Sistemas Distribuídos

Sincronização

António Rui Borges

Resumo

- *Conceitos de tempo*
  - *Hora global*
  - *Horário local*
  - *Tempo lógico*
- *Ajuste da hora local*
  - *Caracterização do problema*
  - *Método cristão*
  - *Algoritmo de Berkeley*
  - *Protocolo de horário de rede*
- *Relógios lógicos*
  - *Relógio lógico escalar*
    - *Ordenação total de eventos*
  - *Relógios lógicos vetoriais*
- *Leitura sugerida*

Conceitos de tempo

Os fenômenos físicos, e em particular a atividade humana, ocorrem no espaço e no tempo. *Tempo* é o elemento chave na caracterização dos eventos, na sua ordenação e na determinação de possíveis relações causais que possam existir entre pares, ou grupos, de eventos.

Tempo em si é pensado de múltiplas maneiras em relação a como o *observador* enfrenta a realidade

- *hora mundial* – tempo percebido por um observador externo
- *horário local* – tempo conforme percebido por cada uma das entidades observadas
- *tempo lógico* – tempo conforme percebido pelo fluxo de informações.

Hora global - 1

Tempo não pode ser medido diretamente.

Sua medição é feita observando o movimento periódico de objetos bem definidos, aproveitando a íntima conexão de *tempo* com *espaço*

- *hora astronômica* – baseado no movimento dos corpos astrais nos céus: especificamente, o movimento circular da Terra em torno do Sol (cada ciclo representa um *ano solar*) e o movimento de rotação da Terra (cada ciclo representa um *dia solar*); o *dia solar* é sucessivamente dividido em horas, minutos e segundos (um *dia* compreende 24 horas, uma hora e 60 minutos e um minuto e 60 segundos); o *segundo solar*, enquanto unidade padrão para medição de tempo, é definida como a fração de 1/86400 do dia solar; e a *ano solar* como sendo aproximadamente 365 dias e 6 horas

Hora global - 2

- *tempo atômico*–o movimento periódico dos corpos astrais não é suficientemente constante para ser usado como padrão quando se consideram intervalos de tempo muito longos; pesquisas realizadas no última década possuir isso o movimento de rotação da Terra tem tornam-se mais lentos com o passar do tempo devido ao atrito produzido pelas marés e ao efeito de arrasto da atmosfera; além disso, o movimento de rotação da Terra é propenso a pequenas oscilações em sua velocidade angular devido, acredita-se, à turbulência no núcleo; por todas estas razões, o*segundo padrão*foi redefinido quando os relógios atômicos surgiram; a definição atual afirma que*segundo*ser igual a*a duração de 9 192 631 770 períodos de radiação correspondentes à transição entre dois níveis hiperfinos do estado de energia mínima do átomo césio 133, em repouso e à temperatura de 0 K*.

Hora global - 3

Hora Atômica Internacional (TAI) representa o padrão prático calculado a partir da média ponderada do tempo medido por mais de 200 relógios atômicos localizados em instituições nacionais ao redor do mundo. Esta tarefa é coordenada pelo Instituto Internacional de Pesos e Medidas (BIPM), através do seu Bureau Internacional da Hora (BIH).

A duração do *segundo padrão* foi criada para garantir a sua coincidência com o *segundo solar* no momento de sua introdução, 1º de janeiro de 1958.

Em 1º de janeiro de 1977, foi introduzida uma correção na convenção, baseada na relatividade geral, para minimizar o efeito de dilatação do tempo produzido pelas variações do campo gravitacional da Terra (os relógios atômicos, usados para calcular a média, estão localizados em diferentes altitudes e , portanto, marque em taxas ligeiramente diferentes).

Hora global - 4

Tempo Universal Coordenado (UTC), com base no *hora atômica internacional*, constitui o principal padrão que define o tempo para as atividades humanas. O sistema é baseado em *segundos padrão* e é mantido o mais próximo possível do *hora astronômica* adicionando ou subtraindo (o último ainda não aconteceu) segundos individuais para compensar a desaceleração e irregularidades ocasionais do movimento de rotação da Terra.

Na escala de tempo UTC, o *segundo* e seus submúltiplos são sempre constantes; seus múltiplos, *dias*, *horas* e *minutos*, no entanto, não são. De tempos em tempos, para manter o ritmo com o *hora astronômica*, um segundo é adicionado ou subtraído ao último minuto de um dia, normalmente no último dia de junho ou dezembro.

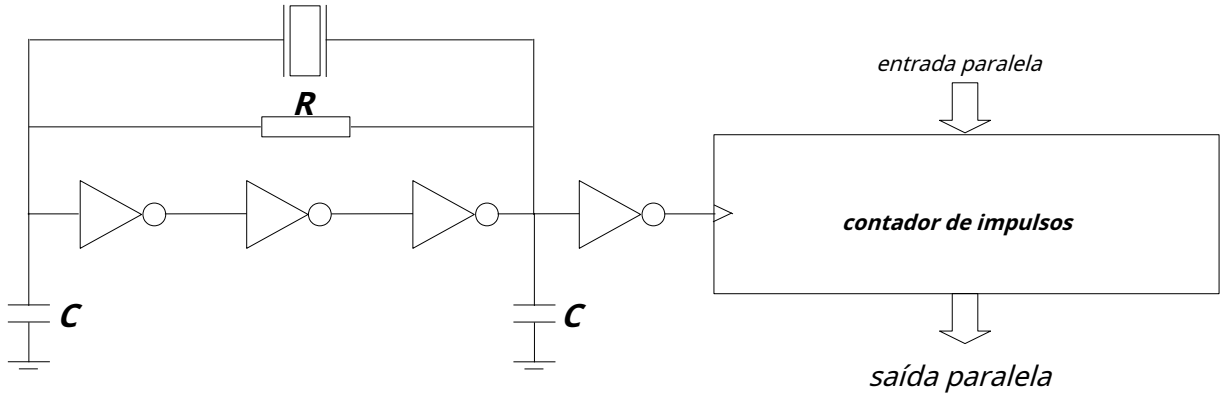
A hora UTC é disponibilizada por diversas fontes, com diferentes incertezas. Entre os mais importantes, estão

- emissores de rádio de ondas curtas: ±10 ms
- sistemas de satélite geoestacionários (GEOS, GPS): ±0,5 ms
- Internet (NTP): ±50 ms.

Hora local – 1

O *relógio* de um sistema computacional consiste em dois elementos: um circuito oscilador, controlado por um cristal de quartzo, e um contador de impulsos. O valor de contagem em um determinado instante pode ser obtido pela leitura do *saída paralela* e o contador pode ser definido para um determinado estado alimentando o *entrada paralela*.

A contagem pode ser convertida em *tempo* desde que uma origem seja definida (a convenção no Unix, por exemplo, corresponde à origem a 0h 0m 0s de 1º de janeiro de 1970).

