# *Sumários das aulas e exames de* Tolerância a Faltas Distribuída 2004-2005

Luís E. T. Rodrigues

Dezembro de 2004

# 1 Introdução

Este documento apresenta o sumário das aulas de Tolerância a Faltas Distribuída e indica qual a bibliografia que aborda os temas leccionados. A leitura deste texto não substitui de modo algum a leitura dos livros aconselhados.

# 2 Objectivos e Programa

## 2.1 Objectivos

A utilização crescente dos sistemas distribuídos numa multitude de áreas de actividade (veja-se o exemplo das aplicações sobre a Internet) introduz duas questões: *i)* o maior número de componentes do sistema requer preocupação com a sua fiabilidade; *ii)* a distribuição geográfica introduz possibilidades atraentes de replicar componentes em diversas máquinas.

A cadeira visa introduzir a área da tolerância a faltas distribuída. Esta disciplina aborda as técnicas que tiram partido da existência de um conjunto de máquinas interligadas em rede para replicar componentes de *software* nessas máquinas. Deste modo consegue-se atingir a confiança no funcionamento de modo mais versátil e menos dispendioso do que recorrendo a maquinaria dedicada. A matéria leccionada abrange conceitos, metodologias e mecanismos (técnicas de programação, protocolos) para a construção de sistemas confiáveis em rede (i.e. cuja probabilidade de falha é muito menor que a de um sistema "normal"). Estes temas são abordados

através das sua facetas teórica (modelos, algoritmos) e prática (sistemas, aplicações).

## 2.2 Programa geral

O programa encontra-se estruturado num conjunto de cinco tópicos, apresentados em blocos de duas ou três aulas teóricas, nomeadamente:

**Introdução à tolerância a faltas** Este bloco inicia-se com a apresentação de um conjunto de noções básicas e definições que apresentam a terminologia básica usada na área. Depois, através da discussão da máquina de estados replicada, e das suas diversas facetas, são motivados os diversos problemas a defrontar. Através do estudo desta técnica emerge a necessidade de alguns mecanismos de suporte básicos tais como: acordo distribuído, comunicação fiável, ordenação de mensagens, filiação em grupo e salvaguardas. Neste contexto, motiva-se também a necessidade de classificar os sistemas consoante o seu nível de sincronia.

**Acordo e ordenação** Este bloco dedica-se ao estudo de um paradigma central à tolerância a faltas distribuída, o problema do acordo distribuído. Soluções para este problema são estudadas para sistemas síncronos e assíncronos e para tolerar diferentes tipos de faltas. A relação deste problema com os problemas da ordenação de mensagens e a confirmação atómica distribuída é discutida.

**Sistemas síncronos** Neste bloco é dado ênfase a problemas que possuem solução em sistemas síncronos como o Acordo Bizantino e sincronização de relógios.

**Salvaguarda e recuperação** Este bloco centra-se no estudo de técnicas de suporte ao que geralmente se designa por *recuperação para trás*. A problemática da obtenção de um estado global coerente é revista, discutindo-se depois diversas técnicas para obter salvaguardas distribuídas e para realizar a recuperação do sistema com base no estado salvaguardado. O caso particular dos sistemas transaccionais é focado em pormenor.

**Exemplos de aplicação** O curso termina com exemplos concretos de aplicação dos conceitos discutidos nos blocos anteriores. Abordam-se as técnicas utilizadas para garantir a disponibilidade de dados replicados. Diversas técnicas de replicação são discutidos.

A aplicação destas técnicas em sistemas de "middleware", como por exemplo a arquitectura CORBA, é também referida. Finalmente, abordam-se as arquitecturas de tipo *cluster*, que são talvez o melhor exemplo da aplicação de diferentes técnicas de tolerância a faltas em sistemas de baixo e médio custo de uso relativamente vulgarizado.

## 2.3 Textos Base

- PAULO VERÍSSIMO AND LUÍS RODRIGUES. Distributed System for System Architects. Kluwer Academic Publishers. ISBN 0-7923-7266-2.

