IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 17)

Consistencia Centrada en el Cliente
¿Qué ve realmente un usuario?

Temas de la clase

1. Consistencia Centrada en el Cliente
2. Sesiones de Garantía
    a. *Monotonic Reads*
    b. *Read Your Writes*
    c. *Monotonic Writes*
    d. *Writes Follow Reads*
3. Tarea 3

Consistencia Centrada en el Cliente

Motivación

Definición

Anteriormente...

- Estudiamos Modelos de consistencia
  - En estricto rigor, fueron *Data-centric consistency models.*
  - Buscan una **vista consistente a nivel de sistema** para todos los procesos que acceden a los datos compartidos de forma concurrente.
  - Estos modelos a menudo requieren mecanismos de sincronización costosos y pueden incurrir en una alta latencia.

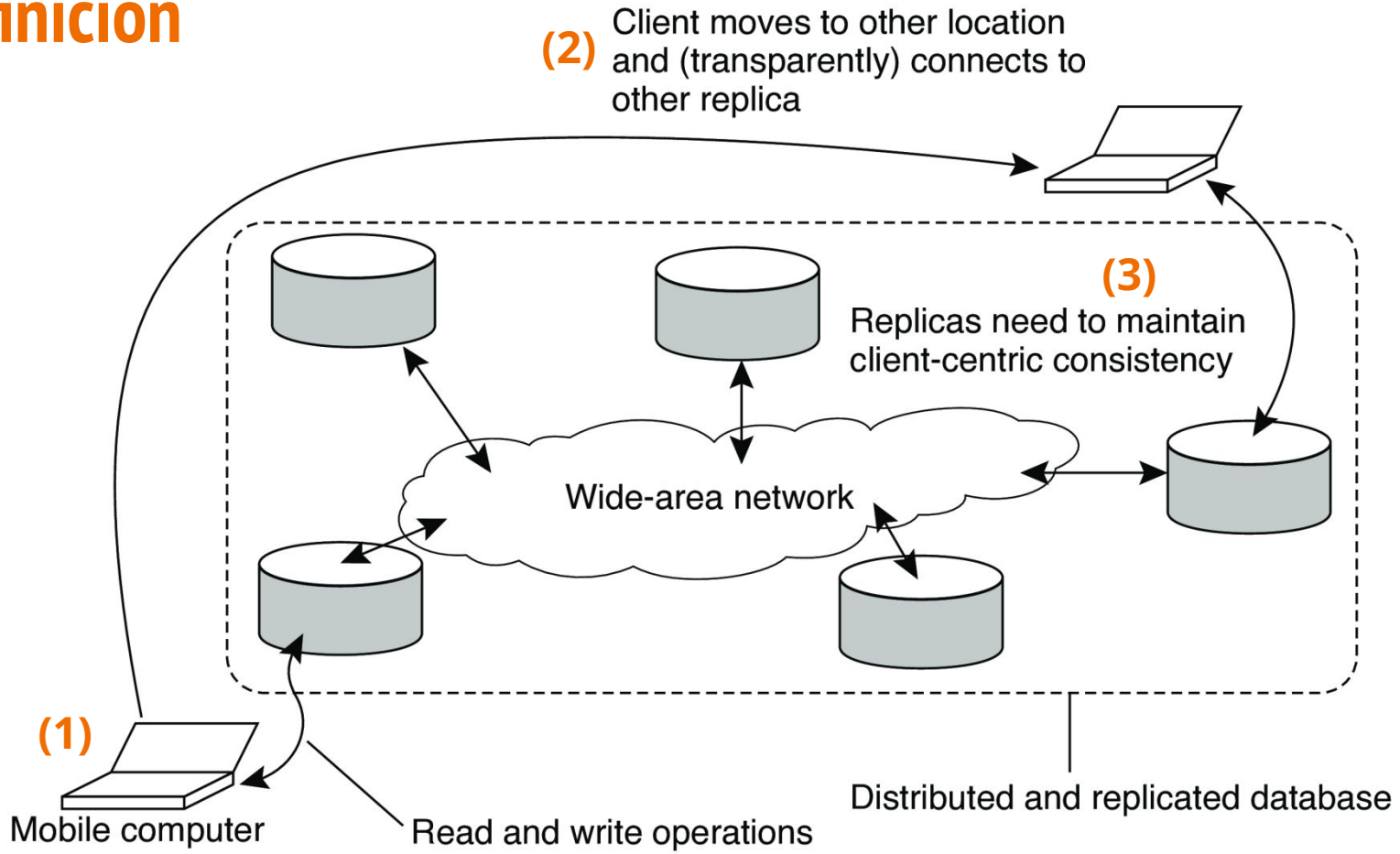
Anteriormente...

