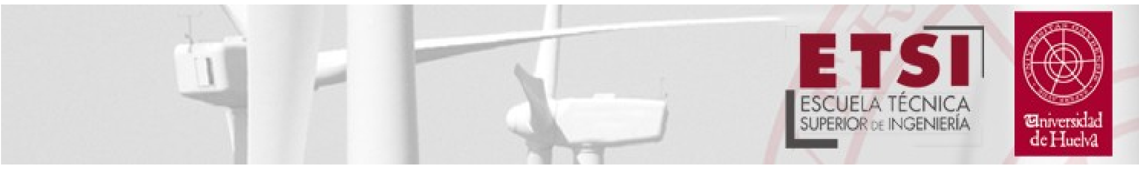


Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

PROBLEMAS DE INTERBLOQUEOS

Problema 1.

Tenemos un sistema operativo donde se están ejecutando 4 procesos y hay 5 recursos. Se conoce que las necesidades máximas de cada proceso son:

Matriz de necesidades máximas

	R1	R2	R3	R4
A	1	1	2	1
B	0	2	1	2
C	2	0	2	1
D	1	2	1	2
E	1	1	1	0

Inicialmente los recursos existentes son E= (3, 2 ,3, 2).

Los procesos tienen asignados los siguientes recursos:

- A tiene 1 recurso de R1
- B tiene 1 recurso de R2
- C tiene 1 recurso de R3
- D tiene 1 recurso de R1
- E tiene 1 recurso de R3

Se está usando como algoritmo de evitación del interbloqueo el Algoritmo del Banquero.

1. ¿En qué estado se encuentra el sistema? → Algoritmo del Banquero
2. Si el proceso D solicita 1 ejemplar de R4. ¿Debe concederse la petición?.

Justifique las respuestas con la técnica adecuada.

Solución

1. Usamos el algoritmo del banquero para ver si el estado es seguro o inseguro

ENUNCIADO

	ASIGNADOS				NEC. MÁXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
A	1	0	0	0	1	1	2	1	0	1	2	1
B	0	1	0	0	0	2	1	2	0	1	1	2
C	0	0	1	0	2	0	2	1	2	0	1	1
D	1	0	0	0	1	2	1	2	0	2	1	2
E	0	0	1	0	1	1	1	0	1	1	0	0

E=(3,2,3,2) → Existentes → ENUNCIADO

A=(2,1,2,0) → Asignados → Suma por columnas Asignados[ ][ ]

L=(1,1,1,2) → E[ ] − A[ ] → Libres

Aplicamos el Banquero: L = L + PENDIENTES[B][ ]

- Puede finalizar B. L = (1,2,1,2) → Como es Candidato → Termina
- Puede finalizar D. L= (2,2,1,2)
- Puede finalizar C. L= (2,2,2,2)
- Puede finalizar A. L= (3,2,2,2)
- Puede finalizar E. L= (3,2,3,2)
- Por tanto el estado es seguro. → Porque han finalizado todos los procesos

$\boxed{L[\,] = E[\,]}$

2. Si se concediese la petición (D solicita 1 de R4) el sistema quedaría en el siguiente estado

	ASIGNADOS				NEC. MÁXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
A	1	0	0	0	1	1	2	1	0	1	2	1
B	0	1	0	0	0	2	1	2	0	1	1	2
C	0	0	1	0	2	0	2	1	2	0	1	1
D	1	0	0	**1**	1	2	1	2	0	2	1	**1**
E	0	0	1	0	1	1	1	0	1	1	0	0

E=(3,2,3,2)

A=(2,1,2,1)

L=(1,1,1,1)

Aplicamos el Banquero:

- Puede finalizar el proceso E. L=(1,1,2,1)
- Puede finalizar el proceso A. L=(2,1,2,1)
- Puede finalizar el proceso C. L=(2,1,3,1)
- Los procesos B y D no pueden finalizar, por tanto el estado es inseguro.

No podemos decir que el estado esté interbloqueado o negarlo. Solo podemos afirmar que si se concede D, el estado se interbloqueará.

Problema 2

En un instante determinado la situación de un sistema con 4 procesos y 4 recursos es la siguiente:

	ASIGNADOS				**NECESIDADES MÁXIMAS**			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	1	1	1	0
P2	0	1	0	1	1	1	0	1
P3	0	1	1	0	0	1	1	1
P4	0	0	1	0	0	0	2	0

Conociendo que existe 1 recurso de R1, 2 recursos de R2, 3 recursos de R3 y 2 recursos de R4:

1. ¿Se encuentra el sistema en un estado seguro? → Banquero
2. ¿Existirá interbloqueo tras una petición del recurso R2 por parte del proceso P1?.

Reducc. Grafo

Use las técnicas apropiadas en cada caso.

Solución

1. Calculamos la matriz de pendientes y los vectores E, A y L

	ASIGNADOS				**NEC. MAXIMAS**				**PENDIENTES**			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	1	1	1	0	0	1	1	0
P2	0	1	0	1	1	1	0	1	1	0	0	0
P3	0	1	1	0	0	1	1	1	0	0	0	1
P4	0	0	1	0	0	0	2	0	0	0	1	0

E= (1,2,3,2)

A= (1,2,2,1) → L= (0,0,1,1)

Buscamos una fila en la matriz de pendientes menor o igual al vector L

P1 → (0,1,1,0) < L → No

P2 → (1,0,0,0) < L → No

P3 → (0,0,0,1) < L → Si (*)

P4 → (0,0,1,0) < L → Si

Se puede seleccionar a P3 o P4, nos decidimos por P3 (es indiferente cual se seleccione). Finalizamos P3 añadiendo sus recursos al nuevo vector L.

L= (0,0,1,1) + (0,1,1,0) = (0,1,2,1)

Volvemos a buscar un fila menor o igual al nuevo L

P1 → (0,1,1,0) < L → Si

P2 → (1,0,0,0) < L → No

P4 → (0,0,1,0) < L → Si (*)

Se puede seleccionar a P1 o P4, nos decidimos por P4 (es indiferente cual se seleccione). Finalizamos P4 añadiendo sus recursos al nuevo vector L.

L= (0,1,2,1) + (0,0,1,0) = (0,1,3,1)

Volvemos a buscar un fila menor o igual al nuevo L

P1 → (0,1,1,0) < L → Si (*)

P2 → (1,0,0,0) < L → No

Se puede seleccionar a P1. Finalizamos P1 añadiendo sus recursos al nuevo vector L.

L= (0,1,3,1) + (1,0,0,0) = (1,1,3,1)

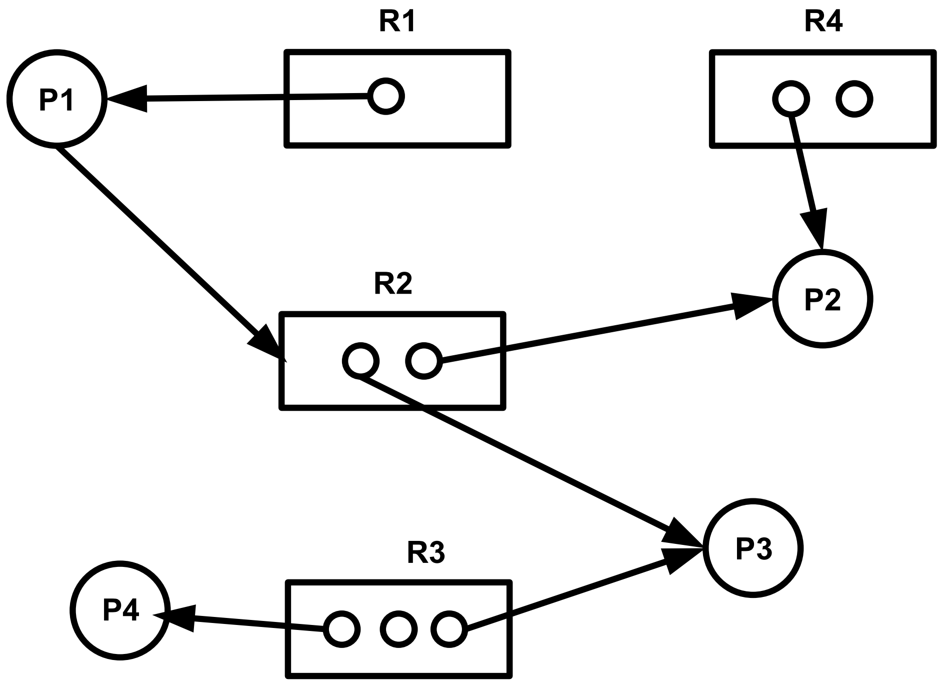
Volvemos a buscar un fila menor o igual al nuevo L

P2 → (1,0,0,0) < L → Si

Como es el único que queda, y puede finalizar puesto que su vector de pendientes es menor o igual que L, todos pueden finalizar y el estado del sistema es un **ESTADO SEGURO**.

