# SKIBIDI - RESUMEN - 11 - DISTRIBUIDOS

Esteban Ortega

Github: @EstebanKiiro

+569 91337509

@esteban...d...luffy

# TEMARIO

## 1 NODO PRINC: COMPUTADOR

- PROCESOS
- THREADS
- CONCURR y SINCRO

## 2 SISTEMAS DISTRIBUIDOS

- TIPOS
- ALTO RENDIMIENTO
- INFO DISTRIBUIDA
- SIST. PERVASIVOS

## 3 COMUNICACION NODOS

- TIPOS y PARADIGMAS
- PROTOCOLOS: TRANSP y APP
- COMUNIC. DIRECTA (SOCKETS)

## 4 COMUNICACIÓN NODOS II

- COMUNIC. INDIRECTA
- COLAS. Publish - subscribe
- INVOCACION. REMOTA: RPC's

## 5 COORDINACION y SINCRO

- ALGORITMOS
- SINC. RELOJES: CRISTIAN / BERKELEY / NTP
- RELOJES IAMPORT /

## 6 VECTORES LÓGICOS

- ESTADOS GLOBALES
- CHANDY IAMPORT

## 7 CONCENSO

- PAXOS
- GEN. BIZANTINOS

## 8 ELECCION LIDER

- CARACT.
- BULLY
- RAFT

## 9 EXCLUSION MUTUA

- ALGORITMOS
- SERVIDOR CENTRAL
- TOKEN RING
- MULTICAST

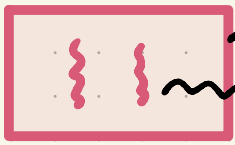
## CONTEXTO:

Fiabilidad, Escalab, Eficiencia

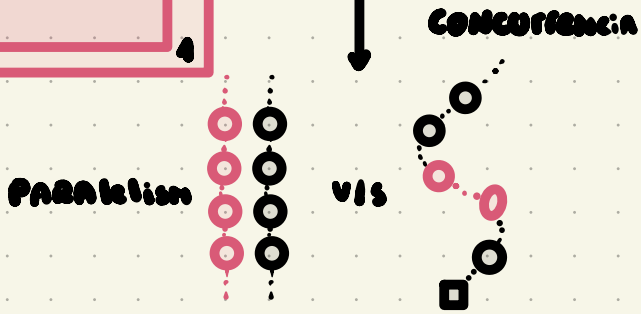
SIST. DIST → Colección Sist informáticos procesos y recursos se distribuyen en Diff COMPUTADORAS

- COMPLEJO → estados globales
- GESTIÓN FALLOS
- LATENCIAS
- PITFALLS → suposición falsa

**NODO PRINCIPAL : COMPUTADOR**



~ PROCESO
~ THREAD




- **Proceso:**
  - Propio espacio memoria
  - Necesitan coordinarse: mensajes
    - Sockets
    - semaforos
    - ...
  - costoso
- **THREAD:**
  - Comparte memoria
  - Vive dentro de un proceso
  - LIGERO
  - Mecanismos coordinación:
    - locks
    - semaforos
    - Eventos
    - join

- **Paralelismo:** EJEC SIMULTANEA
- **Concurrencia:** EJEC va ALTERNANDO

LOCKS: 1 thread se ADUEÑA
Pasadena: SACAR Numero y a fila
Semaforo: [0] thread 1 thread 2
Eventos: Event.wait() Event.set()
Join: Thread espera un tiempo X a otro Thread

## * Pregunta Prueba

Respecto a los procesos y *threads* ¿Cuál de las siguientes afirmaciones es **correcta**?

a. Los *threads* siempre serás más eficiente que los procesos porque trabajan en paralelo.

b. Los procesos no requieren de ningún intermediario para acceder y modificar archivos del disco duro, mientras que los *thread* si.

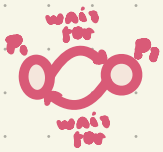
c. El fallo de un *thread* es igual al fallo de un proceso, afecta únicamente al que falló.

d. Tanto los procesos como los *threads* necesitan establecer reglas de comunicación para poder compartir la información entre ellos.

e. Los *threads* son más livianos de crear porque comparten el espacio de memoria del proceso que los creó.