- Estudiamos Modelos de consistencia
  - En estricto rigor, fueron *Data-centric consistency models.*
  - Buscan una **vista consistente a nivel de sistema** para todos los procesos que acceden a los datos compartidos de forma concurrente.
  - Estos modelos a menudo requieren mecanismos de sincronización costosos y pueden incurrir en una alta latencia.
- Estudiamos Teorema CAP y PACELC
  - Cuando priorizamos Consistencia (C) y Tolerancia a la Partición (P), a menudo debemos sacrificar Disponibilidad (A), lo que significa que las operaciones podrían bloquearse o fallar si no se puede garantizar la consistencia en todas las réplicas.

Motivación

- Aquí es donde entra en juego la consistencia centrada en el cliente (*Client-centric consistency models*).
  - No se enfocan en una vista global perfecta para todos los usuarios simultáneamente, sino en la experiencia de un cliente individual.
  - Se oculta inconsistencias de manera eficiente y económica.
  - Acepta una forma de Consistencia (C) más relajada pero aceptable desde la perspectiva del usuario.

Definición



Fuente: Distributed Systems (4ª ed.), Steen, M. van, & Tanenbaum, A. S. (2023). Recuperado de https://www.distributed-systems.net.

Definición

- Los modelos de consistencia centrada en el cliente son cruciales para sistemas donde los usuarios acceden a diferentes réplicas a lo largo del tiempo (por ejemplo, al viajar o usar dispositivos móviles).
- El objetivo es mantener una experiencia de usuario coherente, incluso si la consistencia global es débil (como la consistencia eventual).
  - Esto se logra asegurando que, aunque un cliente pueda interactuar con diferentes réplicas a lo largo del tiempo, la réplica con la que interactúa se actualice con los datos que ese cliente ha manipulado previamente.
- Existen ciertas propiedades que se buscan garantizar en los modelos centrados en el cliente.

Sesiones de Garantía

Monotonic Reads

Read Your Writes

Monotonic Writes

Writes Follow Reads

Session Guarantees

- Las garantías de sesión definen expectativas razonables para una **experiencia consistente de un cliente individual al interactuar con un sistema distribuido a lo largo del tiempo**.
  - Las garantías se espera cumplir cuando hay interacción, por ejemplo, conexión a la réplica.
- Se incluyen cuatro propiedades fundamentales:
  - *Monotonic Reads*
  - *Read Your Writes*
  - *Monotonic Writes*
  - *Writes Follow Reads*

(*) No todas deben cumplirse simultáneamente; pueden aplicarse según el caso de uso.

Fuente: Session Guarantees for Weakly Consistent Replicated Data [Terry et al., 1994]

Session Guarantees

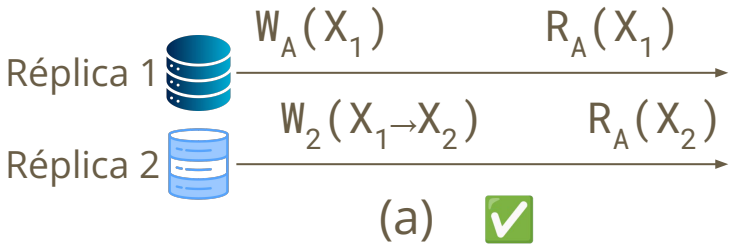
Para describir estos propiedades, utilizamos la siguiente notación:

- X_z — Versión Z Datos X.
- $W_q(X_1 \rightarrow X_2)$ — Indica que el nodo Q escribe versión X_2 *post* saber que existe la versión X_1.
- $W_q(X_1 \mid X_n)$ — Indica que el nodo Q escribió X_n de forma concurrente a la escritura de X_1, la cual pudo haber sido realizada por otro nodo. Esto implica que Q no conocía la existencia de X_1 al momento de escribir X_n, lo que puede llevar a un conflicto de versiones.