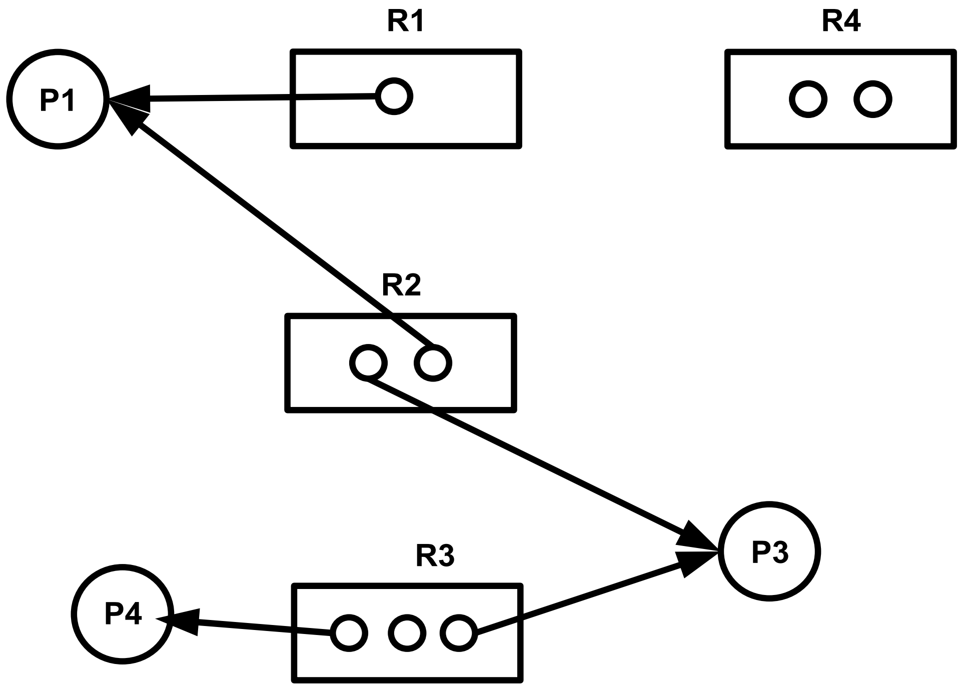
2. Debemos generar el grafo tras dicha petición y comprobar si es reducible. Sólo tenemos en cuenta las peticiones que se han realizado, la matriz de asignados, y la nueva petición. No se usan las necesidades máximas.

2)

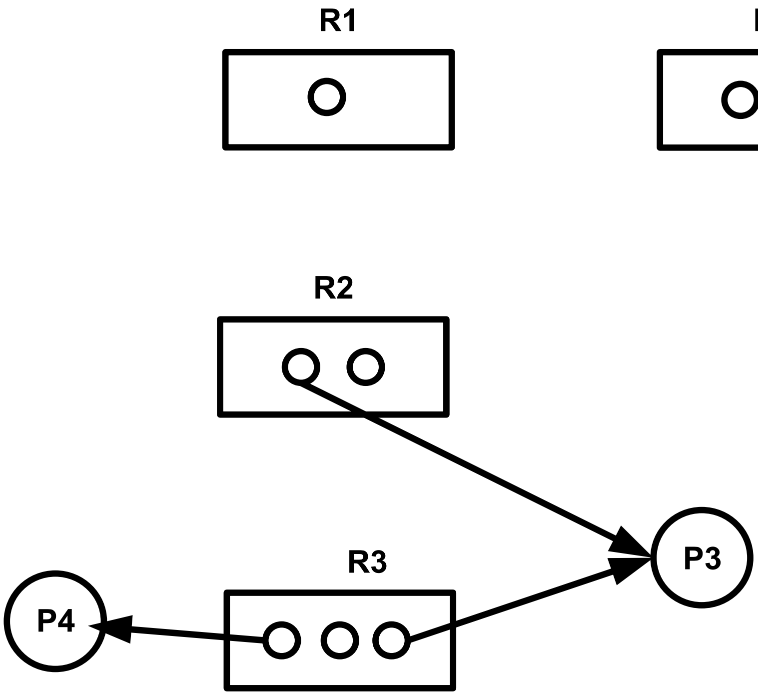


El grafo resultante será:

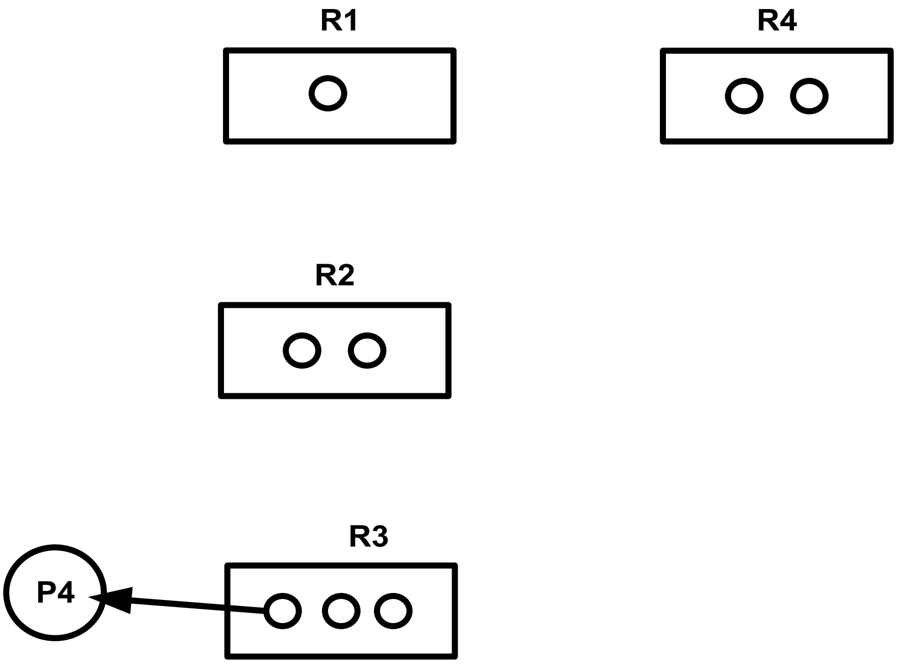
Reducimos por P2, que puede finalizar, puesto que no espera por nadie. Esto libera un ejemplar de R2, que puede ser asignado a P1 que espera por él.



Reducimos por P1



Y termina P4, con lo cual el grafo es totalmente reducido.

