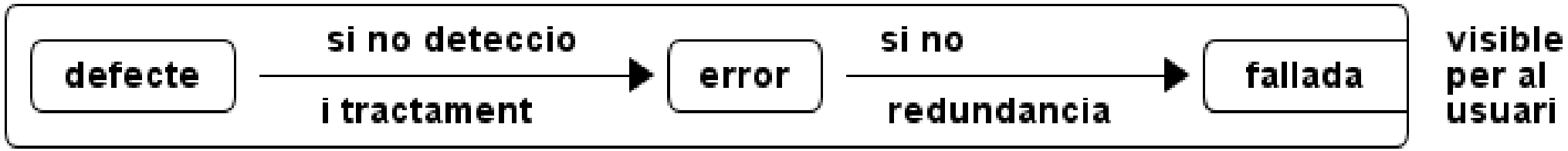
Tema 5.- Gestión de errores

TSR 2021, Juansa Sendra, Grupo B

Introducción.- fallos, replicación, consistencia

- definiciones
  - **defecto** (fault) = condición anómala
  - **error** = manifestación de un defecto (elemento en estado incorrecto)
  - **fallo** (failure)= un elemento del sistema no puede completar su función



Ejemplo.- lectura sector en disco

- defecto.- al leer un sector encontramos un problema de checksum (detección) reintenta por si se trata de un defecto transitorio (ej. polvo, cabezal mal posicionado, ...) (tratamiento)
- error (después de reintentar no desaparece el defecto). Algunos sectores críticos están replicados, o bien usamos un RAID. Si el erróneo está replicado, lee la réplica
- fallo.- no podemos leer el sector $\rightarrow$ memsaje al usuario (visible)

Situaciones de fallo en un sistema distribuido (SD)

- Todo SD debe proporcionar transparencia de errores
  - en caso de producirse errores en algún componente, no ha de ser percibidos por los usuarios $\longrightarrow$ replicación
- También se denomina "fault tolerance" (tolerancia a defectos)
  - El sistema exhibe comportamiento correcto en caso de que aparezcan defectos

Modelos de fallo

- Los fallos pueden tener múltiples causas
- Para diseñar algoritmos distribuidos conviene asumir un modelo de fallos
  - Errores de nodos y de comunicación
  - Errores detectables vs arbitrarios (bizantinos): únicamente consideramos los detectables
  - Errores simples vs múltiples: únicamente consideramos errores simples
    - excepción: también nos interesa el problema de particionado (un grupo de nodos queda aislado del resto del sistema) debido a un error múltiple

Particionado.- Opciones ante una partición

- Sistema particionable
  - Cada grupo aislado puede continuar (posibles inconsistencias)
  - Se necesita un protocolo de reconciliación cuando se recupere la conectividad
- Modelo de partición primaria
  - Únicamente se admite que continue aquel grupo que tenga una mayoría de nodos
  - Puede no existir

Replicación

- Replicación = Varias copias en nodos diferentes
  - Los nodos utilizados no depeden de una misma fuente de defectos
    - red eléctrica, red local, ...
  - Cada réplica falla de manera independente $\rightarrow$ las operaciones en marcha en un réplica errónea pueden completarse en otra réplica

Replicación.- Utilidad

- Mecanismo básico para asegurar la disponibilidad de un componente
- Facilita la recuperación tras un error
  - las réplicas activas se usan como fuente para reconfigurar una réplica reparada
- Mejora el rendimiento si alto % de lecturas
  - ops lectura pueden ejecutarse simultáneamente, cada una por una sola réplica (escalable)
  - ops escritura han de ser aplicadas por todas las réplicas (no escalable)
  - según como se intercalen o propaguen las ops en cada réplica pueden haber divergencias entre sus estados $\longrightarrow$ diferentes modelos de consistencia

Modelos de replicación

- **Pasivo** (una réplica primaria y varias réplicas secundarias)
  - Peticiones cliente a la primaria, que ejecuta la operación
  - Cuando termina propaga las modificaciones a las secundarias y responde al cliente
- **Activo** (máquina de estados)
  - El cliente propaga la petición a todas las réplicas
  - Cada réplica ejecuta directamente la operación y cuando finaliza responde al cliente

Comparativa modelos de replicación

Aspecto	Pasivo	Activo
réplicas procesadoras	1	todas
evita ordenación distrib. peticiones	**si**	no
evita propagación modificaciones	no	**si**
admite ops. no deterministas	**si**	no (inconsist.)
tolera errores arbitrarios	no	**si** (*)
recuperación	lenta (elección primario, reconf)	**inmediata**

