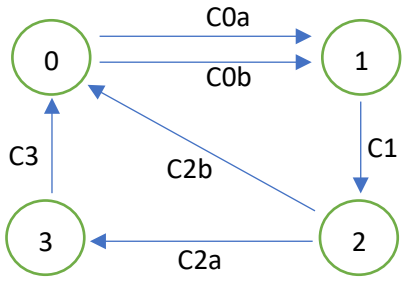
ARQUITECTURA DE COMPUTADORES

Se dispone de la siguiente red estática:



1) Dado el siguiente algoritmo de encaminamiento para la anterior red:

```
Procedure alg1(actual, destino) {
      If (actual==destino) canal=CPU;
      Else if (actual==0) canal=selecciona(C0a,C0b);
      Else if (actual==2)
         If (destino==3) canal=C2a;
         Else canal=C2b;
      Else Canal=Cactual;
}
```

Responde a las siguientes cuestiones:

a) ¿Es determinista o adaptativo? Justifica la respuesta.
b) ¿Busca siempre el camino mínimo (minimal)? Justifica la respuesta.
c) Dibuja el Grafo de Dependencia de Canales y señala los ciclos que observes, si los hay.
d) ¿Dirías que el algoritmo está libre de interbloqueos? Justifica la respuesta.

2) Diseña y muestra, en pseudocódigo como el anterior, un algoritmo de encaminamiento **minimal**, **determinista** y **libre de interbloqueos** para dicha red. A partir del algoritmo diseñado:
   a) Justifica por qué el algoritmo es determinista y minimal.
   b) Dibuja el Grafo de Dependencia de Canales y señala los ciclos que observes, si los hay.
   c) Justifica por qué el algoritmo que has diseñado está libre de interbloqueos.

El ejercicio se hará sobre papel, a mano, y se pasará a pdf para subirlo a Aula Virtual.

SOLUCIÓN

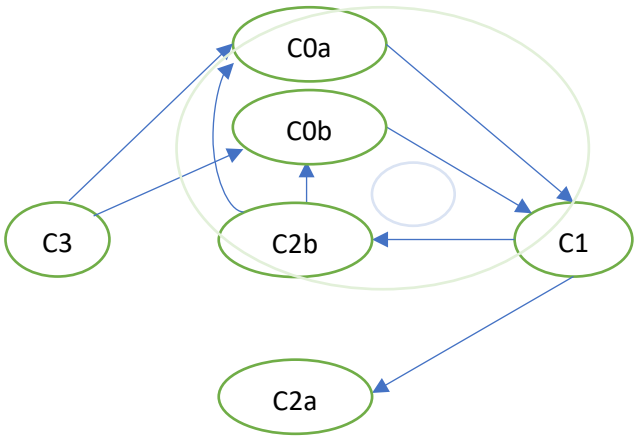
1.a) No es determinista, pues dado un origen y un destino, no siempre se va por el mismo camino (por ejemplo, para ir de 0 a cualquier otro nodo, podemos elegir entre dos caminos). Es adaptativo, pues puedo elegir entre varios caminos. De hecho, si suponemos caminos mínimos,

es completamente adaptativo, pues puedo elegir el camino que quiera de entre todos los posibles. Si no restringiéramos a caminos mínimos, entonces sería parcialmente adaptativo pues, por ejemplo, para ir de 2 a 0, solo me permite un camino de los dos posibles.

1.b) Sí que busca siempre el camino mínimo posible entre dos nodos. La única opción de que fuera por un camino más largo es que se permitiera pasar por 3 cuando va de 2 a 0, pero eso el algoritmo no lo contempla.

1.c) Grafo de dependencias entre canales:

Se observan dos ciclos marcados con óvalos de diferentes colores.



1.d) Aunque el grafo presenta varios ciclos, como es adaptativo, no podemos concluir que va a tener interbloqueos (bloqueos mortales) ni que está libre de ellos. Si fuera determinista, sí que podríamos decir que el algoritmo no está libre de interbloqueos, pero no es el caso.