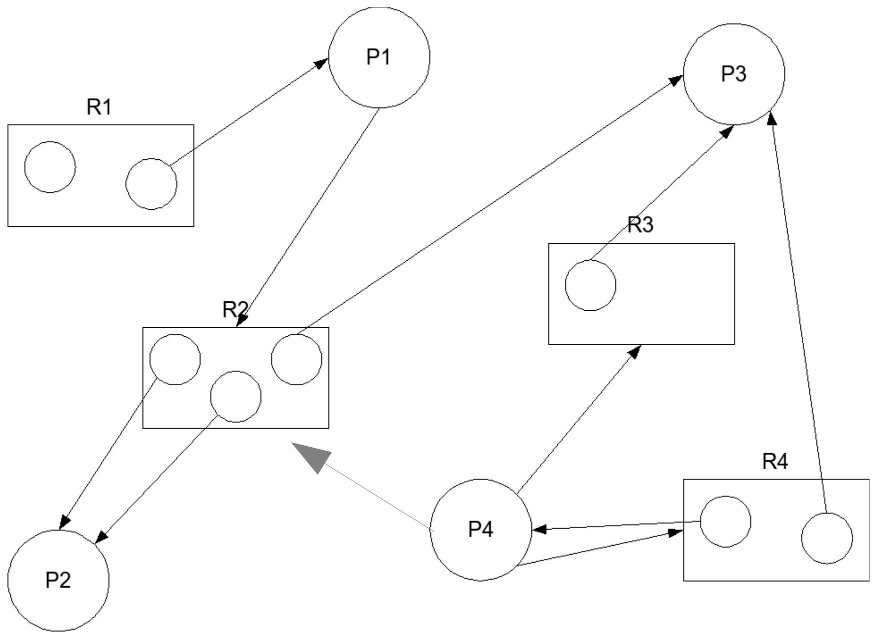


El grafo se puede reducir. Todos los procesos pueden finalizar. Por tanto, no existe interbloqueo.

Se puede decir que, como no forma ningún ciclo, siendo una característica necesaria para que haya interbloqueo, no lo habrá

Problema 3

Dado el siguiente grafo de asignación de recursos:



Y las siguientes necesidades máximas para cada proceso:

	R1	R2	R3	R4
P1	2	1	0	0
P2	1	2	0	0
P3	0	1	1	1
P4	0	1	1	2

Si el proceso P2 solicita un ejemplar del recurso R1:

1. ¿Se le debería conceder la petición sabiendo que estamos usando como técnica de evitación de interbloqueos el algoritmo del banquero?
2. ¿Si se concediese la petición, habría interbloqueo?

Solución

1. Como estamos usando una técnica de evitación, para ver si se puede conceder dicha petición aplicaremos el algoritmo del banquero tras la petición. A partir del grafo calculamos la matriz de asignados, y los vectores A y E. Luego, deducimos las necesidades Pendientes y el vector L. Recurso → Proceso

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	2	1	0	0	1	1	0	0
P2	0	2	0	0	1	2	0	0	1	0	0	0
P3	0	1	1	1	0	1	1	1	0	0	0	0
P4	0	0	0	1	0	1	1	2	0	1	1	1

E= (2,3,1,2)

A= (1,3,1,2) → L= (1,0,0,0)

Cuando el Proceso P2 solicita un recurso de R1 queda:

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	2	1	0	0	1	1	0	0
P2	1	2	0	0	1	2	0	0	0	0	0	0
P3	0	1	1	1	0	1	1	1	0	0	0	0
P4	0	0	0	1	0	1	1	2	0	1	1	1

E= (2,3,1,2)

A= (**2**,3,1,2) → L= (**0**,0,0,0)

Buscamos una fila en la matriz de pendientes menor o igual al vector L

P1 → (1,1,0,0) < L → No

P2 → (0,0,0,0) < L → Si (*)

P3 → (0,0,0,0) < L → Si

P4 → (0,1,1,1) < L → No

Se puede seleccionar a P2 o P3, nos decidimos por P2 (es indiferente cual se seleccione). Finalizamos P2 añadiendo sus recursos al nuevo vector L.

L= (0,0,0,0) + (1,2,0,0) = (1,2,0,0)

Volvemos a buscar un fila menor o igual al nuevo L

P1 → (1,1,0,0) < L → Si

P3 → (0,0,0,0) < L → Si (*)

P4 → (0,1,1,1) < L → No

Se puede seleccionar a P1 o P3, nos decidimos por P3 (es indiferente cual se seleccione). Finalizamos P3 añadiendo sus recursos al nuevo vector L.

L= (1,2,0,0) + (0,1,1,1) = (1,3,1,1)

Volvemos a buscar un fila menor o igual al nuevo L

P1 → (1,1,0,0) < L → Si (*)

P4 → (0,1,1,1) < L → Si

Se puede seleccionar a P1 o P4, nos decidimos por P1 (es indiferente cual se seleccione). Finalizamos P1 añadiendo sus recursos al nuevo vector L.

L= (1,3,1,1) + (1,0,0,0) = (2,3,1,1)

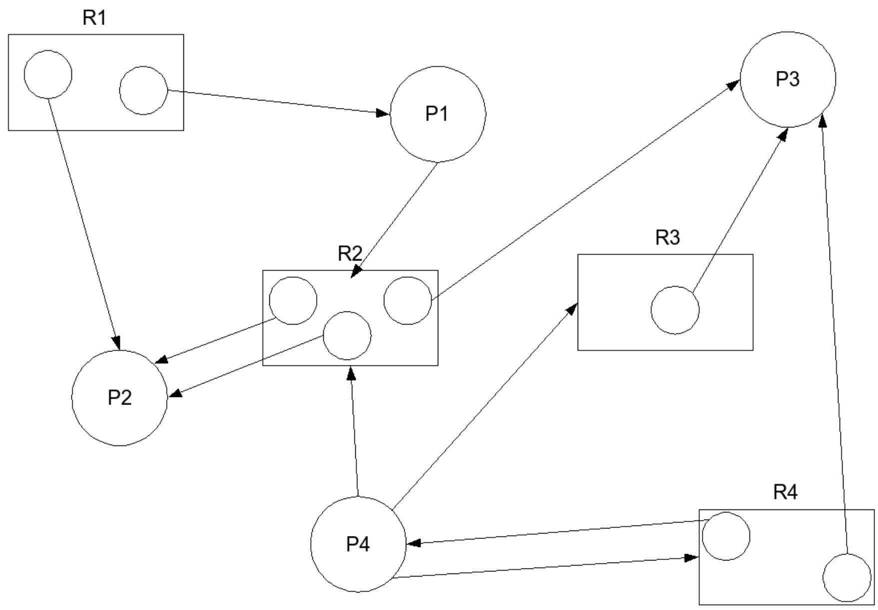
Volvemos a buscar un fila menor o igual al nuevo L

P4 → (0,1,1,1) < L → Si

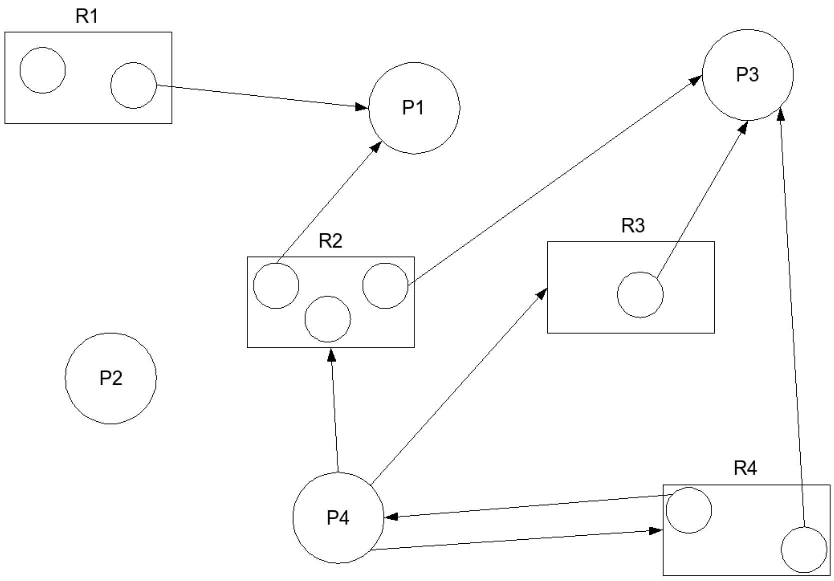
Como es el único que queda, y puede finalizar puesto que su vector de pendientes es menor o igual que L, todos pueden finalizar y el estado del sistema tras la petición será un **ESTADO SEGURO.**

2. Vamos a ver ahora si hay interbloqueo. Para ello, añadiremos la nueva petición al grafo y comprobaremos si es reducible.

