DEPARTAMENTO DE

INGENIERÍA INFORMÁTICA

UNIVERSIDAD DE SANTIAGO DE CHILE

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

ELECCIÓN DISTRIBUDA Y EXCLUSIÓN MUTUA

13169

SISTEMAS DISTRIBUIDOS

Motivación

- Para los sistemas distribuidos resulta fundamental la concurrencia y la colaboración entre diversos procesos.
- Esto implica tener acceso simultaneo a los mismos recursos.
- Para evitar la corrupción y asegurar la consistencia, se necesita encontrar soluciones que garanticen que los procesos tengan acceso mutuamente exclusivo.

Algoritmos de Exclusión Mutua

- Los algoritmos de exclusión mutua se pueden dividir en dos grandes categorías.

- **Soluciones basadas en token**

  - La exclusión mutua se logra pasando un **mensaje especial** conocido como **token**.
  - Sólo hay un token y quien lo tenga puede acceder al recurso compartido.
  - Si un proceso tiene el token pero no quiere usar el recurso, simplemente lo pasa.

- **Soluciones basadas en permisos**

  - El proceso que quiere el acceso al recurso solicita permiso al resto de procesos.
  - Diversas formas de lograr esto

Algoritmos de Exclusión Mutua

Algoritmo Centralizado

- Manera directa. Simula las acciones de un sistema monoprocesador.
- Se selecciona un proceso como coordinador. El resto de procesos envía mensajes al coordinador para poder acceder a un recurso.
- Si el recurso solicitado está libre, entonces se permite el acceso. En caso que un proceso solicite un recurso ocupado, el coordinador encola la petición.

Algoritmos de Exclusión Mutua

Algoritmo Centralizado – Ventajas

- Garantiza **exclusión mutua**.
- Es **justo**, las peticiones se responden en el orden que llegan.
- Es libre de **inanición** (ningún proceso queda esperando por siempre).
- Es simple de implementar y requiere sólo tres tipos de mensajes (petición, autorización y liberación).

Algoritmos de Exclusión Mutua

Algoritmo Centralizado – Desventajas

- Coordinador es el único **punto de fallo**.
- Los procesos solicitantes **no pueden distinguir** un coordinador inactivo respecto a “permiso denegado” (en ambos casos, ningún mensaje regresa).
- Entre más grande el sistema, un único coordinador puede ser un **cuello de botella** en cuanto al rendimiento.

Algoritmos de Exclusión Mutua

Algoritmo Centralizado

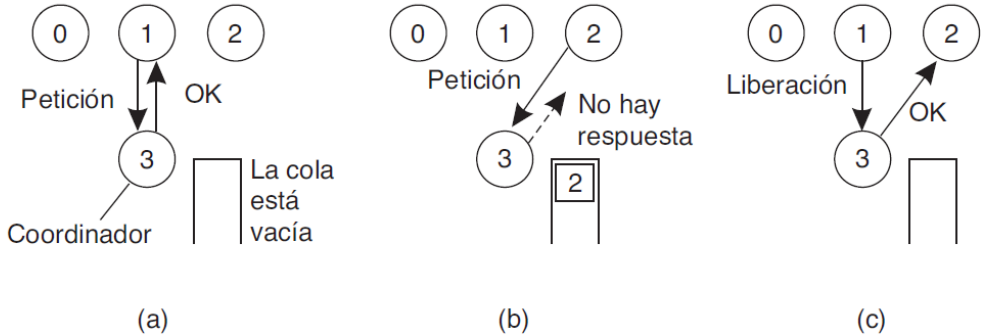


Figura 6-14. El proceso 1 solicita permiso al coordinador para acceder a un recurso compartido. El permiso es otorgado. (b) Luego el proceso 2 solicita permiso para acceder al mismo recurso. El coordinador no responde. (c) Cuando el proceso 1 libera el recurso, éste se lo indica al coordinador, que después le responde a 2.

Algoritmos de Exclusión Mutua

Algoritmo Descentralizado

- Algoritmo de votación ejecutado sobre un sistema basado en DHT.
- Es una extensión del algoritmo centralizado de un solo coordinador.
- Cada recurso se replica *n* veces, donde cada réplica tiene su propio coordinador para controlar el acceso de procesos concurrentes.
- Cuando un proceso quiere acceder a un recurso, debe lograr una votación mayoritaria (más del 50% de los coordinadores debe permitir el acceso al recurso).
- Cuando un coordinador niega el acceso a un recurso, informa al proceso solicitante.

