

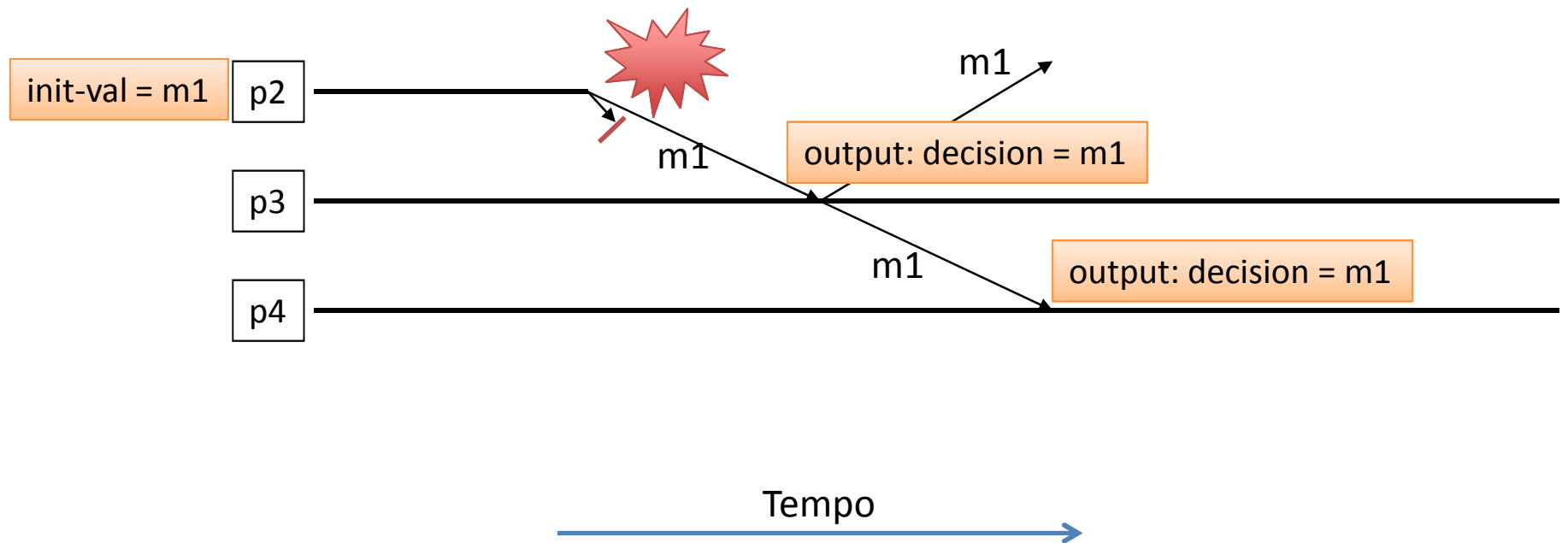
Algoritmos e Sistemas Distribuídos

Aula teórica 4

Ano lectivo 2014/2015

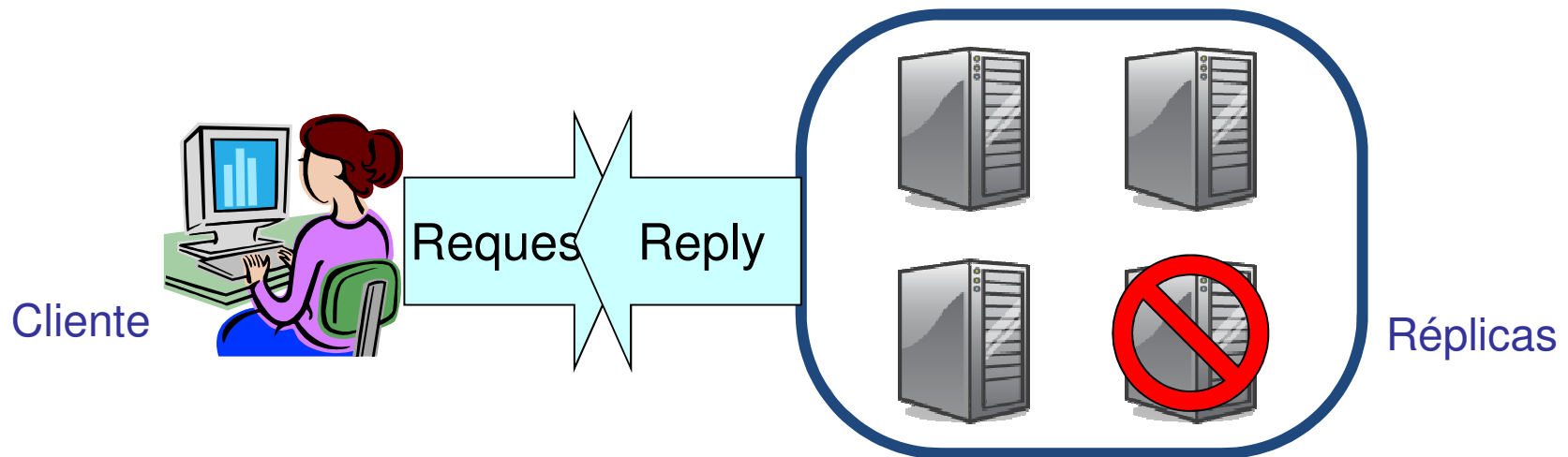
Mestrado Integrado em Engenharia Informática