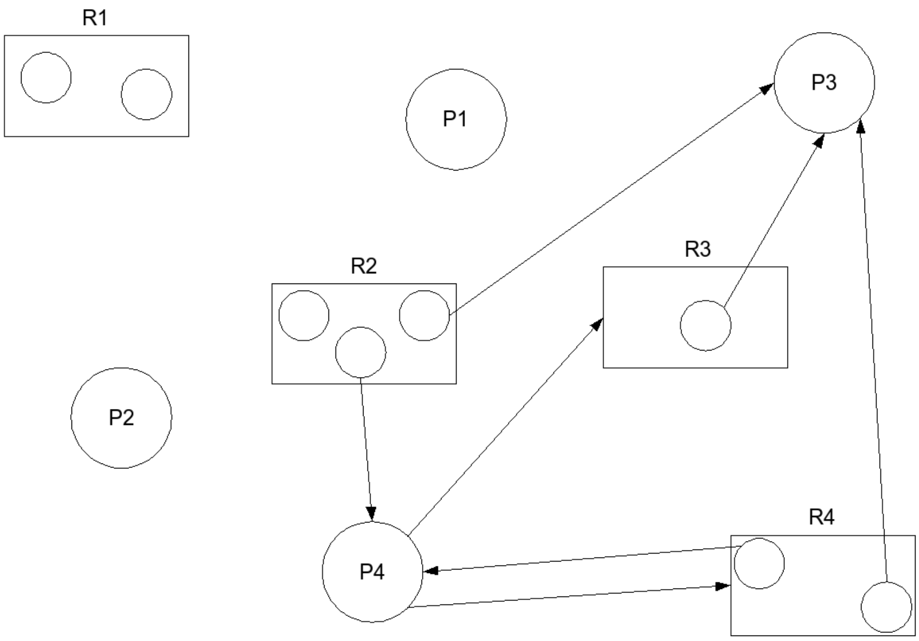
   El grafo con la nueva petición quedará:



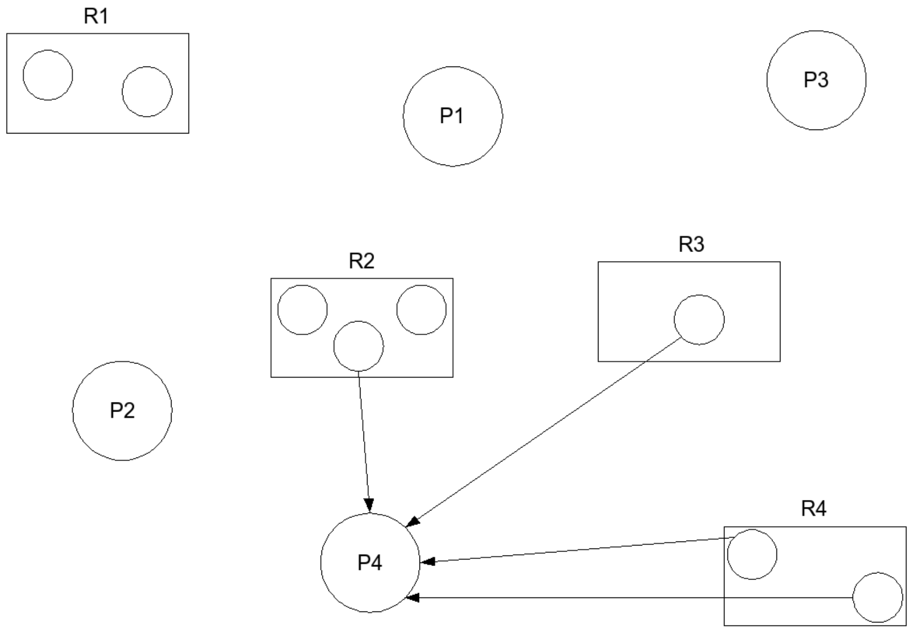
Reducimos por P2, y reasignamos los recursos que se liberan:



Reducimos por P1



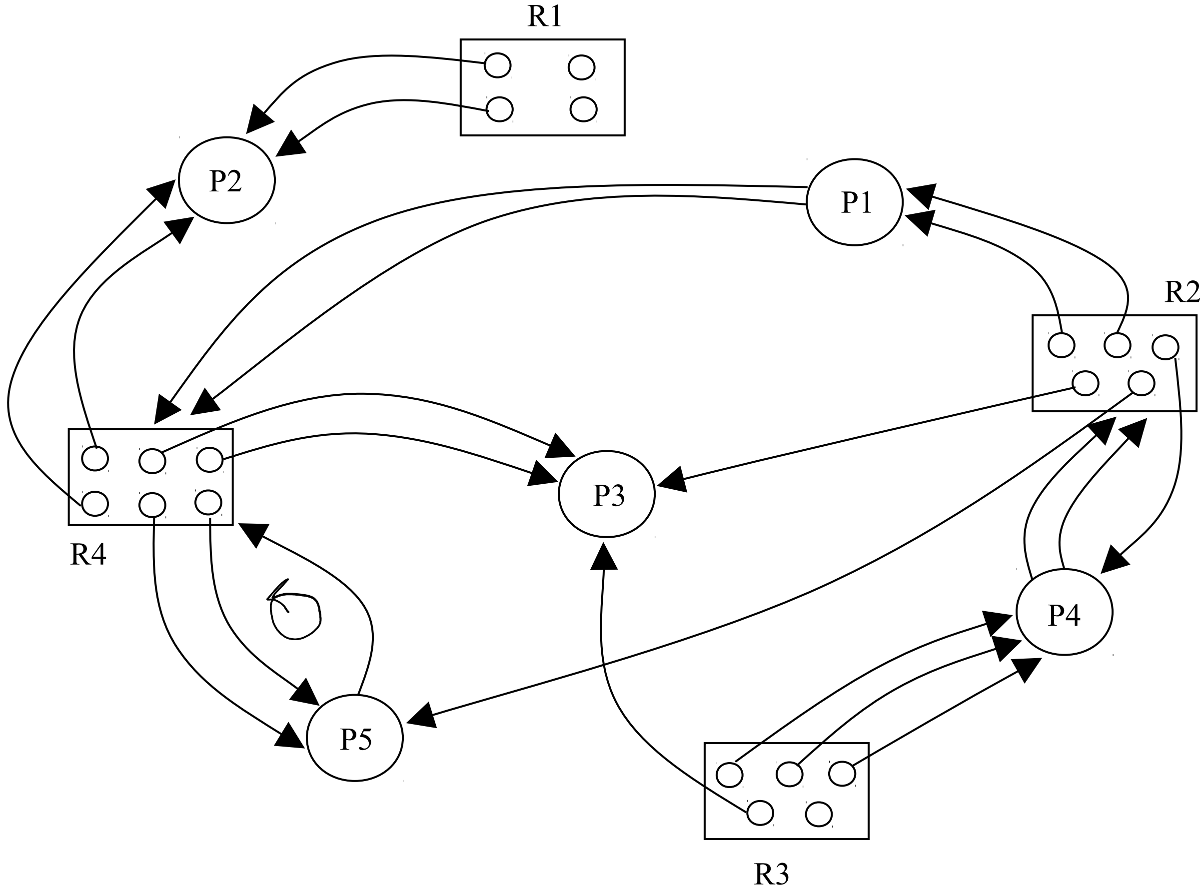
Reducimos por P3 y reasignamos:



Y finalmente finaliza P4, con lo cual el grafo es reducible, y por tanto, no existe interbloqueo.