Session Guarantees - Monotonic Reads

> *Si un cliente lee X_z , cualquier lectura realizada por ese mismo cliente a X devolverá X_z o alguna versión X_d tal que $W_q(X_z \rightarrow X_d)$*

Intuición: Una vez que has visto algo, no esperarías ver una versión anterior o inconsistente lo que viste recién.

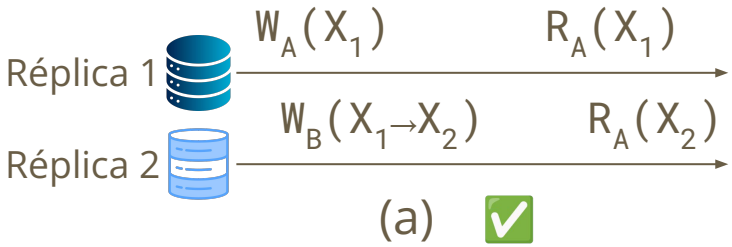


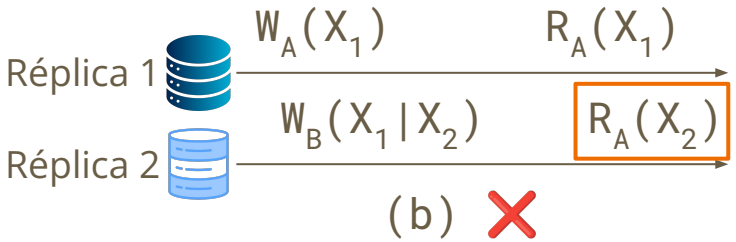
(a) ✅

Session Guarantees - Monotonic Reads

> *Si un cliente lee X_z , cualquier lectura realizada por ese mismo cliente a X devolverá X_z o alguna versión X_d tal que $W_q(X_z \rightarrow X_d)$*

Intuición: Una vez que has visto algo, no esperarías ver una versión anterior o inconsistente lo que viste recién.





Session Guarantees - Monotonic Reads

> *Si un cliente lee X_z , cualquier lectura realizada por ese mismo cliente a X devolverá X_z o alguna versión X_d tal que $W_q(X_z \rightarrow X_d)$*

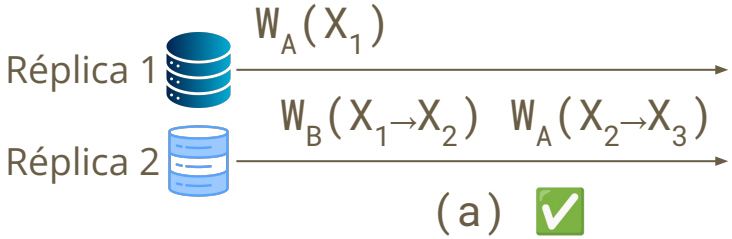
Ejemplo de implementación

En un *load balancer*, se puede usar un *token de sesión* (por ejemplo, un session ID) que dirige todas las lecturas del mismo cliente a la misma réplica o a una que tenga una versión igual o más reciente.

Session Guarantees - Monotonic Writes

Si un cliente escribe X_z , cualquier escritura posterior realizada por ese mismo cliente a X debe ser alguna versión X_d tal que $W(X_z \rightarrow X_d)$

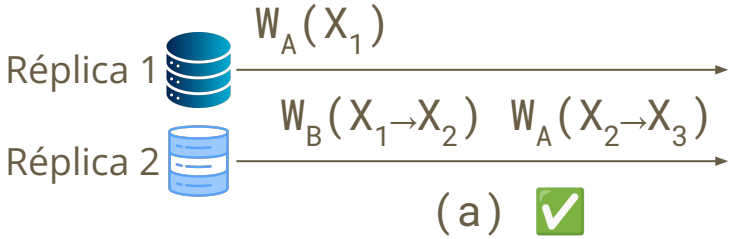
Intuición: Tus propias escrituras deben reflejar una progresión lógica.

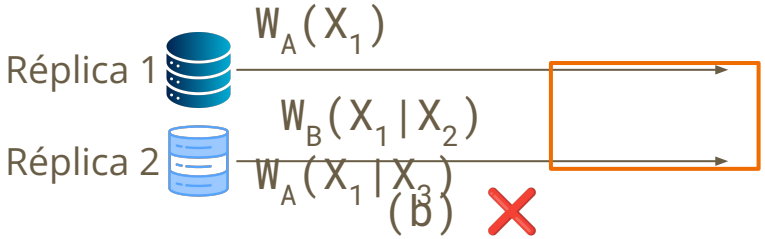


Session Guarantees - Monotonic Writes

Si un cliente escribe X_z , cualquier escritura posterior realizada por ese mismo cliente a X debe ser alguna versión X_d tal que $W(X_z \rightarrow X_d)$

Intuición: Tus propias escrituras deben reflejar una progresión lógica.