(*) para ello requiere que todas las réplicas procesen también las lecturas

Comparativa modelos de replicación.- otros detalles

- En el pasivo (+)
  - mínima carga (sólo una réplica procesa cada petición, puede repartir lecturas)
  - Resulta fácil establecer el orden (numerando los mensajes difundidos por el primario)
  - control de concurrencia local (no necesita alg. de concurrencia distribuidos)
- En el activo (-)
  - las peticiones deben difundirse a todas las réplicas en orden total
    - requeriere consenso, que es un protocolo pesado = caro
  - Cuando interactuan servicios replicados debemos filtrar peticiones
    - Ej. servicios A y B utilizan replicación activa, y A invoca a B
      - B debe filtrar las peticiones recibidas (dejar únicamente 1)
      - y propagar las respuestas a todas las réplicas de A

Comparativa modelos de replicación.- Servicio ejemplo S1

- Queremos desplegar un servicio con los siguientes valores
  - 5 réplicas
  - 1 Gbps de ancho de banda
  - 2ms para la propagación de cada mensaje
  - 200ms para tiempo de respuesta a las operaciones
    - no hay concurrencia interna: para N peticiones, tiempo de respuesta 200*N
  - 10 KB talla media de las modificaciones por operación
    - 10 KB = 80Kb = 0.00008 Gb (a 1Gbps la transferencia necesita 0.08ms)
    - tiempo transferencia modificaciones = 2ms (propagación) + 0.08 ms (transferencia) = 2.08 ms
  - 3 ms para aplicar las modificaciones en las réplicas secundarias

Comparativa modelos de replicación.- Desplegamos 1 servicio S1

Aspecto	Pasivo	Activo
envío petición	2ms	4-6 ms (*)
procesamiento petición	200ms	200ms (en pararelo)
transferencia modificaciones	2.08ms	0ms
Aplicación modificaciones	3ms (secun.)	0ms
Respuesta a modificaciones	2ms	0ms
Respuesta al cliente	2ms	2ms
TOTAL	211.08ms	206..208ms

(*) con un algoritmo de difusión de orden total basado en secuenciador

Comparativa models de replicació.- Despliega 5 servicios S1

En modelo pasivo, cada servicio despliega su réplica primaria en un nodo distinto

Aspecto	Pasivo	Activo
envío petición	2ms	4-6 ms
procesamiento petición	200ms+ 4*3=212ms	200ms*5 = 1000ms
transferencia modificaciones	2.08ms	0ms
Aplicación modificaciones	3ms (secun.)	0ms
Respuesta a modificaciones	2ms	0ms
Respuesta al cliente	2ms	2ms
TOTAL	223.08ms	1006-1008ms

Comparativa modelos de replicación.- Servicio ejemplo S2

- Queremos desplegar un servicio con los siguientes valores
  - 5 réplicas
  - 1 Gbps de ancho de banda
  - 2ms para la propagación de cada mensaje
  - 5ms para tiempo de respuesta a las operaciones
    - no hay concurrencia interna: para N peticiones, tiempo de respuesta 5*N
  - 10 MB talla media de las modificaciones por operación
    - 10 MB = 0.80Gb (a 1Gbps la transferencia necesita 80ms)
    - tiempo transferencia modificaciones = 2ms (propagación) + 80 ms (transferencia) = 82 ms
  - 3 ms para aplicar las modificaciones en las réplicas secundarias

Comparativa modelos de replicación.- Despliega 5 servicios S2

Aspecto	Pasivo	Activo
envío petición	2ms	4-6 ms
procesamiento petición	5ms + 4*3=17ms	5*5ms =25ms
transferencia modificaciones	82ms	0ms
Aplicación modificaciones	3ms (secun.)	0ms
Respuesta a modificaciones	2ms	0ms
Respuesta al cliente	2ms	2ms
TOTAL	108ms	31-33ms

Comparativa modelos de replicación.- Análisis

- Los factores a considerar para elegir modelo son:
  - Tiempo medio de procesamiento de las peticiones
    - Activo apropiado si procesamiento breve
    - Pasivo apropiado si procesament costoso (la carga afecta únicamente a la réplica primaria)
  - Talla de las modificaciones
    - Activo apropiado para modificaciones grandes
    - Passvp apropiado para modificaciones pequeñas (requiere propagación)