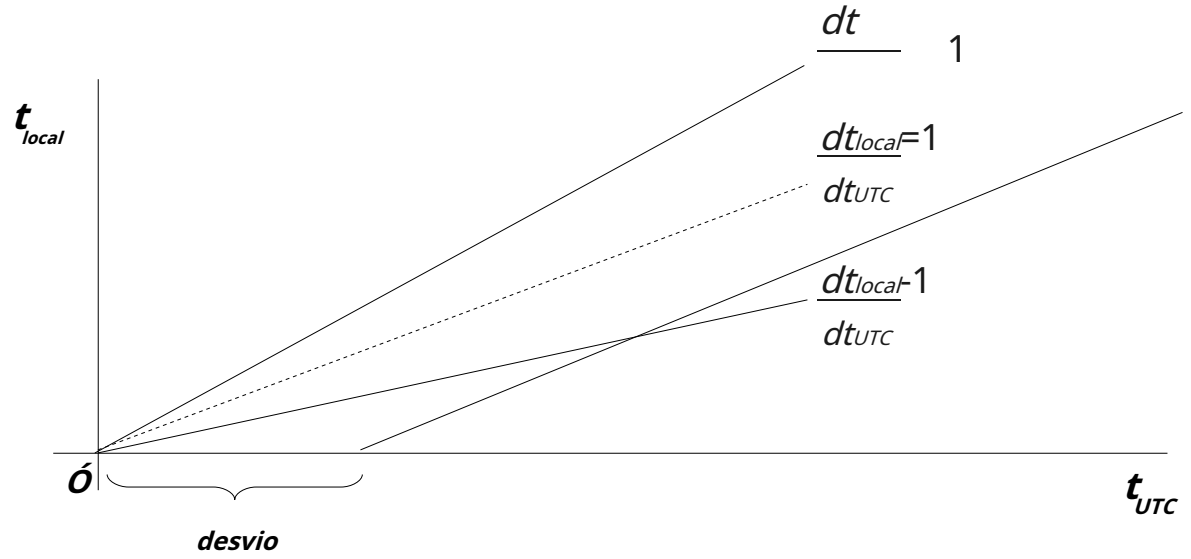


Hora local – 2

O *relógio* pode exibir uma hora incorreta devido a dois fatores diferentes

- *desvio* – o valor de contagem difere do valor correto por um número fixo de impulsos (erro na definição da origem)
- *deriva* – a frequência do oscilador se afasta da nominal, mudando seu valor erraticamente com o passar do tempo devido às condições ambientais, como temperatura e umidade (erro na taxa de contagem).

Normalmente, um oscilador, controlado por um cristal de quartzo, tem um *deriva* da ordem de grandeza de uma parte em 10 por segundo.



Ajuste de tempo – 1

O problema de sincronização dos relógios locais dos sistemas computacionais que compõem os nós de processamento de uma máquina paralela pode ser pensado de duas maneiras diferentes

- *sincronização externa*–dada uma fonte UTC conhecida, $S(t)$, e um intervalo máximo de incerteza tolerada, Δ, entre os relógios locais $Ck_{eu}(t)$, com eu=0,1,..., N-1 e a fonte UTC $S(t)$, garanta sempre t que

$$\forall \qquad |S(t) - Ck(t)| < \Delta$$

- *sincronização interna*–dado um intervalo máximo de incerteza tolerada, Δ, entre os relógios locais $Ck_{eu}(t)$, com eu=0,1,..., N-1, garanta

  tempo todo t que

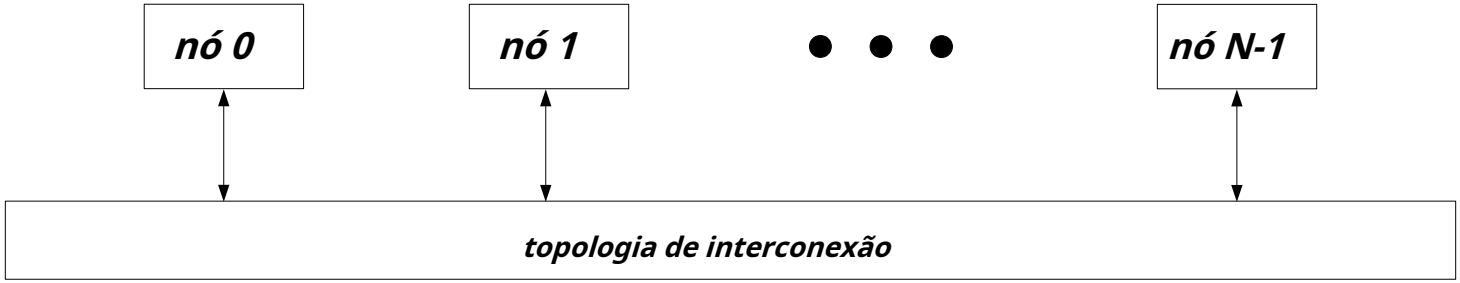
$$\forall_{eu\ j \in \{0,1,\cdots,N-1\}} |Ck_{eu}(t) - Ck_j(t)| < \Delta .$$

Ajuste de tempo – 2

Supõe-se que os nós de processamento da máquina paralela estão conectados por alguma topologia de interconexão e que a comunicação entre eles é realizada através de passagem de mensagens, com tempo de transmissão finito, mas sem limite superior, ou seja,

$$\forall_{eu \in \mathbb{R}^+} \exists_{t_M} \quad t_M > eu$$

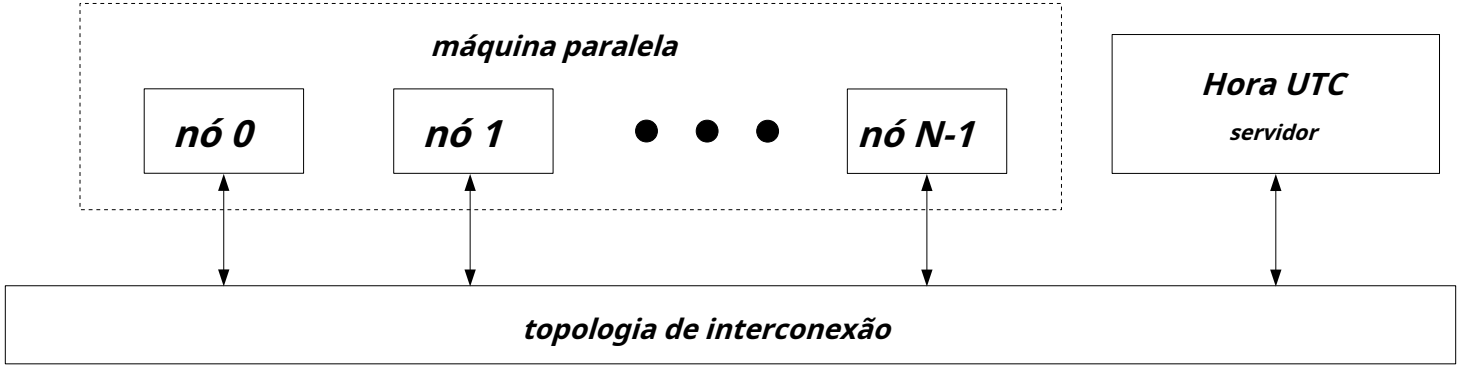
onde t_M representa o tempo de transmissão da mensagem.



O ajuste em si deve ser feito de forma que a monotonicidade da hora local seja sempre preservada. (***Por que***?)

