Ordenação

Sistemas Distribuídos e Tolerância a Falhas

Universidade da Beira Interior 07/08

Benjamim Marques M1440
Daniel Félix M1441
João Duarte a14951

Índice

- Introdução
- Problema
- FIFO
- Ordenação Causal
- Ordenação Total
- Ordenação Temporal
- Algoritmos

Introdução

- A ordenação é essencial em sistemas distribuídos.
- A primeira noção que temos é a de "ordem física" (ordem em relação a tempo).
- Se não considerarmos o tempo, temos apenas uma sequência de eventos.

Introdução (continuação)

- Ordenação:
  - Objectivo – Garantir que os eventos ocorrem sobre uma determinada ordem.
    - Isto é conseguido por protocolos de entrega ordenada.
  - Resultado - Permite indicar quais os eventos que ocorreram primeiro e assumir ou excluir, as causas e efeitos relacionadas entre os mesmos.

Problema

- A ordem que existe mais intuitiva é a ordem física, em que os eventos ocorrem numa linha de tempo real.

Problema (Exemplo)

- Considerando um grupo de utilizadores e uma lista de emails: X, Y, Z e A.
  - O utilizador X envia um mensagem com o assunto *Meeting*.
  - Os utilizadores Y e Z respondem enviando uma mensagem com o assunto *Re:Meeting*.
    - Em tempo real, a mensagem de X foi primeiro enviada para Y.
    - Y lê a mensagem e responde.
    - Z lê ambas as mensagens de X e Y e então envia uma outra resposta que referencia as mensagens de X e de Y.

Problema (Exemplo)

- Mas, devido aos atrasos nas entregas das mensagens, estas podem ser entregues como na figura abaixo e alguns utilizadores poderiam ver estas mensagens de resposta (replies) na ordem errada, como por exemplo, o usuário A poderia ver:

From:	Subject:
Z	Re:Meeting
X	Meeting
Y	Re:Meeting

Problema (Exemplo)

- Se os clocks dos computadores de X, Y e Z pudessem estar sincronizados, então cada mensagem poderia transportar o tempo do clock do computador local (um rótulo de tempo), quando ela fosse enviada.
- Se os clocks são aproximadamente sincronizados, então esses rótulos de tempo, estarão frequentemente na ordem correcta.
- Dado que clocks não podem ser sincronizados na perfeição num Sistema Distribuido, Lamport [1978] propôs um modelo de tempo lógico que pode ser usado para ordenar vários eventos em processos executados em diferentes computadores de um SD.

Precedência

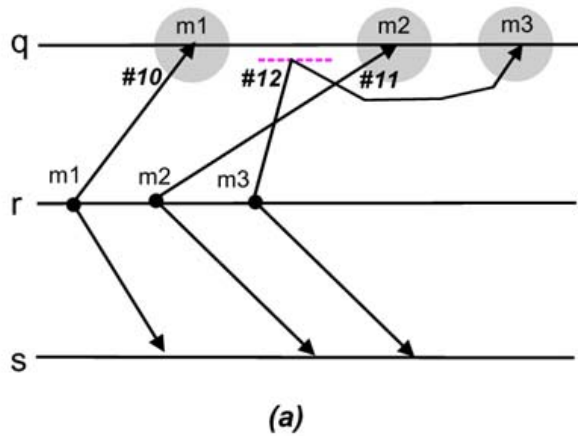
- Se *a* e *b* são eventos no mesmo processo e *a* precede *b*, então $a \rightarrow b$.
- Se *a* é o envio da mensagem *m* e *b* é a recepção da mensagem *m*, então $a \rightarrow b$.
- Se $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$.

FIFO

- Quaisquer duas mensagens enviadas pelo mesmo processo são entregues a qualquer processo pela ordem de envio.
- A ordem FIFO é assegurada pela atribuição de um número de sequência local.
- O receptor entrega as mensagens à aplicação pela ordem dos seus números de sequência.