Consistencia

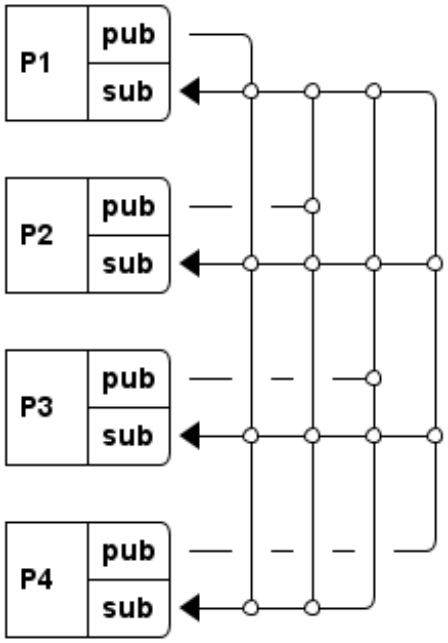
- Replicamos información en múltiples nodos
- Asumimos modelo Pasivo
  - Los clientes realizan la escritura inicialmente en un nodo
    - Dicho nodo propaga el resultado a las otras réplicas
    - La propagación No es instantánea (retardo)
  - Cada proceso lee un único node: lecturas locales
- Modelo de consistencia (centrada en datos) = especifica qué divergencias admite entre los valores de un mismo elemento (variable) en las réplicas
  - El algoritmo de consistencia controla las acciones de lectura y escritura
    - Si permite que lecturas y escituras retornen el control sin esperar la propagación $\rightarrow$ modelo rápido (consistencia débil)
    - Si no, modelo lento (consistencia fuerte)

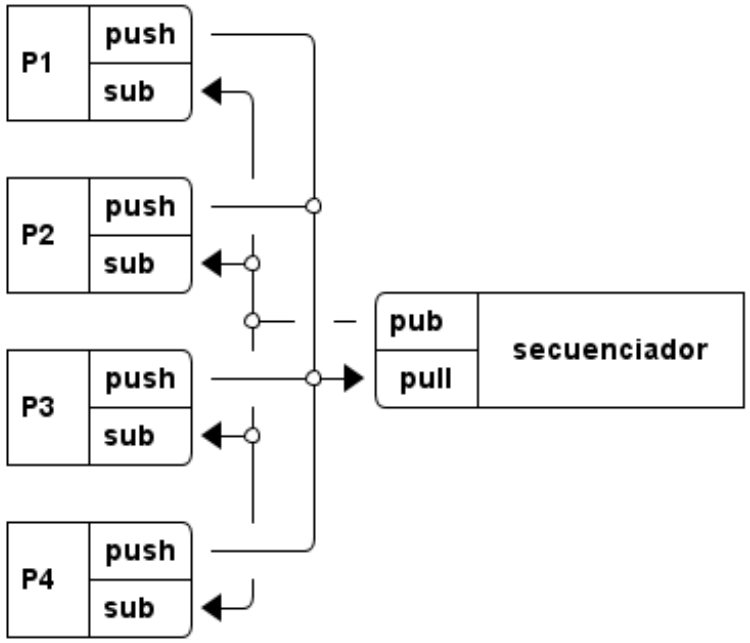
Ejemplo.- Lectures y escrituras

- Imagina nodos comunicados con sockets ZMQ
- Cuando un node escribe una variable (ex. `xWv` significa que en la var. x escribimos el valor v), envia mensaje al resto (propagación)
- Cuando un node recibe el mensaje en su socket sub, la llegada del mensaje corresponde a `xRv` = lee el valor v para x)

Ejemplo.- alternativas para la propagación

- pub/sub (figura izquierda)
  - `xWv` difunde el mensaje por su socket pub, y acaba
- secuenciador (figura derecha)
  - `xWv` envia mensaje al secuenciador, que difunde la operación
  - `xWv` acaba cuando el emisor recibe el mensaje de su propia operación



