CC5303 – Sistemas Distribuidos

6.- Tolerancia a Fallos

Parte 1

Sebastián Blasco V.

Tolerancia a fallas

- Una característica sobresaliente de los sistemas distribuidos que los distingue de los sistemas de una sola máquina es la noción de **falla parcial**.
  - En un sistema distribuido, una falla parcial puede acontecer cuando falla un componente. Esta falla puede afectar la operación de algunos componentes, al tiempo que otros más no se ven afectados en absoluto.
- Por contraste, en un sistema no distribuido, una falla a menudo es total en el sentido de que afecta a todos los componentes y fácilmente puede echar abajo al sistema.

Tolerancia a fallas

Un objetivo importante en el diseño de sistemas distribuidos es construirlos de manera que puedan recuperarse automáticamente de fallas parciales sin que se afecte seriamente el desempeño total.

Tolerancia a las fallas significa que un sistema puede proveer sus servicios incluso en presencia de fallas. En otros términos, el sistema puede tolerarlas y continuar operando con normalidad.

Contenidos

- Conceptos Básicos
- Atenuación del proceso
- Multitransmisión confiable
- Atomicidad (tx distribuidas)
- Cómo recuperarse de una falla

Conceptos Básicos

Tolerancia a fallos, muy relacionado a lo que es un **sistema fiable**

¿Fiabilidad?, comprende

- Disponibilidad
- Confiabilidad
- Seguridad
- Mantenimiento

Conceptos Básicos

¿Fiabilidad?, comprende

- Disponibilidad
  - Propiedad de que un sistema está listo para ser utilizado de inmediato.
  - Se refiere a la probabilidad de que el sistema esté operando correctamente en cualquier momento dado
  - *Un sistema altamente disponible es uno que muy probablemente funcionará en un instante dado.*

Conceptos Básicos

¿Fiabilidad?, comprende

- Confiabilidad
  - Propiedad de que un sistema sea capaz de funcionar de manera continua sin fallar.
  - Se define en función de un intervalo de tiempo en lugar de un instante en el tiempo.
  - *Un sistema altamente confiable es uno que muy probablemente continuará funcionando sin interrupción durante un lapso de tiempo relativamente largo.*

Conceptos Básicos

¿Fiabilidad?, comprende

- Disponibilidad vs. Confiabilidad
  - Si un sistema se viene abajo durante un milisegundo por hora
    - Su disponibilidad es de más del 99.999%, **pero no es confiable**.
  - Un sistema que nunca se congela pero que deja de funcionar dos semanas cada agosto
    - **Es altamente confiable**, aunque sólo esté un 96% disponible.

Estas dos situaciones no son lo mismo.

Conceptos Básicos

¿Fiabilidad?, comprende

- Seguridad
  - Situación en que no acontece nada catastrófico cuando un sistema deja de funcionar correctamente durante un tiempo.
  - Ej. Sistemas de control de procesos en plantas nucleares, armería electrónica médica, etc.
- Mantenimiento
  - Cuán fácil puede ser reparado un sistema que falló.
  - Si las fallas pueden ser detectadas y reparadas en forma automática, el sistema goza también de **alta disponibilidad**.

Conceptos Básicos

Existen distintos tipos de fallas:

- Transitorias
  - Ocurren una vez y luego desaparecen. *Fortuitas*
- Intermitentes
  - Ocurre, luego desaparece por sí sola, después reaparece, y así sucesivamente.
- Permanentes
  - Ocurre, y continúa existiendo hasta que el componente defectuoso es reemplazado.

Conceptos Básicos

Modelos de Fallo

Tipo de falla	Descripción
Falla de congelación	Un servidor se detiene, pero estaba trabajando correctamente hasta que se detuvo
Falla de omisión	Un servidor no responde a las peticiones entrantes
Omisión de recepción	Un servidor no recibe los mensajes entrantes
Omisión de envío	Un servidor no envía mensajes
Falla de tiempo	La respuesta de un servidor queda fuera del intervalo de tiempo especificado
Falla de respuesta	La respuesta de un servidor es incorrecta
Falla de valor	El valor de la respuesta está equivocado
Falla de transición de estado	El servidor se desvía del flujo de control correcto
Falla arbitraria	Un servidor puede producir respuestas arbitrarias en tiempos arbitrarios

Conceptos Básicos

Las más serias son fallas arbitrarias, también conocidas como **fallas bizantinas**. En realidad, cuando ocurren fallas arbitrarias, los clientes deberían estar preparados para lo peor.

