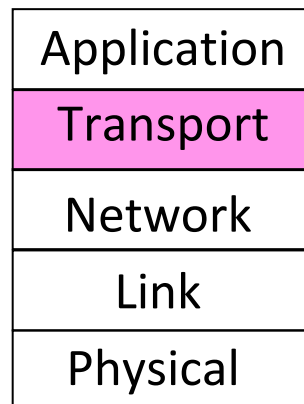


Capítulo 3

Capa de Transporte Liberación de conexiones



Agenda

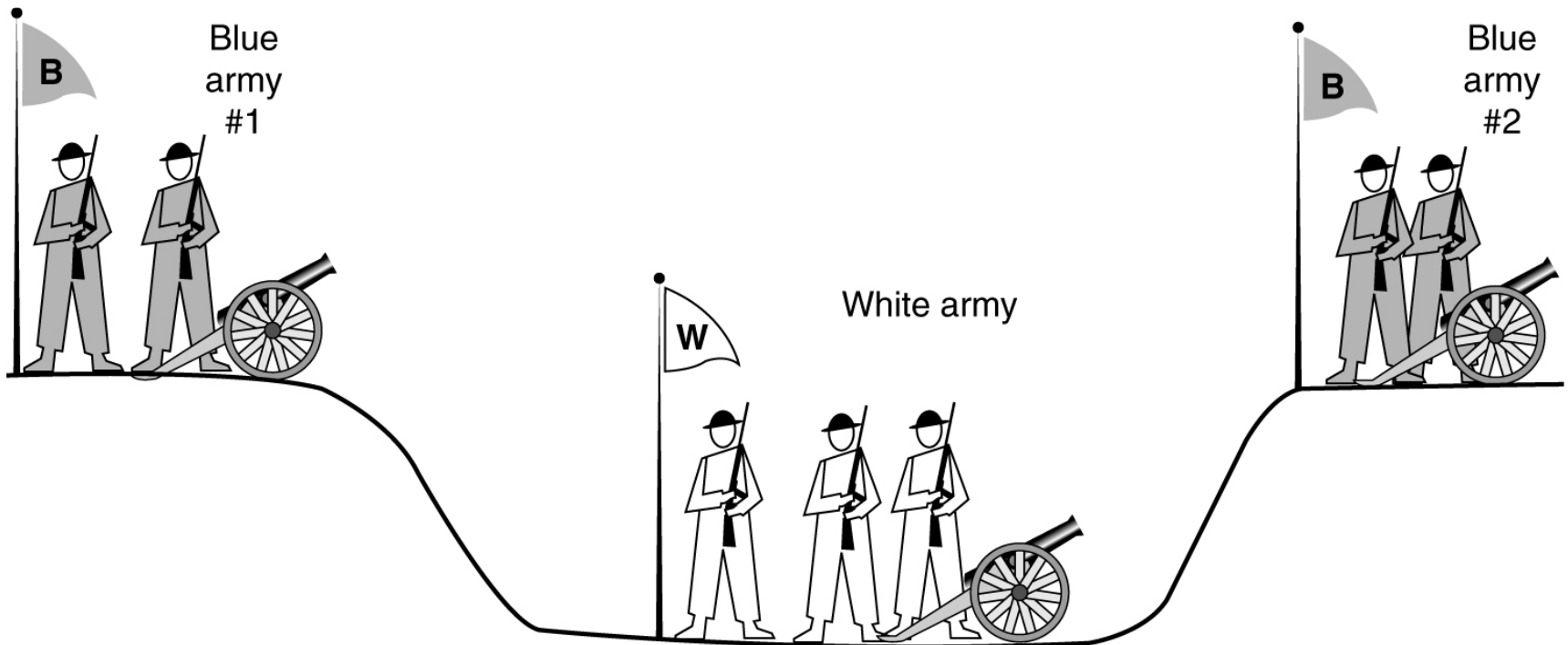
- **Aprenderemos los siguientes asuntos:**
 1. Establecimiento de Conexión
 2. Establecimiento de Conexión en TCP
 - 3. Liberación de Conexiones**
 4. Liberación de conexiones en TCP

Liberación de Conexiones

- **Problema:** ¿Cómo hacer un protocolo para liberación de conexiones?
- **Idea 1:** Podríamos pensar en un protocolo en el que:
 - el host 1 dice “ya terminé ¿terminaste también?”;
 - Si el host 2 responde “Ya terminé también. Adiós”, la conexión puede liberarse con seguridad.
- **Evaluación:** un protocolo así no siempre funciona.
 - Porque existe el **problema de los dos ejércitos**