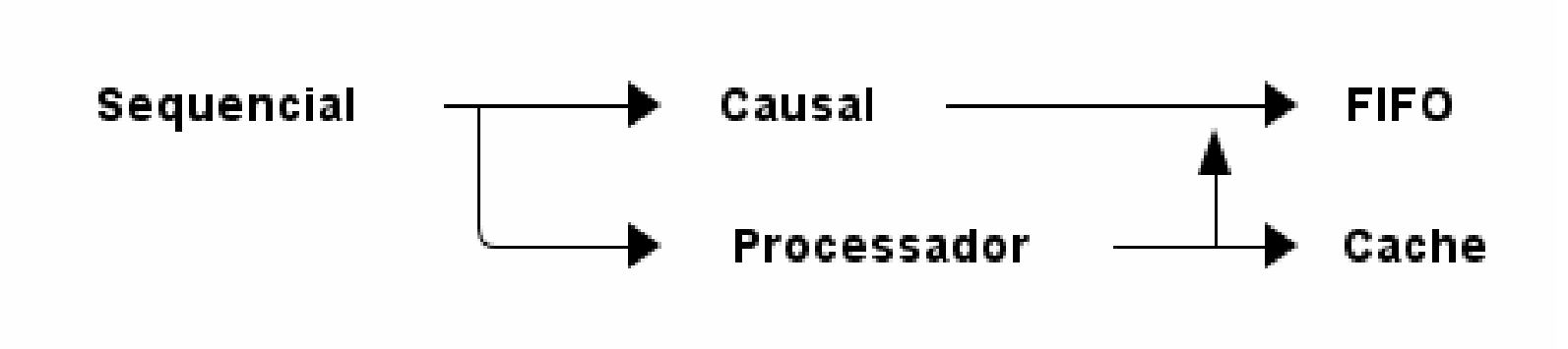


Ejemplo

- Los sockets tcp no pierden ni desordenan mensajes
  - Las escrituras del mismo nodo llegan a todos en el mismo orden
- En la figura de la izquierda
  - No sabemos el orden en que cada node intercala las escrituras de los pub
- En la figura de la derecha
  - Garantiza que todos reciben la misma secuencia de operaciones (mismo orden)

Modelos de consistencia centrada en datos

- Los mas importantes son:
  - teórico (no implantable): **Estricto**
  - Lentos: **secuencial**, **procesador**
  - Rápidos: **cache**, **causal**, **FIFO**



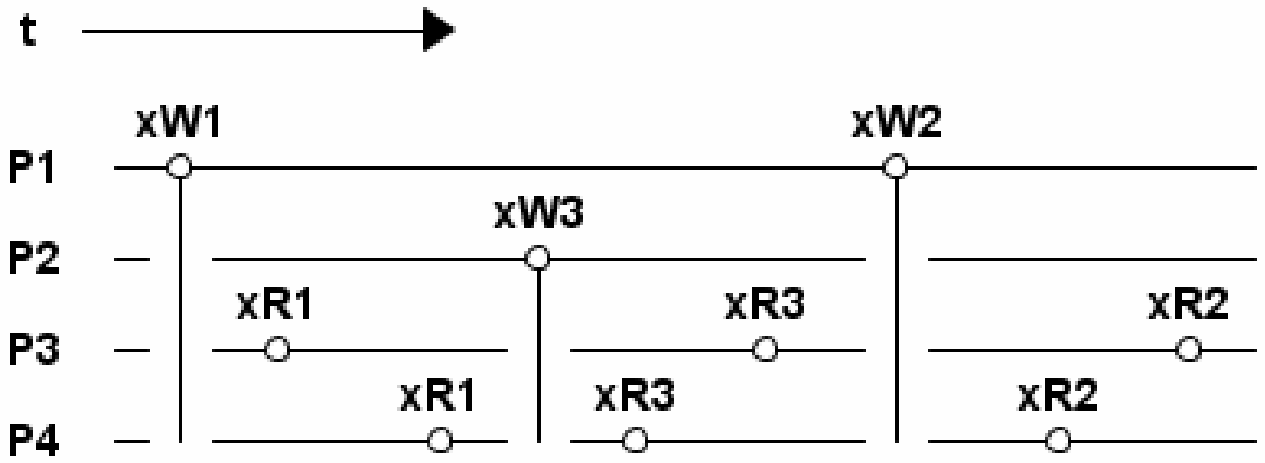
- Si se cumple secuencial, también Processador y Causal
- Si se cumple processador, también Caché
- Si se cumple Causal o Procesador, también FIFO
- Procesador cuando se cumplen simultáneamente los modelos Cache y FIFO

Consistencia final (eventual consistency)

- En los entornos escalables únicamente se exige que las réplicas converjan cuando haya intervalos prolongados sin escrituras
- Mientras hayan escrituras, cada réplica las admitirá sin preocuparse por lo que haga el resto
- Posteriormente se decide cómo intercalarlas y se acuerda qué valor final adoptar.
    - sencillo si las operaciones son conmutativas