R: (c)

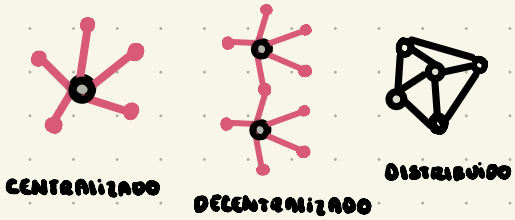
* DEADLOCKS:



# SIST. DISTRIBUIDOS

obj: MEJORAR RENDIMIENTO DE UN SIST. → "A LA VISTA DEL USUARIO": A SOLA COMPUTADORA

## Tipos Sistemas




DESCENTRALIZ: surge de NECESIDAD ← ADMINIST, CONFIANZA, GEOGRAFICA

DISTRIBUIDO: surge de DISEÑO → rendimiento

OBJETIVOS:
1. Compartir recursos
2. TRASPARENCIA
3. Apertura → facil de integrar
4. Confiabilidad → funcionar aun con fallos
5. Seguridad
6. Escalabilidad → seguir rindiendo aunque crezca

## Trasparencia:

(OCULTAR)

- ACCESO → recursos
- Ubicacion
- MIGRACION → sin interrupciones
- REPLICACION → multiples copias
- CONCURRENCIA
- FALLOS

## Escalabilidad:

- TAMAÑO → + users o recursos y buen rendimiento
- Geografica → + Distancia y buen rendimiento
- Funcional → + funcionalidades sin afectar rendimiento

## Tipos Sistemas

1. ALTO RENDIMIENTO: Cluster / Grid
2. INFORM DISTRIBUIDA: operabilidad entre sistemas → correos (Gmail, outlook ...)
3. PERVASIVOS:
   - Ubicuos: en el entorno (Smart tv nieve)
   - Movil
   - Sensores

Una empresa desarrolla un servicio de *streaming* de conciertos que promete funcionar 24/7 a los clientes, incluso si algún servidor se cae, el usuario será redirigido a otro para continuar con el servicio. Para esto, se tiene total confianza en que la comunicación entre los servidores será de la más rápida posible, tal que si cae un servidor, alcanzará a enviar el concierto a otro servidor para que el usuario lo siga viendo sin problemas.

¿Qué par objetivo-problemas de los sistemas distribuidos **ilustra correctamente** este escenario?

a. Compartir recursos - Problemas de latencia y ancho de banda.
b. Confiabilidad - *Pitfalls* de diseño.
c. Seguridad - *Pitfalls* de diseño.
d. Confiabilidad - Problemas de latencia y ancho de banda.
e. Escalabilidad - Complejidad inherente.

# COMUNICACION NODOS

## 1 TIPOS

- ↳ PERSISTENCIA:
  - TRANSITORIA: se pierden (manual)
  - PERSISTENTE: se almacenan (correo)
- ↳ SINCRONIA:
  - Sincronica: bloqueante (ej: request py)
  - Asincronica: no bloqueante (ej: email, zoom)

## 2 ACOPLAMIENTO

- ↳ TEMPORAL:
  - ACOPLADO: al mismo tiempo
  - DESACOPLADO: tiempos distintos
  - } client + recept
- ↳ ESPACIAL (referencial):
  - ACOPLADO: emisor NECESITA CONOCER RECEPTOR
  - DESACOPLADO: NO NECESITA ⇝ transmite por INTERMEDIARIO

## COMUNICACION RED 1

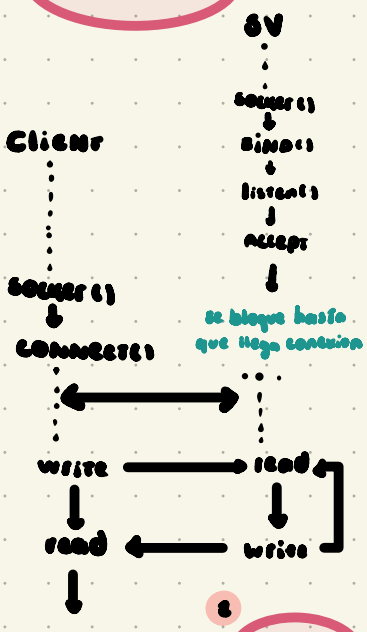
## 3 PARADIGMAS

- DIRECTA: + facil :) / :( acoplado REF AR
- INDIRECTA: INTERMEDIARIO comunica :( DR
- REMOTA: Dif maquinas