Liberación de Conexiones

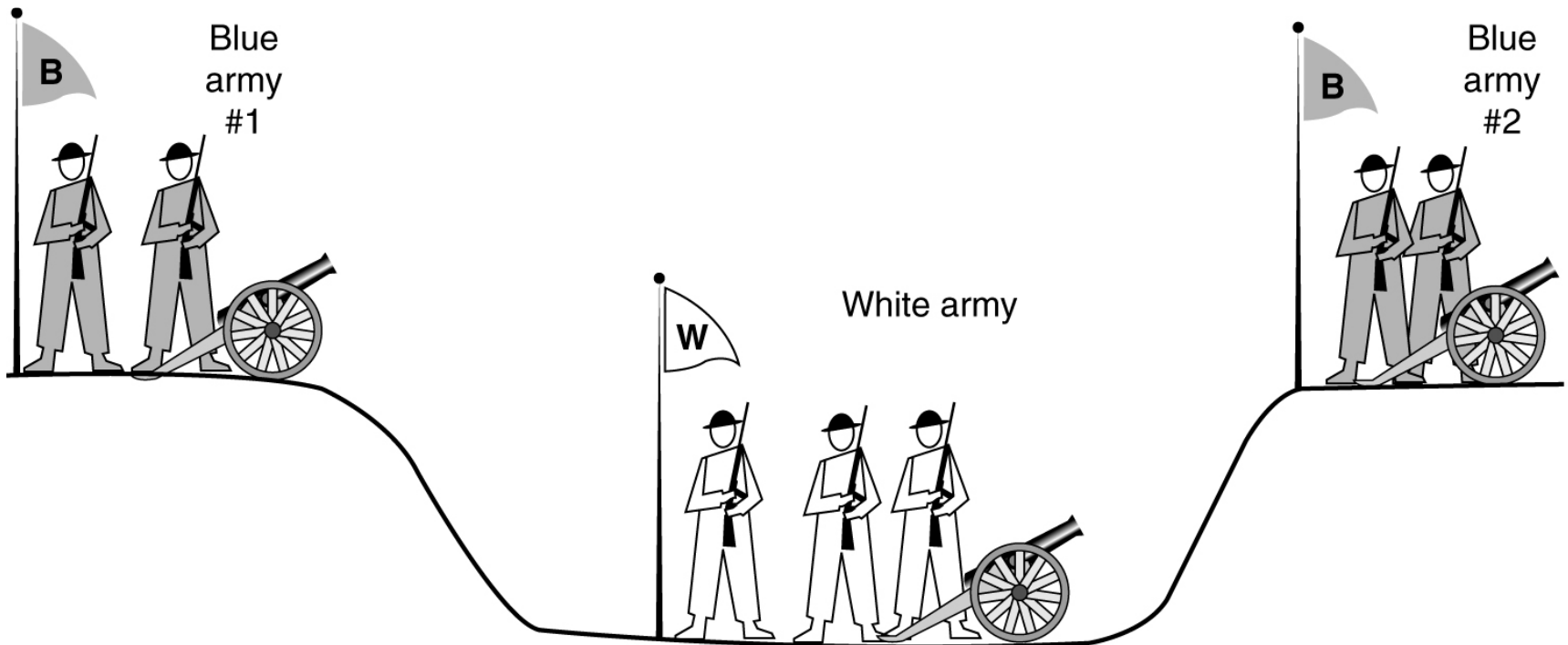
Si los dos ejércitos azules atacan simultáneamente van a ganar. Por eso quieren sincronizar su ataque.



- Suponga que el comandante de ejército azul 1 manda mensaje: “qué le parece que ataquemos en el horario X?”, el mensaje llega y el comandante del ejército azul 2 contesta que está de acuerdo. **¿Va a ocurrir el ataque?**