"Bizantino"

- Se refiere al Imperio Bizantino, una época (330-1453) y un lugar (los Balcanes y la moderna Turquía) donde las conspiraciones interminables, las intrigas, y la desconfianza eran cosa de todos los días en los círculos gubernamentales.

Redundancia

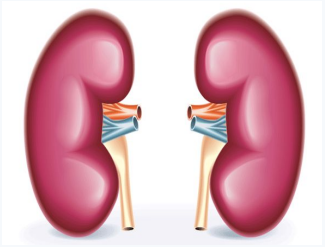
Para que un sistema sea tolerante a fallas, lo mejor que se puede hacer es tratar de ocultar ante otros procesos la ocurrencia de fallas.

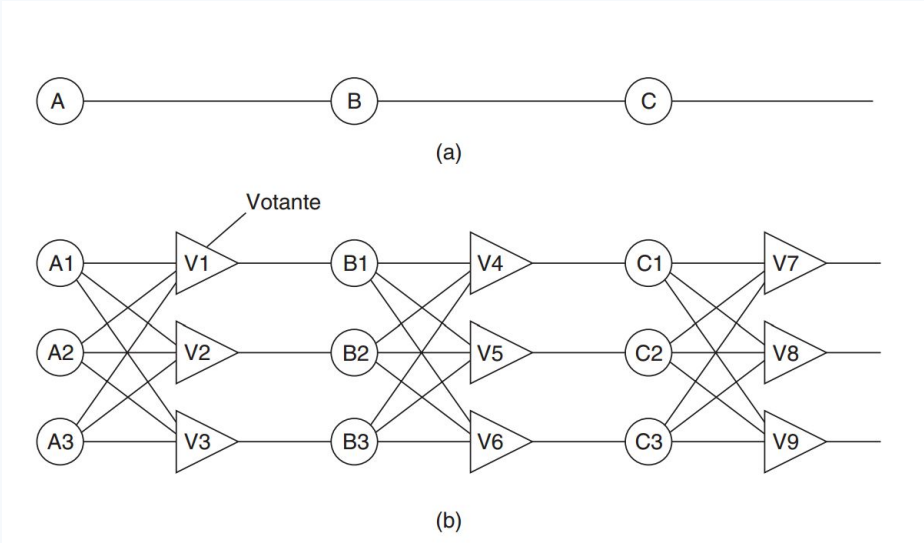
La técnica clave para disfrazar las fallas es utilizar redundancia. Tres clases son posibles:

- Redundancia de información
  - Ej. Se agregan bits adicionales para recuperar los bits mutilados.
- Redundancia de tiempo
  - Ej. Se realiza una acción, y luego, si es necesario, se vuelve a realizar. Txs
- Redundancia física*
  - Ej. Se agrega equipo o procesos adicionales para que el sistema en su conjunto tolere la pérdida o el mal funcionamiento de algunos componentes.
  - (*) en HW o SW

Redundancia

- Redundancia física





Redundancia Modular Triple

Atenuación del proceso

La forma clave de afrontar la tolerancia a un proceso defectuoso es **organizar varios procesos idénticos en un grupo**. Se permite disfrazar uno o más procesos defectuosos presentes en dicho grupo.