## COMUNICACION DIRECTA 3

↳ EMISOR: CONOCE el DESTINO
- DIR
- PUERTO

### 1 SOCKET



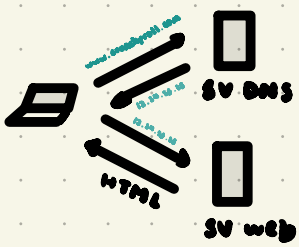

### 2 MPI

→ ESTANDAR EN C, DIST

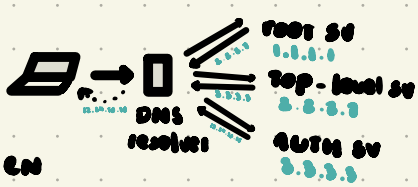
## PROTOCOLOS 2

### 2 APICACION

#### DNS




* respuest cacheadas
↳ sino, busqueda




### 1 TRANSPORTE

#### TCP




:) SEGURO
:( LENTO
- ↳ HTTP
- ↳ FTP
- ↳ SSH
- ↳ SMTP

#### UPP



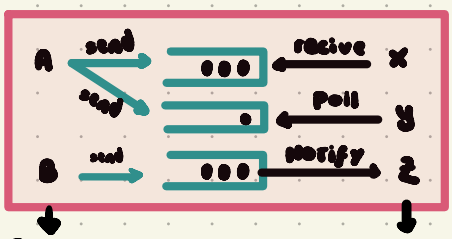

:( INSEGURO
:) RAPIDO
- ↳ DNS
- ↳ NTP

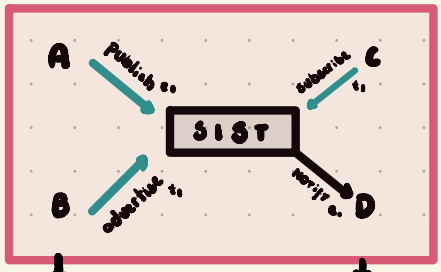
1 GRUPOS : emisor —msg→ proceso grupo —msg→ procesos miembros

2 MEM COMPARTIDA : sector donde TODOS LEEN y ESCRIBEN

3 COLAS MENSAJ

4 PUBLISH-SUBSCRIBE




1 DESACOPLE TEMPORAL

2 COLA → perder msgs

- receive : esperar hasta msg encolado
- Poll : verifica si hay → saca
- Notify : controlador llama a consumer

* Si SIST Asegura persist
⇒ Asegura desacople Temporal

HETEROGENEO : comunica componentes que no estan disen para trabajar juntos

INTERMEDIARIO

COMUNICACIÓN INDIRECTA

(SIN ACOPLAM. REF)

# COMUNICACION NODOS II

INVOCACION REMOTA

↳ EJECUTAR METODOS en OTRA MAQUINA




* SINCRONO (Bloqueante)

* Acople Temporal

RPC

- TRASPARENCIA

ACCESO y UBICACIÓN

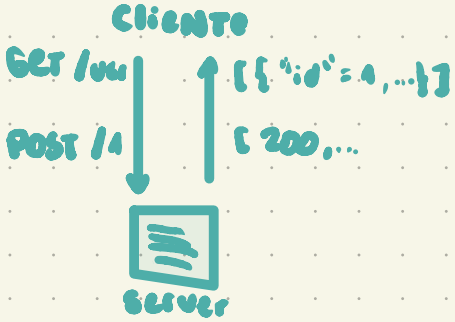
REST

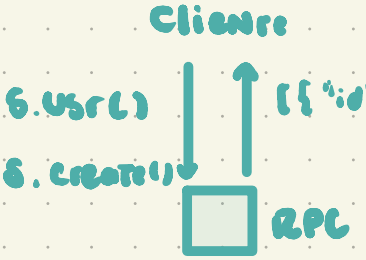
- HTTP Protocol
- CADA EJEC STATELESS

(NO recuerda Anteriores)

Client STUB — SERVER STUB

INTERPRETAN msgs




* RMI : rpc OOP

# UNIDAD 2

## COORDINACION y SINCRONIZACIÓN

- **Asegurar ORDEN TEMPORAL de EVENTOS** ← ?
- **SINC. RELOJES**
  - **FISICOS:** ✱ CUARZO (variación 31.5 seg por año) ✱ ATOMICOS: ✓
  - **LOGICOS:** ORDEN DE EVENTOS
  - **ALGORITMOS**