Liberación de Conexiones

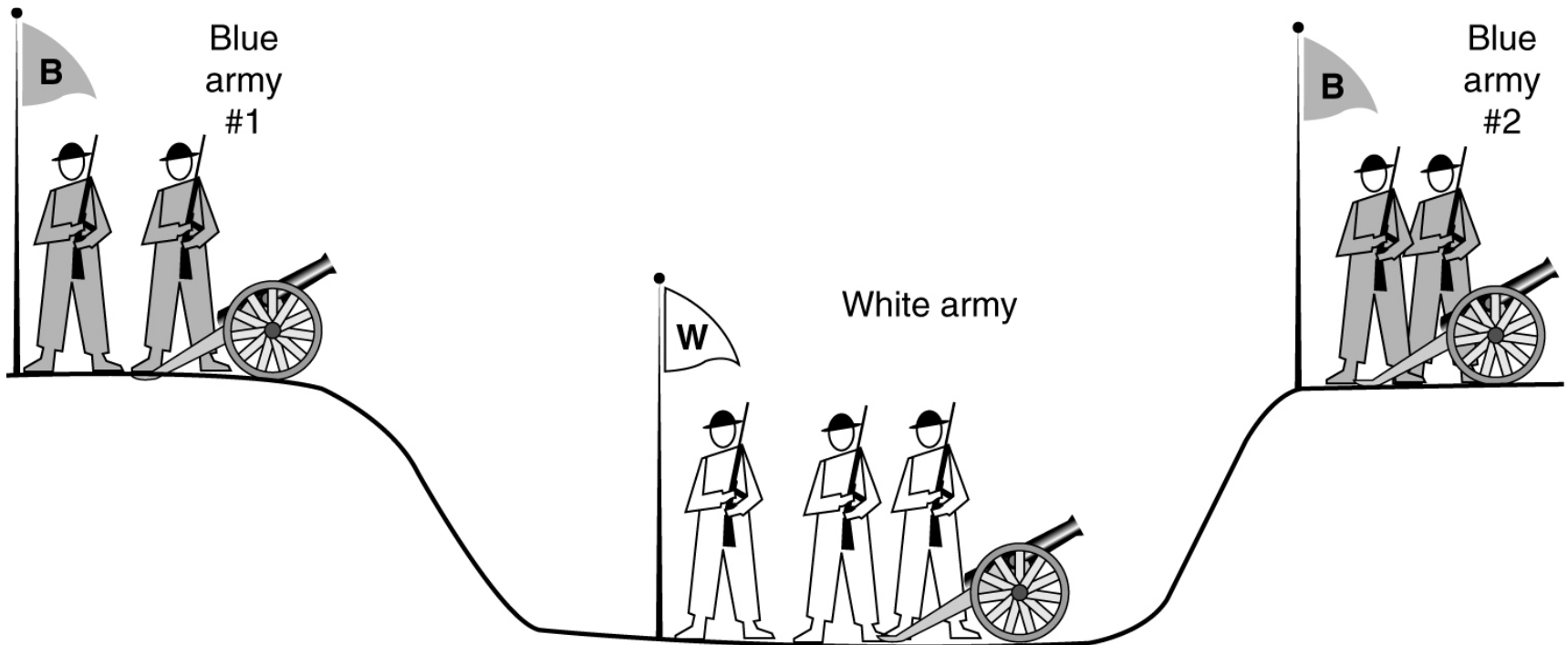
Si los dos ejércitos azules atacan simultáneamente van a ganar. Por eso quieren sincronizar su ataque.



- No porque el comandante del ejército azul 2 no sabe si el mensaje fue recibido por el ejército azul 1.

Liberación de Conexiones

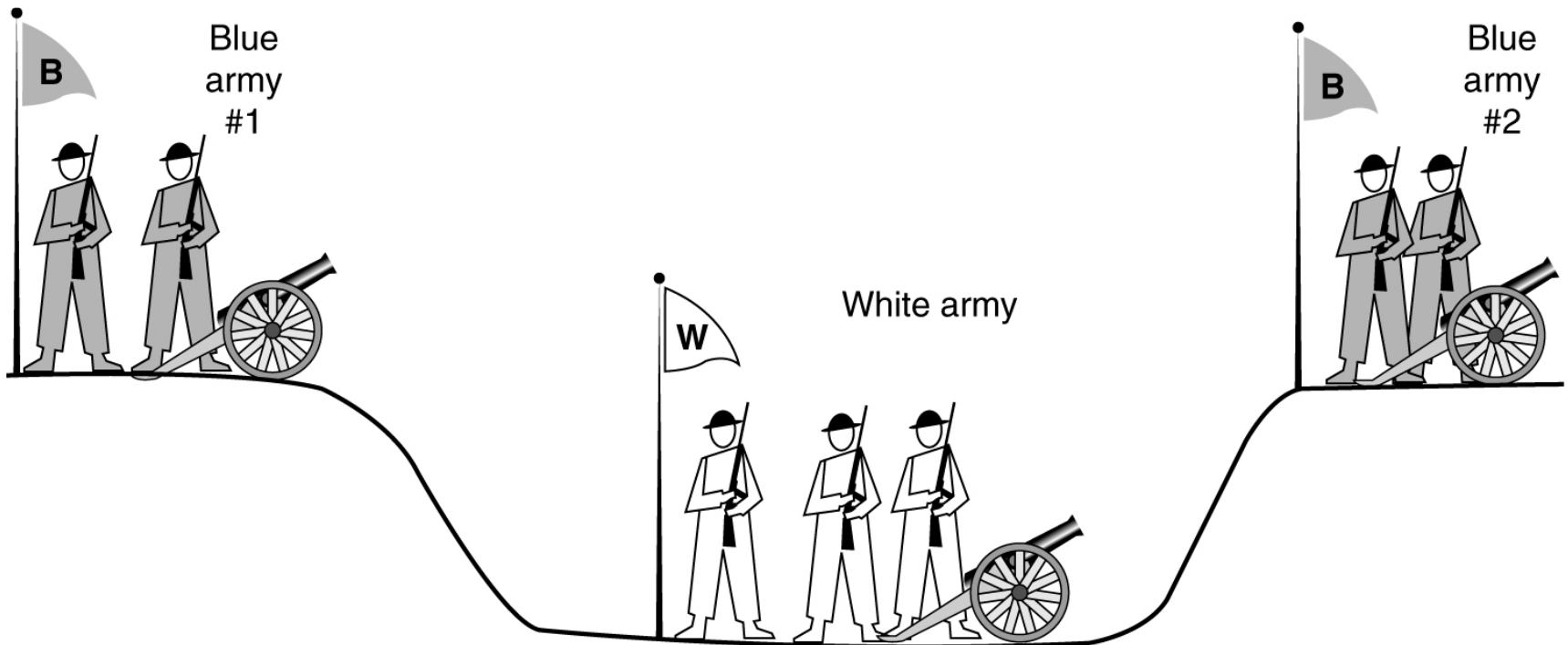
Si los dos ejércitos azules atacan simultáneamente van a ganar. Por eso quieren sincronizar su ataque.