Algoritmos de Exclusión Mutua

Algoritmo Descentralizado – Consideraciones

- Elimina el cuello de botella y el punto único de fallo.
- En caso de fallar un coordinador, se debe reiniciar y olvida las votaciones realizadas.
- En consecuencia, puede otorgar de manera incorrecta acceso a otro proceso.

Algoritmos de Exclusión Mutua

Algoritmo Distribuido

- Propuesto por Ricart y Agrawala (1981).
- Requiere un ordenamiento total de los eventos del sistema, lo cual se puede lograr con relojes lógicos de Lamport (por ejemplo).
- Cuando un proceso desea acceder a un recurso compartido, elabora un mensaje que contiene el nombre del recurso, su número de proceso (identificador) y el tiempo actual (lógico). Tras generar el mensaje, envía dicho mensaje a todos los demás procesos, incluyéndose de manera conceptual.
- Se asume que el envío de mensajes es confiable y no se pierden mensajes en el proceso.

Algoritmos de Exclusión Mutua

Algoritmo Distribuido

- Cuando un proceso recibe un mensaje de petición de otro proceso, toma una acción dependiendo de su propio estado respecto al recurso mencionado en el mensaje.

- Se distinguen tres casos:
  1. Si el destinatario no accede al recurso y no desea acceder a él, envía un mensaje de OK al remitente.
  2. Si el destinatario ya cuenta con acceso al recurso, simplemente no responde. En su lugar, coloca la petición en una cola.
  3. Si el receptor también quiere acceder al recurso, compara el tiempo del mensaje entrante respecto al suyo. El mejor gana. Si el mensaje entrante tiene un tiempo menor, el receptor entrega un mensaje de OK. Si su propio registro es menor, el destinatario encola el mensaje entrante y no envía nada.

Algoritmos de Exclusión Mutua

Algoritmo Distribuido

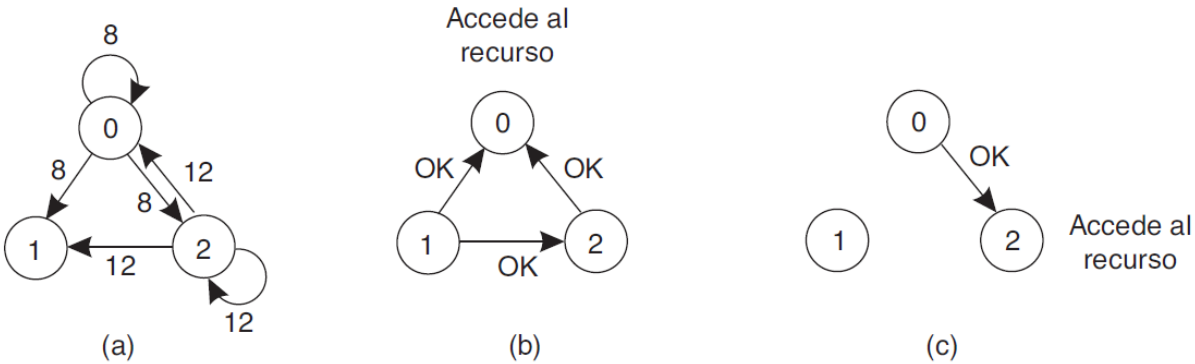


Figura 6-15. (a) Dos procesos desean acceder a un recurso compartido en el mismo momento. (b) El proceso 0 contiene el menor registro de tiempo, de modo que gana. (c) Cuando el proceso 0 se lleva a cabo, envía también un OK, de modo que el 2 puede seguir adelante.

Algoritmos de Exclusión Mutua

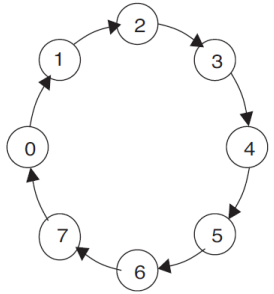
Algoritmo Distribuido

- No existe un único punto de fallo. En su lugar existen *n* potenciales puntos de fallo.
- Si algún proceso falla, no enviará un mensaje, lo cual se interpretará de mala forma.
- Se puede modificar para enviar siempre un mensaje de respuesta.
- No genera cuello de botella, ya que todos los procesos están involucrados en todas las decisiones.
- Fuerte uso del tráfico de la red.