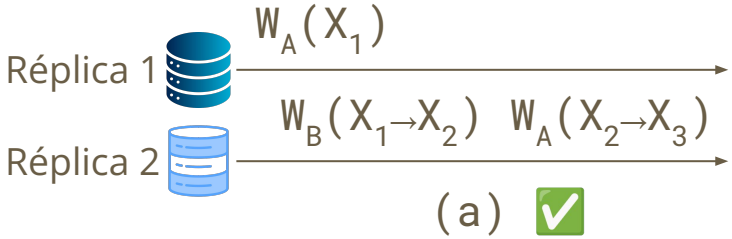


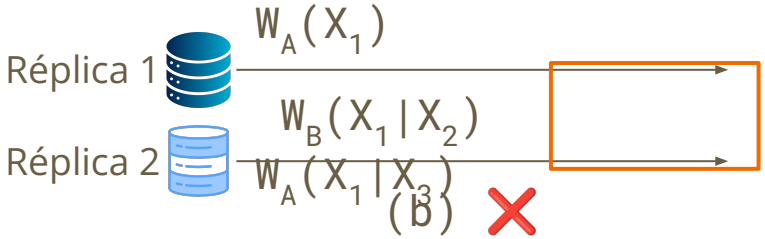


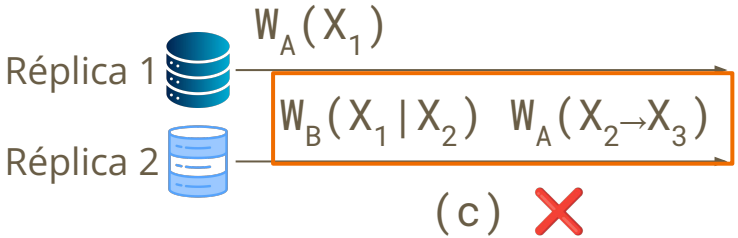
Session Guarantees - Monotonic Writes

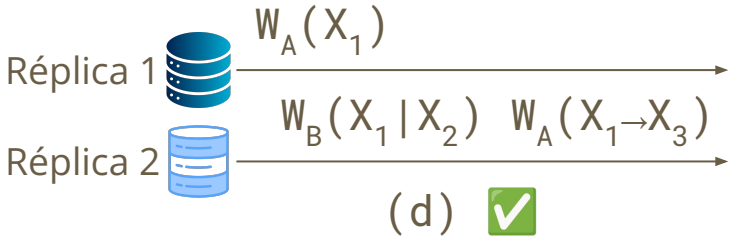
Si un cliente escribe X_z , cualquier escritura posterior realizada por ese mismo cliente a X debe ser alguna versión X_d tal que $W(X_z \rightarrow X_d)$

Intuición: Tus propias escrituras deben reflejar una progresión lógica.









Session Guarantees - Monotonic Writes

Si un cliente escribe X_z , cualquier escritura posterior realizada por ese mismo cliente a X debe ser alguna versión X_d tal que $W(X_z \rightarrow X_d)$

Ejemplo de implementación

El cliente no emite la siguiente escritura hasta que reciba confirmación de la escritura anterior, garantizando orden local.

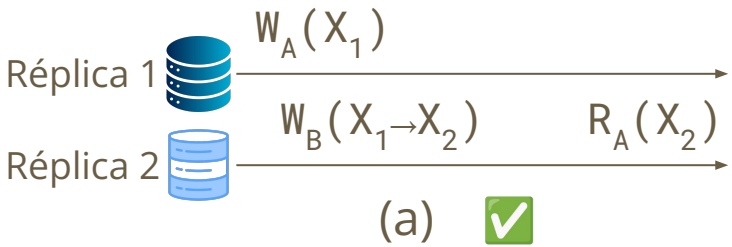
- La confirmación puede implicar esperar que el dato se propague.

