c2:= 1;
end
enddo;
```

Problema!: no se cumple la propiedad de seguridad del algoritmo.

→ Podría darse que ambos procesos entren a la SC

c_1 y c_2 son variables clave

Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusion mutua

Método de refinamiento sucesivo –III



No se cumple la condición de alcanzabilidad
(podrían quedarse ambos procesos interbloqueados)

3A Etapa

Adelantando la
asignación de la
clave la solución
es segura

Proceso P1	Proceso P2
<pre>while true do begin <<resto instrucciones>> c1:= 0;</pre>	<pre>while true do begin <<resto instrucciones>> c2:= 0;</pre>
<pre> while c2=0 do nothing; enddo;</pre>	<pre> while c1= 0 do nothing; enddo;</pre>
<pre> <<seccion critica>> c1:= 1;</pre>	<pre> <<seccion critica>> c2:= 1;</pre>
<pre>end enddo;</pre>	<pre>end enddo;</pre>

Nuevo problema!: el proceso que modifica la clave no sabe si el otro hace lo mismo concurrentemente con el

Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

Método de refinamiento sucesivo –IV



Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

4A Etapa:

Proceso P1

```
while true do
  begin
    <<resto instrucciones>>
    c1:= 0;
  while c2=0 do
    begin
      c1:= 1;
      while c2= 1 do
        nothing;
      enddo;
      c1:= 0;
    end
  enddo;
  <<seccion critica>>
  c1:= 1;
end
enddo;
```

Para indicar que
intenta entrar en S.C.,
cambia su clave

Comprueba la clave
del otro; la vuelve a
cambiar si el otro
tambien intenta entrar,

Proceso P2

```
while true do
  begin
    <<resto instrucciones>>
    c2:= 0;
  while c1=0 do
    begin
      c2:= 1;
      while c1= 1 do
        nothing;
      enddo;
      c2:= 0;
    end
  enddo;
  <<seccion critica>>
  c2:= 1;
end
enddo;
```

*Sigue sin cumplirse la condición de alcanzabilidad.
Concluimos que la lectura de variables
clave no nos va a dar la solución*

Método de refinamiento sucesivo –V



5A Etapa: Algoritmo de Dekker

Es fusión de 1 y 4

	Proceso P1	Proceso P2
	<pre>while true do begin <<resto instrucciones>> c1:= 0;</pre>	<pre>while true do begin <<resto instrucciones>> c2:= 0;</pre>
El proceso intenta entrar en S.C.		
comprueba la clave del otro	<pre>while c2= 0 do if turno= 2 then begin c1:= 1; while turno= 2 do nothing; enddo; c1:= 0; end endif enddo;</pre>	<pre>while c1= 0 do if turno= 1 then begin c2:= 1; while turno= 1 do nothing; enddo; c2:= 0; end endif enddo;</pre>
si no tiene el turno hace espera activa, despues de cambiar su clave	<pre><<seccion critica>> turno:= 2; c1:= 1; end enddo;</pre>	<pre><<seccion critica>> turno:= 1; c2:= 1; end enddo;</pre>

Hay un caso en el que el algoritmo no cumple la vivacidad

Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusion mutua



Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua

Verificación de propiedades de seguridad

Exclusión mutua:

- 1 P_i entra en sección crítica sólo si $c[j] == 1$
- 2 P_i comprueba la clave del otro, $c[j]$, sólo después de asignar su propia clave
Luego, cuando P_i entra se cumple $c[j] == 1 \wedge c[i] == 0$

Método de refinamiento sucesivo: verificación de propiedades –II



Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua

Verificación de propiedades de seguridad

Alcanzabilidad de la sección crítica:

Si P_i y P_j intentan entrar en sección crítica y $\text{turno} == i$:

Discusión sobre la *equidad* de la solución dada por el Algoritmo de Dekker

Generalización a N procesos: Algoritmo de Dijkstra



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

```
c: array[0..n-1] of (pasivo,solicitando,en_SC)
turno: 0.. n-1;
```

```
repeat
```

```
  repeat
```

```
    E2:c[i]:= solicitando;
```

1A barrera
detiene a los
procesos si
el que posee
el turno no
esta pasivo

```
    while turno <> i do
      E3:if c[turno]= pasivo
        then turno:= i
        endif;
      enddo;
```

La comprobacion
del estado del que
tiene el turno y
el cambio de este
no se hace
atomicamente

Si un grupo de
procesos se ve
obligado a ciclar
nuevamente,
el que posee el
turno no puede
estar pasivo