Última aula: difusão



Replicação de máquinas de estados (State machine replication = SMR)

1. Tornar o serviço determinístico (máquina de estados)
2. Replicar o servidor
3. Garantir que as réplicas correctas seguem a mesma sequência de transições entre estados
4. Votar nos outputs de forma a tolerar falhas



Requisito central da SMR

Todas as réplicas correctas têm de receber todas as operações pela mesma ordem

- Difusão fiável não garante esta propriedade

Consenso

- Inputs: cada processo tem como input na variável v_i uma proposta inicial
- Outputs: cada processo tem como saída uma variável $decision_i$, inicialmente *null*
- C1 [Validade] Se todos os processos tiverem $v_i = v$, então v é o único valor do output permitido
- C2 [Acordo] Dois processos correctos não podem decidir valores diferentes
- C3 [Terminação] Todos os processos correctos têm de decidir a certo ponto (“eventually”)
- Opcional: C4[integridade] Se um processo correcto decide v , então v era a proposta inicial de algum processo

Nota: definição tem de ser adaptada para o caso de falhas Bizantinas

Solução para consenso síncrono

states_i:

v_i - input, proposta inicial

decision_i - output, decisão final

vals_i - inicialmente $\{v_i\}$

rounds_i - inteiro que representa o número do round, inicialmente 0

msgs_i:

if (rounds_i ≤ f)

forall (j in V)

send_i(j, vals_i); // send to all

}

trans_i:

rounds_i = rounds_i + 1

forall (j in V - {i})

receive(S) from j

vals_i = vals_i ∪ S

if (rounds_i == f + 1)

decision_i = min(vals_i)

Hoje

- Número mínimo de rounds para o consenso num sistema síncrono
- Impossibilidade do consenso num sistema assíncrono com uma única falha