Algoritmos de Exclusión Mutua

Algoritmo de Anillo de Token

- En un anillo, se da una distribución lógica de los componentes. No tiene un orden en particular, lo único importante es saber el proceso que va a continuación de él.
- Cuando se inicia el anillo, al proceso 0 se le asigna un ***token***.
- El *token* se mueve desde el proceso *k* al proceso *k+1*. Cuando un proceso adquiere el *token*, verifica si necesita acceder al recurso compartido, realiza sus labores y entrega el *token* al siguiente proceso.
- Si un proceso recibe el *token* y no necesita usar los recursos compartidos, sólo lo pasa a través del anillo. En consecuencia, cuando ningún proceso necesita el recurso, el *token* circula a alta velocidad a través del anillo.



Algoritmos de Exclusión Mutua

Algoritmo de Anillo de Token – Problemas

- Si el *token* se pierde, en algún momento debe reponerse. Es complejo detectar la pérdida, dado que no se puede adelantar cuánto tiempo utilizará el *token* un proceso particular.

- Si un proceso falla también es un problema, aunque más fácil de recuperar ya que simplemente se puede ignorar el proceso conflictivo al pasar el *token*.

Algoritmos de Exclusión Mutua

Comparación de Algoritmos

- El algoritmo centralizado es el más simple y más eficiente. Sólo requiere mensajes de petición, permiso para entrar y liberación para salir. Espera 2 mensajes para acceder al recurso.

Algoritmo	Mensajes por entrada/salida	Retraso antes de la entrada (durante el tiempo del mensaje)	Problemas
Centralizado	3	2	Falla el coordinador
Descentralizado	3mk, k = 1, 2, …	2 m	Innanición, baja eficiencia
Distribuido	2(n − 1)	2(n − 1)	Falla de cualquier proceso
Anillo de token	1 a ∞	0 a n − 1	Pérdida del token, falla del proceso

Figura 6-17. Comparación de tres algoritmos de exclusión mutua.

Algoritmos de Exclusión Mutua

Comparación de Algoritmos

- Para el algoritmo descentralizado, cada uno de los *m* coordinadores debe mandar los mensajes, pero es posible que se necesiten *k* intentos para lograrlo. Espera 2 mensajes por cada coordinador para acceder al recurso.

Algoritmo	Mensajes por entrada/salida	Retraso antes de la entrada (durante el tiempo del mensaje)	Problemas
Centralizado	3	2	Falla el coordinador
Descentralizado	3mk, k = 1, 2, …	2 m	Innanición, baja eficiencia
Distribuido	2(n − 1)	2(n − 1)	Falla de cualquier proceso
Anillo de token	1 a ∞	0 a n − 1	Pérdida del token, falla del proceso

Figura 6-17. Comparación de tres algoritmos de exclusión mutua.

Algoritmos de Exclusión Mutua

Comparación de Algoritmos

- El algoritmo distribuido requiere *n - 1* mensajes de petición y *n - 1* mensajes adicionales de autorización.

Algoritmo	Mensajes por entrada/salida	Retraso antes de la entrada (durante el tiempo del mensaje)	Problemas
Centralizado	3	2	Falla el coordinador
Descentralizado	3mk, k = 1, 2, …	2 m	Innanición, baja eficiencia
Distribuido	2(n − 1)	2(n − 1)	Falla de cualquier proceso
Anillo de token	1 a ∞	0 a n − 1	Pérdida del token, falla del proceso

Figura 6-17. Comparación de tres algoritmos de exclusión mutua.

Algoritmos de Exclusión Mutua

Comparación de Algoritmos

- Para el *token* en anillo, el número de mensajes es variable. En ocasiones el *token* puede circular por horas sin que ningún proceso se interese por él.

