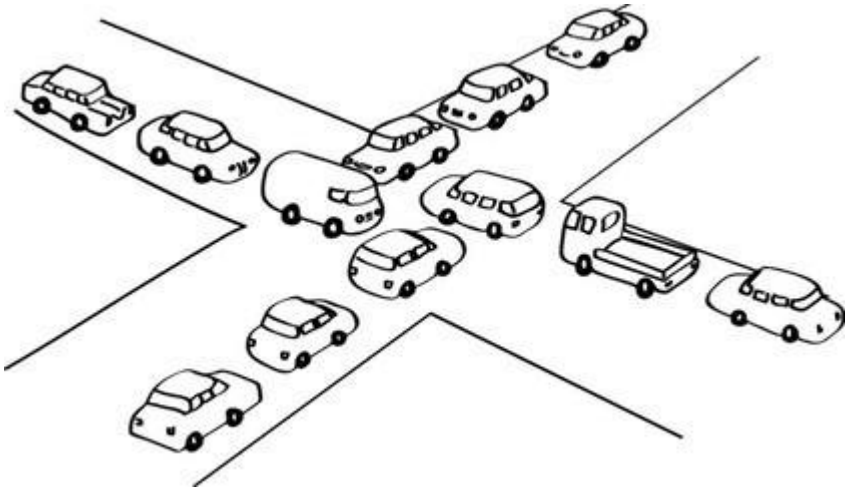
Interbloqueos



¿Qué es un interbloqueo?

Es un conjunto de procesos que se encuentra en un bloqueo mutuo si cada proceso en el conjunto está esperando un evento que sólo puede ser ocasionado por otro proceso en el conjunto.

EL ALGORITMO DE LA AVESTRUZ

Hace analogía a meter la cabeza en la arena y hacer como que si no lo veo, no pasa nada. Así que de esto salen dos formas de ver las cosas.
1-La perspectiva matemática: Considera el interbloqueo como inaceptable y argumenta que debe prevenirse a toda costa.
2-La perspectiva de los ingenieros: Evalúa la frecuencia y la gravedad del interbloqueo en comparación con otros fallos del sistema. Si el interbloqueo es infrecuente en comparación con otros problemas de rendimiento, algunos ingenieros pueden priorizar mantener el rendimiento del sistema en lugar de eliminar por completo el riesgo de interbloqueo.

Cuando dos procesos intentan acceder a recursos mutuamente excluyentes, como una unidad de CD-ROM y una impresora, lleva a un estancamiento. Bloquear o devolver una clave de error son dos posibilidades vistas en clase para evitar esto.

DETECCIÓN Y RECUPERACIÓN DE UN INTERBLOQUEO

Otra técnica es la detección y recuperación. Cuando se utiliza esta técnica, el sistema no trata de evitar los interbloqueos. En vez de ello, intenta detectarlos cuando ocurran y luego realiza cierta acción para recuperarse después del hecho.

Algunas de las formas en que se pueden detectar los interbloqueos, y ciertas maneras en que se puede llevar a cabo una recuperación de los mismos son las siguientes.

¿Cómo detectar interbloqueos?

Para interbloqueos con un recurso de cada tipo:

Considere un sistema con siete procesos, A a G, y seis recursos, R a W. El estado de cuáles recursos están contenidos por algún proceso y cuáles están siendo solicitados es el siguiente: (Se produce interbloqueos en D, E y G)

1. El proceso A contiene a R y quiere a S.
2. El proceso B no contiene ningún recurso pero quiere a T.
3. El proceso C no contiene ningún recurso pero quiere a S.
4. El proceso D contiene a U y quiere a S y a T.
5. El proceso E contiene a T y quiere a V.
6. El proceso F contiene a W y quiere a S.
7. El proceso G contiene a V y quiere a U

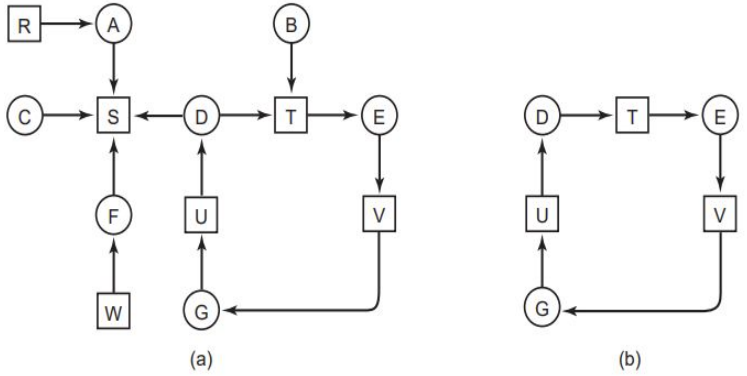


Figura 6-5. (a) Un gráfico de recursos. (b) Un ciclo extraído de (a).