Problema 4

Dado el siguiente grafo de asignación de recursos:



y sabiendo que:

- A P1 le queda por pedir 1 ejemplar del recurso R2 y
- A P3 le queda por pedir 1 ejemplar del recurso R2 y otro ejemplar del recurso R3

Se pide:

1. ¿Se encuentra el sistema en estado seguro o inseguro? → Banquero
2. Si P3 realiza la petición de 1 ejemplar del recurso R3. ¿Se concederá la petición si estamos en un sistema que usa un algoritmo de evitación del interbloqueo?
3. Si se añaden al grafo las 3 solicitudes que faltaban. ¿Existiría interbloqueo?

Solución

Calculamos las matrices:

(Proceso → Rec)+
(Enunciado)
(A+P)

R → Proceso

	Asignados				Nec. Máximas				Pendientes			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	2	0	0	0	3	0	2	0	1	0	2
P2	2	0	0	2	2	0	0	2	0	0	0	0
P3	0	1	1	2	0	2	2	2	0	1	1	0
P4	0	1	3	0	0	3	3	0	0	2	0	0
P5	0	1	0	2	0	1	0	3	0	0	0	1

E = (4, 5, 5, 6) A = (2, 5, 4, 6) L = (2, 0, 1, 0)

1. Hay que aplicar el algoritmo del banquero

P1 → No
P2 → Si → L = (4, 0, 1, 2)
↓
P1 → No
P3 → No
P4 → No
P5 → Si → L = (4, 1, 1, 4)
↓
P1 → Si → L = (4, 3, 1, 4)
↓
P3 → Si → L = (4, 4, 2, 6)
↓
P4 → Si → L = (4, 5, 5, 6)

Secuencia que hace terminar a todos los procesos: P2, P5, P1, P3 y P4 → **Estado SEGURO**

2. Volvemos a aplicar el algoritmo del banquero tras incorporar la nueva solicitud a las tablas:

	Asignados				Nec. Máximas				Pendientes			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	2	0	0	0	3	0	2	0	1	0	2
P2	2	0	0	2	2	0	0	2	0	0	0	0
P3	0	1	**2**	2	0	2	2	2	0	1	**0**	0
P4	0	1	3	0	0	3	3	0	0	2	0	0
P5	0	1	0	2	0	1	0	3	0	0	0	1

E = (4, 5, 5, 6) A = (2, 5, **5**, 6) L = (2, 0, **0**, 0)

Ejemplares de cada recurso.

P1 → No

P2 → Si → L = (4, 0, 0, 2)

↓

P1 → No
P3 → No
P4 → No
P5 → Si → L = (4, 1, 0, 4)

↓

P1 → Si → L = (4, 3, 0, 4)

↓

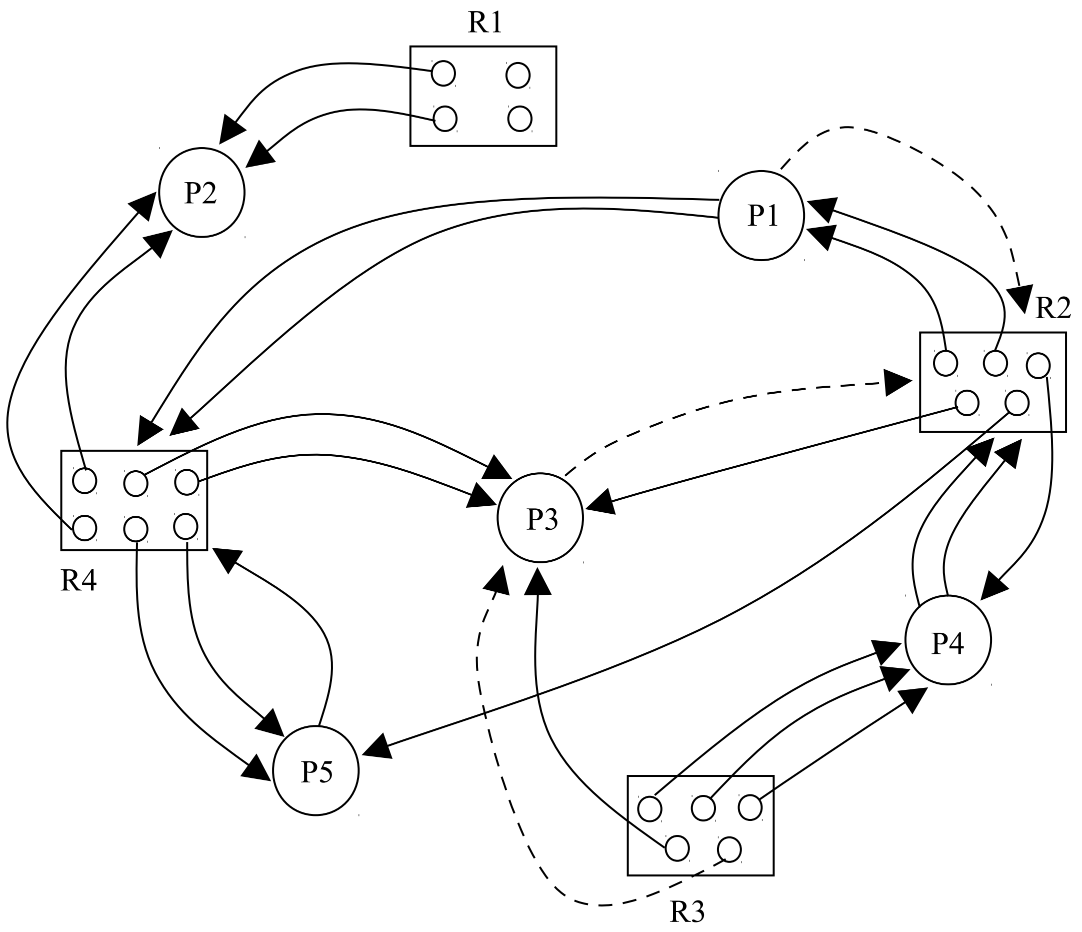
P3 → Si → L = (4, 4, 2, 6)

↓

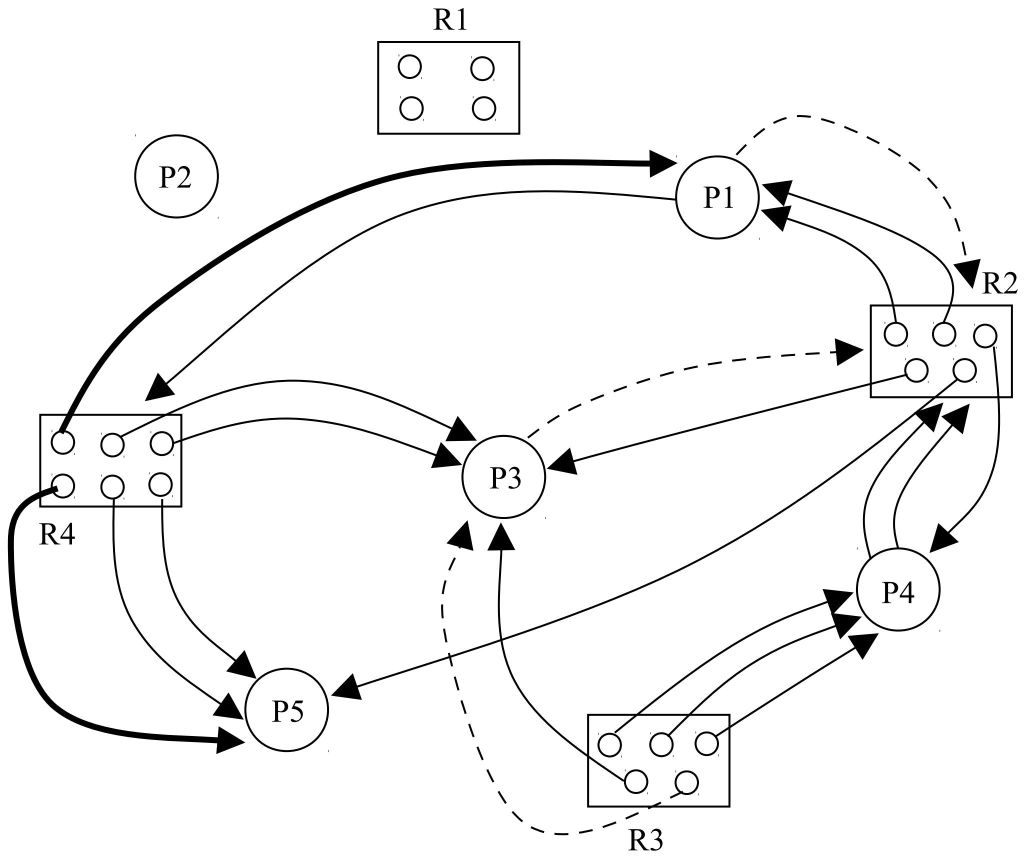
P4 → Si → L = (4, 5, 5, 6)