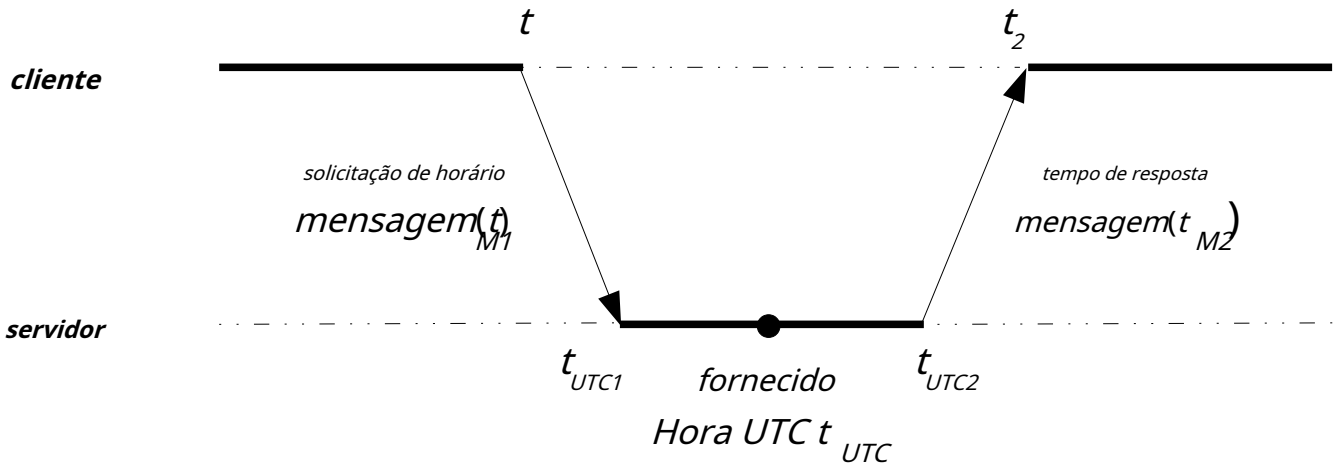
Método cristão – 1

É um método de sincronização externa onde se assume a disponibilidade de um servidor de horário UTC. Periodicamente, de forma proativa, para ajustar seu próprio relógio local, cada nó (agindo como cliente) se dirige ao servidor através do envio de uma mensagem onde o *certo* o tempo é solicitado. O servidor responde enviando-o em um formato predefinido.



Método cristão – 2

- decomposição de operação
  - a solicitação é feita no horário local t_1 através do envio de uma mensagem com tempo de transmissão t_{M1}, a mensagem é recebida no horário UTC do servidor t_{UTC1}
  - A Hora t_{UTC}, que será retornado, é ajustado para corresponder aproximadamente ao meio do intervalo de processamento do servidor
  - a resposta é enviada no horário UTC t_{UTC2} através de uma mensagem com tempo de transmissão t_{M2}, a mensagem é recebida na hora local t_2.



Método cristão – 3

- o cliente tem à sua disposição os horários t_1, t_2 e t_{UTC} para ajustar o relógio local
- o cliente assume que o *desvio* do relógio local produz uma variação insignificante do intervalo de tempo $t_2 - t_1$
- expressando o *desvio* entre a hora UTC e a hora local em $desligado = S(t) - Ck(t)$, a estimativa de deslocamento é dada por

$$desligado_{Husa} = t_{UTC} - \frac{t_1 + t_2}{2}$$

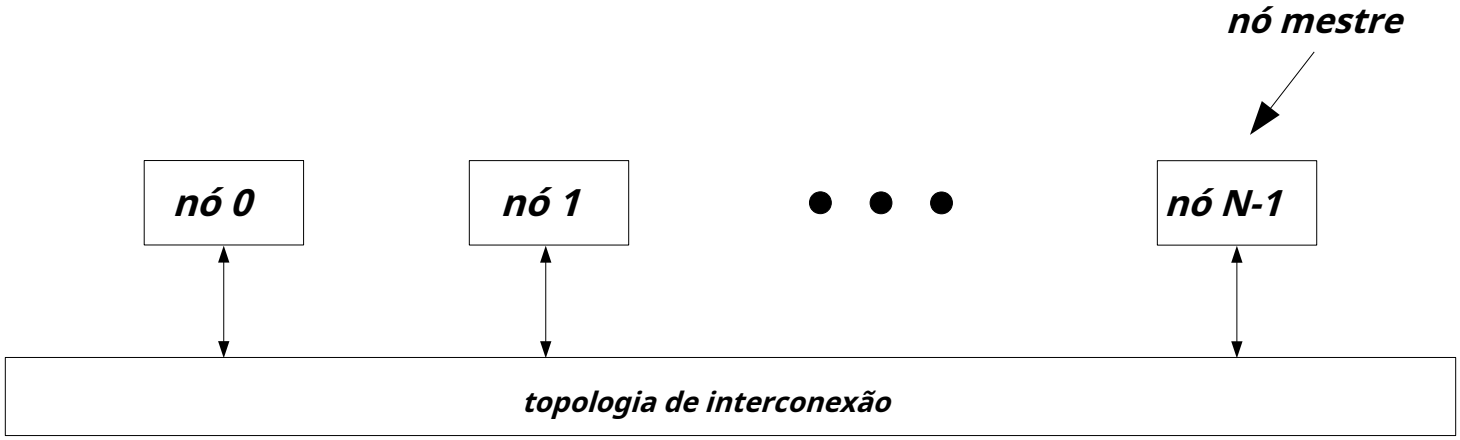
- a incerteza associada a este valor no pior caso, Δ_{Husa}, assume que $t_2 - t_1 \approx t_{M1} + t_{MÍNIMO}$ (o tempo de processamento no servidor é insignificante quando comparado com o tempo de transmissão da mensagem e uma delas é transmitida em tempo mínimo)

$$\Delta_{Husa} = \frac{t_2 t_1 - t}{2}_{MÍNIMO}$$

- se a incerteza estimada for maior que o valor nominal aceito, como no caso do canal de comunicação sofrer variações aleatórias de carga ou o servidor estar muito ocupado, o cliente poderá rejeitar a *desvio* estimativa e tente novamente mais tarde.

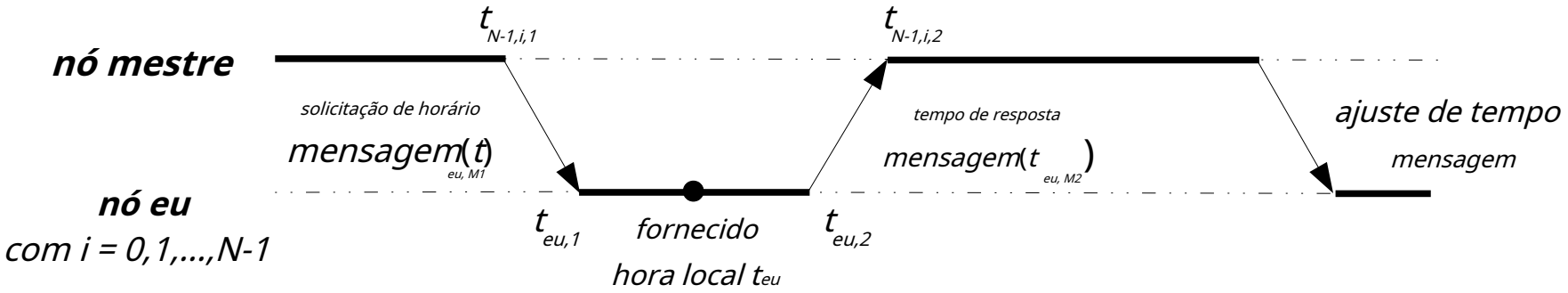
Algoritmo de Berkeley – 1

É um método de sincronização interna a ser aplicado quando um servidor de horário UTC não está disponível. Assim, o único objetivo é manter sincronizados entre si os relógios locais dos nós de processamento da máquina paralela. De tempos em tempos, um dos nós, chamado por esse motivo de *mestre* nó, aborda de forma proativa todos os nós da máquina paralela, inclusive ela, para obter informações sobre a hora local. Em seguida, calcula o valor médio dos offsets entre os demais relógios locais e o seu próprio e envia a todos os nós, inclusive ele, a correção que deve ser introduzida para ajustar a hora local a um valor comum.