- Supongamos que mejoramos el protocolo de la filmína previa haciendo que el comandante del ejército azul 1 confirme la recepción del mensaje del comandante del ejército azul 2. **¿Va a ocurrir el ataque?**

Liberación de Conexiones

Si los dos ejércitos azules atacan simultáneamente van a ganar. Por eso quieren sincronizar su ataque.



- No porque el comandante del ejército azul 1 no sabe si la confirmación de recepción fue recibida por el ejército azul 2.

Liberación de Conexiones

- **Proposición:** No existe un protocolo que resuelva el problema de los dos ejércitos:
- **Demostración:**
 - Supongamos que existiera un protocolo; o el último mensaje del protocolo es esencial o no lo es.
 - Si no lo es, hay que eliminarlo y así seguir hasta que quede un protocolo en el que todos los mensajes son esenciales.
 - ¿Qué ocurre si el mensaje final no pasa?
 - Si se pierde el ataque no ocurrirá; dado que el emisor del mensaje nunca puede estar seguro de su llegada, no se arriesgará a atacar; el otro ejército azul sabe esto, por lo que no atacará tampoco.

Liberación de Conexiones

- **Aplicación del problema de los dos ejércitos a liberación de conexiones:**
 - Para el caso de liberación de conexión “atacar” equivale a “desconectar”.
 - Si ninguna de las partes está preparada para desconectarse hasta estar convencida que la otra está preparada para desconectarse también, nunca ocurrirá la desconexión.