```
    E4:c[i]:= en_SC;
    j:= 0;
```

2A barrera
asegura que
se cumple la
propiedad
de seguridad

```
    while(j<n) and (j=i or c[j] <> en_SC) do
      j:= j+1;
    enddo;
  until j>= n;
```

```
  <<Seccion Critica>>
```

```
  E1:c[i]:= pasivo;
```

```
  <<Resto de instrucciones>>
```

```
until false
```

Verificación de las propiedades del A. Dijkstra

Verificación de las propiedades de seguridad

Exclusión mutua:

demostración similar a la del A. Dekker



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

Alcanzabilidad de la sección crítica:

① `turno` es una variable compartida, mantendrá el valor i del último P_i que lo asigne

② Sean $\{P_1 \dots P_i \dots P_m\}$ tales que $c[i] = \text{en_SC}$ y $\text{turno} == k$ con $1 \leq k \leq m$, entonces

P_k entrará en su sección en tiempo finito y el resto

$P_i : 1 \leq i \leq m \wedge i \neq k$ se quedará ciclando en el primer bucle (1A) del protocolo



Verificación de las propiedades de vivacidad del A. Dijkstra

El A. Dijkstra satisface seguridad pero no evita el peligro de inanición de los procesos del programa

posición/acción	c[1]	c[2]	c[3]	turno
Inic.: P_1, P_2, P_3 en E1	pasivo	pasivo	pasivo	3
$P_1 : E_1 \rightarrow E_2$	solicitando	pasivo	pasivo	3
$P_2 : E_1 \rightarrow E_2$	solicitando	solicitando	pasivo	3
$P_1 : E_2 \rightarrow E_3$	solicitando	solicitando	pasivo	3
$P_2 : E_2 \rightarrow E_3$	solicitando	solicitando	pasivo	3
$P_1 : E_3 \rightarrow E_4$	en_SC	solicitando	pasivo	3
$P_2 : E_3 \rightarrow E_4$	en_SC	en_SC	pasivo	3
...

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC
Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

```

c: array[0..n-1] of (pasivo,solicitando,en_SC)
turno: 0..n-1;

repeat
  repeat
    E2:c[i]:= solicitando;

1A barrera      while turno <> i do
detiene a los   E3:if c[turno]= pasivo
procesos si    then turno:= i
el que posee   endif;
el turno no   enddo;
esta pasivo    E4:c[i]:= en_SC;
               j:= 0;

2A barrera      while (j<n) and (j=i or c[j] <> en_SC) do
asegura que    j:= j+1;
se cumple la   enddo;
propiedad     until j>= n;
de seguridad

<<Seccion Critica>>
E1:c[i]:= pasivo;
<<Resto de instrucciones>>
until false
  
```

La comprobacion del estado del que tiene el turno y el cambio de este no se hace atómicamente

Si un grupo de procesos se ve obligado a ciclar nuevamente, el que posee el turno no puede estar pasivo

Algoritmo de Knuth para N procesos



1A barrera
detiene a los
procesos si
el que posee
el turno no está
pasivo

2A barrera
asegura que
se cumple la
propiedad
de seguridad