Client tokens con vectores de versión: los servidores aplican la escritura sólo si incluye todas las escrituras previas del cliente y utiliza el vector para resolver posibles conflictos.

Session Guarantees - Read your writes

> *Si un cliente escribe X_z , cualquier lectura posterior realizada por ese mismo cliente a X debe ser X_z o alguna versión X_d tal que $W(X_z \rightarrow X_d)$*

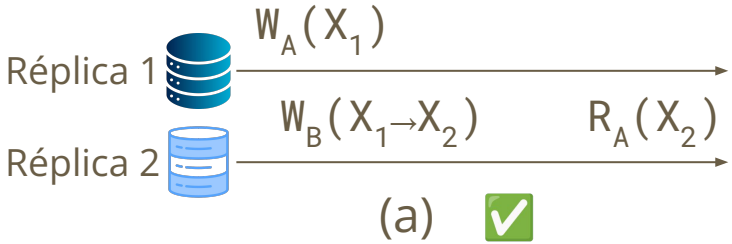
Intuición: Si escribes algo, deberías poder leerlo inmediatamente después.

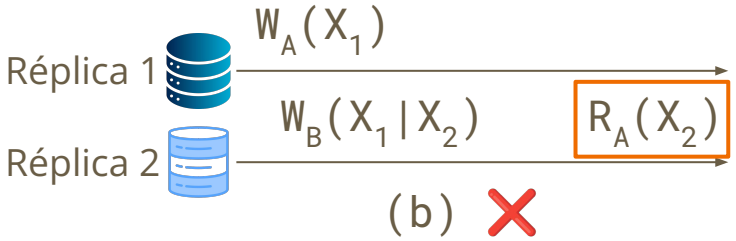


Session Guarantees - Read your writes

> *Si un cliente escribe X_z , cualquier lectura posterior realizada por ese mismo cliente a X debe ser X_z o alguna versión X_d tal que $W(X_z \rightarrow X_d)$*

Intuición: Si escribes algo, deberías poder leerlo inmediatamente después.





Session Guarantees - Read your writes

> *Si un cliente escribe X_z , cualquier lectura posterior realizada por ese mismo cliente a X debe ser X_z o alguna versión X_d tal que $W(X_z \rightarrow X_d)$*

Ejemplo de implementación

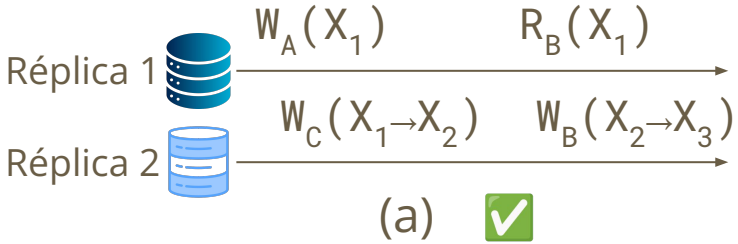
Session token para redirigir consulta a solo servidores que tengan la escritura propagada.

Confirmación de escritura.

Session Guarantees - Writes Follow Reads

Si un cliente lee X_z , cualquier escritura posterior realizada por ese mismo cliente a X debe ser alguna versión X_d tal que $W(X_z \rightarrow X_d)$

Intuición: Si lees algo y luego lo modificas, tu modificación debe basarse en la versión que leíste.

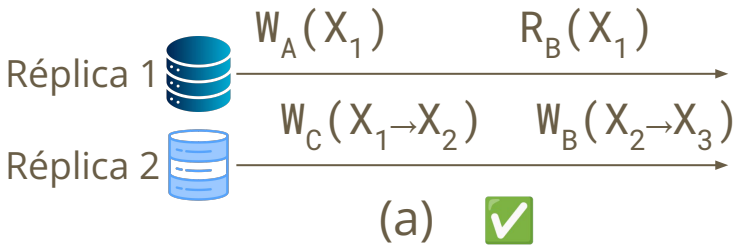


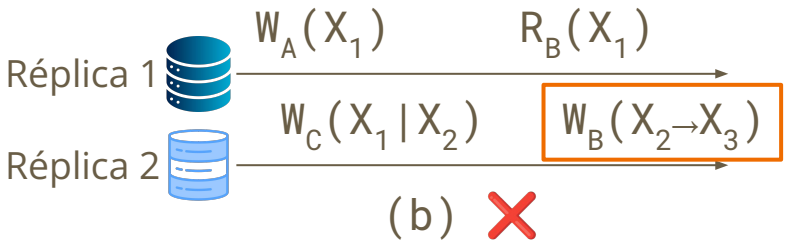
(a) ✅

Session Guarantees - Writes Follow Reads

Si un cliente lee X_z , cualquier escritura posterior realizada por ese mismo cliente a X debe ser alguna versión X_d tal que $W(X_z \rightarrow X_d)$

Intuición: Si lees algo y luego lo modificas, tu modificación debe basarse en la versión que leíste.