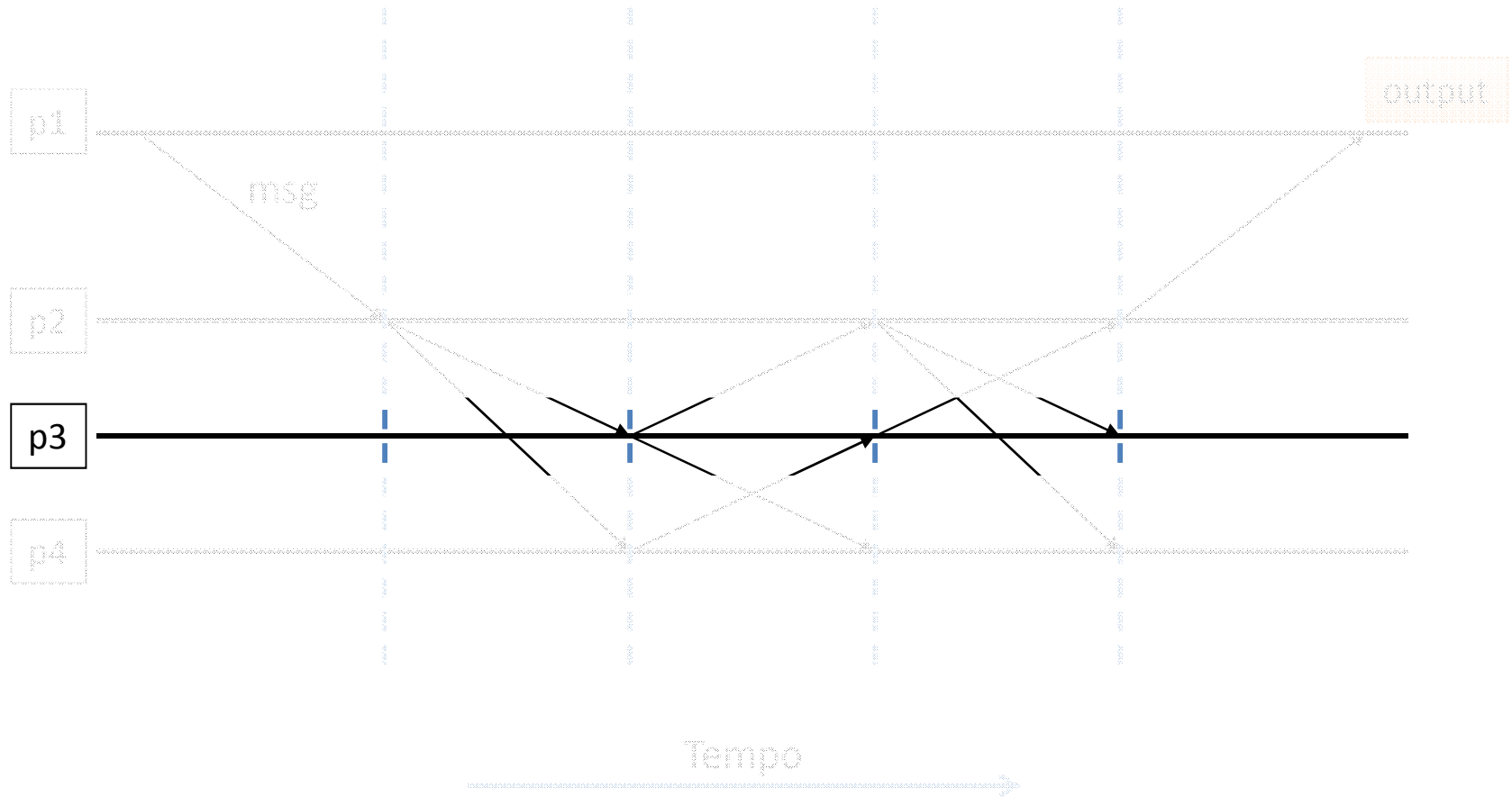
Limites à eficiência

- Teorema: Não existe nenhum algoritmo que resolva o problema do consenso num sistema síncrono em menos de $f+1$ rounds, na presença de f falhas, usando $n > f+1$ réplicas
- Para simplificar vamos demonstrar para $f=1$
 - Provar que é impossível resolver em um round

Definição: vista

- Dada uma execução e , dizemos que a vista do processo i nessa execução, denominada $e|i$, é a subsequência de e que consiste nos estados percorridos por i , juntamente com as mensagens recebidas por i

Exemplo: vista



Similaridade

- Dadas duas execuções e_1 , e_2 do mesmo algoritmo, dizemos que elas são similares para o processo i , denominado $e_1 \sim_i e_2$, se $e_1|_i = e_2|_i$
- O que podemos garantir sobre duas execuções similares para o processo i ?
- Processo i vai enviar as mesmas mensagens e efectuar os mesmos outputs

Execuções similares de consenso

- Lema: se i é correcto e $e1 \sim_i e2$ então i decide o mesmo valor no consenso em ambas execuções
- Considerando o fecho transitivo de $\sim$, dizemos que: $e1 \approx e2$ se existirem execuções $x_0, \dots, x_k$ tais que: $e1 = x_0 \sim_i x_2 \sim_j \dots \sim_k x_k = e2$
 - Chamamos a este fecho transitivo uma cadeia de similaridade
- Lema: se $e1 \approx e2$ então $\text{dec}(e1) = \text{dec}(e2)$
 - em que $\text{dec}(e)$ é o valor decidido por todos os processos (tem de ser o mesmo, porquê?)

Ideia da demonstração (do teorema do número mínimo de rounds)

- Contradição: supor que existe solução
- Considerar execução em que todos os processos inicialmente propõem valor 0
- Considerar execução em que todos os processos inicialmente propõem valor 1
- Mostrar que existe uma cadeia de similaridade entre as duas execuções
 - Qual é a contradição?