```
repeat
  repeat
    E0: c[i]:= solicitando;
    j:= turno; --variable local
    E1: while j <> i do
      if c[j] <> pasivo then j:= turno
      else j:= (j-1) MOD n
      endif;
    enddo;
    E2: c[i]:= en_SC;
    k:= 0
    while (k<n) and (k=i pr c[k]<>en_SC) do
      k:= k+1;
    enddo;
  until k>= n;
  E3: turno:= i;
  << Seccion Critica >>
  turno:= (i-1) MOD n;
  E4: c[i]:= pasivo;
  E5: <<resto de instrucciones>>
until false;
```

Si un grupo de
procesos se ve
obligado a ciclar
nuevamente,
el que posee el
turno no puede
estar pasivo

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

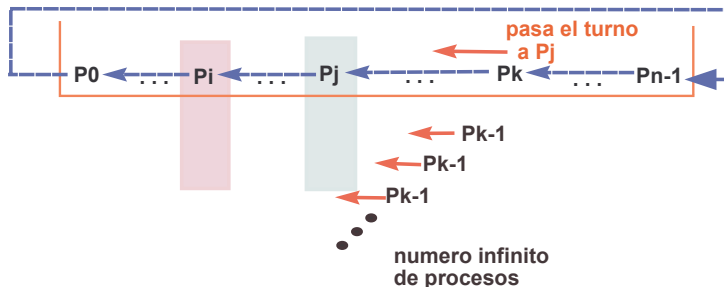
Algoritmo de Knuth para N procesos –II



Imposibilidad de la inanición de los procesos si se supone que existe un número finito de ellos en el algoritmo

Escenario de inanición: P_j se adelanta continuamente a P_i

Turno circular



Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

Algoritmo de Knuth para N procesos –III



Escenario que muestra la espera maxima para 4 procesos con el A. Knuth

E0: $\langle P_4 \rangle$

$\langle c[i] == \text{solicitando} \rangle$

E1: 1A Barrera

$\langle \text{turno} == i \rangle$

E2:

2A Barrera

$\langle c[i] == \text{pasivo} \rangle$

$\langle \langle \text{Seccion Critica} \rangle \rangle$

E5:

E2: P1, P2, P3
(solicitando)

P1, P3
(solicitando,
turno==1)

P1
(solicitando,
turno==2)

P2, P1
(solicitando,
turno==2)

P1
(solicitando,
turno==1)

1 2

3

4 5

6

7 8 9

10 11

E5:

P2 (pasivo,
turno==1)
P1 (pasivo,
turno==4)

P3 (pasivo,
turno==2)
P1 (pasivo,
turno==4)
P2 (pasivo,
turno==4)

P2 (pasivo,
turno==2)
P3 (pasivo,
turno==2)

P2 (pasivo,
turno==1)
P1 (pasivo,
turno==4)
P3 (pasivo,

P1 (pasivo,
turno==4)
P2 (pasivo,
turno==4)
P3 (pasivo,
turno==4)

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

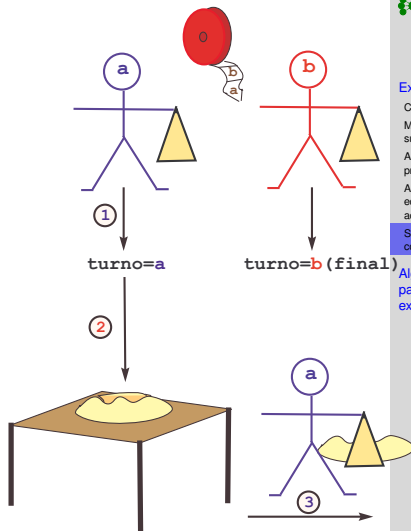
Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusion mutua

Solución para 2 procesos

```
var
  solicitado: array[0..1] of boolean;
  turno: 0..1;
```

```
Pi ::=
  ...
  solicitado[i] := true; --j=2, i=1
  turno := i;
  while (solicitado[j] and turno=i) do
    nothing;
  enddo;
  <<seccion critica>>
  solicitado[i] := false;
  ...
```



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

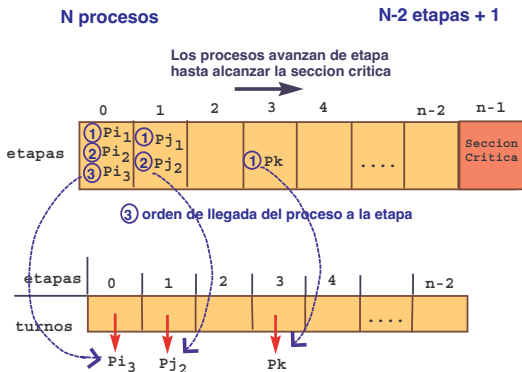
Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

Solución para N procesos - Idea general:



Variables compartidas entre los procesos:

```
type etapas= -1..n-2; var c: array[0..n-1] of etapas;
procesos= 0..n-1;      turno: array[0..n-2] of procesos;
```



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua

Algoritmo de Peterson-III

Solución para N procesos



Exclusión mutua

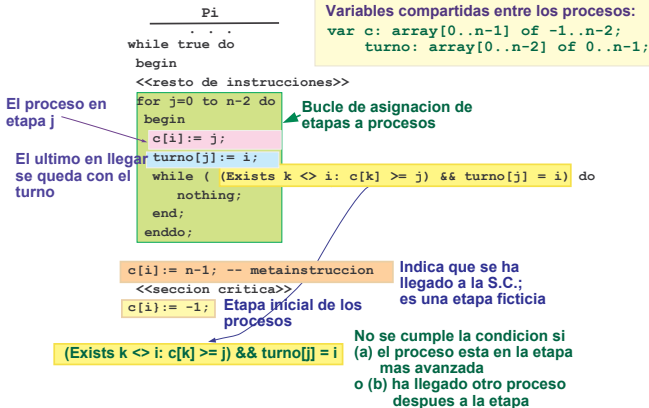
Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusión mutua





Se dice que P_i precede a un proceso P_j sii $c[i] > c[j]$

L1 Un proceso que precede a todos los demás puede avanzar al menos una etapa

(Exists $k < i$: $c[k] \geq j$) && turno[j] = i

Ya que no se cumple la condicion del segundo bucle si:

(a) el proceso esta en la etapa mas avanzada

o (b) ha llegado otro proceso despues a la etapa

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua



L2 Un proceso que pasa de la etapa j a la $j+1$ ha de verificar alguna de estas condiciones:

- 1 *Precede a todos los demás*
- 2 *No estaba solo en la etapa j*

Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusión mutua

L3 Si existe al menos dos procesos en la etapa j , entonces existe al menos un proceso en cada una de las etapas anteriores

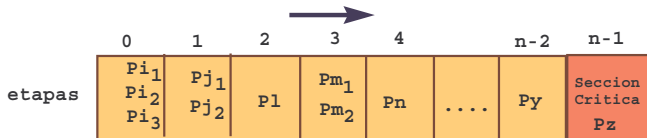
- *La demostración se hace por inducción sobre la variable que representa la etapa j*

L4 El número máximo de procesos que puede haber en la etapa j es $n-j$, con $0 \leq j \leq n-2$

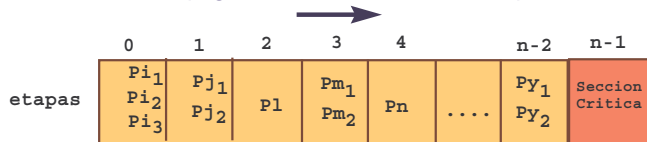
- *La demostración se hace aplicando el **Lema 3***
- *Por tanto, a la etapa $n-2$ llegarán como máximo 2 procesos*

Verificación de las propiedades de seguridad del A. Peterson-IV

El algoritmo de Peterson cumple con la exclusion mutua en el acceso a la seccion critica



P_y no puede avanzar a la siguiente etapa mientras la seccion critica este ocupada (segun las condiciones del Lema 2)



Segun el Lema 2, solo 1 de los 2 procesos en la etapa (n-2) podra avanzar a la seccion critica



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusion mutua

Algoritmo de Peterson-IV

```
var c: array[0..N-1] of -1..N-2;  
    turno: array[0..N-2] of 0..N-1;  
while (true) do  
    begin  
        Resto de las instrucciones;  
        (1) for j=0 TO N-2 do  
            (2)     begin  
                (3)         c[i]:= j;  
                (4)         turno[j]:= i;  
                (5)         for k=0 TO N-1 do  
                    (6)             begin  
                        (7)                 if (k=i) then continue;  
                        (8)                 while(c[k]>=j and turno[j]=i) do  
                            (9)                     nothing;  
                        (10)                enddo;  
                    (11)             end;  
                (12)     end;  
                (13) c[i]:= n-1; /*meta-instruccion*/  
                (14) <seccion critica>  
                (15) c[i]:= -1  
            end  
end
```



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua



I. Alcanzabilidad de la SC:

demostración por *reducción al absurdo*

- Todos los procesos se quedan bloqueados al llegar a una etapa y no avanzan más (hipótesis de incorrección)

II. Propiedad de equidad

demostración:

- El número máximo de turnos que un proceso cualquiera tendría que esperar con el algoritmo de Peterson es de
$$r(n) = n - 1 + r(n - 1) = \frac{n \times (n - 1)}{2} \text{ turnos}$$

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusión mutua

Problemática para la implementación de los algoritmos en sistemas distribuidos



Miscelánea

- Los algoritmos no pueden utilizar sincronización global entre los procesos sólo utilizando variables en memoria compartida
- Se utilizan operaciones atómicas de paso de mensajes para sincronizarlos
- La red de comunicaciones ha de cumplir las siguientes condiciones:
 - 1 Red de comunicaciones completamente conectada
 - 2 Transmisión de mensajes sin errores
 - 3 Retraso variable en la entrega de mensajes con tiempo acotado
 - 4 Posibles *desencuenciamientos* en la entrega de mensajes en transmisión

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusión mutua

variables
locales

Proceso P(i)