Secuencia que hace terminar a todos los procesos: P2, P5, P1, P3 y P4 → **Estado SEGURO. La petición puede concederse.**

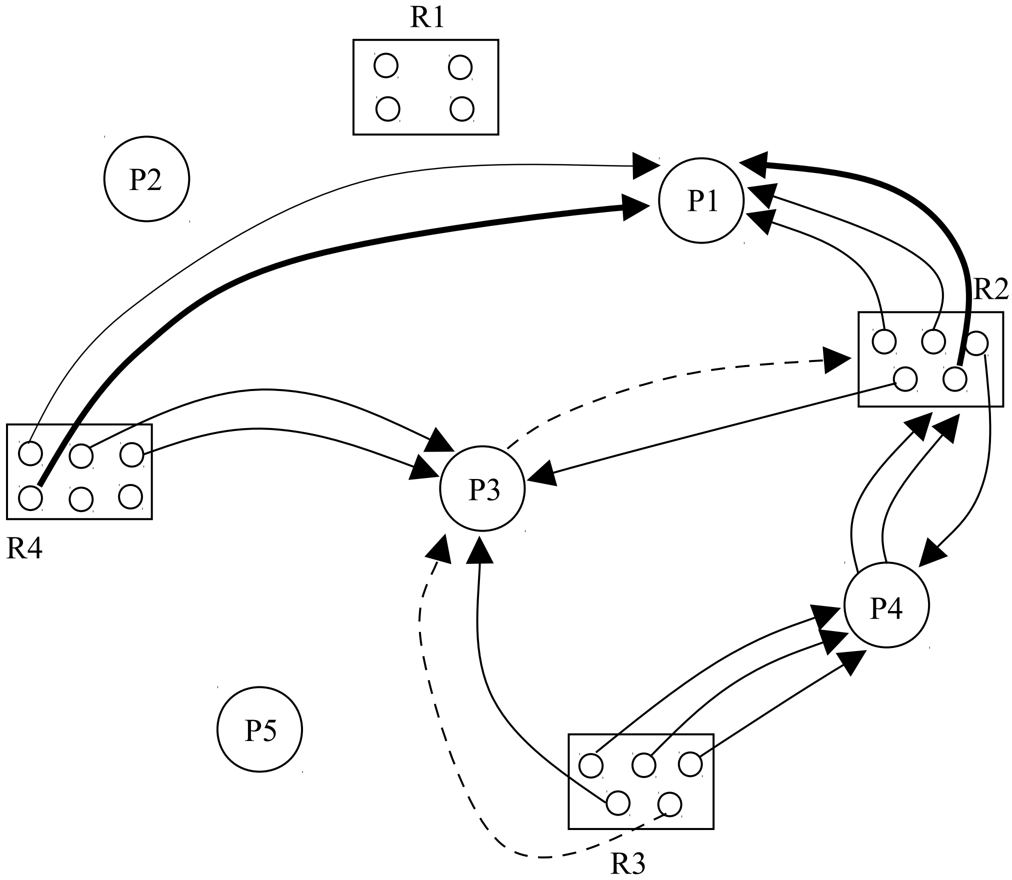
3. Se añaden las solicitudes al grafo (en línea discontinua) y se trata de reducir:



Se reduce P2. Los dos recursos que deja libre de R4 se conceden uno a P5 y otro a P1 (en negrita). Es importante advertir que si los dos recursos se conceden a P1 el grafo no se podría reducir (pero si encontramos un camino por el que el grafo se reduce entonces no está interbloqueado). El grafo resultante es



Se reduce P5. Libera 3 recursos de R4 y 1 recurso de R2. Se asignan a P1 y el grafo resultante es:



Se reduce P1. Libera 2 recursos de R4 y 3 recursos de R2. Tanto P3 como P4 pueden obtener todos los recursos que necesitan y podrían finalizar.

El grafo se puede reducir ➔ **NO HAY INTERBLOQUEO.**

Problema 5

Tenemos un sistema operativo donde se están ejecutando 4 procesos y hay 4 recursos. Se conoce que las necesidades máximas de cada proceso son:

	R1	R2	R3	R4
P1	1	1	0	1
P2	1	2	0	0
P3	0	0	2	1
P4	1	1	1	0

Inicialmente los recursos existentes son E= (2, 2 ,4, 2), y todos están libres.

Los procesos van solicitando los siguientes recursos:

1. P1 solicita R1
2. P2 solicita R2
3. P4 solicita R2
4. P2 solicita R1
5. P4 solicita R3
6. P3 solicita R3
7. P3 solicita R4
8. P1 solicita R4
9. P1 solicita R2
10. P2 solicita R2
11. P4 solicita R1
12. P3 solicita R3

Demuestre si se llega o no a una situación de interbloqueo.

En el caso de que el interbloqueo ocurra:

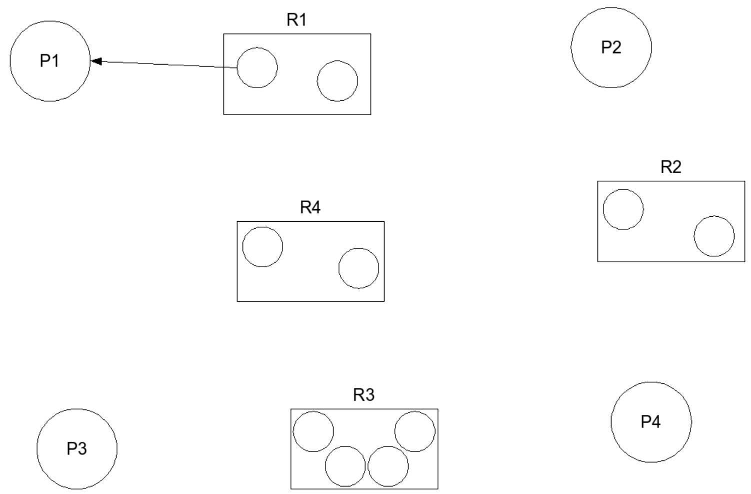
1. ¿Qué solicitud lo provoca? ¿A qué procesos afecta?
2. ¿Cómo podría haberse evitado el interbloqueo? ¿Qué solicitud debería haberse denegado?