Algoritmo de Berkeley – 2

- decomposição de operação
  - a solicitação é feita pelo nó mestre na hora local $t_{N-1,i,1}$ através do envio de uma mensagem para cada nó da máquina paralela com tempo de transmissão $t_{eu,\,M1}$, uma mensagem é recebida por cada nó no horário local $t_{eu,1}$, com i = 0,1,... ,N-1
  - A Hora t_{eu}, que será enviado por cada nó, é ajustado para corresponder aproximadamente ao meio do intervalo de processamento
  - a resposta é enviada por cada nó na hora local $t_{eu,2}$, com i = 0,1, ... ,N-1, através de uma mensagem com tempo de transmissão $t_{eu,\,M2}$, a mensagem é recebida pelo nó mestre na hora local $t_{N-1,i,2}$
  - o nó mestre calcula os desvios à sua hora local e envia uma mensagem a todos os nós com o ajuste que deve ser introduzido em cada caso.



Algoritmo de Berkeley – 3

- o *mestre* nó tem à sua disposição os tempos $t_{N-1,i,1}$, $t_{N-1,i,2}$ e t_{eu}, com i = 0,1,...,N-1, para estimar o deslocamento e a incerteza associada de cada um dos relógios locais
- o *mestre* nó assume que o *derivado* do seu relógio local produz uma variação insignificante dos intervalos de tempo $t_{N-1,i,2} - t_{N-1,i,1}$, com i = 0,1,...,N-1
- as estimativas individuais são calculadas, de acordo com o método de Cristian, por

$$desligado_{Husa}(eu) = t_{N-1,eu,1} + \frac{t_{eu,2}}{2} \qquad desligado_{Husa}(eu) = t_{N-1,eu,2} - \frac{t_{N-1,eu,1} - t_{MÍNIMO}}{2}$$

com $eu = 0,1,\cdots, N-1$

- o *mestre* nó calcula a seguir a média das estimativas do *desvio*, $desligado_{está\ med}$, rejeitando nos cálculos os casos em que as incertezas estimadas são maiores que o valor nominal aceito
- o *mestre* nó envia no final para cada um dos nós o ajuste que deve ser introduzido em cada caso, $desligado_{está\ med} - desligado_{Husa}(eu)$
- deve-se observar que, como o ajuste é um valor diferencial, o tempo de transmissão desta última mensagem não introduz qualquer incerteza adicional.

Protocolo de tempo de rede (NTP) - 1

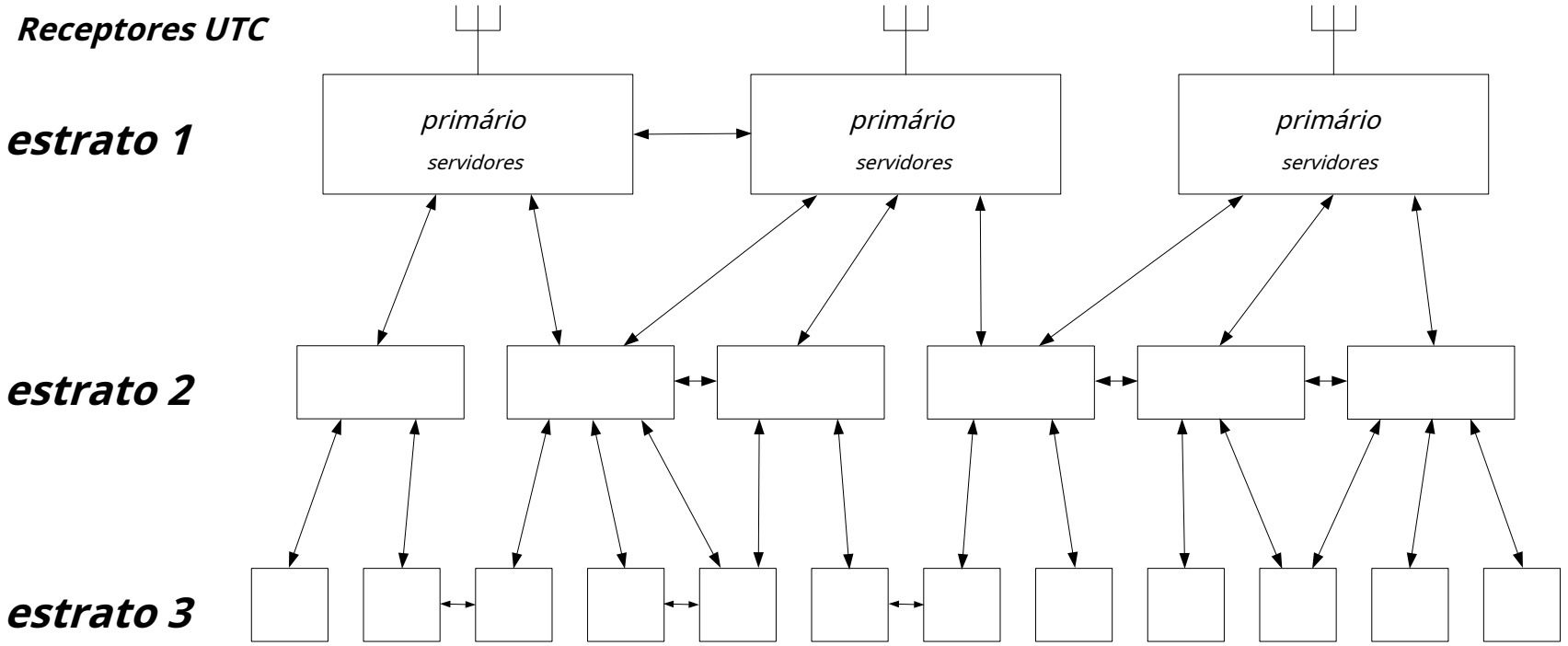
Os métodos descritos visam a sincronização dos relógios locais de sistemas informáticos integrados em redes locais. O *Protocolo de horário de rede* (NTP), por outro lado, trata da própria Internet.

O objetivo é

- permitir que cada sistema de computador conectado à Internet ajuste seu relógio local com precisão razoável e faça o ajuste a uma taxa alta o suficiente para evitar sérias discrepâncias de tempo devido a *deriva*
- garantir que o serviço prestado possa sobreviver às perdas mais ou menos longas de conectividade de servidores específicos, mantendo uma disponibilidade permanente
- para fornecer proteção contra interferências específicas.

