Este laboratório prático explorou diferentes mecanismos de tempo em um sistema distribuído simulado.

1. Foi implementada uma simulação em Java com 3 processos  que trocam mensagens entre si para representar um sistema distribuído.
2. Na primeira parte, foram utilizados relógios físicos (System.currentTimeMillis()) para registrar os eventos, com o objetivo de observar possíveis inconsistências na ordem temporal.
3. Na segunda parte, foi implementado o relógio lógico de Lamport para garantir a consistência causal, onde o timestamp de um recebimento é sempre maior que o do envio correspondente.
4. Na terceira e última parte, foram implementados relógios vetoriais, que, além de garantirem a causalidade, permitem determinar se dois eventos são concorrentes.

**Resultados da Simulação com Relógios Vetoriais**

--- Iniciando Simulação com Relógio: VETORIAL ---
Processo P0 | Evento: ENVIOU para P1 | Relógio (VETORIAL): [2, 0, 0]
Processo P1 | Evento: RECEBEU de P? | Relógio (VETORIAL): [2, 2, 0]
Processo P2 | Evento: ENVIOU para P0 | Relógio (VETORIAL): [0, 0, 2]
Processo P0 | Evento: RECEBEU de P? | Relógio (VETORIAL): [3, 0, 2]
Processo P1 | Evento: ENVIOU para P2 | Relógio (VETORIAL): [2, 3, 0]
Processo P2 | Evento: RECEBEU de P? | Relógio (VETORIAL): [2, 3, 3]

**Análise e Conclusão**

A prática demonstrou que relógios físicos não são confiáveis para ordenar eventos em sistemas distribuídos. O relógio de Lamport resolveu o problema da ordem causal, enquanto o relógio vetorial se mostrou o mais completo, permitindo também a identificação de eventos concorrentes (ex: [2, 0, 0] e [0, 0, 2]). Conclui-se que a escolha do mecanismo de relógio é fundamental para garantir a consistência e a corretude de um sistema distribuído.