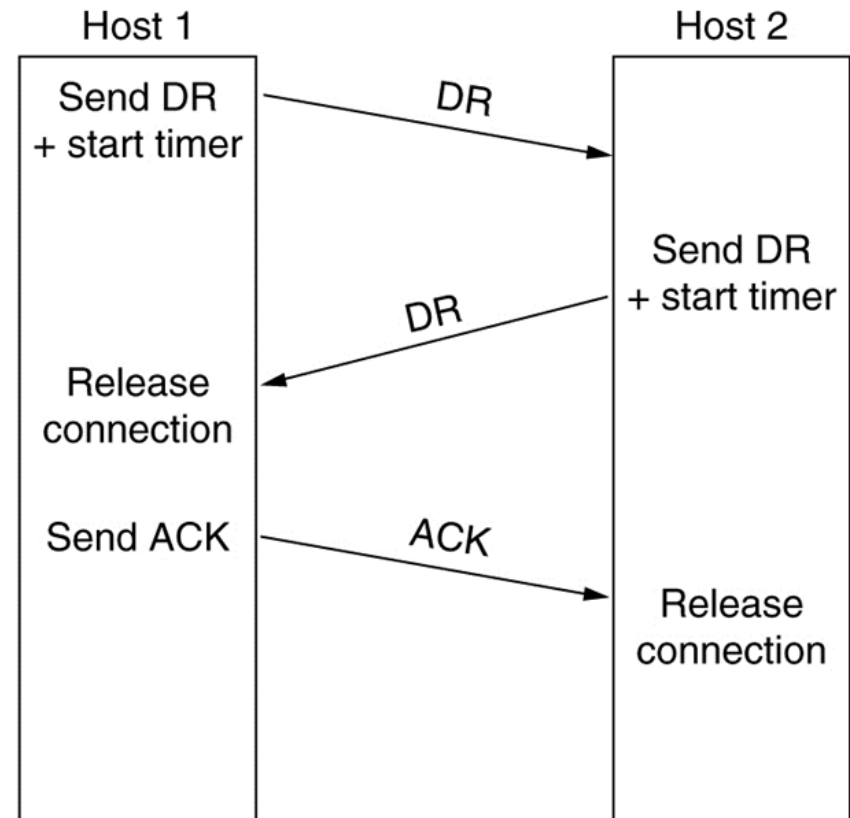
Liberación de Conexiones

- **Idea 2:** Vamos a permitir que cada parte decida cuando la conexión está terminada. Este es un problema más sencillo.
- Ahora estudiamos cuatro escenarios de liberación de conexión usando un **acuerdo de 3 vías**.
 - Aunque este protocolo no es infalible, generalmente es adecuado.
- **La liberación de conexión en un host significa:**
 - que la ET remueve la información sobre la conexión de su tabla de conexiones abiertas y avisa de alguna manera al dueño de la conexión (i.e. el usuario de transporte).
- **¿Cómo sería el caso normal de acuerdo de 3 vías? (usar idea 2 y que se envían mensajes DR – de pedido de desconexión.)**

Liberación de Conexiones

- **Caso normal:** (DR significa DISCONNECTION REQUEST)

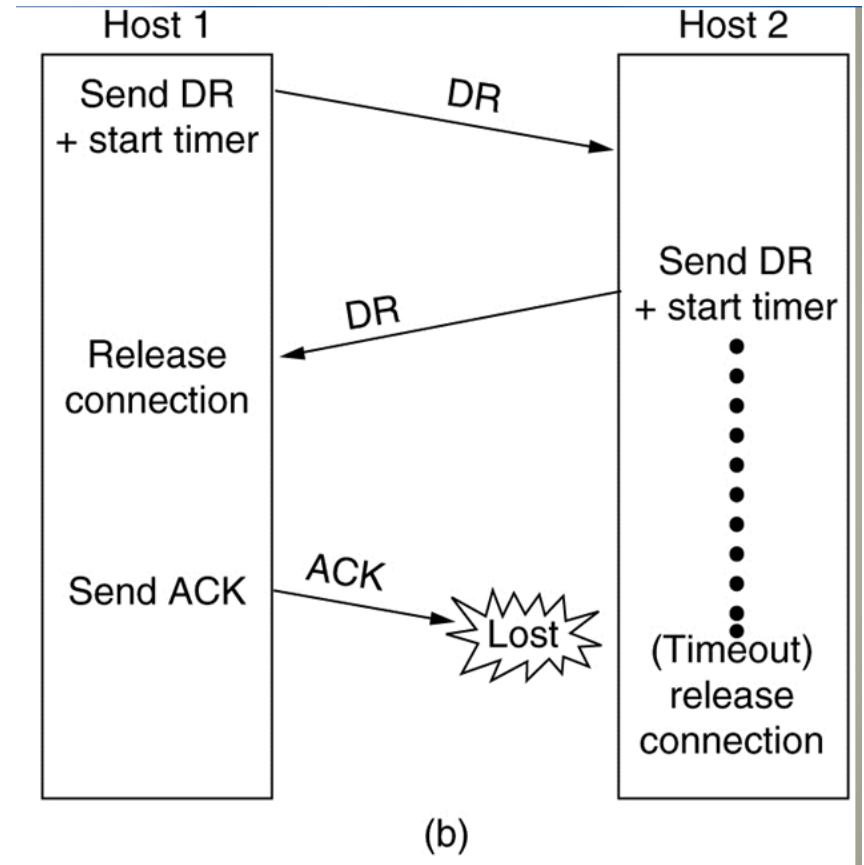
1. Host 1 envía un segmento DR e inicia un temporizador para el caso que no llegue DR de host 2;
2. al llegar DR al host 2, éste emite un segmento DR e inicia un temporizador para el caso que no llegue respuesta de host 1;
3. al llegar esta DR el host 1 envía de regreso un segmento ACK y libera la conexión;
4. cuando el segmento ACK llega el host 2 también libera la conexión.