### LAMPORT

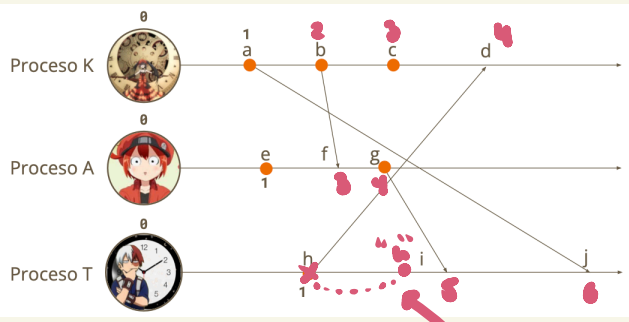
↳ orden PARCIAL de EVENTOS
- IDENTIF: EVENT CONCURRENTES

Proceso $P_i$,
Tiene su reloj $C(P_i)$

**REGLAS**

$R_1$: Ocurre evento en $P_2 \rightarrow C(P_2) += 1$

$R_2$: $P_i$ envía mensj → se adjunta $C(P_i)$
Se actualiza $C(P_2)$: $\max(C(P_i), C(P_2)) + 1$




$C(B) < C(j) \Rightarrow b \rightarrow j$

h ocurre ANTES que b,c,f,g?

¡No necesariamente!

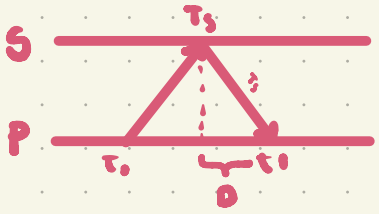
### CRISTIAN → BERKELEY → NTP

↳ UN SERVIDOR AJUSTA SU RELOJ SEGUN OTRO

↳ Existe DELAY

1. $D_{IDA} = D_{VUELTA}$
2. "RESPUESTA INMEDIATA"

$$T_p = T_s + D$$

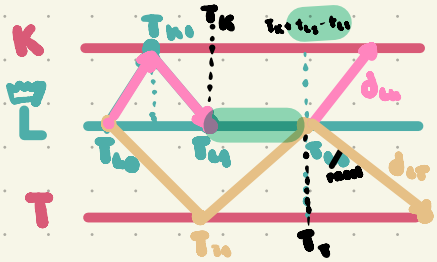



$$T_p = T_s + \frac{(t_1 - t_0)}{2}$$

↳ USA CRISTIAN UN NODO LIDER

↳ CALC. PROMEDIO y

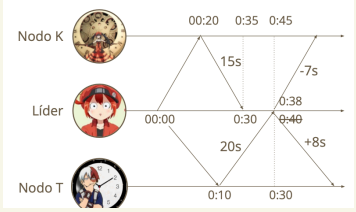
1. AJUSTA SU RELOJ
2. ENVIA T a AJUSTAR



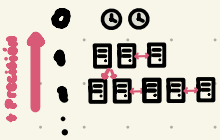

$T_K = T_{K_0} + (T_{K_1} - T_{K_0})/2$

$T_T = T_{T_0} + (T_{K_1} - T_{K_0})/2$

$T_{t_2} \rightarrow$ T PROM

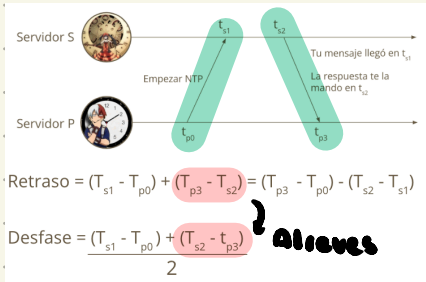


PREVIOUSLY : **NTP**




- RETRASO : Tiempo IDA y VUELTA de 2 msgs
- DESFASE : DESPLAZ entre relojes

NTP realiza esto muchas veces y ELIGE el MENOR RETRASO para aplicar su DESFASE al servidor