```
ns:0..+INF; mns:0..INF:=0;  
solicitaSC, prioridad:boolean;  
num_repesperadas: 0..n-1;  
repretrasadas:array[1..N]of boolean;
```

Hebra Main(i) ::=

```
solicitaSC:=true;  
ns:= mns+1;  
num_repesperadas:=n-1;
```

```
for j:=1 to n do  
  if j<>i then  
    send(j,pet,ns,i);  
  endif;  
enddo;
```

```
wait(sinc);  
<<Seccion  
  Critica>>
```

```
solicitaSC:=false;
```

```
for j:=1 to no do  
  if repretrasadas[j]  
  then  
    begin  
      repretrasadas[j]:=false;  
      send(j,rep);  
    end;  
  endif;  
enddo
```

Hebra Rep(i) ::=

```
receive(rep);  
num_repesperadas-=1;  
if num_repesperadas=0  
then  
  signal(sinc);  
endif;
```

Hebra Pet(i) :

```
receive(pet,k,j)  
mns:=max(ns,k);  
prioridad:=solicitaSC and  
  (k>ns or (k=ns and i<j));  
if prioridad then  
  repretrasadas[j]:=true  
  else send(j, rep)  
endif;
```

secciones
de codigo
exclusion
mutua

Algoritmo de Ricart-Agrawala-II



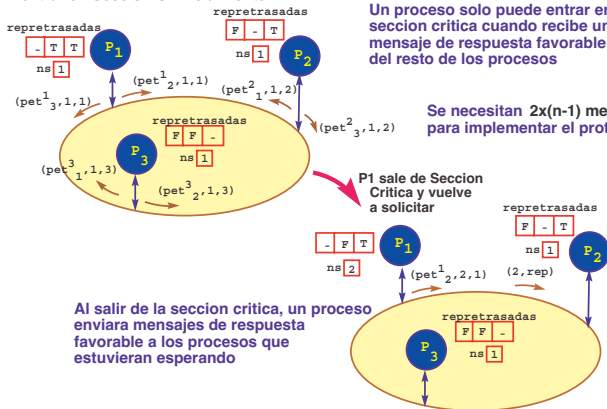
Escenario en el que 3 procesos intentan entrar en Sección C. inicialmente

Un proceso solo puede entrar en seccion critica cuando recibe un mensaje de respuesta favorable del resto de los procesos

Se necesitan $2x(n-1)$ mensajes para implementar el protocolo

P1 sale de Seccion Critica y vuelve a solicitar

Al salir de la seccion critica, un proceso enviara mensajes de respuesta favorable a los procesos que estuvieran esperando



Exclusión mutua

Condiciones de Dijkstra

Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusion mutua

Verificación de las propiedades del algoritmo de Ricart-Agrawala



Exclusión mutua entre procesos al acceder a la sección crítica

- P_i y P_j consiguen entrar a la vez en S.C. (hipótesis de incorrección)
- $\Leftrightarrow P_{i,j}$ ha transmitido su petición a $P_{j,i}$, recibiendo contestación favorable
 - 1 P_i ha enviado contestación favorable antes de generar su número de secuencia $\Rightarrow P_j$ pospone la respuesta
 - 2 P_j ha enviado contestación favorable antes de generar su número de secuencia $\Rightarrow P_i$ pospone la respuesta
 - 3 Después de generar su número de secuencia es imposible que cada proceso envíe contestación favorable al otro

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua

Alcanzabilidad de la sección crítica

Debido a la ordenación total de las peticiones, es imposible que se retrase indefinidamente la contestación favorable a algún proceso

Algoritmo de Suzuki-Kasami-Ricart-Agrawala



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusión mutua

Proceso P(i)

Variables locales a P(i)

```
token_presente: boolean;  
en_SC: boolean;  
token: array[1..N] of 0..+INF;  
peticion: array[1..N] of 0..+INF;
```

```
Hebra pet(i) ::=  
  receive(pet,k,j);  
  peticion[j] := max(peticion[j],k)  
  if token_presente and  
    not en_SC then  
    for j:=i+1 to n, 1 to i-1 do  
      if peticion[j]>token[j] and  
        token_presente then  
        begin  
          token_presente:= false;  
          send(j,acceso,token);  
        end;  
      endif;  
    enddo
```

secciones
de código
exclusión
mutua

Hebra main(i) ::=