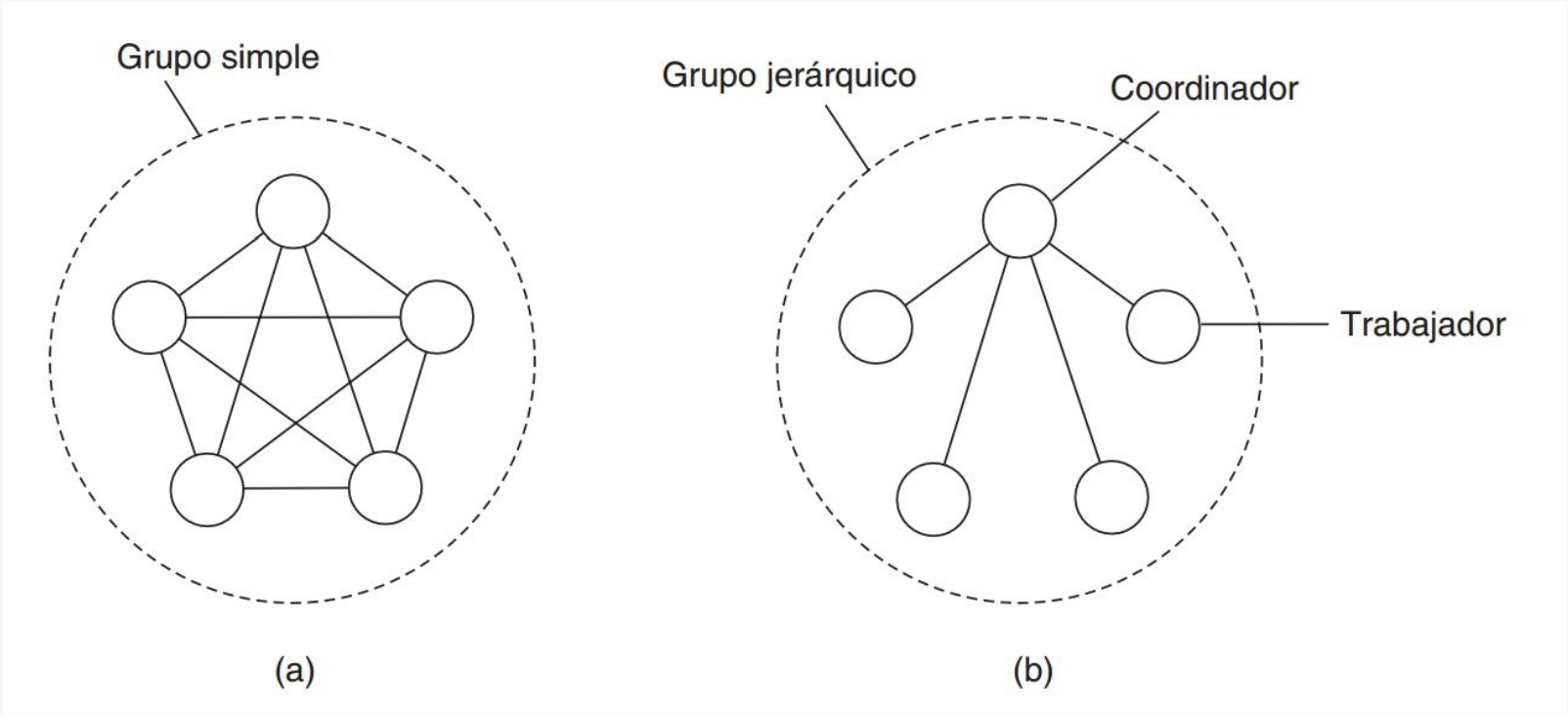
- Grupos de procesos
  - Cuando un mensaje es enviado al grupo, **todos los miembros de éste lo reciben**.
  - Si en un grupo **falla un proceso**, alguno de los demás procesos puede **hacerse cargo** de él.
  - Son dinámicos
    - Se crean y se destruyen. Un proceso puede unirse o abandonar en plena ejecución, además de pertenecer a varios grupos simultáneamente.
    - **Se requieren mecanismos adecuados para gestionar los grupos y la membresía a un grupo.**

Atenuación del proceso

- Grupos de procesos
  - Los grupos pueden ser
    - abiertos: no-miembros pueden enviar al grupo
    - cerrados: solo los miembros pueden enviar al grupo
  - Los miembros del grupo pueden ser iguales (peer) , o bien existe un miembro coordinador o líder
  - De existir, los envíos se hacen al coordinador, que decide a qué miembro reenviar

Atenuación del proceso

- Grupos de procesos



Atenuación del proceso

- Grupos de procesos
  - **Membresía**: Se requiere cierto método para la creación y eliminación de grupos, así como permitir a los procesos que se unen o dejen grupos (Mecanismo distribuido vs. Servidor de grupos)
    - Un miembro crashea => Debe dejar de pertenecer al grupo.
      - Los demás miembros deben descubrir en forma experimental que ya no pertenece.
    - La salida o entrada a un grupo debe sincronizarse con el envío de mensajes
    - Cuando se une al grupo debe recibir todos los mensajes.
      - Sol: cuando se une, enviar un mensaje a todos , Hola! :)
    - Si hay muchos procesos que fallan y el grupo no funciona se necesita protocolo de reconstrucción del grupo. Algún proceso toma la iniciativa, ¿Qué pasa si mas de uno lo hace?