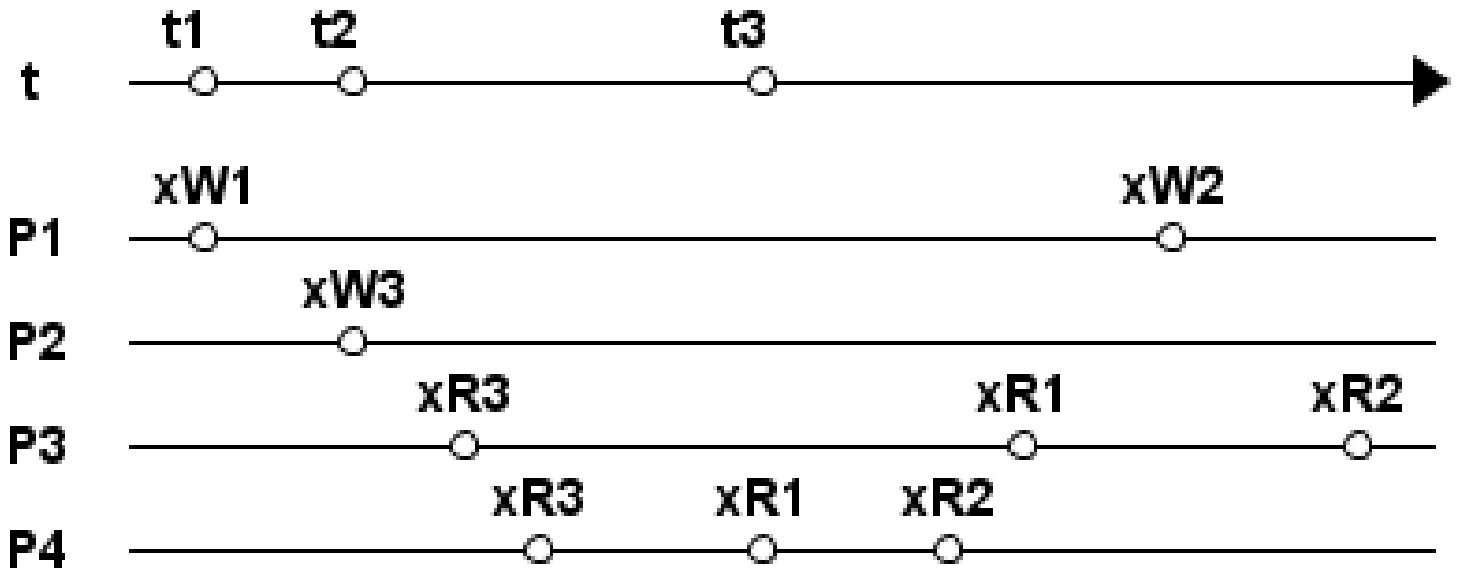
consistencia centrada en datos: Modelo Estricto

- Asume un reloj global que puede etiquetar cada evento $\longrightarrow$ No se puede implantar
  - No pueden ocurrir dos escrituras a la vez en todo el sistema
  - Propagación inmediata de las escrituras
  - Cada lectura siempre devuelve el valor de la última escritura realizada sobre esa variable