FIFO (continuação)

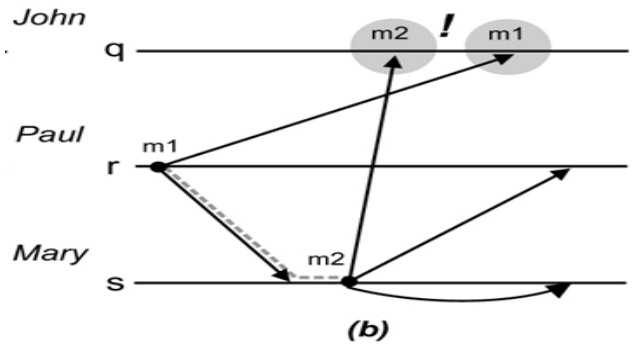
- Para isso o receptor deve ter um *buffer* temporário de mensagens recebidas fora de ordem e caso exista alguma em falta pedir a retransmissão da mesma.



(a)

FIFO (continuação)

- A ordenação FIFO pode ser insuficiente.
- Como John está à espera de receber as mensagens por ordem de execução das tarefas.
- Mas como recebe as mensagens de processos diferentes e o protocolo FIFO ignora a relação existente entre "r" e "s" existe um conflito e m2 é recebido antes de m1 (prococado também por um atraso na entrega de m1).



Ordenação causal

- Consiste na garantia que as mensagens enviadas por processos diferentes são entregues pela ordem correcta no receptor.

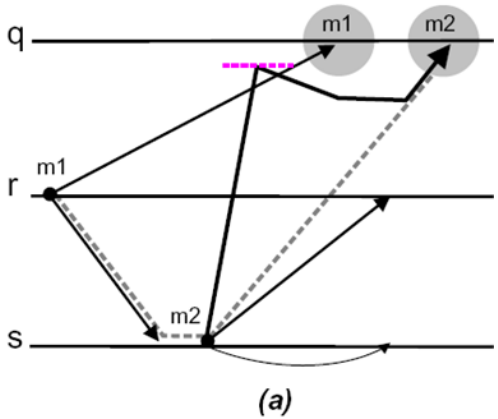
  - *Causal delivery*

    $envio_p(m) \rightarrow envio_q(n)$
    $entrega_r(m) \rightarrow entrega_r(n)$.

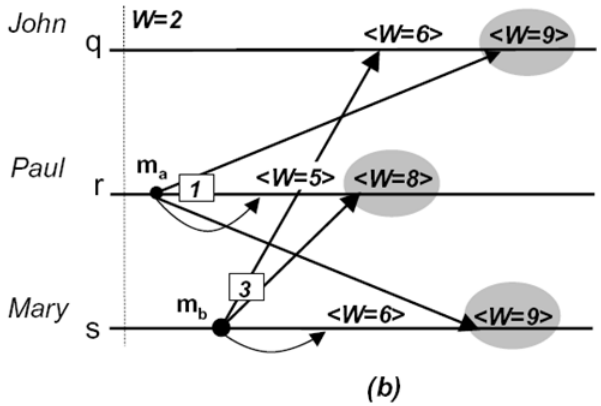
Ordenação causal (continuação)

- Limitações:
  - Não garante que as mensagens dos processos concorrentes sejam ordenadas.

Solução do FIFO



Problema Causal



Ordenação Total

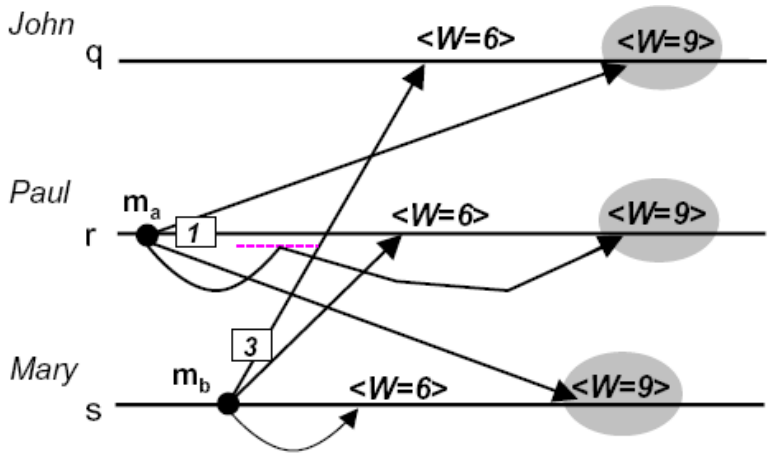
- Garante que quaisquer duas mensagens entregues a qualquer par de receptores são entregues na mesma ordem para ambos.