Atenuación del proceso

Se dice que un sistema es **tolerante a *k* fallas** si puede sobrevivir a las fallas de k componentes y continuar satisfaciendo sus especificaciones.

- El problema del consenso:
  - Cada proceso *Pi* empieza con estado ***undecided***
  - Cada proceso *Pi* propone un valor *Vi*
  - Los procesos *Pi* intercambian mensajes hasta fijar una variable de decisión *di*, y entrar en el estado ***decided***. *di* queda fijo.

El consenso es importante para determinar nodos vivos y funcionales!

Consenso

Consenso con *N* procesos

- Cada proceso *Pi* elige un valor *Vi* y lo envía a los demás.
- Cuando cada proceso *Pi* ha coleccionado los *N* valores, calcula el que más se ha repetido, o algún valor *null* si no hay ninguna mayoría.
- Todos terminan en estado **decided** con *di = majority(v1, v2, . . . , vN)*, o bien *di = null*.

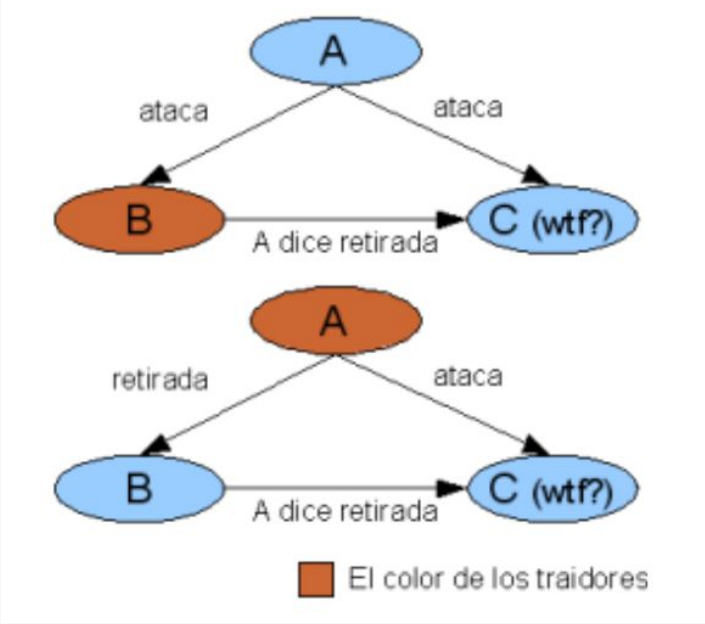
- ¿Qué ocurre ante una caída?
  - El algoritmo podría no terminar.
- ¿Qué pasa si un proceso ”miente”?
  - Se puede llegar a consenso en un valor incorrecto

Consenso

- Si los componentes fallan silenciosamente
  - Es suficiente con tener *k + 1* de ellos para proveer la tolerancia *k* a fallas.
- Frente a fallas bizantinas donde un proceso falla y sigue funcionando y enviando respuestas erróneas o aleatorias
  - Se requiere un mínimo de *2k + 1* réplicas para lograr la tolerancia a *k* fallas.
  - En el peor de los casos, los *k* procesos que fallan, podrían generar la misma respuesta. No obstante, los *k + 1* procesos restantes también darán la misma respuesta, así que el cliente o el votante sólo pueden creerle a la mayoría.