consistencia centrada en datos: Modelo Secuencial

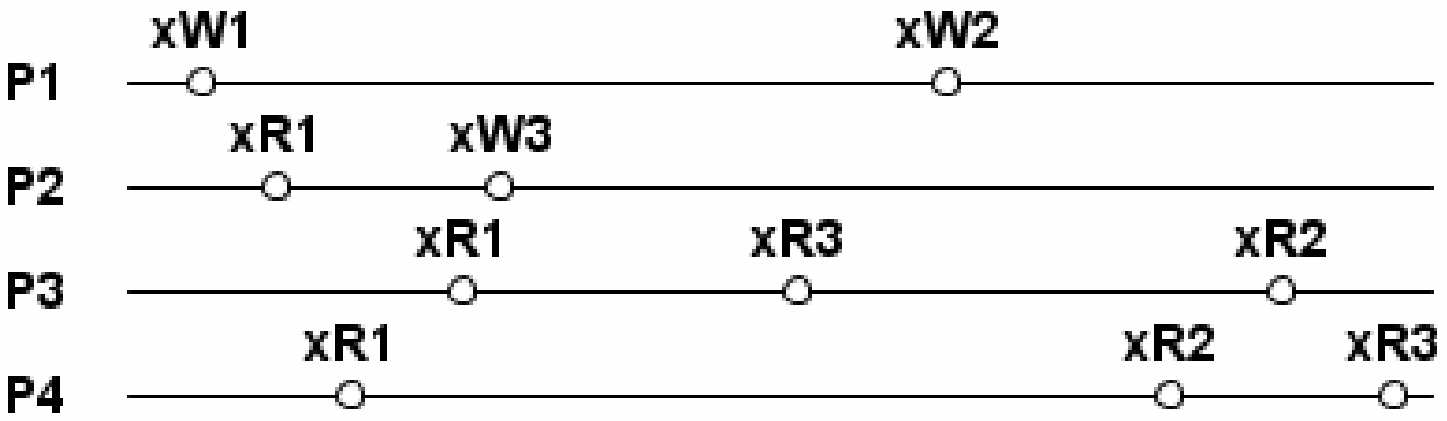
- Todos los procesos aplican las escrituras en el mismo orden, pero cada uno a su ritmo



- P3 y P4 aplican las escrituras en el mismo orden (xR3 xR1 xR2)
- NO es estricta: en t3 tendría que leer 3 (xR3)

consistencia centrada en datos: Modelo Causal

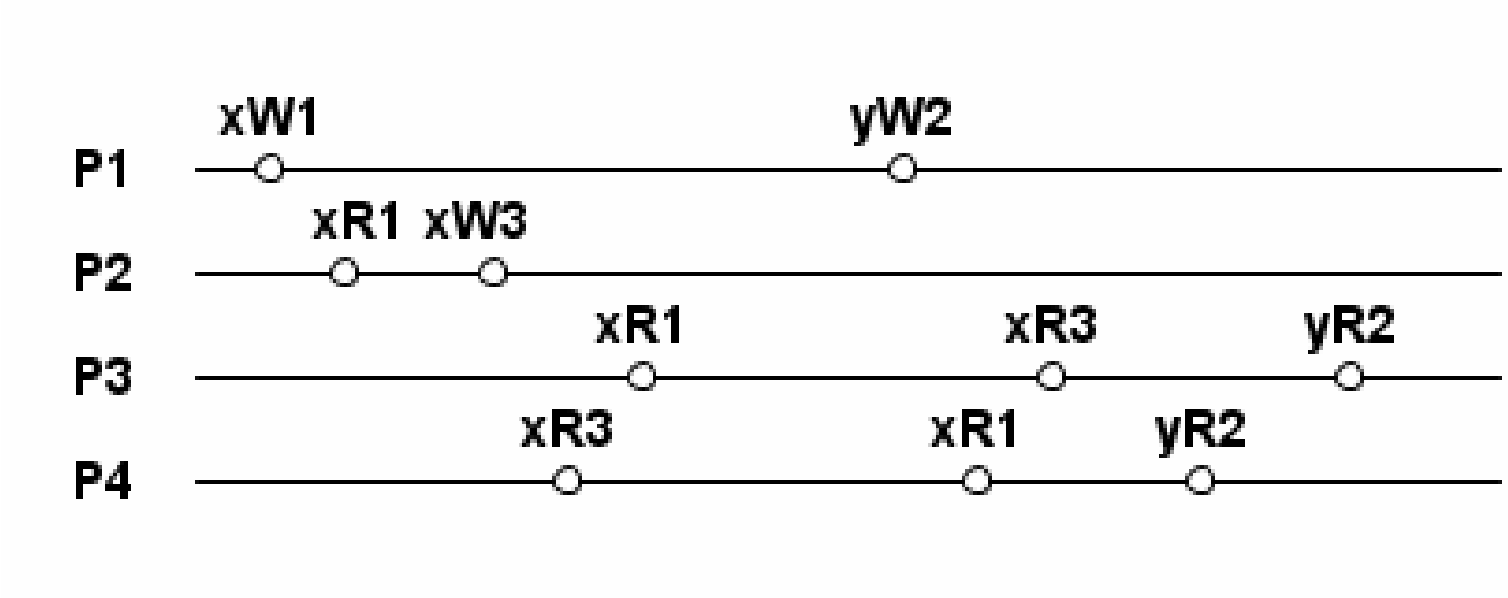
- Se respeta la relación de orden “happens before” (a $\rightarrow$ b) definida por Lamport
  - Utilizada para definir los relojes lógicos
  - En la consistencia causal se cumple xWa $\rightarrow$ xRa