¿Cómo detectar interbloqueos?

Para detectar este interbloqueo con un recurso de cada tipo se utiliza un algoritmo simple que inspecciona un gráfico y termina al haber encontrado un ciclo, o cuando ha demostrado que no existe ninguno. Utiliza cierta estructura dinámica de datos: L, una lista de nodos, así como la lista de arcos. Durante el algoritmo, los arcos se marcarán para indicar que ya han sido inspeccionados, para evitar repetir inspecciones.

Detección de interbloqueos con un recurso de cada tipo

El algoritmo opera al llevar a cabo los siguientes pasos, según lo especificado:

1. Para cada nodo N en el gráfico, realizar los siguientes cinco pasos con N como el nodo inicial.

2. Inicializar L con la lista vacía y designar todos los arcos como desmarcados.

3. Agregar el nodo actual al final de L y comprobar si el nodo ahora aparece dos veces en L. Si lo hace, el gráfico contiene un ciclo (listado en L) y el algoritmo termina.

4. Del nodo dado, ver si hay arcos salientes desmarcados. De ser así, ir al paso 5; en caso contrario, ir al paso 6.

5. Elegir un arco saliente desmarcado al azar y marcarlo. Después seguirlo hasta el nuevo nodo actual e ir al paso 3.

6. Si este nodo es el inicial, el gráfico no contiene ciclos y el algoritmo termina. En caso contrario, ahora hemos llegado a un punto muerto. Eliminarlo y regresar al nodo anterior; es decir, el que estaba justo antes de éste, hacerlo el nodo actual e ir al paso 3.

Detección de interbloqueos con un recurso de cada tipo

El algoritmo utilizado realiza una búsqueda a través de los nodos del gráfico, esperando encontrar un ciclo. Comienza desde un nodo específico y sigue un recorrido profundo en busca de ciclos. Si vuelve a un nodo ya visitado, detecta un ciclo. Si agota todas las ramificaciones desde un nodo sin encontrar ciclos, regresa al nodo anterior. Si completa un recorrido sin encontrar ciclos, el subgrafo desde ese nodo está libre de ciclos.

Para ver cómo funciona en la práctica, se aplica al recorrer los nodos del gráfico en un orden específico. Cada nodo se procesa siguiendo una secuencia, y si se detecta un ciclo, el algoritmo se detiene. Se describe cómo se ejecuta el algoritmo en el gráfico de recursos mediante una exploración de los nodos, registrando las visitas a través de una lista L. Si se encuentra un ciclo, se detiene el algoritmo para identificar y resolver el interbloqueo.

Detección del interbloqueo con varios recursos de cada tipo

Cuando hay múltiples copias de los recursos, se requiere un método diferente para detectar interbloqueos. Se presenta un algoritmo basado en matrices para la detección de interbloqueos entre n procesos, de P1 a Pn. Este algoritmo utiliza un vector de recursos existentes (E), que muestra el número total de instancias de cada recurso en el sistema. Por ejemplo, si la clase 1 son las unidades de cinta(para almacenamiento de datos) y E1 es 2, significa que hay dos unidades de cinta en el sistema.

Además, se utiliza un vector de recursos disponibles (A), donde cada valor proporciona la cantidad de instancias del recurso que están disponibles en un momento dado. Por ejemplo, si ambas unidades de cinta están asignadas, A1 será 0.

El algoritmo hace uso de dos matrices: C (matriz de asignaciones actuales) y R (matriz de peticiones). La fila i-ésima de la matriz C muestra cuántas instancias de cada clase de recurso contiene el proceso Pi en un momento dado. Por ejemplo, Cij representa el número de instancias del recurso j que están contenidas por el proceso i. De manera similar, Rij representa el número de instancias del recurso j que el proceso Pi desea.