Session Guarantees - Writes Follow Reads

> *Si un cliente lee X_z , cualquier escritura posterior realizada por ese mismo cliente a X debe ser alguna versión X_d tal que $W(X_z \rightarrow X_d)$*

Ejemplo de implementación

Cada *read* devuelve un identificador de causalidad (por ejemplo, un *version vector*). El siguiente *write* incluye este identificador para asegurar que se aplica solo en nodos que ya tienen esa versión o posteriores.

Session Guarantees - Fallas comunes I

🔴 *Read Your Writes*

El usuario publica un comentario en una red social, recargas... y no aparece.

⚠️ 🚨 El usuario piensa: “¿No se publicó? ¿Debo escribirlo de nuevo?”

🔴 *Monotonic Reads*

El usuario revisa su correo desde el teléfono y ves un correo nuevo. Luego abre el cliente web... ¡y el correo no está!

⚠️ 🚨 El usuario cree que algo "se perdió" o que el sistema es inestable.

Session Guarantees - Fallas comunes II

🔴 *Monotonic Writes*

El usuario manda un mensaje por Telegram desde tu computador y luego lo edita desde tu celular.

Cuando vuelve al PC para editar el mensaje, esta tercera edición se basa en la versión inicial del computador y no en la editada desde el celular.

⚠️ 🚨 El usuario creía que sus cambios se guardaron correctamente, pero en realidad el sistema no propagó el cambio del celular.

Session Guarantees - Fallas comunes III

🔴 *Writes Follow Reads*

El usuario ve en la agenda que hay una reunión programada para el martes, pero rechaza porque está ocupado ese día y propone moverla al miércoles.

Pero justo antes, alguien más ya había movido la reunión al miércoles y el usuario no tenía reflejado ese cambio al momento de enviar su propuesta.

⚠️ 🚨 La propuesta del usuario se basa en información desactualizada, generando conflicto o incoherencia en el sistema.

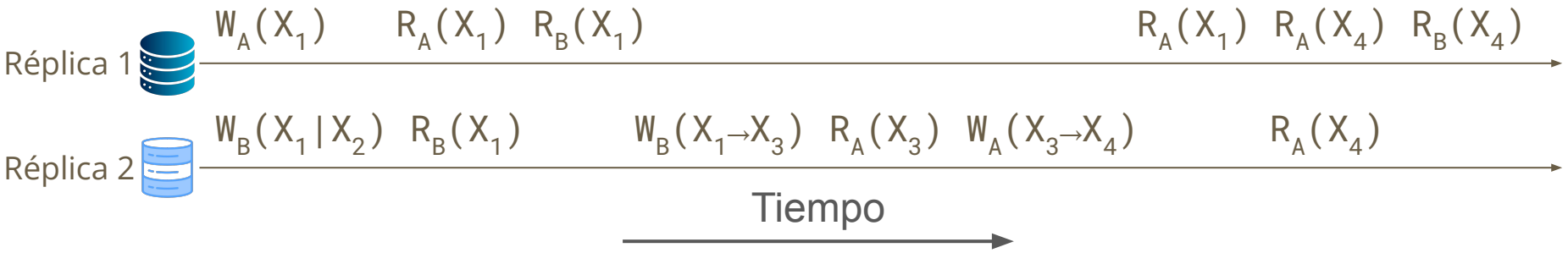
Session Guarantees

🤔 ¿Por qué esto importa? 🤔

- La percepción de confiabilidad del sistema depende de que sus acciones tengan efecto visible y coherente.
- "Lo que hice, se guardó" y "lo que veo, tiene sentido" son expectativas básicas.
- La consistencia eventual no es suficiente para una buena UX.
- Compromiso práctico:
  - No siempre puedes ofrecer consistencia fuerte global.
  - Pero sí puedes ofrecer una experiencia consistente para cada cliente.