Algoritmo	Mensajes por entrada/salida	Retraso antes de la entrada (durante el tiempo del mensaje)	Problemas
Centralizado	3	2	Falla el coordinador
Descentralizado	3mk, k = 1, 2, …	2 m	Innanición, baja eficiencia
Distribuido	2(n − 1)	2(n − 1)	Falla de cualquier proceso
Anillo de token	1 a ∞	0 a n − 1	Pérdida del token, falla del proceso

Figura 6-17. Comparación de tres algoritmos de exclusión mutua.

Algoritmos de Elección

- Muchos de los algoritmos de elección requieren un proceso que actúe como coordinador.
- En general, no importa el proceso que se seleccione, lo importante es la existencia del coordinador.
- Si todos los procesos son exactamente iguales, entonces no hay forma de seleccionar a uno en particular. Para ayudar a los algoritmos, supondremos que cada proceso tiene un identificador (como su dirección de red) y los algoritmos intentarán seleccionar el proceso con mayor *id*.
- Los procesos conocen el *id* del resto y se busca que todos los procesos sepan cuál es el coordinador al terminar el proceso de elección.

Algoritmos de Elección

Algoritmos del Abusón (Bully)

- Cuando cualquier proceso advierte que el coordinador ya no está respondiendo peticiones, inicia una elección.

- Un proceso *P* inicia una elección de la siguiente forma:

  1. *P* envía un mensaje de ***ELECCIÓN*** a todos los procesos con *id* superior.
  2. Si ningún proceso responde, *P* ha ganado la elección y se convierte en el coordinador.
  3. Si uno de los procesos con *id* superior responde, toma el mando.

Algoritmos de Elección

Algoritmos del Abusón (Bully)

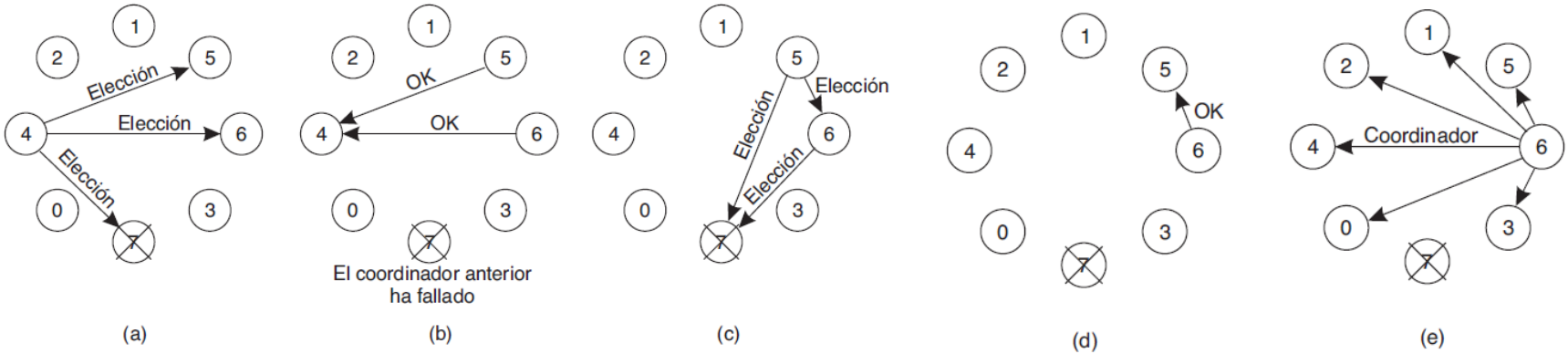


Figura 6-20. Algoritmo del abusón. (a) El proceso 4 celebra la elección. (b) Los procesos 5 y 6 responden, indicándole a 4 que se detenga. (c) Ahora 5 y 6 celebran una elección. (d) El proceso 6 le indica al 5 que se detenga. (e) El proceso 6 gana, y lo comunica a todos.

Algoritmos de Elección

Algoritmo de Anillo

- A diferencia de otros algoritmos de anillo, no utiliza un *token*.
- Suponemos que los procesos están físicamente o lógicamente ordenados, de tal forma que cada proceso sabe cuál es su sucesor.
- Cuando un proceso nota que el coordinador falló, elabora un mensaje de ***ELECCIÓN*** con su propio *id* y envía el mensaje a su sucesor.
- En cada paso del mensaje, el remitente agrega el *id* de su proceso hasta llegar al proceso que inició la elección.
- Una vez llega el mensaje al proceso que inició la elección, envía un mensaje por el anillo con el ***COORDINADOR*** informando cuál es el nuevo coordinador y los miembros del nuevo anillo.