Protocolo de tempo de rede (NTP) – 2

Arquitetura do serviço



Protocolo de tempo de rede (NTP) – 3

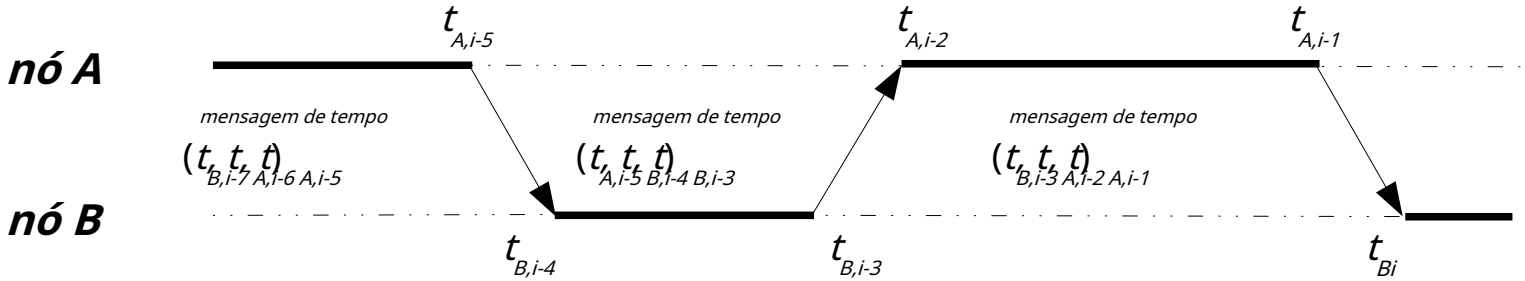
- o serviço pressupõe uma organização hierárquica dos sistemas informáticos em diferentes níveis, denominada *estratos*
- os sistemas informáticos no primeiro nível, *estrato* 1, estão conectados diretamente a uma fonte UTC, por isso são chamados *servidores primários*
- os restantes sistemas informáticos, localizados nos outros *estratos*, são chamados *servidores secundários* e ajustar seus relógios locais com servidores pertencentes a um *estrato* imediatamente acima na hierarquia
- os sistemas informáticos de cada *estrato* também pode coordenar suas informações de tempo com outros servidores pertencentes ao mesmo *estrato* fornecer informações globalmente mais estáveis e robustas

Protocolo de tempo de rede (NTP) – 4

- à medida que se desce na hierarquia, o grau de incerteza da informação de tempo produzida aumenta porque os erros introduzidos em cada estágio de sincronização são cumulativos
- sempre que a subárvore de ajuste for reconfigurada dinamicamente
  - sempre que um *servidor primário* não consegue acessar sua fonte UTC, ele passa para o *estrato* 2, tornando-se um *servidor secundário*
  - sempre que um servidor costumava ajustar o relógio local de um sistema de computador em um determinado *estrato* fica indisponível, outro servidor é procurado.

Protocolo de tempo de rede (NTP) – 5

- ocorre uma troca de mensagens horárias continuamente entre pares de nós (A e B, por exemplo) para ajuste do relógio local do nó B, se este pertencer a um nível inferior.*estrato*na hierarquia, ou para ajuste mútuo dos relógios locais de ambos os nós, se pertencerem ao mesmo*estrato*
  - cada mensagem encaminhada inclui três carimbos de data/hora: os horários locais de transmissão e recepção da última mensagem recebida, $t_{B,i-3}$ e $t_{A,i-2}$, e a hora local de transmissão do mensagem presente, $t_{A,i}$
  - após a recepção da mensagem, o nó de destino salva o horário de recepção local, t_{Bi}
  - as quatro vezes, $t_{B,i-3}$, $t_{A,i-2}$, $t_{A,i-1}$ e t_{Bi}, são então usados para calcular uma estimativa do*desvio* e sua incerteza.



Protocolo de tempo de rede (NTP) – 6

- o nó B assume que o *desvio* s de seu relógio local e o relógio local do nó A produzem variações desprezíveis dos intervalos de tempo $t_{Bi} - t_{B,i-3}$ e $t_{A,i-1} - t_{A,i-2}$, respectivamente
- expressando o *desvio* entre os horários locais de ambos os nós como $desligado = Ck_A(t) - Ck_B(t)$, sua estimativa é dada por

$$desligado_{Husa} = \frac{t_{Ah,\,eu-1} + t_{Ah,\,eu-2}}{2} - \frac{t_{B,\,eu-2} + t_{B,\,eu-3}}{2}$$

- a incerteza associada a este valor no pior caso, Δ_{Husa}, assume que o tempo de transmissão de Um de doissucessivo mensagem es é mínimo

$$\Delta_{Husa} = \frac{t_{Ah,\,eu-2} - t_{B,\,eu-3} + t_{B,\,eu} - t_{Ah,\,eu-1} - t_{MÍNIMO}}{2}$$

- o nó B aplica a seguir um algoritmo de filtragem estatística aos pares sucessivos ($desligado_{Husa}$, $-_{Husa}$) que são calculados para produzir uma estimativa final
- o procedimento é realizado com mais de um servidor, os resultados são comparados e podem levar à alteração do(s) servidor(es) a que se dirigem.

A variabilidade introduzida nas leituras dos relógios locais dos nós de processamento da máquina paralela, pelos algoritmos de ajuste de tempo, impossibilita a utilização de informações de tempo para sincronizar as atividades dos diferentes processos que formam uma aplicação distribuída. O melhor que se pode esperar é gerar um grau de incerteza na faixa de milissegundos entre as leituras de tempo dos pares de relógios locais, o que significa que cerca de um milhão de instruções podem ser executadas por cada processador em um intervalo de tempo desta magnitude.

Por outro lado, Lamport (1978) mostrou que se dois processos residentes em nós distintos não interagem, não é estritamente necessário que os seus relógios locais acelerem: a incompatibilidade não é observável e, portanto, não surgirão conflitos. Assim, o que realmente importa não é que todos os processos envolvidos concordem sobre o valor da *real* tempo, mas ordenam da mesma maneira os eventos relevantes que ocorrem, ou seja, aqueles que estão envolvidos em algum tipo de interação.

A exploração deste conceito leva ao chamado *relógios lógicos* que apenas retratam o fluxo de informações que ocorre.

Relógio lógico escalar - 1

Um define um *evento* como qualquer atividade relevante que ocorre durante a execução de um processo. Entre todas as atividades que ocorrem, as atividades de comunicação são especialmente significativas no que diz respeito à sincronização de processos, ou seja, transmissão de mensagens. *enviando* e mensagem *recebendo*.

A ordenação dos eventos que ocorrem em uma aplicação distribuída é baseada em duas observações quase triviais

- quando dois eventos ocorrem no mesmo processo, eles ocorrem na ordem percebida pelo processo
- quando uma mensagem é trocada entre processos, a mensagem *enviando* evento ocorre necessariamente *antes* o *recebendo* evento da mesma mensagem.

Pares de eventos são classificados para fins de ordenação como

- *sequencial* – se for possível afirmar qual ocorreu *antes*
- *simultâneo* – de outra forma.