- RACHID GUERRAOUI AND LUÍS RODRIGUES. Introduction to Distributed Algorithms. Livro em preparação. Disponível on-line e na reprografia do DI. (Nota: não divulgar fora do contexto da cadeira. O livro está inacabado e é alvo de constantes correcções e alterações).

## 2.4 Outros Livros

1. J.-C. GEFFROY AND G. MOTET, Design of Dependable Computing Systems, Kluwer Academic Publishers 2002.

2. A. TANENBAUM AND M. VAN STEEN, Distributed Systems: Principles and Paradigms Prentice Hall, 2002.

3. G. COULOURIS AND J. DOLLIMORE AND T. KINDERBERG, *Distributed Systems, Concepts and Design, Third Edition*, Addison-Wesley, 2001.

4. K. BIRMAN, *Building Secure and Reliable Network Applications* , Manning, 1997. **Disponível em formato electrónico na página da cadeita.**

5. P. JALOTE, *Fault Tolerance in Distributed Systems*, Prentice Hall, 1994.

6. M. SINGHAL AND N. SHIVARATRI, *Advanced Concepts In Operating Systems, Distributed, Database and Multiprocessor Operating Systems*, McGraw-Hill. 1994.

7. S. MULLENDER, editor, *Distributed Systems, 2nd Edition*, ACM-Press. Addison-Wesley, 1993.

## 2.5 Artigos Científicos

Os textos supra-mencionados são complementados por um conjunto de artigos que servem de base a uma componente de avaliação contínua, sendo apresentados pelos alunos de pós-graduação durante as aulas. Este ano foram seleccionados os seguintes artigos:

1. ZHEN XIAO AND KEN BIRMAN, A Randomized Error Recovery Algorithm for Reliable Multicast, IEEE Infocom 2001, April 2001, Alaska.

   Este artigo ilustra como é possível aumentar a capacidade de escala de protocolos de difusão fiável recorrendo a abordagens probabilistas.

2. A. SOUSA, J. PEREIRA, F. MOURA, R. OLIVEIRA, Optimistic Total Order in Wide Area Networks. IEEE Intl. Symp. on Reliable Distributed Systems (SRDS'2002), October 2002.

   Este artigo ilustra um conjunto de investigação recente que pretende ultrapassar algumas da limitações de desempenho dos sistemas de grande-escala geográfica.

3. B. KEMME AND G. ALONSO., A Suite of Database Replication Protocols based on Group Communication Primitives. 18th International Conference on Distributed Computing Systems (ICDCS 98), Amsterdam, The Netherlands, May 1998.

   Este artigo mostra como as abstracções leccionadas na cadeira podem ser usadas para replicar bases de dados.

4. P. NARASIMHAN, L.E. MOSER, P.M. MELLIAR-SMITH., Strong Replica Consistency for Fault-Tolerant CORBA Applications. Journal of Computer System Science and Engineering (Spring 2002).

   Um sistema que fornece tolerância a faltas em sistemas CORBA. Muito perto da norma adoptada pela OMG, o sistema usa intensivamente as abstracções discutidas na cadeira e deu origem a um produto comercial.

# 3 Programa Pormenorizado

A cadeira encontra-se organizada em torno de aulas teóricas e aulas teórico-práticas (que são leccionadas exclusivamente aos alunos da licenciatura).

## 3.1 Sumários das Aulas Teóricas

**Aula 1: Fundamentos** A aula introduz os conceitos fundamentais sobre tolerância a faltas. Discutem-se quais as fontes de problemas e motiva-se a distinção entre falta, erro e falha. Abordam-se os objectivos da tolerância a faltas: a obtenção de confiança no funcionamento e introduzem-se algumas das métricas utilizadas para a sua aferição tais como a fiabilidade e a disponibilidade.

A aula prossegue com uma panorâmica sobre as técnicas que permitem obter tolerância a faltas e a importância da redundância neste contexto. Aborda-se a distinção entre as técnicas basedaas na detecção e recuperação de faltas e as técnicas baseadas no mascaramento. Discute-se também a necessidade de caracterizar o modelo de faltas para poder definir uma política correcta de tolerância.

