Relojes de Lamport

GOMEZ EDGAR, Universidad Central de Venezuela, Venezuela

Los relojes lógicos de Lamport constituyen un mecanismo fundamental para el ordenamiento causal en sistemas distribuidos, permitiendo establecer relaciones de precedencia entre eventos sin requerir sincronización temporal física. Este documento presenta un ejercicio práctico que ilustra la aplicación paso a paso del algoritmo de Lamport en un escenario de comunicación entre tres nodos (A, B y C). A través de la secuencia de eventos propuesta, donde los mensajes M_1 a M_7 se intercambian siguiendo reglas específicas de envío y recepción, se demuestra cómo los contadores lógicos se actualizan mediante las operaciones de incremento local y maximización al recibir mensajes. La solución detallada muestra la evolución de los relojes lógicos desde su inicialización en cero hasta alcanzar valores finales ($A = 9$, $B = 10$, $C = 6$), destacando cómo el algoritmo maneja situaciones donde el orden físico de recepción difiere del orden causal (como la recepción de M_2 antes que M_1). Los resultados finales proveen las marcas temporales asignadas a cada mensaje durante su envío, ofreciendo una comprensión clara de la implementación y utilidad de los relojes de Lamport para garantizar la coherencia lógica en sistemas distribuidos asíncronos.

CCS Concepts: • **Organización de sistemas informáticos → Sistemas Distribuidos**; *Relojes de Lamport*.

Additional Key Words and Phrases: Sistemas Distribuidos, Asincronía

ACM Reference Format:
GOMEZ EDGAR. 2026. Relojes de Lamport. *Proc. ACM Hum.-Comput. Interact.* 1, 1, Article 1 (January 2026), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1. Introduction

Los relojes o tiempos lógicos de Lamport, propuestos por Leslie Lamport en 1978, son un mecanismo utilizado en sistemas distribuidos para ordenar eventos causalmente sin requerir sincronización física exacta. Utilizan contadores que se incrementan en cada evento y se actualizan al recibir mensajes, estableciendo una relación de sucesos.

El algoritmo sigue las siguientes reglas:

- Un proceso incrementa su contador antes de cada evento que ocurra en ese proceso. - Cuando un proceso envía un mensaje, este incluye su contador en el envío. - Al recibir un mensaje, se actualiza el contador del receptor si es necesario, al mayor entre su propio contador y la marca de tiempo recibida en dicho mensaje.

En pseudocódigo el algoritmo para enviar un mensaje es:

```
reloj = reloj + 1;
marca_temporal_mensaje = reloj;
enviar(mensaje, marca_temporal_mensaje);
```

Algoritmo para la recepción del mensaje:

```
recibir() = (mensaje, marca_temporal_mensaje);
reloj = max(marca_temporal_mensaje,reloj) + 1;
```

Author's Contact Information: GOMEZ EDGAR, Universidad Central de Venezuela, Caracas, Venezuela.

2026. ACM 2573-0142/2026/1-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2. Ejercicio de Funcionamiento de relojes de Lamport

Reloj lógico de Lamport: Tres computadoras, A, B y C, se comunican mediante un protocolo que implementa relojes lógicos de Lamport (incluyen la hora de su reloj en los mensajes). Al principio del tiempo, las tres computadoras comienzan con su reloj lógico a cero. Posteriormente, se produce la siguiente secuencia de eventos:

- A envía el mensaje M1 a B
- C envía el mensaje M2 a B
- B recibe M2 antes que M1
- B responde primero el mensaje M3 a C y luego el mensaje M4 a A
- Tras recibir el mensaje M3, C envía el mensaje M5 a A
- Tras recibir el mensaje M5, A envía el mensaje M6 a B
- Tras recibir el mensaje M6, B envía el mensaje M7 a A
- El último mensaje que A recibe es M4

* Indique la hora que acompaña a los mensajes a medida que se envían en cada paso.