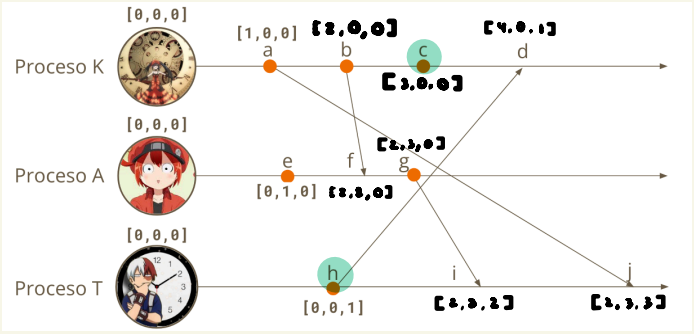



# VECTORES LOGICOS y ESTADOS GLOBALES

## VECTORES

**Relojes Vectoriales - Reglas**

- Cuando pasa un evento en el proceso $P_i$, se aumenta en una unidad el $V_i[i]$.
- Un evento es algo ocurrido únicamente en el proceso, el envío de un mensaje a otro proceso o la recepción de un mensaje.
- Cuando el proceso $P_i$ recibe un vector $V_z$, primero se aumenta en una unidad el $V_i[i]$. Luego, se actualiza el vector $V_i$ del siguiente modo:
  - $V_i[x] = \max(V_i[x], V_z[x])$ para $x = 1,2,...,N$ → componente a componente deja los maximos
- Con esto, se puede comparar 2 eventos (**a** y **b**) según sus vectores.
  - $V^a_i < V^b_k$ El evento b en $P_k$ ocurrió después que el evento a en $P_i$
  - En otro caso, el evento a y b son "concurrentes".




¿Qué podemos decir del evento h vs c?
**Son concurrentes**

¿Y del evento a vs i?
**a ocurre antes que i**

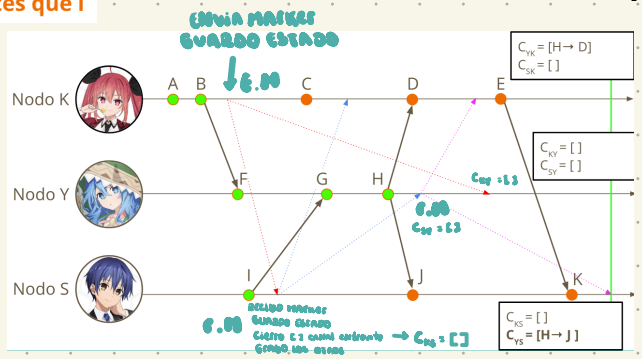
## EST. GLOBALES

- Consiste en los ESTADOS LOCALES y los MENSAJES en TRANSITO

## ALG CHANDY LAMPORT

1. NODO $P_x$ INICIA Algoritmo
   - GUARDA ESTADO LOCAL
   - ENVIA MARKER a TODOS
   - EMPIEZA A GRABAR MENSAJES

2. CASO 1 : $P_y$ recibe por primera vez MARCADOR de $P_z$
   - GUARDA ESTADO LOCAL
   - ENVIA MARKER a TODOS CANALES
   - $C_{zy}$ queda VACÍO
   - EMPIEZA A GRABAR MENSAJES EN LOS DEMÁS CANALES



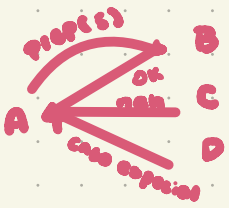
MUITIPLES NODOS ← ? **CONCENSO** → AGREEMENT : de todos los nodos
TERINATION : ACEPTAN OP
FAUIT TOLERANCE

ACUERDAN DESICIONES
(AUN si hay fallos)

**CONCENSO** → **AIGORITMOS**

## PAXOS

1. Proponente : Propone Accion a consolidar
2. Aceptantes: Aceptan Accion y GUARDAN LOCAL
3. aprendiz : consolidan si +50% NODOS : misma Accion




### PAXOS

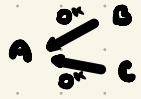
Garantiza Tolerancia a FALLOS
$(n-1)/2$ NODOS pueden FALLAR
Si la mayoría está ⇒ FUNCIONA

ok : Primer Prepare
nah : ANTES Acepto $n_2$ y $n_2 > n$
caso * : ANTES Acepto $n_2$ y $n_2 < n$

↳ COMPARTE la Accion + ID




FALLIDA



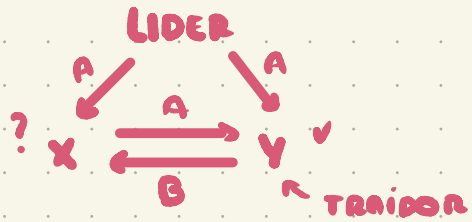

EXITOSA




### CASOS

1. INTERRUMPIR :  consolida $A_2$
2. SOBRESCRIBIR :  consolida $A_2$
3. LIVELOCK : 

## BIZANTINOS




### SOLUCIONES

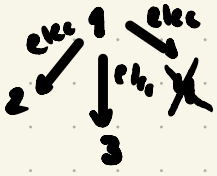
1. OM : requiere $3n+1$ NODOS para por mayoría elegir msg
2. SM : El LIDER FIRMA el msg
   * Permite ENCONTRAR TRAIDORES LIDERES
3. Proof of Work : computan Problemas y los mas rapidos deciden
4. Proof of Stake