Solución

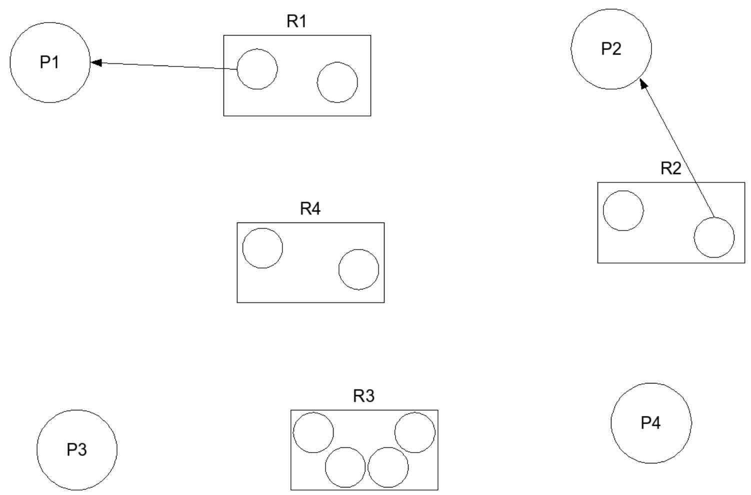
1. Para ver si se llega al interbloqueo, vamos a ir generando el grafo solicitud a solicitud, e ir comprobando si el grafo resultante se puede reducir. La solicitud que provoque que el grafo no sea reducible, es la que produce el interbloqueo.

   Sólo se comprobará si el grafo es reducible cuando se localice un ciclo, en caso contrario no hace falta.