Relógio lógico escalar - 2

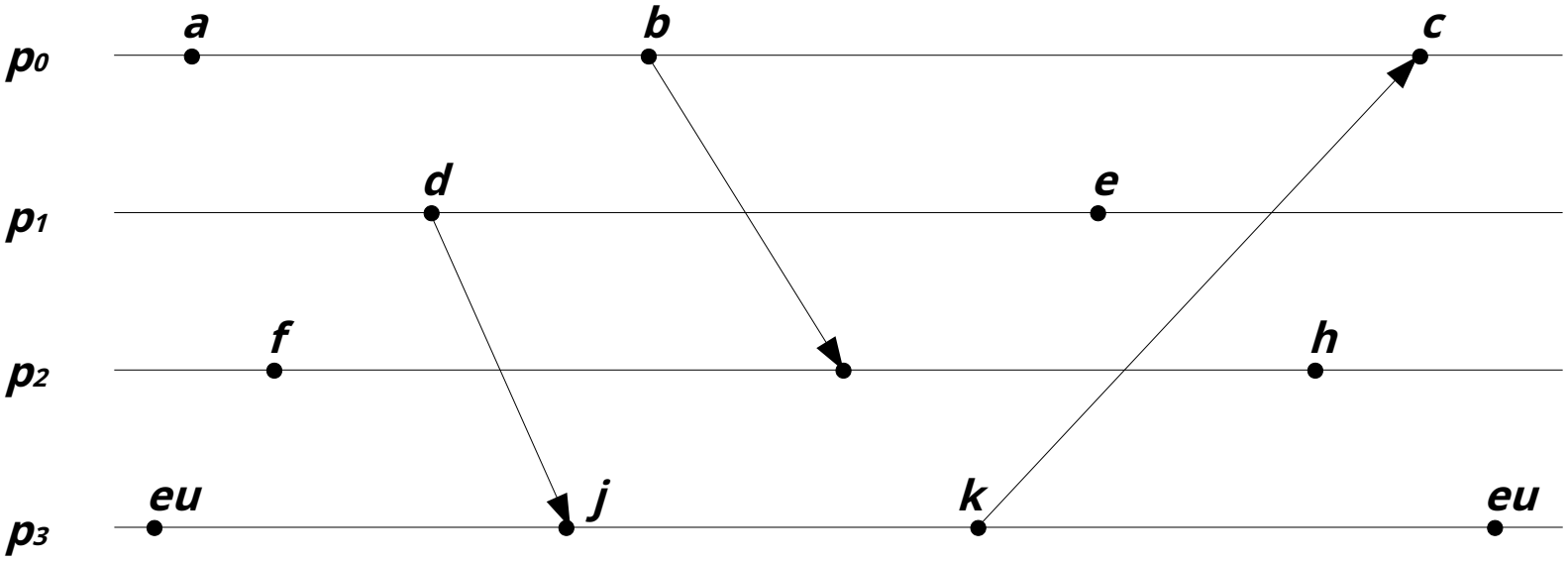
Lamport (1978) nomeado *ocorreu antes*, e denotado por - , a ordem parcial dos eventos que surge da generalização das observações que acabaram de ser feitas.

A relação *ocorreu antes* pode ser formalmente definido da seguinte maneira

- $\exists_{p_{eu} \in \{p_0, p_1, \cdots, p_{N-1}\}} e e' \Rightarrow e e'$
- $\forall_{mensagem}$ enviar(*eu*) -receber(*eu*)
- $e \text{-} e' \wedge e' \text{-} e'' \Rightarrow e \text{-} e''$

onde p_{eu}, com eu=0,1,...,N-1, são os processos que coexistem na aplicação distribuída e que são executados em nós distintos da máquina paralela, e e, e' e e'' são eventos genéricos.

Relógio lógico escalar - 3



eventos sequenciais

$f\text{-}g \wedge g\text{-}h \Rightarrow f\text{-}h$

$d\text{-}j \wedge j\text{-}k \wedge k\text{-}c \Rightarrow d\text{-}c$

eventos simultâneos

$\neg(f\text{-}c) \wedge \neg(c\text{-}f) \Rightarrow f\text{-}c \; \neg(eu\text{-}e) \wedge \neg(e\text{-}eu) \Rightarrow eu\text{-}e$

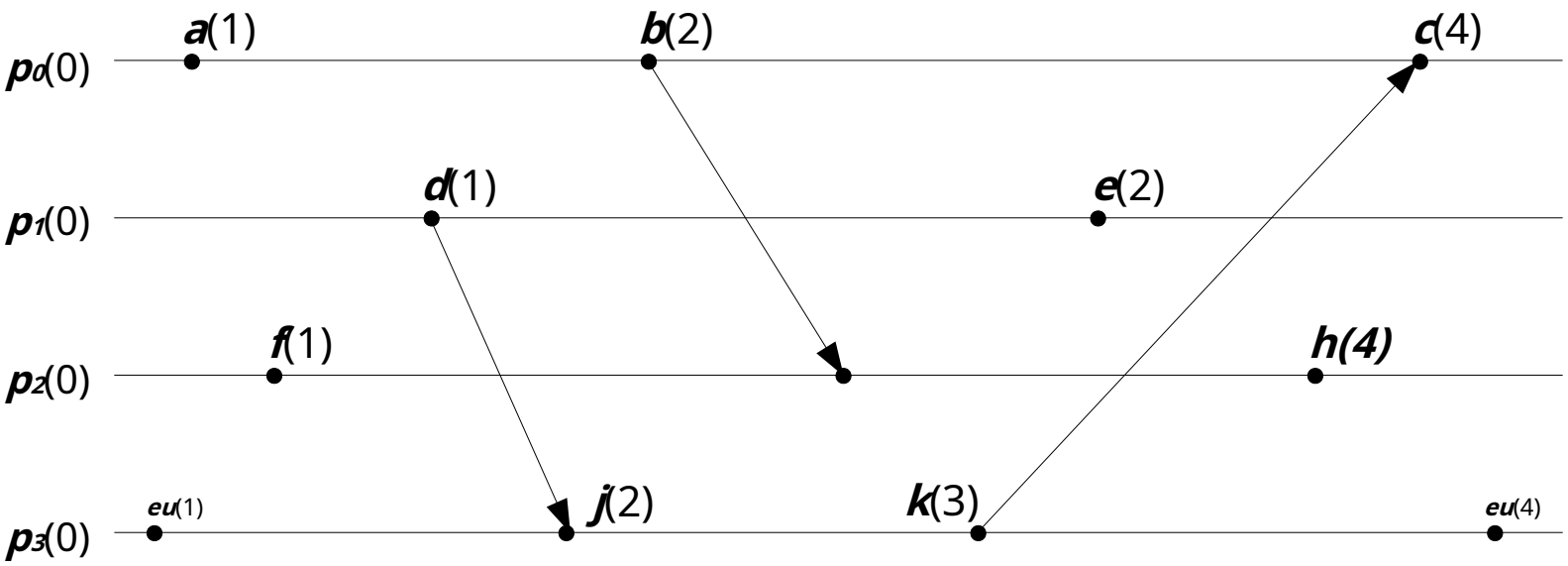
Relógio lógico escalar - 4

Lamport sugeriu um dispositivo, que ele chamou *relógio lógico*, para explicitar numericamente a ordenação resultante da relação *ocorreu antes*. A *relógio lógico escalar* é essencialmente um contador local de eventos, aumentando monotonicamente, que não tem associação direta com o *real* tempo.

Em uma aplicação distribuída, cada processo *p*, com *eu*=0,1,...,N-1, usa um relógio lógico como seu próprio relógio local, *Ck*, e marca os eventos de acordo com as seguintes regras

- *inicialização*: $Ck_{eu}=0$
- *ocorrência de um evento localmente relevante*: atualização do relógio local, $Ck_{eu}=Ck_{eu}+\alpha_{eu}$, onde α_{eu} é uma constante numérica geralmente igual a 1
- *envio de mensagem*: inserção de um carimbo de data/hora *ts*, de valor igual a Ck_{eu}, na mensagem a ser enviada após a atualização ocorrer
- *recepção de mensagem*: ajuste do relógio local ao valor *máx.*(Ck_{eu}, *ts*) antes de atualizar seu valor com o evento de recepção.