Detección del interbloqueo con varios recursos de cada tipo

Hay una importante invariante que se aplica para estas cuatro estructuras de datos. En especial, cada recurso está asignado o está disponible. Esta observación significa que:

La sumatoria de matriz de asignaciones actuales(C)

más la matriz de recursos disponibles
es igual a la matriz de recursos existentes.

$$\sum_{i=1}^{n} C_{ij} + A_j = E_j$$

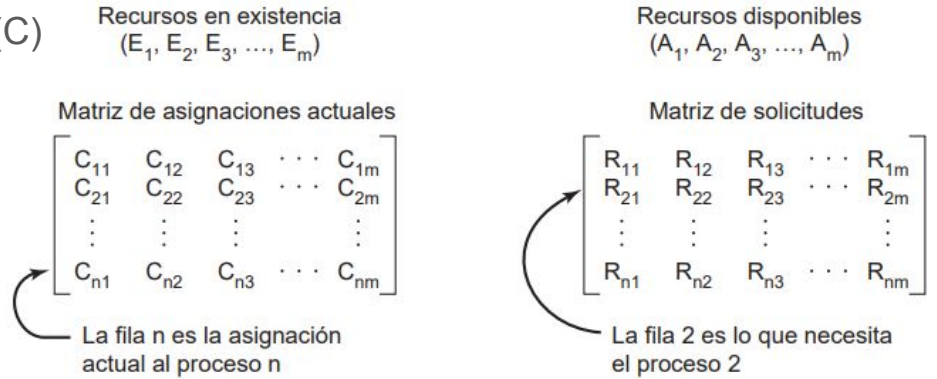


Figura 6-6. Las cuatro estructuras de datos que necesita el algoritmo de detección de interbloqueos.

Detección del interbloqueo con varios recursos de cada tipo

El algoritmo de detección de interbloqueos se muestra a continuación:

1. Buscar un proceso desmarcado, Pi , para el que la i-ésima fila de R sea menor o igual que A.

2. Si se encuentra dicho proceso, agregar la i-ésima fila de C a A, marcar el proceso y regresar al paso 1.

3. Si no existe dicho proceso, el algoritmo termina. Cuando el algoritmo termina, todos los procesos desmarcados (si los hay) están en interbloqueo

Detección del interbloqueo con varios recursos de cada tipo

```
R= 0
Repetir{
  Buscar Pi no includo en R tal que R[i]<=D;
       Si Encontrado{
                    Reducir por Pi:A =A +C[i];
                    Añadir A a R;
                     }

}Mientras(Encontrado)
Si(R==P) No hay interbloqueo
Si no: Procesos en P-R están en interbloqueo
```

Unidades de cinta, Trazadores, Escáneres, Unidades de CD-ROM

$E = (4 \quad 2 \quad 3 \quad 1)$

Unidades de cinta, Trazadores, Escáneres, Unidades de CD-ROM

$A = (2 \quad 1 \quad 0 \quad 0)$

Matriz de asignaciones actuales

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Matriz de peticiones

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Figura 6-7. Un ejemplo para el algoritmo de detección de interbloqueos.

Detección del interbloqueo con varios recursos de cada tipo

Para ejecutar el algoritmo de detección de interbloqueos, buscamos un proceso cuya petición de recursos se pueda satisfacer.

La primera no se puede satisfacer debido a que no hay unidad de CD-ROM disponible.

La segunda tampoco se puede satisfacer, ya que no hay escáner libre. Por fortuna la tercera se puede satisfacer, por lo que el proceso 3 se ejecuta y en un momento dado devuelve todos sus recursos, para obtener A (2 2 2 0). En este punto se puede ejecutar el proceso 2 y devolver sus recursos, con lo cual obtenemos A (4 2 2 1) Ahora se puede ejecutar el proceso restante. No hay interbloqueo en el sistema.

¿Cómo recuperar estos interbloqueos?

Supongamos que nuestro algoritmo de detección de interbloqueos ha tenido éxito y detectó un interbloqueo. ¿Qué debemos hacer ahora? Se necesita alguna forma de recuperarse y hacer funcionar el sistema otra vez.