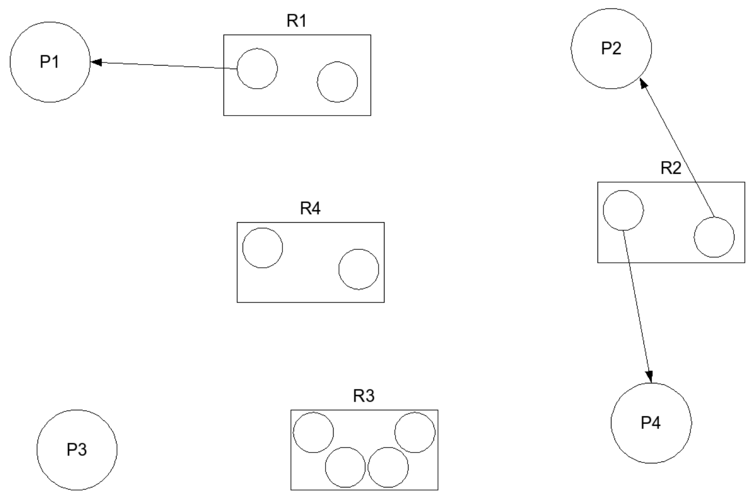
   Solicitud 1: P1 solicita R1



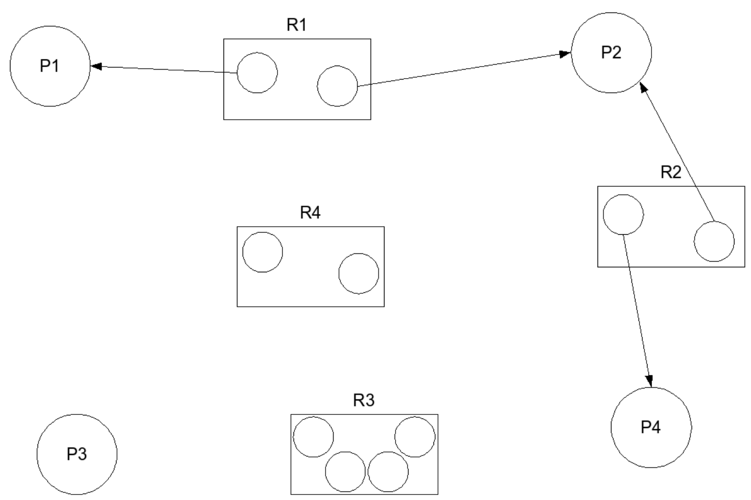
   Solicitud 2: P2 solicita R2



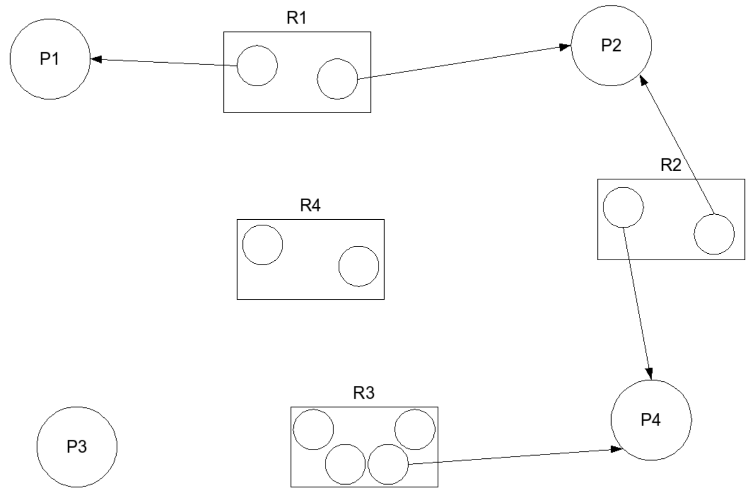
Solicitud 3: P4 solicita R2



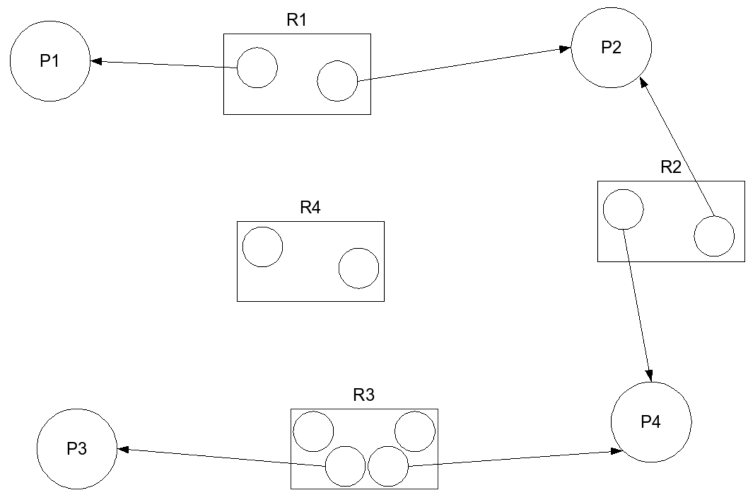
Solicitud 4: P2 solicita R1



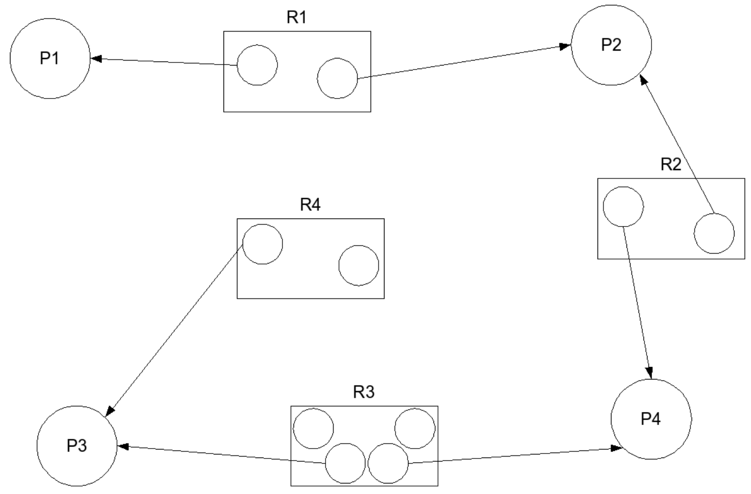
Solicitud 5: P4 solicita R3



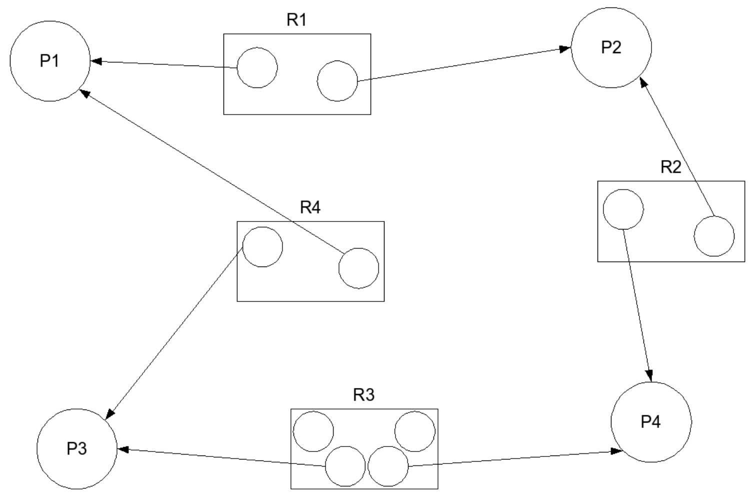
Solicitud 6: P3 solicita R3



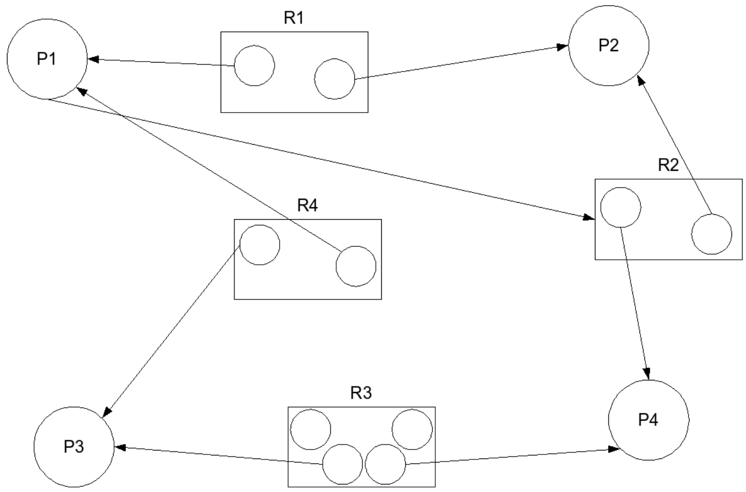
Solicitud 7: P3 solicita R4



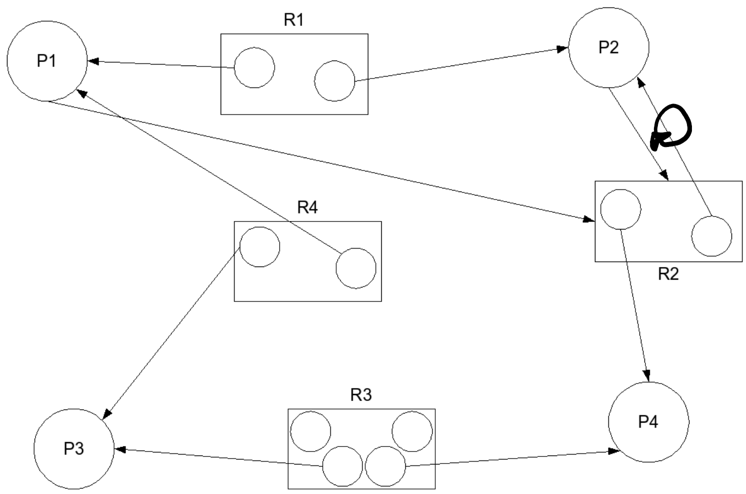
Solicitud 8: P1 solicita R4



Solicitud 9: P1 solicita R2

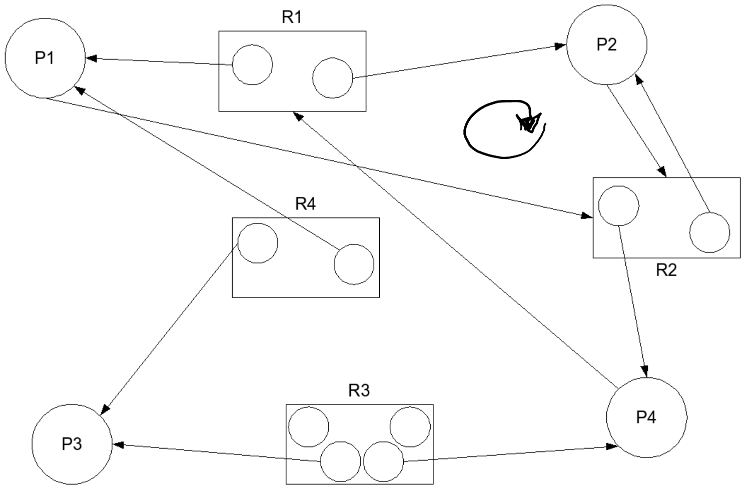


Solicitud 10: P2 solicita R2



Si reduzco P4, quede Libre R2 y deshace el ciclo.

Solicitud 11: P4 solicita R1



Tras esta solicitud se localizan dos ciclos: P1, R2, P4, R1, P1 y otro entre P2, R2, P4, R1, P2. Para saber si existe interbloqueo hay que reducir el grafo.

Este grafo no es reducible. Se podría reducir P3, pero luego P1, P2 y P4 no se puede. Con lo cual, esta solicitud es la que produce el interbloqueo (11.- P4 solicita R1). Los procesos interbloqueados son P1, P2 Y P4.

Aplicar DyV

2. Para ver cómo podría haberse evitado el interbloqueo, vamos a usar un algoritmo de evitación (el del banquero) y comprobar que solicitud produce el estado inseguro.

Calculamos las matrices de asignados, necesidades máximas y pendientes, así como los vectores E, A y L:

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	0	0	1	1	0	1	1	1	0	1
P2	0	0	0	0	1	2	0	0	1	2	0	0
P3	0	0	0	0	0	0	2	1	0	0	2	1
P4	0	0	0	0	1	1	1	0	1	1	1	0

E= (2,2,4,2)

A= (0,0,0,0) → L= (2,2,4,2)

- Tras P1 solicita R1 queda:

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	1	1	0	1	0	1	0	1
P2	0	0	0	0	1	2	0	0	1	2	0	0
P3	0	0	0	0	0	0	2	1	0	0	2	1
P4	0	0	0	0	1	1	1	0	1	1	1	0

E= (2,2,4,2)

A= (1,0,0,0) → L= (1,2,4,2)

y el estado es seguro.

- Tras P2 solicita R2 queda:

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	1	1	0	1	0	1	0	1
P2	0	1	0	0	1	2	0	0	1	1	0	0
P3	0	0	0	0	0	0	2	1	0	0	2	1
P4	0	0	0	0	1	1	1	0	1	1	1	0

E= (2,2,4,2)

A= (1,1,0,0) → L= (1,1,4,2)

y el estado es seguro.

- Tras P4 solicita R2 queda:

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	1	1	0	1	0	1	0	1
P2	0	1	0	0	1	2	0	0	1	1	0	0
P3	0	0	0	0	0	0	2	1	0	0	2	1
P4	0	1	0	0	1	1	1	0	1	0	1	0

E= (2,2,4,2)

A= (1,2,0,0) → L= (1,0,4,2)

y el estado es seguro.

- Tras P2 solicita R1 queda:

	ASIGNADOS				NEC. MAXIMAS				PENDIENTES			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	1	0	0	0	1	1	0	1	0	1	0	1
P2	1	1	0	0	1	2	0	0	0	1	0	0
P3	0	0	0	0	0	0	2	1	0	0	2	1
P4	0	1	0	0	1	1	1	0	1	0	1	0

E= (2,2,4,2)

A= (2,2,0,0) → L= (0,0,4,2)

y en esta situación podemos finalizar P3, pero luego no se puede localizar ninguna fila que sea menor o igual que L, con lo cual **el estado es inseguro**, y por tanto esa solicitud no debería haberse concedido para evitar el interbloqueo.

Tras finalizar P3 se comprueba que el nuevo vector de libres es L=(0, 0, 4 , 2) y que los vectores pendientes de P1, P2 y P4 son:

P1 → (0,1,0,1) < L → No

P2 → (0,1,0,0) < L → No

P4 → (1,0,1,0) < L → No

Con lo cual, el estado es inseguro.