Algoritmos de Elección

Algoritmo de Anillo

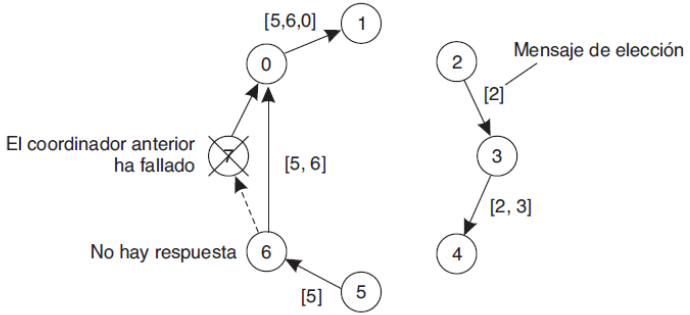


Figura 6-21. Algoritmo de elección que utiliza un anillo.

En la figura 6-21 vemos lo que ocurre si dos procesos, 2 y 5, descubren simultáneamente que el coordinador anterior, el proceso 7, ha fallado. Cada proceso elabora un mensaje de *ELECCIÓN* y comienza a circular su mensaje, de modo independiente uno de otro. En algún momento, ambos mensajes recorrerán todo el camino, y tanto el proceso 2 como el 5 los convertirán en mensajes de *COORDINADOR* con exactamente los mismos miembros y en el mismo orden. Cuando ambos hayan hecho el recorrido nuevamente serán eliminados. Tener más mensajes en circulación no ocasiona daños; a lo sumo, consume un poco de ancho de banda, pero no se considera un desperdicio.

Algoritmos de Elección

Elección en Ambiente Inalámbrico (WAN)

- Los algoritmos vistos se basan en que el paso de mensajes es confiable y que la topología de la red no cambia.
- Para elegir un líder, cualquier nodo de la red (denominado fuente) puede iniciar una elección enviando un mensaje de ***ELECCIÓN*** a sus vecinos inmediatos.
- Cuando un nodo recibe el mensaje de elección, designa al remitente como su padre y posteriormente envía un mensaje de elección a sus vecinos inmediatos a excepción del padre.
- Cuando un nodo recibe un mensaje de otro nodo que no sea su padre, simplemente acusa recibo.

Algoritmos de Elección

Elección en Ambiente Inalámbrico (WAN)

- Cuando un nodo *R* ha designado el nodo *Q* como su padre, este reenvía el mensaje a sus vecinos inmediatos (a excepción de *Q*) y espera acuse de recibo antes de enviar acuse de recibo del mensaje *ELECCIÓN* de *Q*.

- La espera tiene una importante consecuencia. Si los vecinos ya tienen un padre, responden de inmediato a *R*. Básicamente, *R* sería un nodo hoja. Los nodos vecinos reportarán la información sobre la capacidad de los recursos en la red.

- Esta información permitirá a *Q* comparar las capacidades de *R* con la de otros nodos y seleccionar el mejor para ser líder.

Algoritmos de Elección

Elección en Ambiente Inalámbrico (WAN)