**Aula 2: A máquina de estados distribuída** A aula pretende motivar os alunos para os principais problemas da tolerância a faltas distribuída através da análise de uma técnica concreta que se apresenta como bastante intuitiva numa primeira análise: a máquina de estados distribuída.

A máquina de estados distribuída é caracterizada e a sua utilização como técnica base de tolerância a faltas é abordada de modo informal. As propriedades das primitivas de comunicação que facilitam a concretização deste paradigma são discutidas.

**Aula 3: Difusão fiável e uniforme** Na sequência da aula anterior, discute-se a noção de comunicação atómica. Através deste exercício emerge a distincão entre difusão uniforme e não uniforme, o problema do grau de síncronismo do problema, e a necessidade de considerar (ou não) a existência de partições na rede. Todos estes aspectos se cruzam quando se tenta definir o que é um elemento "correcto".

Posteriormente, introduzem-se os aspectos de dinâmica na filiação. A noção de vista de grupo e de serviço de filiação são discutidos. Posteriomente, re-equaciona-se a noção de difusão fiável na presença de filiação dinâmica e introduz-se o conceito de sincronia na vista, que ao ordenar mensagens em relação a vistas permite definir o conjunto de elementos aos quais cada mensagem deve ser entregue. Os aspectos de transferência de estado são também discutidos.

**Aula 4: Ordenação de mensagens** Nesta aula discute-se em que medida

os protocolos de ordenação de mensagens podem facilitar o desenvolvimento de aplicações tolerantes a faltas. Em particular, discutem-se as vantagens da ordenação total de mensagens para concretizar a técnica de repliuação activa. Através da apresentação de algoritmos concretos para realizar a ordem total ilustram-se as vantagens da sincronia na vista.

**Aula 5: Consenso** A aula introduz o problema do consenso e aborda a sua solução em sistemas com detectores de falhas perfeitos. É discutida a diferença entre acordo uniforme e não uniforme. É dado um algoritmo concreto para resolver este problema.

**Aula 6: Utilização do acordo** A aula demonstra como o acordo distribuído pode ser usado como bloco para construção de outros serviços. Relembrando o problema da máquina de estados distribuída, mostra-se como é possível resolver o problema da ordenação total de mensagens usando o acordo como uma caixa-preta.

A relação com o problema da confirmação atómica distribuída é discutida. À semelhança do que foi anteriormente feito para a ordem total, ilustra-se também como é possível obter difusão atómica com base no acordo distribuído. É introduzido o problea da difusão com terminação e apresentado um algoritmo para resolver este problema.

**Aula 7: Registos** É feita a introdução à replicação de dados considerando o tipo de dados mais simples: um registo com apenas um valor. Discute-se qual o comportamento "esperado" do registo, expresso por um modelo de coerência. Neste contexto apresenta-se a noção de registos regulares e atómicos. Discutem-se as dificuldades introduzidas pela replicação do estado do registo em cenários gradualmente mais complexos: apenas um escritor e um leitor, um escritor e vários leitores e, finalmente, vários escritores e vários leitores.

**Aula 8: Acordo em sistemas assíncronos** A aula centra-se sobre a resolução do problema do acordo distribuído em sistemas assíncronos. Uma explicação intuitiva das razões pelos quais o problema é insolúvel em sistemas assíncronos puros é apresentada. Posteriormente, discute-se o modelo de sistemas assíncronos aumentados com detectores de falhas não fiáveis. Definem-se algumas classes de detectores de falhas. Um algoritmo para resolver este problema, baseado no algoritmo de *Paxos* do Lamport é apresentada.

**Aula 9: Acordo bizantino/sincronização de relógios** A aula discute o problema do acordo em sistemas síncronos. Mostra-se que nestes sistemas, é possivel tolerar falhas arbitrárias desde que exista suficiente redundância no sistema. Através de exemplos ilustra-se a dificuldade do problema e descreve-se uma solução para o problema do acordo Bizantino.