- Recuperación por medio de apropiación
- Recuperación a través del retroceso
- Recuperación a través de la eliminación de procesos

Recuperación por medio de apropiación

En ocasiones, es posible reasignar temporalmente un recurso de un proceso en ejecución a otro proceso. Sin embargo, este procedimiento a menudo puede requerir intervención manual, especialmente en sistemas operativos de procesamiento por lotes que se ejecutan en mainframes. Por ejemplo, para reasignar una impresora láser, un operador puede recolectar las hojas impresas, suspender el proceso actual (marcándolo como no ejecutable) y asignar la impresora a otro proceso.

Una vez que el segundo proceso ha finalizado, se pueden reanudar el proceso original y reinsertar las hojas impresas. La posibilidad de reasignar recursos sin que los procesos afectados lo noten depende en gran medida de la naturaleza del recurso, y a menudo es difícil o imposible de lograr. La selección del proceso a suspender se basa en la facilidad con la que los recursos pueden ser retirados de ese proceso.

Recuperación a través del retroceso

En situaciones donde se prevé la posibilidad de interbloqueos, los diseñadores de sistemas y operadores pueden implementar puntos de comprobación en los procesos de forma periódica. Estos puntos de comprobación implican la creación de instantáneas del estado del proceso, incluyendo la imagen de la memoria y la asignación de recursos en un archivo para su reinicio posterior. Para optimizar su efectividad, los nuevos puntos de comprobación no reemplazan a los anteriores, sino que se guardan en nuevos archivos, formando una secuencia acumulativa a lo largo del tiempo de ejecución del proceso.

Recuperación a través del retroceso

Cuando se detecta un interbloqueo, se identifican los recursos necesarios para resolver la situación. Para la recuperación, un proceso que posee un recurso crucial vuelve a un estado anterior, anterior a la adquisición de dicho recurso, utilizando uno de sus puntos de comprobación previos. Todo el progreso posterior al punto de comprobación se pierde (por ejemplo, la salida impresa), ya que se restablece a un momento previo donde no tenía el recurso, el cual se asigna a uno de los procesos involucrados en el interbloqueo. Si el proceso reiniciado intenta adquirir el recurso nuevamente, debe esperar hasta que esté disponible una vez más.

Recuperación a través de la eliminación de procesos

En situaciones de interbloqueo, una estrategia cruda para romper el ciclo es eliminar uno o más procesos. Esta solución puede lograrse eliminando uno de los procesos involucrados en el ciclo, lo que con suerte permitiría que los demás continúen su ejecución. Esta acción se puede repetir hasta que se rompa el ciclo.

Recuperación a través de la eliminación de procesos

Otra alternativa es elegir como "víctima" a un proceso que no está en el ciclo para liberar sus recursos. Es importante seleccionar cuidadosamente el proceso a eliminar, ya que podría estar conteniendo recursos necesarios por un proceso involucrado en el ciclo de interbloqueo. Por ejemplo, al eliminar un proceso que posee recursos que no están siendo utilizados en el ciclo, se pueden liberar esos recursos y romper el interbloqueo.

En resumen: Siempre es preferible eliminar un proceso que pueda reiniciarse sin consecuencias dañinas. Por ejemplo, una compilación de código puede reiniciarse sin alterar el resultado final, ya que lee un archivo de código fuente y genera un archivo de código objeto. En contraste, un proceso que actualiza una base de datos puede presentar problemas si se reinicia a mitad de la operación, ya que podría generar inconsistencias en los datos al ejecutarse parcialmente y luego de nuevo.

¿Cómo evitar interbloqueos?

¿Hay algún algoritmo que siempre pueda evitar un interbloqueo al realizar la elección correcta todo el tiempo?

La respuesta es un sí calificado: podemos evitar los interbloqueos, pero sólo si hay cierta información disponible de antemano.

¿Cómo evitar interbloqueos?

Estados seguros e inseguros:

Los principales algoritmos para evitar interbloqueos se basan en el concepto de los estados seguros.

-Estados seguros: Se dice que un estado es seguro si hay cierto orden de programación en el que se puede ejecutar cada proceso hasta completarse, incluso aunque todos ellos solicitaran de manera repentina su número máximo de recursos de inmediato.

-Estado inseguros: La diferencia con el estado seguro es que, desde un estado seguro, el sistema puede garantizar que todos los procesos terminarán; desde un estado inseguro, no se puede dar esa garantía.