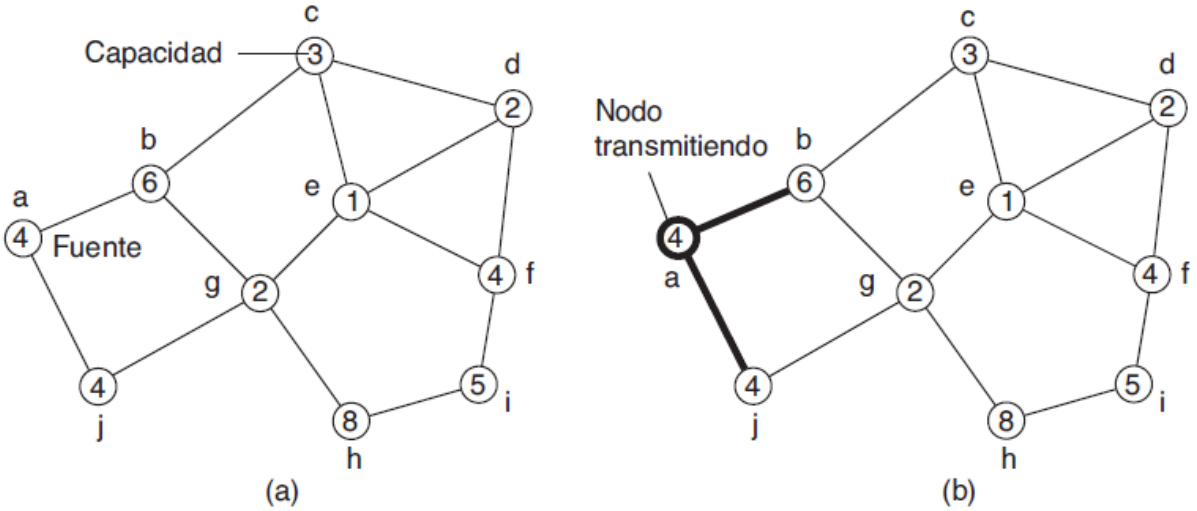


Figura 6-22. Algoritmo de elección en una red inalámbrica, con el nodo *a* como fuente. (a) Red inicial. (b) a (e) Fase de construcción de árbol (el último paso de transmisión de los nodos f e i no se muestra). (f) Se reporta a la fuente sobre el mejor nodo.

Algoritmos de Elección

Elección en Ambiente Inalámbrico (WAN)

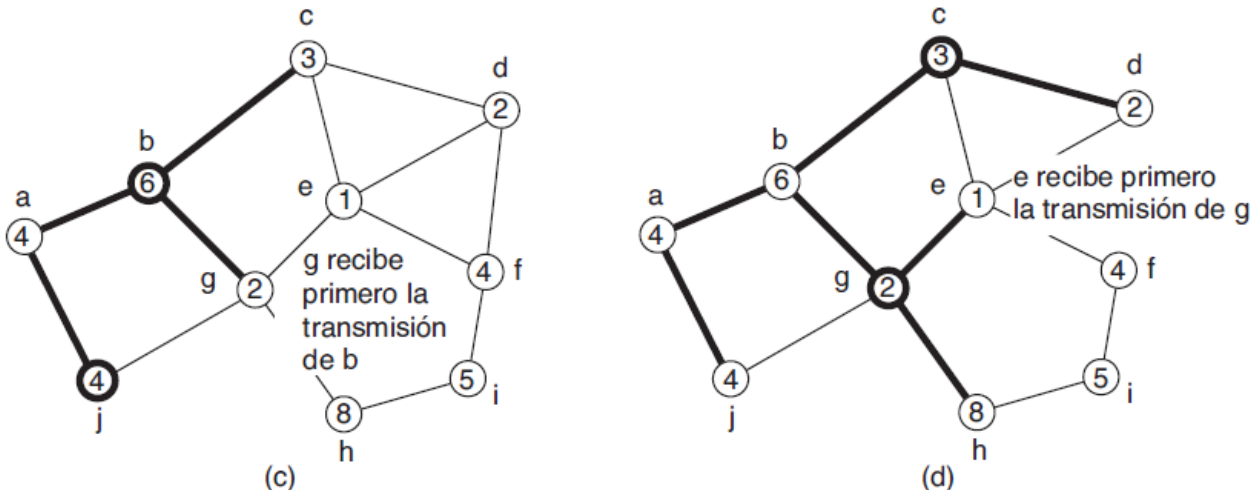


Figura 6-22. Algoritmo de elección en una red inalámbrica, con el nodo *a* como fuente. (a) Red inicial. (b) a (e) Fase de construcción de árbol (el último paso de transmisión de los nodos f e i no se muestra). (f) Se reporta a la fuente sobre el mejor nodo.