Relógio lógico escalar - 5



É fácil provar por indução matemática que

$$e \rightarrow e' \Rightarrow Ck_{eu}(e) < Ck_j(e') \quad \text{, com } eu\ j = 0,1,\cdots,N-1 \quad .$$

O inverso, porém, não é verdade.

Ordenação total de grupos de eventos - 1

Lamport mostrou, no entanto, que grupos de eventos relacionados podem estar sujeitos a uma operação de *pedido total* e ser percebido na mesma ordem por processos residentes em diferentes nós de uma máquina paralela, se relógios lógicos do tipo que ele prescreve forem utilizados na geração dos carimbos de hora incluídos nas mensagens trocadas.

Deixar e_j, com j=0,1,...,K-1, sejam eventos associados à troca de mensagens eu_j entre os processos p, com eu=0,1,...,N-1, então os eventos e_j pode ser *totalmente ordenado* se e somente se for possível estabelecer uma correspondência um-a-um entre cada evento e um ponto na linha numérica [reta] através de uma propriedade associada (o carimbo de data/hora inserido em cada mensagem).

Ordenação total de grupos de eventos - 2

Como nada impede que os carimbos de data/hora associados a duas mensagens distintas sejam iguais, $ts(eu_p) = ts(eu_q)$, com $p, q = 0,1, \ldots, K-1$ e $p \neq q$, Lamport definiu uma estrutura de dados que chamou *carimbo de data/hora estendido*, $(ts(eu), eu\ ia(eu))$, que consiste no par ordenado formado pelo carimbo de data/hora da mensagem e pela identificação do remetente, e prescreveu a seguinte regra para ordenar os *carimbos de data/hora*

$$(ts(eu_p), eu\ ia(eu_p)) < (ts(eu_q), eu\ ia(eu_q)) \Leftrightarrow ts(eu_p) < ts(eu_q) \vee$$

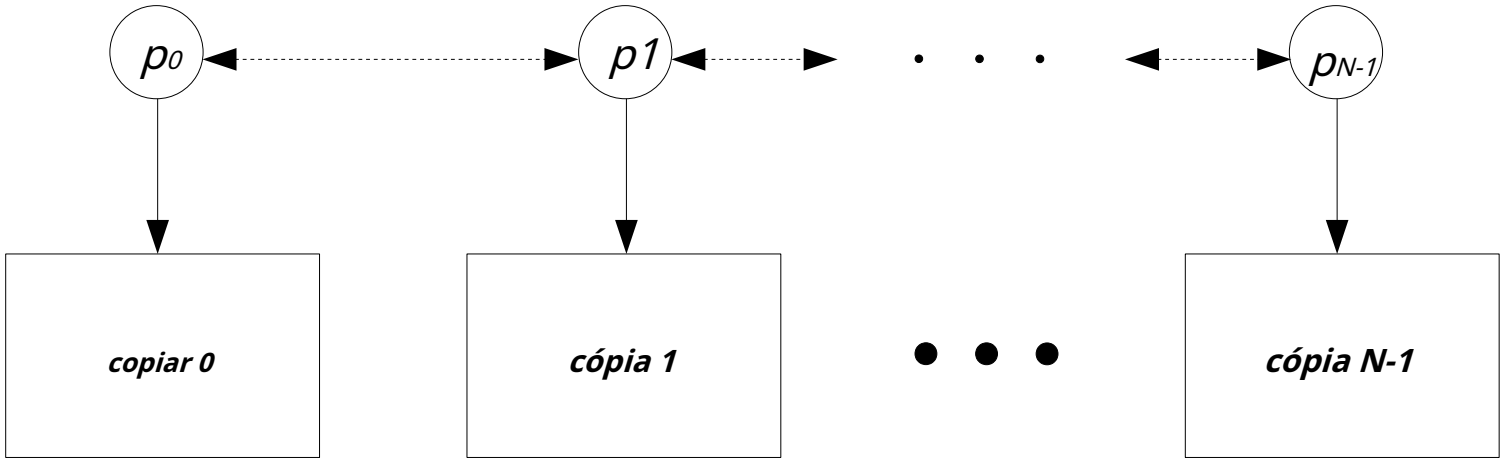
$$ts(eu_p) = ts(eu_q) \wedge eu\ ia(eu_p) < eu\ ia(eu_q).$$

Ordenação total de grupos de eventos - 3

Suponha que, em uma determinada aplicação distribuída, existam N cópias da mesma região de dados localizadas em locais geograficamente distintos. Cada cópia é acessada por um processo específico p_{eu}, com eu=0,1,...,N-1. Cada processo p_{eu} realiza operações de *escrita* e *leitura* sobre os registros de sua cópia levando à alteração de seu conteúdo.

Pergunta

Como organizar as operações para que as diferentes cópias fiquem permanentemente sincronizadas, ou seja, apresentem sempre o mesmo valor em todos os seus registradores?



Ordenação total de grupos de eventos - 4

A sincronização permanente das cópias exige que

- sempre que um processo quiser modificar o valor de um registro de sua cópia local, a operação deve ser propagada primeiro para os processos que gerenciam o acesso a todas as outras cópias
- a ordem de execução dessas operações deve ser a mesma em todos os lugares.

Para que isso seja feito com sucesso, é preciso assumir que

- os processos p_{eu} são mantidos em correto funcionamento, ou seja, não *falhas catastróficas* ocorrer
- não há perda de mensagem.

Ordenação total de grupos de eventos - 5

Lamport propôs o seguinte algoritmo para resolver o problema

- cada processo p_{eu}, ao afirmar que a próxima operação modificará o valor de um registro de sua cópia local, constrói uma mensagem com todos os dados referentes à operação e anexa um carimbo de data/hora com o valor de seu relógio local marcando o evento

- a mensagem é enviada a todos os membros do grupo, incluindo ele mesmo
- ao receber a mensagem, cada processo *p* ajusta seu relógio local de acordo com as regras prescritas por Lamport e insere a mensagem em uma fila local em ordem crescente de sua *carimbo de data/hora estendido*
- uma mensagem de *reconhecer* é enviado a todos os membros do grupo, incluindo ele mesmo
- as operações descritas nas mensagens armazenadas em cada fila local são executadas por cada processo p_{eu} na ordem pré-estabelecida quando todos os membros do grupo tiverem reconhecido a operação.

Relógio lógico vetorial - 1

Mattern (1989) e Fidge (1991) introduziram outro tipo de relógio lógico com o objetivo de superar a limitação presente no relógio lógico escalar de Lamport.

$\neg[\forall e_{eu}, e'_j \quad Ck(e_{eu}) < Ck(e') \Rightarrow e \qquad e']$, com $eu\ j = 0,1,\cdots,N-1$ e $eu \neq j$,

isto é, quando dois eventos e e e' ocorrem em processos distintos, o fato de que o valor de o carimbo de data/hora associado ao primeiro ser menor que o valor do carimbo de data/hora associado ao segundo não significa o primeiro evento *já ocorreu antes* o segundo.