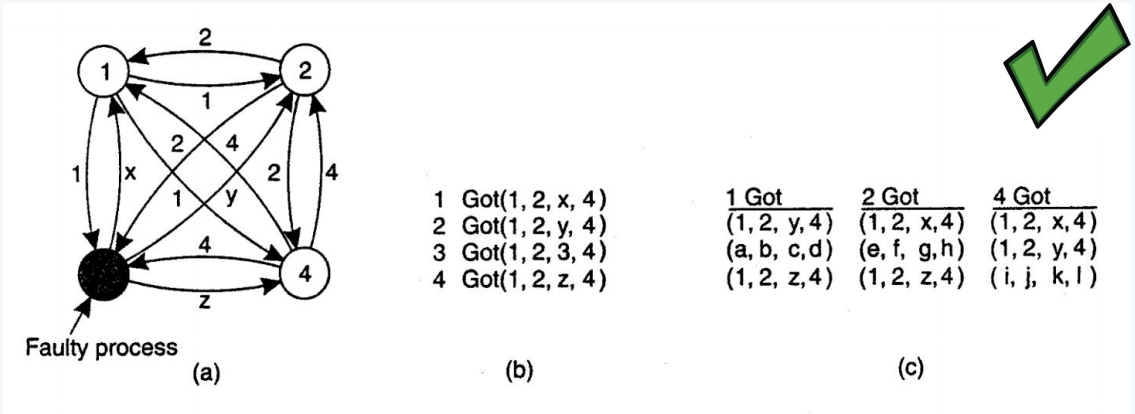
Consenso

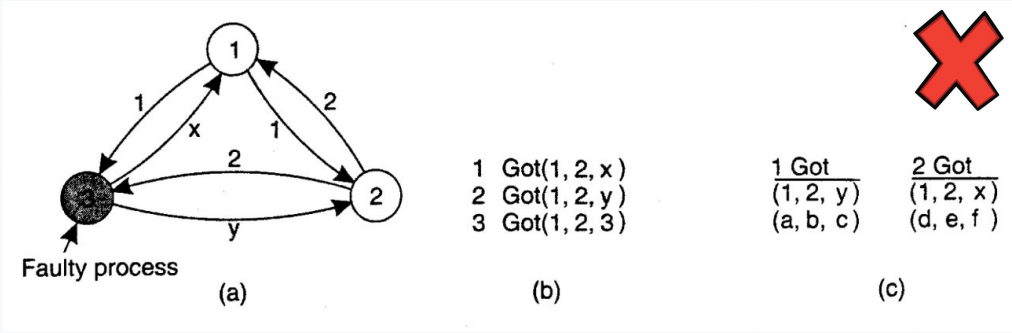
- El problema de los generales Bizantinos
  - Tres o más generales deben decidir entre atacar ó retirarse. Pero uno o más de los generales pueden ser un traidor, y propagar una orden incorrecta.