A aula faz também uma breve introdução à necessidade e utilidade deste serviço. Mostra-se como é possível que um processo correctos obtenha uma estimativa do valor do relógio noutro processo correcto e como a variação nos atrasos da rede introduzem erros nesta medições. Seguidamente demonstra-se como é possível obter sincronização recorrendo a algoritmos baseados em convergência iterativa e acordo iterativo. Os algoritmos probabilistas são também abordados.

**Aula 10: Transacções atómicas** Os sistemas de transações são obordados no contexto dos sistemas tolerantes a faltas com a capacidade de tolerar falhas e recuperações (*crash-recovery*). A necessidade de salvaguardar o estado do algoritmo em pontos chave da sua execução, de modo a assegurar a correcta recuperação do sistema é discutida. Os sistemas de confirmação atómica em sistemas transaccionais são dados como exemplo concreto de sistemas que utilizam esta técnica.

**Aula 11: Replicação de dados pessimista** Discute-se agora a replicação de dados em sistemas sujeitos a partições. A noção de *quorum* é introduzida. Discute-se a votação ponderada, e as vantagens de atribuir votos diferentes a cada réplica. Métodos sistemáticos de calcular estes votas são referidos de modo superficial. Faz-se também uma pequena panorâmica de outras técnicas que complementam a técnica de votação ponderada, como por exemplo os quorum em grelha.

**Aula 12: Estado coerente** Nesta aula abordam-se as técnicas que permitem fazer recuperação-para-trás, nomeadamente as soluções baseadas em salvaguardas periódicas do estado. Coloca-se o problema do estado global coerente e das técnicas que permitem a sua obtenção. Faz-se a distinção entre o estado global coerente (sem mensagens recebidas mas nunca enviadas) e fortemente coerente (também sem mensagens enviadas mas não recebidas). Discute-se as diferentes alternativas para realizar salvaguardas, em particular a utilização de protocolos coordenados e não ccordenados. Discutem-se também alguns aspectos práticos como qual a periodicidade para fazer salva-

guardas e os mecanismos que permitem capturar o estado dos processos. A utilização de históricos de mensagens, para aleviar o custo de fazer uma salvaguarda total dos processos é também discutida.

**Aula 13: Aplicações** O curso termina com uma breve panorâmica das técnicas utilizadas em sistemas comerciais. São abordadas as seguintes aplicações: sistemas CORBA tolerantes a faltas, replicação de bases de dados, sistemas de elevado desempenho e disponibilidade (geralmente designados por *clusters*). A aula ilustra como os conceitos discutidos ao longo do curso são utilizados nestes sistemas.

## 3.2 Sumários das Aulas Teórico-Práticas

**Aula 1: Apresentação da avaliação.** Apresentação dos vários componentes da avaliação. Discussão da calendarização. Panorâmica das ferramentas a utilizar. Apresentação e discussão do enunciado do projecto.

**Aula 2: Como fazer relatórios.** Apresentação da estrutura dos relatórios e artigos científicos. Apresentação dos aspectos mais relevantes na forma e conteúdo deste tipo de textos. Projecção destes conceitos nos objectivos da avaliação prática da cadeira.

**Aula 3: Introdução ao sistema Appia.** É feita uma panorâmica sobre as principais funcionalidades do sistema Appia. Esta aula foi leccionada pelo Dr. Nuno Carvalho.

**Aula 4: Orador convidado.** Devido a uma "ponte" não se realizou a aula originalmente planeada. Os alunos foram convidados a assistir uma palestra do Prof. José Pereira da Universidade do Minho, sobre difusão probabilista.