- xW1 $\rightarrow$ xW2, xW1 $\rightarrow$ xW3
- xW2 || xW3 (los procesos pueden ordenarlas como quieran)
- NO es secuencial (orden xR1 xR3 xR2 en P3, orden xR1 xR2 xR3 en P4)

consistencia centrada en datos: Modelo FIFO

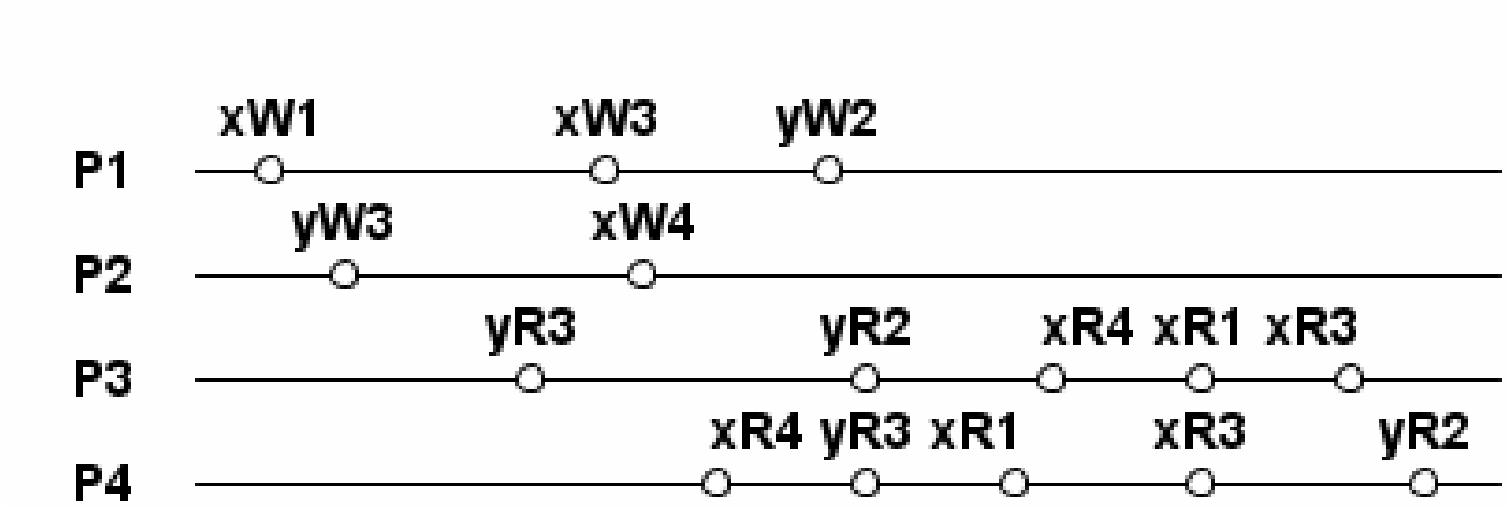
- Requiere que las escrituras realizadas por un proceso se lean en orden de escritura por el resto de procesos
- No impone restricciones a la hora de “mezclar” lo que han hecho distintos escritores



- La única restricción es que todos los lectors obtenen antes el valor 1 que el valor 2
  - Ya que ambos valores van ser escritos por P1
  - Pero el valor 3 puede obtenerse en cualquier punto de la secuencia

consistencia centrada en datos: Modelo Caché

- Las escrituras realizadas sobre una misma variable han de ser vistas en el mismo orden por todos los procesos (secuencial para cada variable)
    - Si un mismo proceso ha realizado varias escrituras sobre una misma variable, estarán en el orden en que las ha realizado el proceso
- NO impone restricciones para "mezclar" lo que se ha hecho sobre distintas variables



- En este ejemplo el orden de escritura sobre `x` es 4,1,3, sobre `y` es 3,2

Estrategia para determinar el modelo de consistencia a partir de la traza

- Descartar estricta (contraejemplo). Si es Estricta, hemos terminado
- Descartar secuencial (contraejemplo). Si es Secuencial, hemos terminado
- Descartar causal (contraejemplo). Si es Causal, hemos terminado
- Comprobar si cumple FIFO
- Comprobar si cumple Cache
    - Si cumple FIFO y Cache es Procesador