(a)

Liberación de Conexiones

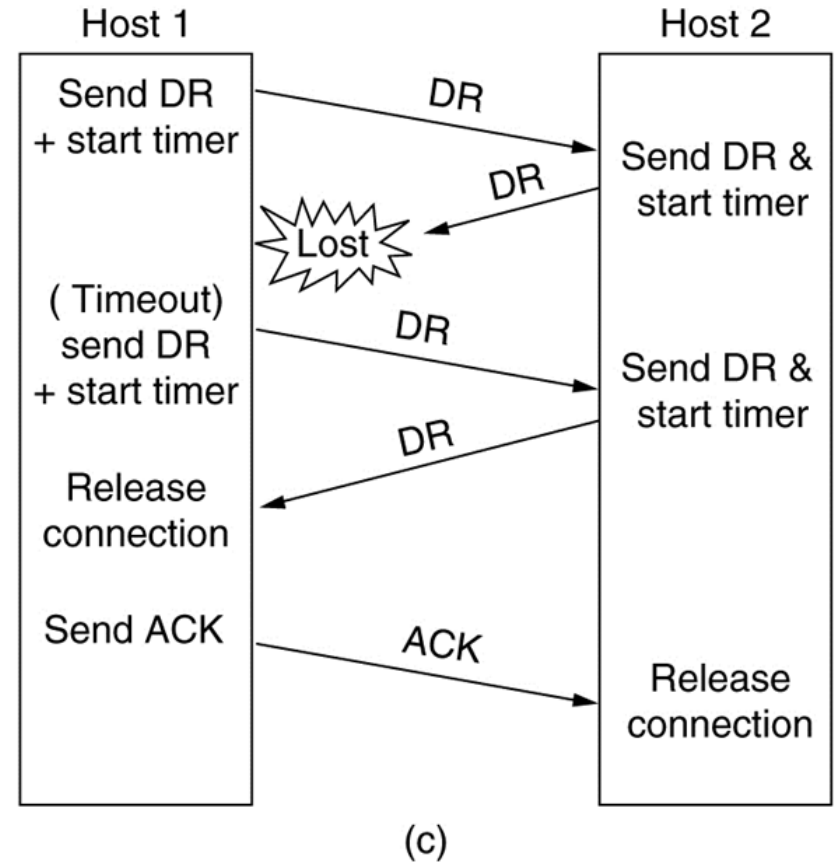
- **Caso 2: Si se pierde el último segmento ACK**
 - al expirar el temporizador la conexión se libera de todos modos.



Liberación de Conexiones

- **Caso 3: Si se pierde el segundo DR qué pasa:**

- el host 1, no recibirá la respuesta esperada, su temporizador expirará y todo comenzará de nuevo.

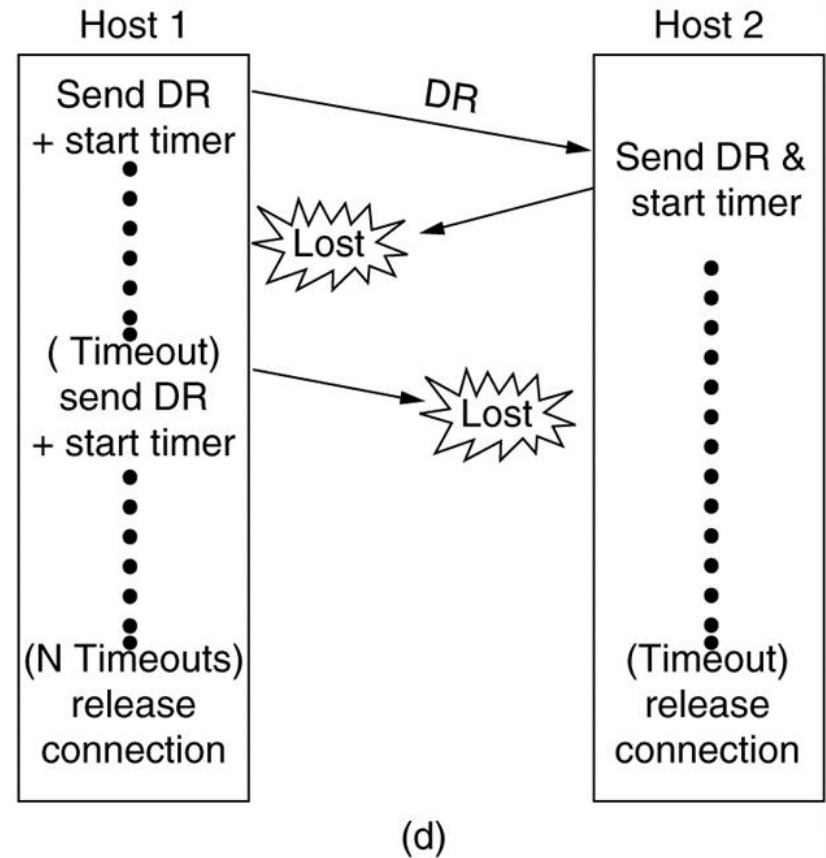


Liberación de Conexiones

- **Caso 4: Respuesta perdida y DRs subsiguientes perdidos.**

Supongamos que todos los intentos repetidos de retransmitir la DR también fallan debido a la pérdida de segmentos.