- Consiste na garantia que a entrega das mensagens se fazem em todos os processos pela mesma ordem.

  - Se um processo entregar m1 entes de entregar m2, então qualquer outro processo que entregar m2 irá entregar antes m1.

Ordenação Total (continuação)

- Não implica nem a ordenação FIFO nem a Causal, a ordem de entrega pode ser arbitrária, desde que seja a mesma em todos os processos.



Ordenação Temporal

- Uma mensagem m1 precede temporalmente m2 e será recebida por essa ordem por todos os membros do grupo se e só se m1 for enviada antes de m2 com um certo intervalo de tempo (Δt).

- Este tempo é o tempo aproximado que demora uma mensagem a ser entregue dentro de um sistema, ou o tempo de execução de um processo, ou a tempo de resposta de um controlo num processo físico.

Algoritmos de ordenação Causal

- Objectivo:
  - Assegurar que as mensagens são entregues à aplicação de forma a respeitar a ordem causal, se m1 e m2 devem ser entregues ao mesmo processo e m1 precede m2 então m2 é entregue depois de m1.

Algoritmos de ordenação Causal (continuação)

- Uma das maneiras mais intuitivas é fazer com que cada mensagem carregue o sua própria lista "passado".
- Para isto é necessário que cada processo mantenha uma lista de mensagens enviadas.

Algoritmos de ordenação Causal (continuação)

- Protocolo:
  - Quando uma mensagem é enviada, leva a lista "passado" do seu processo emissor no campo de controlo.
  - Depois de enviar a mensagem, o emissor adiciona essa mensagem á sua lista.
  - Quando uma mensagem é recebida é verificado o seu campo de controlo. As mensagens que se encontram nessa lista que ainda não foram entregues podem ser imediatamente entregues mesmo que não tenham sido recebidas ainda.

Algoritmos de ordenação Causal (continuação)

- Depois de essas mensagens terem sido entregues á aplicação a mensagem recebida é adicionada a lista "passado" de mensagens recebidas do receptor.

- Isto garante que as mensagens enviadas serão todas entregues mesmo que as anteriores se percam, pois a última carrega todas as outras como garantia.

- Um ponto negativo deste protocolo é o tamanho exagerado do campo de controlo, pois este pode crescer indefinidamente podendo conter muitas mensagens. O que torna este protocolo impraticável.

Algoritmos de ordenação Causal (continuação)

- Deve ser completado com um mecanismo fundamental para eliminar informação obsoleta.

- Por exemplo no envio de 4 mensagens as 2 primeiras são entregues antes das 2 duas últimas serem enviadas, e então os receptores enviam uma mensagem a confirmar que receberam de forma a que as 2 últimas mensagens não carreguem as 2 primeiras.

Algoritmos de ordenação Causal (continuação)

- Uma forma de resolver este problema é reduzir o tamanho do campo de controlo guardando na lista "passado" apenas os identificadores das mensagens, em vez das mensagens completas.

- Nesta ideia assume-se que um 3º componente é responsável pela garantia de entrega.

- Ou seja, se uma mensagem for perdida, é de alguma forma retransmitida até ser recebida por todos os receptores pretendidos.