```
  if NOT token_presente then  
    begin  
      ns:= ns+1;  
      broadcast(pet,ns,i);  
      receive(acceso,token);  
      token_presente:= true;  
    end;  
  endif;  
  en_SC:=true;
```

```
<<Seccion  
  Critica>>  
  token[i] := ns;  
  en_SC:= false;
```

```
  for j:=i+1 to n, 1 to i-1 do  
    if peticion[j]>token[j] and  
      token_presente then  
      begin  
        token_presente:= false;  
        send(j,acceso,token);  
      end;  
    endif;  
  enddo
```

Verificación del algoritmo de Suzuki-Kasami-Ricart-Agrawala



- El que todo proceso acceda en exclusión mutua es equivalente a demostrar el siguiente aserto: "globalmente, el número de variables *token_presente* = *true* es idénticamente igual a la unidad"
 - 1 El aserto anterior se satisface inicialmente
 - 2 El aserto anterior se cumple cada vez que se transmite el token. El protocolo necesita **n** mensajes.

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Alcanzabilidad de la sección crítica

- Si ningún proceso posee el token, en un momento de la ejecución del algoritmo, este ha de estar necesariamente en transmisión

Algoritmos distribuidos
para el problema de
exclusión mutua

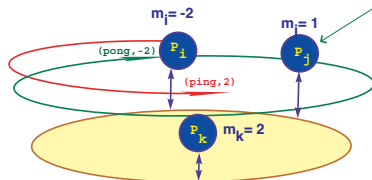
Propiedad de equidad

- Se obliga a que P_j transmita el token al primero que lo solicitó en el orden $\{j + 1, j + 2, \dots, n, n - 1, \dots\}$
- ¿Qué pasa si se pierden mensajes?

Algoritmo de regeneración de token de Misra

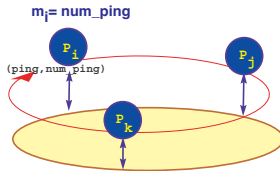


Actualización de tokens



se han encontrado aquí
la primera vez

Caso de pérdida de 1 token



Relacion invariante durante toda la ejecución

$$\text{num_ping} + \text{num_pong} = 0$$

Inicialmente, ambos tokens asignados
a un solo nodo con:

$$\text{num_ping} = 1, \text{ num_pong} = -1$$

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

Algoritmo de regeneración de token de Misra-II



Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento
sucesivo

Algoritmo de Dijkstra y
problemas de vivacidad

Algoritmo de Knuth y
equidad relativa en el
acceso a la SC

Solución totalmente
correcta para N procesos

Algoritmos distribuidos
para el problema de
exclusion mutua

Proceso (i)

```
while true do      variable local mi: int:=0;
  Seleccionar
  Cuando recibido(ping,num_ping) hacer
    if mi= num_ping then
      begin -- se ha perdido pong, hay que recuperarlo
        num_ping := (num_ping+1)mod n+1;
        num_pong := -num_ping;
      end
    else mi:= num_ping;
    endif;
  endhacer;
  Cuando recibido(pong,num_pong) hacer
    if mi= num_pong then
      begin -- se ha perdido ping, hay que recuperarlo
        num_pong := -((-num_pong+1)mod n+1);
        num_ping := -num_pong;
      end
    else mi:= num_pong;
    endif;
  endhacer;
  Cuando se encuentran(ping,pong) hacer
    begin -- |num_ping|=|num_pong|,llevan el "num.colisiones"
      num_ping := (num_ping+1)mod n+1;
      num_pong := -num_ping;
    end
  endhacer;
endseleccionar;
enddo;
```



Para más información, ejercicios, bibliografía adicional, o "simplemente inspiración" sobre la temática, se puede consultar:

Capel, M.I.(2022). "Programación Concurrente y en Tiempo Real". capítulo 2

Michel Raynal (1986). Algorithms for mutual exclusion. Cambridge: MIT Press.

Michel Raynal (2013). Distributed algorithms for message-passing systems. Springer.

Exclusión mutua

Condiciones de Dijkstra
Método de refinamiento sucesivo

Algoritmo de Dijkstra y problemas de vivacidad

Algoritmo de Knuth y equidad relativa en el acceso a la SC

Solución totalmente correcta para N procesos

Algoritmos distribuidos para el problema de exclusión mutua