A ideia deles era manter as informações sobre os eventos, não apenas recuperadas do próprio relógio lógico, mas também todas as informações disponíveis recuperadas dos relógios lógicos dos demais processos do grupo, mesmo que não estivessem atualizadas. Desta forma, pode-se captar o *causalidade potencial* que pode existir entre eventos que ocorrem em processos residentes em nós distintos da máquina paralela.

Relógio lógico vetorial - 2

O dispositivo sugerido operando em um sistema de processo N é basicamente uma coleção de contadores de eventos crescentes monotonicamente, que novamente não têm conexão com o *real* tempo.

A coleção de contadores é organizada como um array *V*, onde cada elemento representa um relógio local semelhante ao Lamport.

Em uma aplicação distribuída, cada processo p_{eu}, com eu=0,1, relógio lógico, tem seu próprio vetor $V_{eu}[1..N]$. Variedade V_{eu} os elementos são interpretados da seguinte maneira

- $V_{eu}[eu] = Ck_{eu}$, é o relógio local do processo p_{eu}
- $V_{eu}[j] = Ck_j$, com $j \neq eu$, representa o processo de percepção p_{eu} tem sobre a evolução do relógio local do processo p_j (processo p_j entretanto pode ter marcado mais eventos, mas o processo p_{eu} ainda não recebeu nenhuma informação sobre eles nos carimbos de hora das mensagens recebidas até agora).

Relógio lógico vetorial - 3

A atualização do relógio vetorial local é realizada de acordo com as seguintes regras

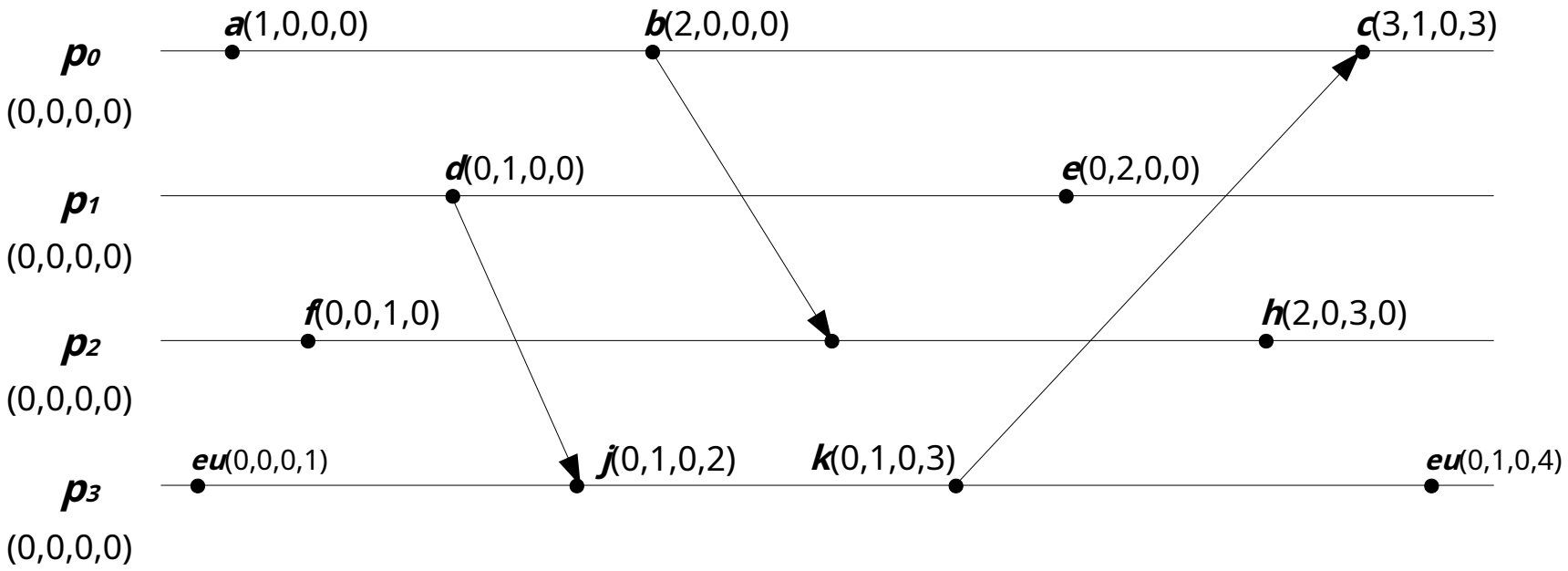
- *inicialização*: $V_{eu}[j] = 0$, com $j=0,1,\ldots,N-1$
- *ocorrência de um evento localmente relevante*: atualização do relógio local $V_{eu}[eu] = V_{eu}[eu]+\alpha_{eu}$, onde α_{eu} é uma constante numérica geralmente igual a 1
- *envio de mensagem*: inserção de um carimbo de data/hora *ts*, de valor igual a V_{eu}, na mensagem a ser enviada após a atualização ocorrer
- *recebimento de mensagem*: ajuste de cada elemento do relógio vetorial local ao valor $máx.(V_{eu}[j], ts[j])$, com $j=0,1,\ldots,N-1$ e $eu \neq j$, antes de atualizar seu próprio valor $V_{eu}[eu]$ com o evento de recepção.

Relógio lógico vetorial - 4

Deixar V e V' sejam carimbos de data/hora vetoriais, então seus valores serão comparados de acordo com as seguintes regras

- $V = V' \Leftrightarrow \forall_{0 \le j < N} V[j] = V'[j]$
- $V \le V' \Leftrightarrow \forall_{0 \le j < N} V[j] \le V'[j]$
- $V < V' \Leftrightarrow V \le V' \wedge V \ne V'$.

Relógio lógico vetorial - 5



eventos sequenciais

$f\text{-}h \quad \Rightarrow \quad V_2(f) < V_2(h)$

$d\text{-}c \quad \Rightarrow \quad V_1(d) < V_3(c)$

eventos simultâneos

$f\text{-}c \quad \Rightarrow \neg[V_2(f) < V_0(c)] \wedge \neg[V_0(c) < V_2(f)]$ $\neg[V_3(eu) <$

$eu\text{-}e \Rightarrow \quad V_1(e)] \wedge \neg[V_1(e) < V_3(eu)]$

Relógio lógico vetorial - 6

É fácil provar por indução matemática que

$$e \rightarrow e' \Leftrightarrow V_{eu}(e) < V_j(e') \quad \text{, com } eu\ j = 0,1,\cdots,N-1 \quad .$$

O inverso agora também é verdadeiro.

$$V_{eu}(e) < V_j(e') \Rightarrow e \rightarrow e' \quad \text{, com } eu\ j = 0,1,\cdots,N-1.$$

Desafio

Prove!

É precisamente este facto que permite, através da comparação dos carimbos temporais associados, captar o *causalidade potencial* que podem existir entre eventos que ocorrem em processos localizados em nós distintos de uma máquina paralela e ordená-los.

Leitura sugerida

- *Sistemas Distribuídos: Conceitos e Design, 4ºEdição, Coulouris*, Dollimore, Kindberg, Addison-Wesley
  - Capítulo 11:*Tempo e estados globais*
    - Seções 11.1 a 11.4
- *Sistemas Distribuídos: Princípios e Paradigmas, 2ª Edição,*Tanenbaum, van Steen, Pearson Education Inc.
  - Capítulo 6:*Sincronização*
    - Seções 6.1 e 6.2