No obstante, si encontramos una **configuración de interbloqueo**, es decir, una situación donde se produce un interbloqueo, entonces podemos concluir que el algoritmo no está libre de interbloqueos. En este caso es posible imaginar una situación de interbloqueo de la siguiente manera: 1 envía a 0, 2 envía a 1, 0 envía 2 y 3 envía a 2. Con el envío de 1 a 0, ocupamos C1 y luego quiero el C2b, con el envío de 2 a 1, ocupo el canal C2b (bloqueando al paquete que está en C1). Al mismo tiempo, 0 envía a 2 con lo que ocupa C0a (por ejemplo) pero se para pues C1 está ocupado. También al mismo tiempo, el paquete que había salido ya de 3, había llegado a 0 y ahora sale hacia 2, ocupando C0b, pero se bloquea pues C1 está ocupado. En ese instante se produce una configuración de interbloqueo y ningún paquete puede continuar. Por lo tanto, **el algoritmo no está libre de interbloqueos.**

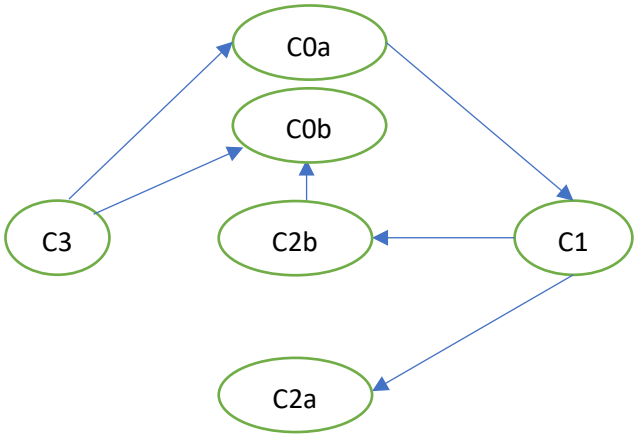
2) Hay más de una solución para obtener un algoritmo libre de interbloqueos dada la red propuesta. Lo más sencillo es crear un algoritmo cuyo grafo de dependencias no tenga ciclos. Para más sencillez se puede optar por un algoritmo determinista que solo nos permita una opción en cada caso. Para romper los dos ciclos del grafo anterior, podemos restringir si ir por C0a o C0b dependiendo del destino. Por ejemplo, si estando en 0, solo permito ir por C0b cuando el destino es 1, desaparece directamente la dependencia entre C0b y C1 rompiéndose el ciclo más pequeño (azul). Pero además, como cualquier paquete que viaje por C2b tiene su destino en 0 o en 1, nunca tomará el enlace C0a, por lo que también desaparece la dependencia entre C2a y C0a, rompiéndose el ciclo externo (verde). El algoritmo sería entonces:

```
Procedure alg2(actual, destino) {
      If (actual==destino) canal=CPU;
      Else if (actual==0)
         If (destino==1) canal=C0b;
         Else canal=C0a;
      Else if (actual==2)
         If (destino==3) canal=C2a;
         Else canal=C2b;
      Else Canal=Cactual;
}
```

2.a) El algoritmo es determinista pues dado un origen y un destino siempre elige el mismo canal de salida, no permite elegir entre varios. Es minimal, pues al igual que antes siempre elige caminos mínimos.

2.b) Grafo de dependencias entre canales. Es como el de antes, pero tal como se comentaba, desaparece la dependencia entre C0b y C1 (si un paquete va por C0b es porque su destino es 1 y no puede seguir por C1). También desaparece la dependencia entre C2b y C0a, pues si un paquete va por C2b es porque, o bien su destino es 0, en cuyo caso no sigue ni por C0a ni por C0b, o bien porque su destino es 1, en cuyo caso va por C0b, nunca por C0a.

El grafo resultante no tiene ciclos:



2.c) Como el grafo resultante no tiene ciclos, podemos concluir que el algoritmo de encaminamiento correspondiente está libre de interbloqueos.