- Tras N reintentos el emisor se da por vencido y libera la conexión.
- Mientras tanto también termina el temporizador del receptor y también se sale.



Liberación de Conexiones

- **El protocolo anterior falla** si se pierde la DR inicial y N retransmisiones.
 - El emisor se dará por vencido y liberará la conexión, pero el otro lado no sabrá nada sobre los intentos de desconexión y seguirá plenamente activo.
 - Esta situación origina una **conexión abierta a medias**.
- **Problema: ¿Cómo hacer para que no ocurran las conexiones abiertas a medias?**

Liberación de Conexiones

- **Solución 1:** Pudimos haber evitado este problema no permitiendo que el emisor se diera por vencido tras N reintentos, sino obligándolo a seguir insistiendo hasta recibir una respuesta.
- **Problema de esta solución:**
 - si se permite que expire el temporizador en el otro lado, entonces el emisor continuará eternamente, pues nunca aparecerá una respuesta.
- **¿Cómo acabar con las conexiones abiertas a medias de los dos lados?**

Liberación de Conexiones

- **Solución 2:** Una manera de matar conexiones abiertas a medias es:
 - si no ha llegado ningún segmento durante una cierta cantidad de segundos al host 2, se libera automáticamente la conexión en el host 2.
 - Luego el host 1 detectará la falta de actividad y también se desconectará.
 - Esta solución también resuelve el caso que la red “se rompió” y los hosts ya no pueden conectarse.
- **¿Cómo se puede implementar la solución anterior?**

Liberación de Conexiones

- **Implementación:** Es necesario que cada ET tenga un temporizador que se detenga y se reinicie con cada envío de un segmento.
- **Evaluación del protocolo obtenido:** no se puede garantizar **absolutamente** que cuando se libera una conexión no ocurre pérdida de datos.
 - Pero si se puede limitar mucho que esto suceda.

Liberación de Conexiones

La **liberación simétrica**:

- Cada parte se cierra por separado, independientemente de la otra
- Una de las partes emite un DISCONNECT *porque ya no tiene más datos por enviar y aun está dispuesta a recibir datos de la otra parte.*
- Una conexión se libera cuando ambas partes han emitido una primitiva DISCONNECT.

Liberación de Conexiones

- **La liberación simétrica:**
 - Es ideal cuando cada proceso tiene una cantidad fija de datos por enviar y sabe con certidumbre cuándo los ha enviado.
 - En otras situaciones la determinación de si se ha efectuado o no todo el trabajo o si debe terminarse o no la conexión no es tan obvia.
- TCP trabaja con liberación simétrica.

Agenda

- **Aprenderemos los siguientes asuntos:**
 1. Establecimiento de Conexión
 2. Establecimiento de Conexión en TCP
 3. Liberación de Conexiones
 4. **Liberación de conexiones en TCP**