Construção da cadeia de execuções similares

- $e_0 \rightarrow$ todos propõem zero e não há falhas
- $e_1 \rightarrow$ todos propõem um e não há falhas
- x_i : Execuções sem falhas, em que os primeiros $i-1$ processos (com identificador inferior a i) propõem um
 - Ainda não provámos que $e_0 \approx e_1$ – temos de continuar a construção, ligando os x_i entre si, ou seja, provar que $x_i \approx x_{i+1}$

Demonstrar que $x_i \approx x_{i+1}$

- Partindo de x_i , vamos construir um conjunto de execuções x_{ij} , onde $0 \leq j \leq n-1$ da seguinte forma:
 - x_{ij} é igual a x_i , excepto que um crash de i faz com que se percam as mensagens de i para os j processos com identificadores mais elevados (excluindo i)

Demonstrar que $x_i \approx x_{i+1}$

- Do ponto de vista de todos os processos excepto j -ésimo id mais elevado, $x_{ij} \sim x_{ij+1}$
- Prosseguindo até $j=n-1$, ficamos com $x_i \approx x_{i_crash}$, onde o processo i não envia nenhuma mensagem
- Podemos trocar o valor de decisão inicial do processo i $0 \rightarrow 1$ (similar para todos os outros processos)
- Depois construir uma cadeia de forma semelhante, em que repomos as mensagens de i , uma de cada vez, até chegar a x_{i+1}
 - Trabalho para casa: representar as diferentes execuções da demonstração num diagrama temporal

Conclusão

- Provámos que $x_i \approx x_{i+1}$, ligando desta forma $e_0 \approx e_1$
- Em e_0 a única decisão possível é zero
- Em e_1 a única decisão possível é um
- Pelo lema sobre execuções similares do consenso, a decisão em e_0 e em e_1 tem de ser a mesma $\rightarrow$ contradição

Generalização

- Fácil generalizar para f arbitrário
- Ideia base é semelhante: cadeia de execuções similares que une uma execução com todos os valores iniciais a um com outra com todos a zero.
- Cada execução executa em f rounds
- É possível portanto falhar um processo por round
- Qual é o problema ao aplicar a ideia da demonstração que vimos?
- Ideia: aplicar a construção de forma recursiva (ver livro da Prof. Nancy Lynch para demonstração)

Mais resultados sobre o consenso

- Resultado fundamental (FLP):
- Não existe nenhum protocolo determinístico para resolver o consenso num sistema assíncrono em que um único processo possa ter uma falha por crash
 - Fisher, Lynch, and Paterson. Impossibility of distributed consensus with one faulty process. JACM, Vol. 32, no. 2, April 1985, pp. 374-382
- Vamos fazer demonstração mais simples do que a original, mas apenas para dois processos e uma falha
 - (se tiverem curiosidade, vejam a demonstração original no livro da cadeira: Nancy Lynch – Distributed Algorithms)

Prova da impossibilidade do consenso

- Por absurdo, vamos considerar que existe um algoritmo que resolve consenso (e derivar uma contradição)
- Vamos considerar 3 execuções desse algoritmo com condições particulares dos atrasos na rede
 - (Notar que qualquer atraso na rede é possível num sistema assíncrono)
- Chamar aos processos: A, B

Execução #1

- Ambos os processos têm proposta inicial 0
- Processo B falha (crash) no início da execução
- Pelas propriedades da especificação do consenso, processo A decide o valor 0 ao fim de um certo tempo
 - Chamemos esse tempo o instante t_1

Execução #2

- Ambos os processos têm proposta inicial 1
- Processo A falham (crash) no início da execução
- Pelas propriedades da especificação do consenso, processo B decide o valor 1 ao fim de um certo tempo
 - Chamemos esse tempo o instante t_2

Execução #3

- Mensagens de A para B e vice-versa têm um atraso até serem entregues superior a $\max(t1, t2)$
- Por similaridade à execução #1, processo A decide 0 no tempo $t1$
- Por similaridade à execução #2, processo B decide 1 no tempo $t2$
- Contradição! (Qual?)

Como dar a volta ao FLP?

- Temos de enfraquecer a especificação do problema
- Enfraquecer a propriedade de terminação
 - Apenas garantir terminação em períodos de sincronia: se a rede entregou as mensagens atempadamente durante um certo intervalo
 - Usar algoritmos aleatórios para garantir terminação com elevada probabilidade

Como dar a volta ao FLP? (cont.)

- Enfraquecer as propriedades de acordo/ validade
 - consenso num “k-set”: tem de existir um subconjunto W dos valores iniciais com k elementos tal que todos os valores decididos pertencem a W
- Fortalecer o modelo do sistema
 - Introduzia abstracção de “detectores de falhas” que permitem decidir se um processo falhou ou não