Algoritmos de Elección

Elección en Ambiente Inalámbrico (WAN)

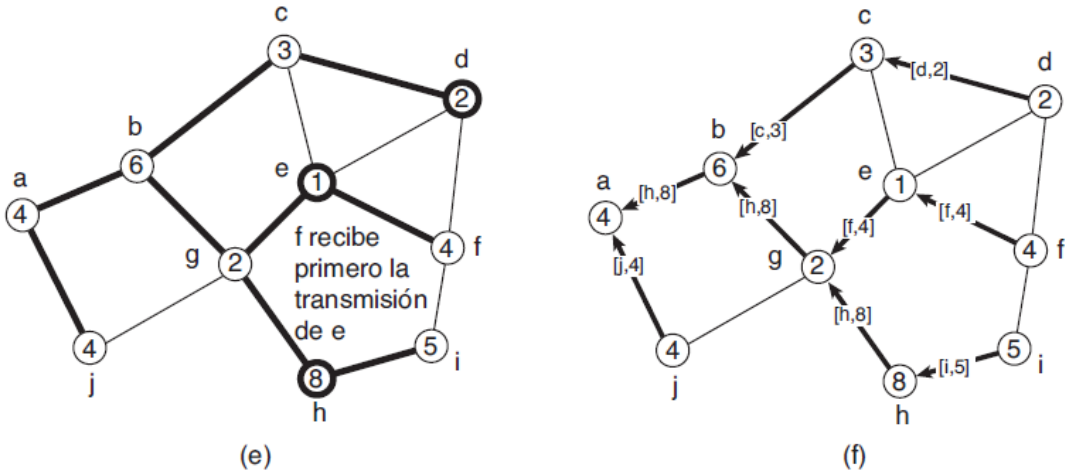


Figura 6-22. Algoritmo de elección en una red inalámbrica, con el nodo *a* como fuente. (a) Red inicial. (b) a (e) Fase de construcción de árbol (el último paso de transmisión de los nodos f e i no se muestra). (f) Se reporta a la fuente sobre el mejor nodo.

Algoritmos de Elección

Elección en Sistemas de Gran Escala

- Los algoritmos presentados funcionan, en general, para sistemas relativamente pequeños. De hecho, los algoritmos se centran en la selección de un solo nodo.

- Hay situaciones en que se debe seleccionar varios nodos, como el caso de los **super peers (super puntos)**.

- Existen requerimientos para seleccionar un super peer.
  - Los nodos normales deben tener acceso de baja latencia a los super peers.
  - Los super peers deben distribuirse uniformemente a través de la red superpuesta.
  - Debe existir una cantidad predefinida de super peers, en virtud del tamaño de la red.
  - Cada super peer debe servir a un número máximo de nodos normales.

Algoritmos de Elección

Elección en Sistemas de Gran Escala

- Supongamos que necesitamos colocar uniformemente *N* super puntos a través de la red sobrepuesta.
- La idea es simple, un total de *N* tokens se propagan por los *N* nodos seleccionados al azar.
- Ningún nodo puede mantener más de un token.
- Cada token representa una “fuerza de rechazo” por la que otro token prefiere retirarse. Si todos los tokens tienen la misma fuera de rechazo, se alejarán unos de otros y se propagarán uniformemente en el espacio geométrico.

Algoritmos de Elección

Elección en Sistemas de Gran Escala

- Los tokens se irán moviendo dependiendo de un umbral. Cuando un token es mantenido por un nodo durante un cierto tiempo, entonces se transforma en un super peer.

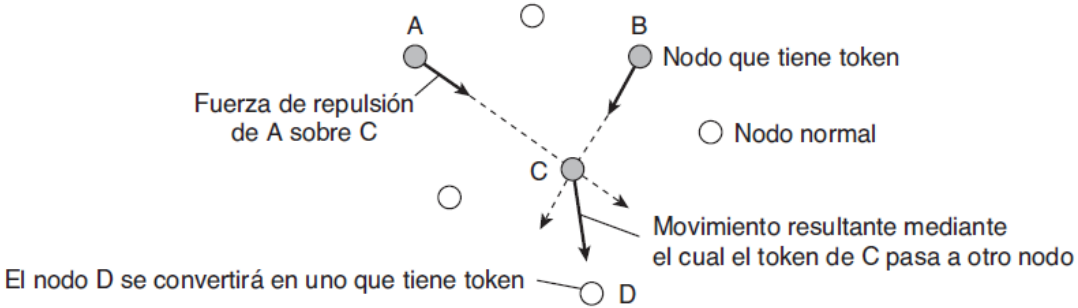


Figura 6-23. Tokens en movimiento en un espacio bidimensional que utilizan fuerzas de repulsión.

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Departamento de Ingeniería Informática
Universidad de Santiago de Chile

¿CONSULTAS?