Consenso

- El problema de los generales Bizantinos



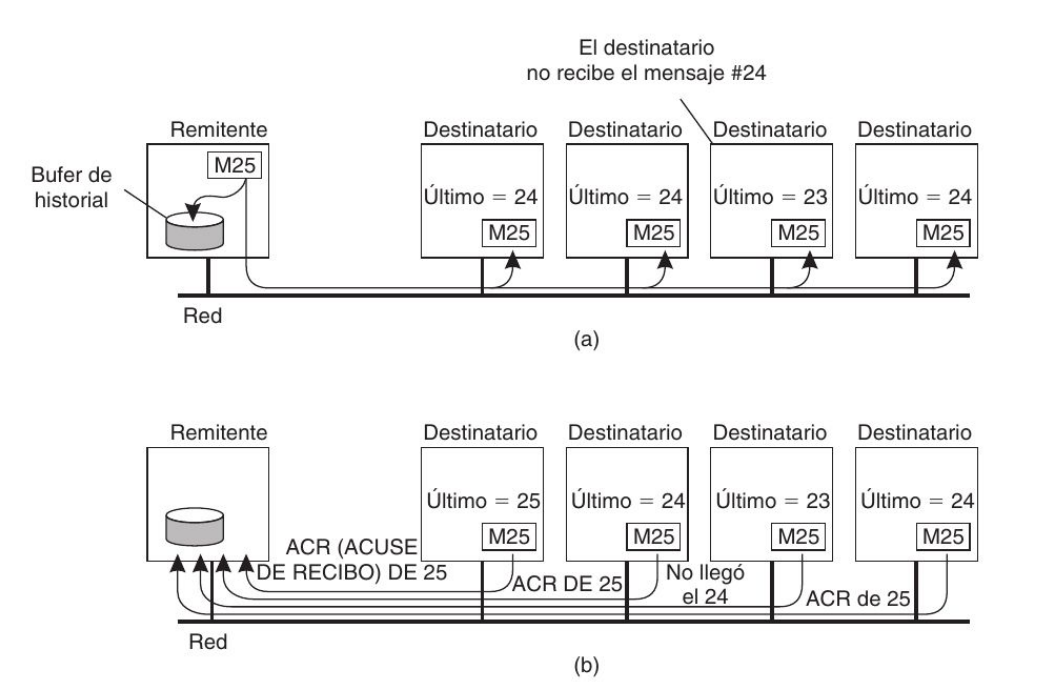


Posible con $N \geq 3f + 1$

Consenso

- El problema de los generales Bizantinos
  - El escenario anterior es válido sólo bajo comunicación **Síncrona**
  - Es imposible garantizar consenso en un sistema asíncrono, aún con una única caída de un proceso.
    - Al ser asíncrono, es imposible distinguir una caída de un proceso lento y no se puede establecer un timeout.

Multitransmisión Confiable



Sistema cuando todos los destinatarios son conocidos y se asume que no fallan.

¿Escalabilidad? Nop…
Implosión de retroalimentación

¿Idea de solución? Acuse recibo solo en error.

-No da garantías de que no habrá implosión por retroalimentación

-¿Nadie piensa en el remitente?

Multitransmisión Confiable Escalable

- Los destinatarios nunca confirman la entrega exitosa de un mensaje multi transmitido, sino que informan sólo cuando no reciben un mensaje.
- Siempre que un destinatario advierta que le falta un mensaje, multi transmite su retroalimentación al resto del grupo.
- Un remitente R que no recibió un mensaje m programa un mensaje de retroalimentación con cierta demora aleatoria.