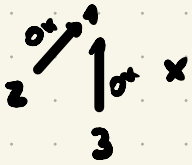
# ELECCION LIDER

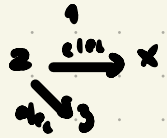
- Safety: NUNCA 2 lideres
- Liveness: TODOS participan

## BULLY

- ELECTION
- OK
- COORDINATOR



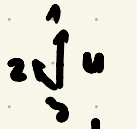









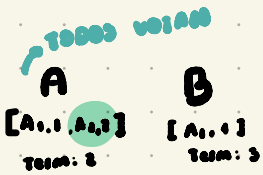

COORDINATOR

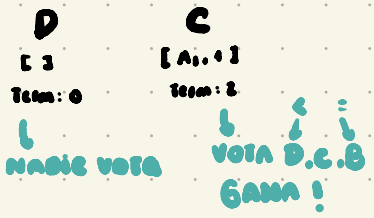
3: LIDER

## RAFT

- SEGUIDOR
- CANDIDATO
- LIDER

**FASE 1**: Democracia




**FASE 2**: Dictadura

- LIDER replica sus logs
  - ↳ SOBREESCRIBE !!!

CONSOLIDACION: +50%

### RAFT

Garantiza Tolerancia a FALLOS
(n-1)/2 NODOS pueden FALLAR
Si la mayoría está ⇒ FUNCIONA

## RING-BASED




Si RECIBO
Mi ID
(soy LIDER)

ENVIO ELECTED = 4

* **Peor caso**:

3n - 1 MENSAJES

ID MAYOR: UNO ANTES del que inicia la elección

# PREGUNTAS DE CIASES

**1** Una empresa desarrolla un servicio de *streaming* de conciertos que promete funcionar 24/7 a los clientes, incluso si algún servidor se cae, el usuario será redirigido a otro para continuar con el servicio. Para esto, se tiene total confianza en que la comunicación entre los servidores será de la más rápida posible, tal que si cae un servidor, alcanzará a enviar el concierto a otro servidor para que el usuario lo siga viendo sin problemas.

¿Qué par objetivo-problemas de los sistemas distribuidos **ilustra correctamente** este escenario?

a. Compartir recursos - Problemas de latencia y ancho de banda.
b. Confiabilidad - *Pitfalls* de diseño.
c. Seguridad - *Pitfalls* de diseño.
d. Confiabilidad - Problemas de latencia y ancho de banda.
e. Escalabilidad - Complejidad inherente.

**2** **Supongamos** que su profesor hará una conferencia por Zoom para hablar de anime. Pero, como no tiene Zoom Pro, no va a grabar esa conferencia. Además, para mantener la privacidad de usuarios, Zoom será un intermediario que recibirá la información y reenviará la transmisión a todos los presentes sin que el profesor sepa realmente sus identidades.

¿Cuál de las siguientes afirmaciones describe de forma más precisa la naturaleza del intercambio de información?

a) Es una comunicación persistente, asincrónica y acoplado referencialmente.

b) Es una comunicación acoplada referencialmente y utiliza protocolo TCP para el envío de cada *frame*.

c) Es una comunicación transitoria, asincrónica y acoplada temporal.

d) Es una comunicación desacoplada temporal y referencialmente.

e) Es una comunicación persistente y que utiliza el protocolo UDP para el envío de cada *frame*.

**3** Un sistema de vigilancia está compuesto por múltiples cámaras que reportan datos en tiempo real a través de RPC hacia un servidor. Este servidor luego reenvía los datos a distintos clientes mediante un sistema de colas. ¿Qué **ventaja clave aporta** este diseño híbrido?

a. Aumentar la latencia en el traspaso de información.
b. Garantiza que los sensores puedan comunicarse incluso si el servidor está inactivo.
c. Asegura que todos los clientes reciban todos los datos al mismo tiempo.
d. Permitir que los clientes reciben los datos cuando ellos quieran.
e. Disminuir la latencia en el traspaso de información.

**4** ¿Con cuál de los siguientes mecanismos **es más seguro** que los nodos queden sincronizados lo más cercano al Tiempo Universal Coordinado (UTC)?

a. Relojes lógicos de Lamport.
b. Algoritmo de Berkeley.
c. Protocolo de Tiempo de Red.
d. Algoritmo de Raft.
e. Método de Cristian.

RESPUESTAS: 1.b 2.c 3.d 4.c