¿Cómo evitar interbloqueos?

El algoritmo del banquero para un solo recurso y varios recursos:

Dijkstra (1965) ideó un algoritmo de programación que puede evitar interbloqueos; este algoritmo se conoce como el algoritmo del banquero.

El algoritmo del banquero maneja un conjunto de recursos disponibles y el número de recursos que un proceso puede solicitar. Se basa en el supuesto de que cada proceso debe declarar de antemano su máxima demanda de cada tipo de recurso y no puede exceder este límite.
El algoritmo del banquero considera cada petición a medida que va ocurriendo, y analiza si al otorgarla se produce un estado seguro. Si es así, se otorga la petición; en caso contrario, se pospone hasta más tarde. Para ver si un estado es seguro, el banquero comprueba si tiene los suficientes recursos para satisfacer a algún cliente. De ser así, se asume que esos préstamos volverán a pagarse y ahora se comprueba el cliente más cercano al límite, etcétera. Si todos los préstamos se pueden volver a pagar en un momento dado, el estado es seguro y la petición inicial se puede otorgar.

Codificación

```
struct estado
{
    int recursos[m];
    int disponibles[m];
    int necesidad[n][m];
    int asignacion[n][m];
}
```

(a) estructuras de datos globales

```
if (asignacion [i,*] + peticion [*] > necesidad [i,*])
    < error >;                                    /* petición total > necesidad */
else if (peticion [*] > disponibles [*])
    < suspender al proceso >;
else                                              /* simular asignación */
{
    < definir nuevo_estado como:
    asignacion [i,*] = asignacion [i,*] + peticion [*];
    disponibles [*] = disponibles [*] - peticion [*] >;
}
if (seguro(nuevo_estado))
    < llevar a cabo la asignación >;
else
{
    < restaurar el estado original >;
    < suspender al proceso >;
}
```

(b) algoritmo de asignación de recursos

```
boolean seguro (estado E)
{
    int disponibles_actual[m];
    proceso resto[<número de procesos>];
disponibles_actual = disponibles;
resto = {todos los procesos};
    posible = verdadero;
    while (posible)
    {
        <encontrar un proceso Pk en resto tal que
            necesidad [k,*] - asignacion [k,*] <= disponibles_actual;>
        if (encontrado)                                  /* simular ejecución de Pk */
        {
            disponibles_actual = disponibles_actual + asignacion [k,*];
            resto = resto – {Pk};
        }
        else
            posible = falso;
    }
    return (resto == null);
}
```

(c) algoritmo para comprobar si el estado es seguro (algoritmo del banquero)

¿Cómo evitar interbloqueos?

Resumen algoritmo Banquero:

Por desgracia, aunque en teoría el algoritmo es maravilloso, en la práctica es en esencia poco práctico, debido a que los procesos raras veces saben de antemano cuáles serán sus máximas necesidades de recursos. Además, el número de procesos no está fijo, sino que varía en forma dinámica a medida que los nuevos usuarios inician y cierran sesión. Por otro lado, los recursos que se consideraban disponibles pueden de pronto desvanecerse (las unidades de cinta se pueden descomponer). En la realidad, son escasos los sistemas que emplean el algoritmo del banquero para prevenir interbloqueos.

Otro tipo de interbloqueo

Bloqueo de dos fases:

Los métodos generales para evitar interbloqueos no son muy prometedores, existen algoritmos especializados que son efectivos para aplicaciones específicas. Uno de estos métodos, conocido como bloqueo de dos fases, es utilizado en sistemas de bases de datos. En la primera fase, el proceso intenta adquirir los bloqueos de los registros necesarios. Si encuentra un registro bloqueado, libera todos los bloqueos adquiridos y comienza de nuevo la primera fase. Esta estrategia se asemeja a la solicitud previa de recursos antes de realizar acciones irreversibles.

Sin embargo, el bloqueo de dos fases no es universalmente aplicable. En sistemas de tiempo real o control de procesos, donde la interrupción del proceso a mitad de su ejecución no es aceptable, este enfoque no es viable. Este algoritmo solo funciona en situaciones donde el programa puede detenerse en cualquier punto durante la primera fase y reanudarse sin inconvenientes, lo cual no es factible para muchas aplicaciones que no pueden estructurarse de esta manera.