Multitransmisión Confiable Escalable

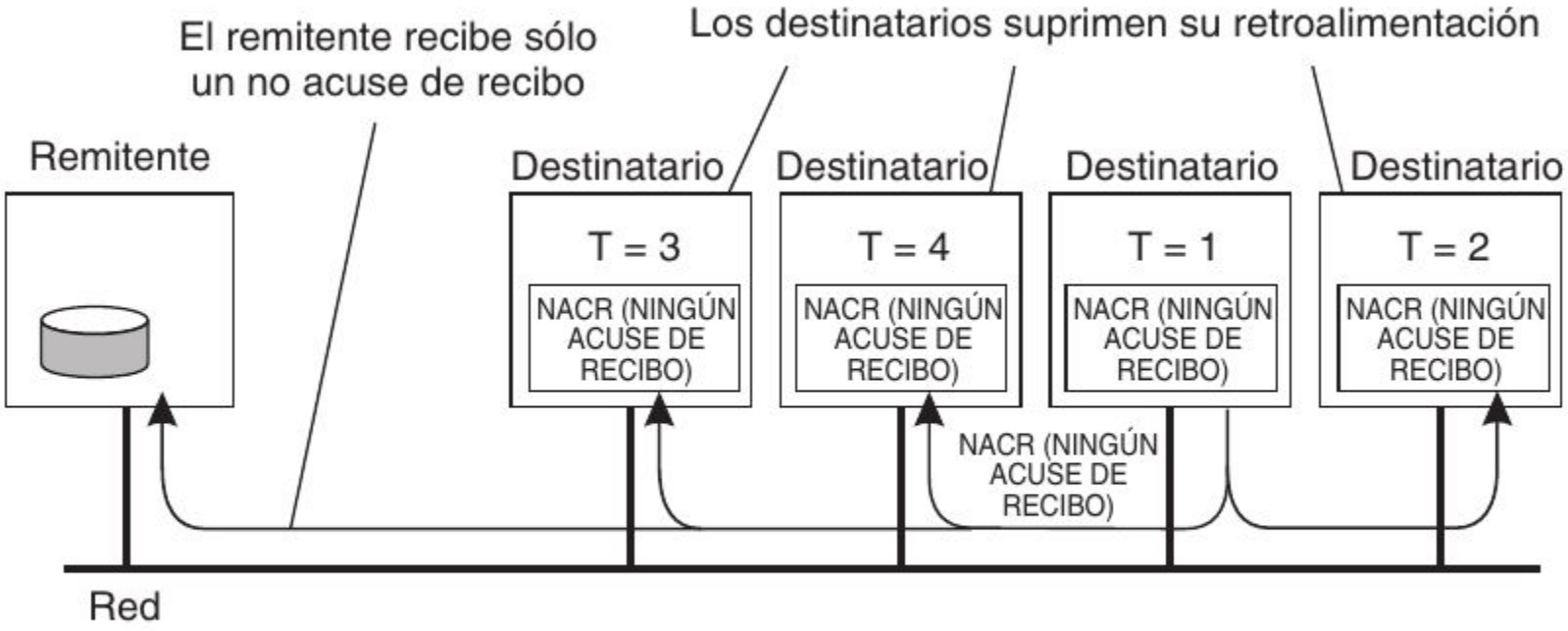
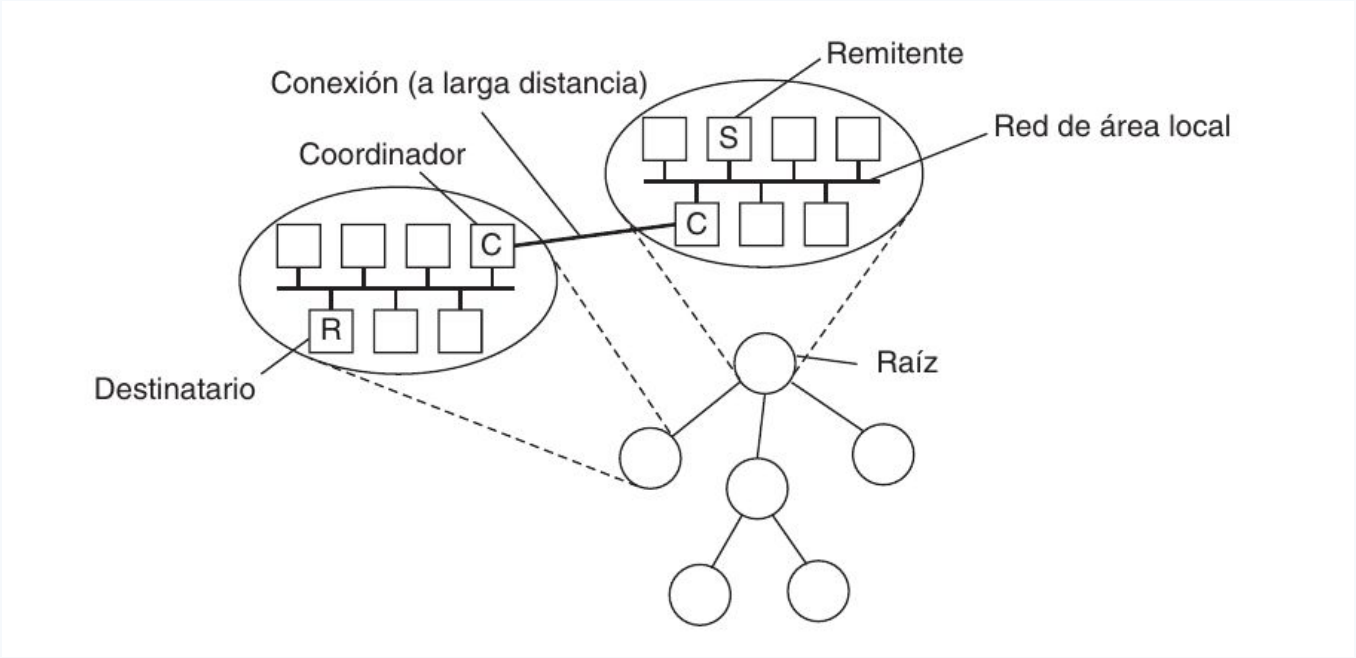


Figura 8-10. Varios destinatarios han programado una solicitud de retransmisión, pero la primera solicitud conduce a la supresión de las demás.

Control de retroalimentación jerárquico

Lograr la escalabilidad para grupos de destinatarios muy grandes requiere la adopción de métodos jerárquicos.



Esencia de la multitransmisión confiable jerárquica. Cada coordinador local remite el mensaje a sus hijos, y posteriormente se ocupa de las solicitudes de retransmisión.