**Aula 5: Apresentação e discussão do artigo 1.**

**Aula 6: Apresentação e discussão do artigo 2.**

**Aula 7: Consenso ("*Kata*").** A aula apresentou um exercício que consiste na resolução do consenso com um detector de falhas forte (isto é, um detector em que se tem a certeza que pelo menos um processo correcto nunca é suspeitado):

**Aula 8: Apresentação e discussão do artigo 3.**

**Aula 9: Transformações com registos ("*Kata*").** A aula apresentou um exercício que consiste em obter registos mais potentes através da composição de registos mais simples. Por exemplo, como transformar vários registos (1-N)-Atómico num registo (N-N)-Atómico.

**Aula 10: Apresentação e discussão do artigo 4.**

**Aula 11: Acrodo probabilista.** Nesta aula foi apresentado um protocolo de acordo distribuído probabilista. Mostrou-se a importância da utilização da aleatoriedade para assegurar a terminação do algoritmo com probabilidade que tende para 1 (com o aumento do número de turnos).

**Aula 12: Discussões.** Visualização e discussão dos trabalhos.

# 4 Exemplos de Exames

Os exames anteriores são fornecidos a título de exemplo. Os alunos não devem inferir que o exame deste ano deve necessariamente seguir fielmente este modelo. Saliente-se que a componente teórico-prática da cadeira tem vindo a ser reforçada de modo sustentado ao longo dos anos. Do mesmo modo, o material de estudo que suporta a aprendizagem dos algoritmos tem vindo a ser melhorado. Existe pois uma tendência natural para reforçar a avaliação deste componente na avaliação localizada.

## 4.1 Época de 1999-2000

**Conceitos**

**Questão 1** *(*1 valor*) Diga que entende por* falta, erro *e* falha

Duas técnicas básicas de tolerância a faltas são a "recuperação para trás" e o *mascaramento de faltas*.

**Questão 2** *(*2 valores*) Diga o que entende por cada uma destas técnicas. Use pelo menos um exemplo para ilustrar o seu modo de funcionamento.*

Existem dois modelos extremos para caracterizar o comportamento de um sistema distribuído no domínio do tempo: os sistemas assíncronos e os sistemas síncronos.

**Questão 3** *(*1 valor*) Como é que o problema da detecção de falhas se relaciona com estes modelos?*

**Comunicação em grupo**

Considere que replicava um determinado serviço usando comunicação em grupo. Considere também que a satisfação de um determinado pedido depende do estado do sistema (por exemplo, reserva de bilhetes). É também possível testar a possibilidade de satisfazer o pedido sem o executar. O pseudo-código do servidor é apresentado de seguida:

```
quando recebe M ponto-a-ponto do cliente:
caso M seja:
  inicia-transaccao:
    lista = novalista();
    envia-cliente (ok);
  pedido (p):
    se (testapedido(p)==OK)
        lista = lista+p;
        envia-cliente (ok);
    senao
        envia-cliente (aborta);
  fim-transaccao:
    envia-grupo (transaccao, lista);

quando recebe M do grupo:
caso M seja:
  transaccao (lista):
    se (para todos p na lista:
        (testapedido(p)==OK))
        executapedido (p);
        envia-cliente (confirma);
    senao
        envia-cliente (aborta);
```

**Questão 4** *(*1 valor*) Considere que possuia três tipos de ordenação de mensagens para a primitiva "envia-grupo": FIFO, causal e total. Que qualidade de serviço seria necessário usar neste caso? Justifique?*

**Questão 5** *(*1 valor*) É possível que um cliente receba ok a todos os seus pedidos e depois receba um "aborta" quando solicita o fim da transacção? Justifique.*

**Questão 6** *(*2 valores*) Proponha uma solução em que os clientes enviam os pedidos em difusão para todos os elementos do grupo.*

**Questão 7** *(*1 valor*) Compare a sua solução com a solução anterior.*

**Sincronização de relógios**

Considere um sistema com três nós, um dele exibindo falhas arbitrárias. O valor dos relógios de cada um dos nós é o seguinte: $N1 = 10$, $N2 = 12$, $N3 =$bizantino.