- Enviar(M1, _)
- Enviar(M2, _)
- Enviar(M3, _)
- Enviar(M4, _)
- Enviar(M5, _)
- Enviar(M6, _)
- Enviar(M7, _)

2.1. Solución

- **Regla 1:** Antes de enviar un mensaje, se incrementa el contador local en 1 y se envía ese valor con el mensaje.
- **Regla 2:** Al recibir un mensaje con tiempo t_m, se actualiza el contador local:

$$t_{\text{nuevo}} = \max(t_{\text{local}}, t_m) + 1$$

1. Inicializacion de relojes en cero
   Estado: A=0, B=0, C=0
2. Primer evento: A envía M1 a B
   - A incrementa: A = 0 + 1 = 1
   - Envia M1 con tiempo = 1

   ENVIAR(M1, 1)
   Estado: A=1, B=0, C=0
3. Segundo evento: C envía M2 a B
   - C incrementa: C = 0 + 1 = 1
   - Envia M2 con tiempo = 1

   ENVIAR(M2, 1)
   Estado: A=1, B=0, C=1
4. Tercer evento: B recibe M2 antes que M1
   - B recibe M2 con tiempo =1
   - B = max(0,1) + 1 = 2

   Estado: A=1, B=2, C=1
5. Cuarto evento: B recibe M1
   - B recibe M1 con tiempo =1

- B = max(2,1) + 1 = 3

Estado: A=1, B=3, C=1

6. Quinto evento: B envía M3 a C
   - B incrementa: B = 3 + 1 = 4
   - Envia M3 con tiempo = 4

   ENVIAR(M3, 4)
   Estado: A=1, B=4, C=1
7. Sexto evento: B envía M4 a A
   - B incrementa: B = 4 + 1 = 5
   - Envia M4 con tiempo = 5

   ENVIAR(M4, 5)
   Estado: A=1, B=5, C=1
8. Septimo evento: C recibe M3
   - M3 llega con tiempo = 4
   - C = max(1,4) + 1 = 5

   Estado: A=1, B=5, C=5
9. Octavo evento: C envía M5 a A
   - C incrementa: C = 5 + 1 = 6
   - Envia M5 con tiempo = 6

   ENVIAR(M5, 6)
   Estado: A=1, B=5, C=6
10. Noveno evento: A recibe M5
    - M5 llega con tiempo = 6
    - A = max(1,6) + 1 = 7

    Estado: A=7, B=5, C=6
11. Decimo evento: A envía M6 a B
    - A incrementa: A = 7 + 1 = 8
    - Envia M6 con tiempo = 8

    ENVIAR(M6, 8)
    Estado: A=8, B=5, C=6
12. Undécimo evento: B recibe M6
    - M6 llega con tiempo = 8
    - A = max(5,8) + 1 = 9

    Estado: A=8, B=9, C=6
13. Duodécimo evento: B envía M7 a A
    - B incrementa: B = 9 + 1 = 10
    - Envia M7 con tiempo = 10

    ENVIAR(M7, 10)
    Estado: A=8, B=10, C=6
14. Decimotercer evento: A recibe M4
    - M4 llega con tiempo = 5
    - A = max(8,5) + 1 = 9

    Estado: A=9, B=10, C=6

* Hora que acompaña a los mensajes a medida que se envían en cada paso.

- Enviar(M1, 1)

- Enviar(M2, 1)
- Enviar(M3, 4)
- Enviar(M4, 5)
- Enviar(M5, 6)
- Enviar(M6, 8)
- Enviar(M7, 10)

3. Conclusión

La implementación práctica del algoritmo de Lamport demostró su eficacia para establecer un orden causal lógico en sistemas distribuidos sin requerir sincronización temporal física. A través del ejercicio con tres procesos, se verificó que las reglas básicas de incremento local y maximización al recibir mensajes garantizan la consistencia del tiempo lógico, incluso cuando el orden físico de recepción difiere del orden de envío. Los resultados obtenidos (M1=1, M2=1, M3=4, M4=5, M5=6, M6=8, M7=10) validan que el algoritmo proporciona un mecanismo sencillo pero robusto para rastrear relaciones de precedencia, constituyendo así la base fundamental para técnicas más avanzadas de sincronización en sistemas distribuidos asíncronos.

Referencias

[1] Wikipedia. *Tiempos lógicos de Lamport*. Recuperado de https://es.wikipedia.org/wiki/Tiempos_l%C3%B3gicos_de_Lamport. Último acceso: 25 de enero de 2026.