Otro tipo de interbloqueo

Otro tipo de interbloqueo puede ocurrir en los sistemas de comunicaciones (como las redes), en donde dos o más procesos se comunican mediante el envío de mensajes. Un arreglo común es que el proceso A envía un mensaje de petición al proceso B, y después se bloquea hasta que B envía de vuelta un mensaje de respuesta. Suponga que el mensaje de petición se pierde. A se bloquea en espera de la respuesta. B se bloquea en espera de una petición para que haga algo. A esta situación se le conoce como **interbloqueo de comunicación.**

Para solucionar esta problemática, cada vez que se envía un mensaje del que se espera una respuesta, también se inicia un temporizador. Si el temporizador termina su conteo antes de que llegue la respuesta, el emisor del mensaje asume que éste se ha perdido y lo envía de nuevo (una y otra vez, si es necesario). De esta forma se evita el interbloqueo.

Otro tipo de interbloqueo

En el caso de que el mensaje original no se perdió pero la respuesta simplemente se retrasó, el receptor destinado puede recibir el mensaje dos o más veces, tal vez con consecuencias indeseables. Piense en un sistema bancario electrónico en el que el mensaje contiene instrucciones para realizar un pago. Es evidente que no se debe repetir (y ejecutar) varias veces, sólo porque la red es lenta o el tiempo de espera es demasiado corto. El diseño de las reglas de comunicación para que todo funcione bien, que se conocen como **protocolo.**

Otro tipo de interbloqueo

Los interbloqueos no se limitan exclusivamente a sistemas de comunicación o de recursos de Sistemas operativos, ya que también pueden ocurrir interbloqueos de recursos en redes. Tomemos como ejemplo una red simplificada, representada en la Figura de abajo, que simula la estructura básica de Internet. En esta red, existen dos tipos principales de computadoras: hosts (servidores) y enrutadores.

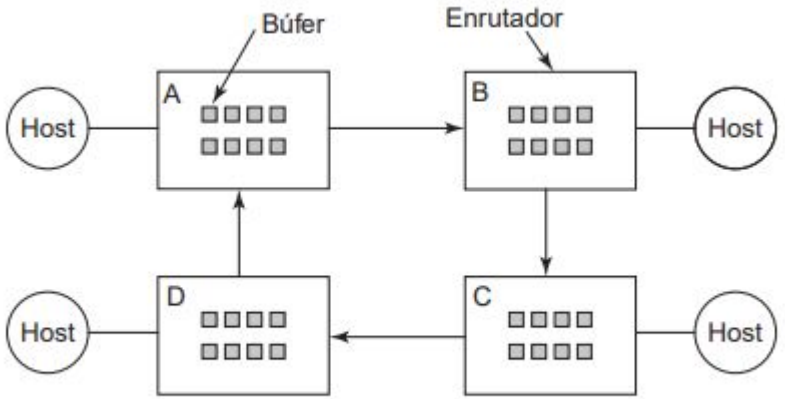


Figura 6-15. Un interbloqueo de recursos en una red.

Otro tipo de interbloqueo

En la práctica los enrutadores poseen millones de búferes, asi mismo, la naturaleza del interbloqueo potencial persiste. Si se presenta un escenario donde todos los paquetes en un enrutador deben dirigirse al siguiente en un ciclo cerrado (por ejemplo, A a B, B a C, C a D y D de vuelta a A), se origina un interbloqueo clásico de recursos. En esta situación, ningún paquete puede moverse más allá de su enrutador actual debido a la falta de búfer en el enrutador de destino, generando un interbloqueo en medio de un sistema de comunicaciones.

Otro tipo de interbloqueo

La solución a esta problemática es:

- Utilizar algoritmos de enrutamiento que eviten la formación de ciclos o permitan redirigir el tráfico de manera más eficiente podría ser una solución para evitar los ciclos que causan interbloqueos.

- Priorización de tráfico: Establecer prioridades en el procesamiento de paquetes, permitiendo que los paquetes críticos o urgentes tengan preferencia para evitar bloqueos que puedan impactar servicios esenciales.

- Monitoreo y gestión activa de red: Implementar sistemas de monitoreo en tiempo real para detectar posibles interbloqueos y tomar medidas correctivas rápidas, como redirigir el tráfico o balancear la carga.