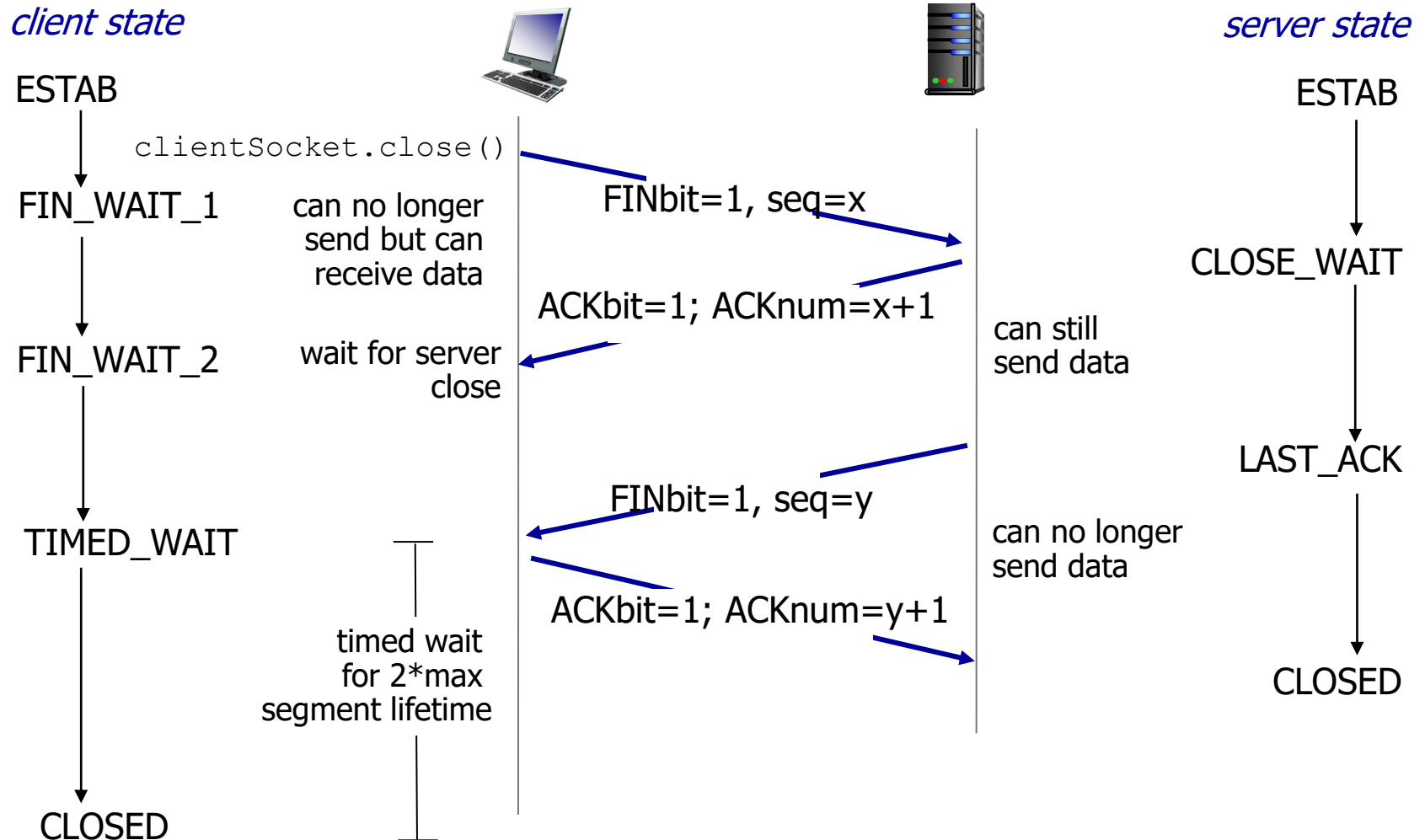
Liberación de una conexión TCP

- Campo usado por TCP para liberación de conexiones:
- **FIN:** especifica que el emisor no tiene más datos que transmitir.
 - Tras cerrar una conexión, un proceso puede **continuar recibiendo** datos indefinidamente.
 - Ambos segmentos, SYN y FIN, tienen n° de secuencia, y por tanto, tienen la garantía de procesarse en el orden correcto.

Liberación de una conexión TCP

- **Ejercicio:** Dar la secuencia de mensajes para la liberación de una conexión TCP considerando:
 - Se usa liberación simétrica.
 - Se usa el campo FIN.
 - Los segmentos con FIN prendido son confirmados.
 - En los segmentos usar los campos: AckNum, AckBit, y Seq.

Liberación de una conexión TCP



Liberación de una conexión TCP

- **Resumen**

- Para liberar una conexión cualquiera de las partes puede enviar un segmento TCP con el **bit *FIN* establecido**, lo que significa que no tiene más datos por transmitir, pero todavía ***puede recibir*** datos del otro lado.
- Al confirmarse la recepción del *FIN*, ese sentido se apaga (el receptor del ack no va a enviar más).
 - Sin embargo puede continuar un flujo de datos indefinido en el otro sentido.

Liberación de una conexión TCP

- Cuando ambos sentidos se han apagado, se libera la conexión.
- Normalmente se requieren 4 segmentos TCP para liberar una conexión: un *FIN* y un *ACK* para cada sentido.
 - Sin embargo es posible que el primer *ACK* y el segundo *FIN* estén contenidos en el mismo segmento, reduciendo la cuenta total a 3.

Liberación de una conexión TCP

- Una vez que el cliente manda el Ack al servidor, entra en un estado de espera llamado TIMED-WAIT.
- El tiempo gastado en `TIMED_WAIT` es de dos tiempos de vida de paquete.
 - TCP espera esta cantidad para garantizar que todos los paquetes de la conexión han muerto, en el caso que el Ack final se haya perdido
- Luego de la espera la conexión se cierra formalmente y todos los recursos del lado del cliente son liberados.

Liberación de una conexión TCP

- Ambos extremos de una conexión TCP pueden enviar segmentos *FIN* al **mismo tiempo**.
 - La recepción de ambos se confirma de la manera normal, y se apaga la conexión.
 - **No hay diferencia** entre la liberación secuencial o simultánea por parte de los hosts.