Poniendo a prueba lo que hemos aprendido 👀

¿Cuales de las siguientes garantías se están **cumpleando** para el **cliente A**? Responder con Sí o No a cada propiedad.



a. *Monotonic Reads*

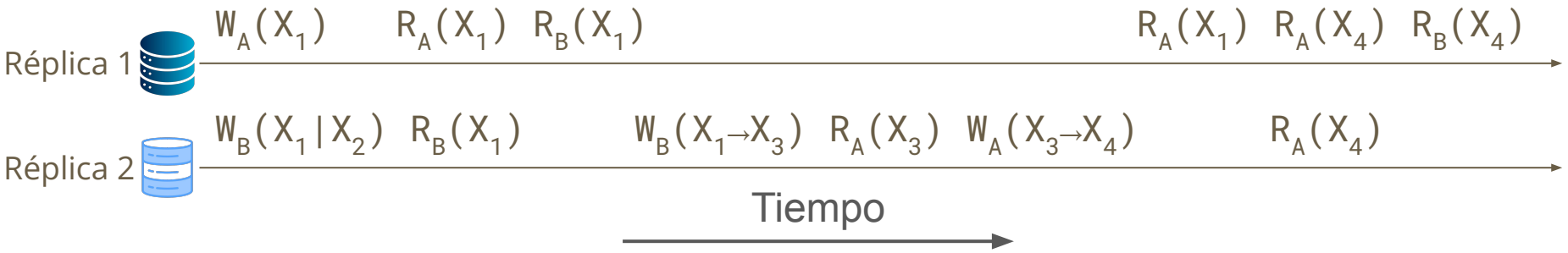
b. *Read Your Writes*

c. *Monotonic Writes*

d. *Writes Follow Reads*

Poniendo a prueba lo que hemos aprendido 👀

¿Cuales de las siguientes garantías se están **cumpleando** para el **cliente A**? Responder con Sí o No a cada propiedad.



a. *Monotonic Reads* ❌ Lee X_3 en una réplica y luego volvió lee una versión pasada.

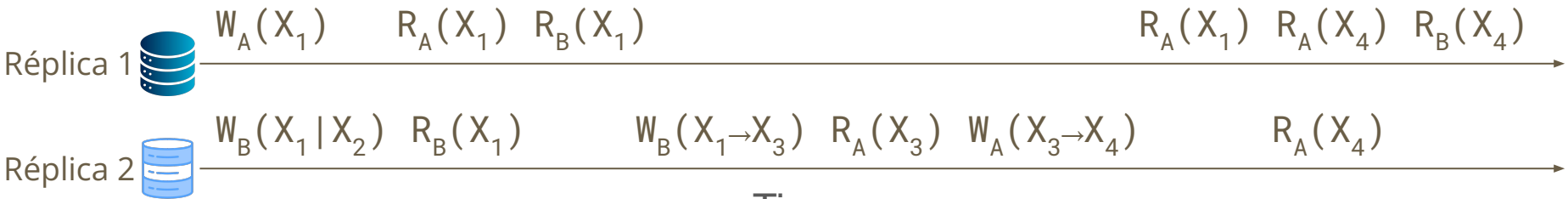
b. *Read Your Writes* ❌ Escribe X_4 pero luego no lee esa versión.

c. *Monotonic Writes* ✅ Sus escrituras muestran una progresión.

d. *Writes Follow Reads* ✅ Escribe siempre en base a lo que vió.

Poniendo a prueba lo que hemos aprendido 👀

¿Cuales de las siguientes garantías se están **cumpleando** para el **cliente A**? Responder con Sí o No a cada propiedad.



a. *Monotonic Reads* ❌ Lee X y luego vol

b. *Read Your Writes* ❌ Es lo

c. *Monotonic Writes* ✅ Su una p

d. *Writes Follow Reads* ✅ Esc se a lo que vió.

¿Qué podemos decir del cliente B?

Tarea para la casa

Próximos eventos

Próxima clase

- Última unidad del curso: seguridad computacional
- Partiremos con una introducción a algunos conceptos.

Evaluación

- Última tarea del curso: Tarea 3.
- Control 5 se publica el otro miércoles.

IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 17)

Créditos (animes utilizados)

Kusuriya no Hitorigoto