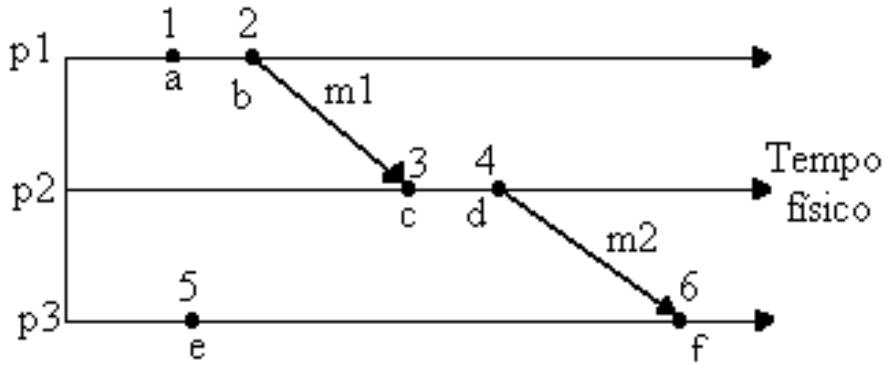
Algoritmos de ordenação Causal (continuação)

- Nesta forma a diferença é que quando uma mensagem é entregue e o campo de controlo é verificado se alguma mensagem ainda não tiver sido entregue a última é posta em espera até todas as outras chegarem e serem entregues.

- Só depois a última é entregue e o seu identificador adicionado a lista “passado” do receptor.

Algoritmos de ordenação Causal(continuação)

- Lamport
- Regras
  - O relógio local é incrementado antes de cada evento no processo em 1 unidade.
  - Quando um processo envia uma mensagem envia o número do clock do processo.
  - Quando um processo recebe uma mensagem compara o valor do clock local cm o da mensagem e actualiza o clock local para o máximo entre os dois mais uma unidade.

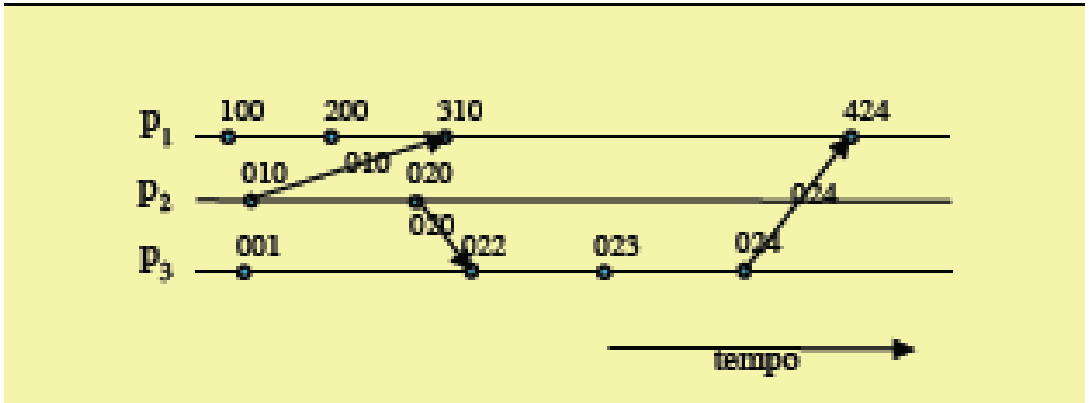


Algoritmos de ordenação Total

- O objecto da ordenação total é assegurar que todas as mensagens são entregues a todos os receptores pela mesma ordem.

Algoritmos de ordenação Total (continuação)

- Relógios vectoriais



- Se no final todos os vectores forem iguais não se consegue ver a ordenação dos eventos e então associa-se, por exemplo, o numero do processo a cada um para ordenar.

Algoritmos de ordenação Total (continuação)

- Sequenciador
  - Esta ideia consiste em seleccionar um processo especial e atribuir-lhe a tarefa de ordenar todas as mensagens.
  - Todos os emissores enviam as suas mensagens para o sequenciador.
  - Que por sua vez atribui um número de sequência único a todas mensagens e posteriormente irá retransmiti-las a todos os receptores pretendidos.

Algoritmos de ordenação Total (continuação)

- A ordenação total é a ordem pela qual o sequenciador processa as mensagens que chegam.
- Neste protocolo, a falha do sequenciador traz o problema da indisponibilidade e pode até requerer processos de recuperação complexos para assegurar o ordenamento correcto das mensagens.