Não considere o erro de leitura dos relógios remotos (isto é, considere que consegui ler cada um destes relógio com a máxima precisão). Naturalmente, não se sabe à partida qual dos relógio apresenta um comportamento Bizantino. Pressuponha também que o algoritmo escolhido para realizar a sincronização de relógios é o seguinte: descarta os dois valores extremos e escolhe a média dos valores restantes.

**Questão 8** *(*1 valor*) Mostre através de um exemplo concreto, porque é que o nó bizantino pode impedir os nós correctos de aumentarem a precisão dos seus relógios.*

Considere agora que tinha quatro nós, com os seguintes valores: $N1 = 10$, $N2 = 11$, $N3 = 12$, $N4 =$bizantino e que aplica o mesmo algoritmo.

**Questão 9** *(*1 valor*) No pior caso, quais os valores possíveis que resultam no afastamento máximo entre os relógio após a sincronização?*

**Questão 10** *(*1 valor*) Como é que o problema do acordo bizantino pode ser aplicado à sincronização de relógios?*

**Replicação de dados/votação**

Considere um sistema que mantém dados replicados usando uma técnica de votação ponderada. Assuma que existem 5 réplicas com os seguintes pesos: R1=1, R2=2; R3=3, R4=3, R5=2.

**Questão 11** *(*1 valor*) Qual é o quorum de escrita mínimo? Justifique.*

**Questão 12** *(*1 valor*) Se o quorum de escrita fosse 8, qual seria o quorum de leitura mínimo?*

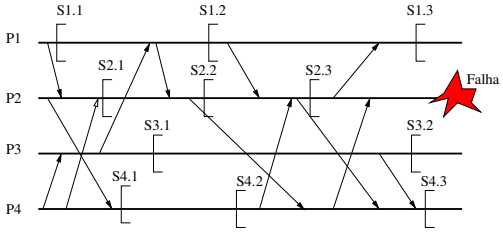
**Recuperação/votação**

Considere um sistema com cinco réplicas inicialmente, R1, R2, R3, R4 e R5. Cada réplica mantém um número de versão que é actualizado sempre que se realiza uma actualização. Quando um nó falha o sistema continua com as réplicas disponíveis. Considere que as réplicas falham por ordem (R1 falha primeiro e R5 em último) e recuperam de acordo com a seguinte sequência: R1, R5, R2, R3 e R4.

**Questão 13** *(*2 valores*) Será possível iniciar o sistema a partir da versão mais recente sem esperar que todas as réplicas recuperem? Descreva as estruturas de dados nencessários para o efeito.*

### Salvaguardas

A figura seguinte ilustra interacções entre quatro processos (P1,P2; P3 e P4) e diversas salvaguardas locais a cada processo.

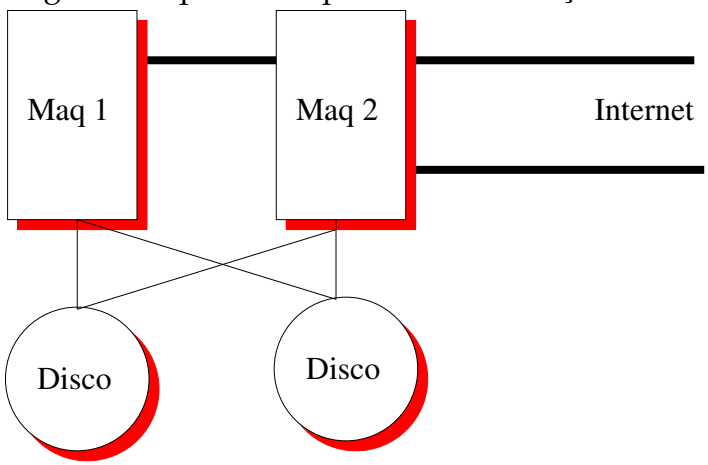



**Questão 14** *(*1 valor*) Após a falha do processo P2, diga qual o conjunto de salvaguardas coerentes para o qual seria possível retroceder (rollback).*

**Questão 15** *(*1 valor*) Diga quais as diferenças entre um sistema de salvaguarda distribuído coordenado e não coordenado.*

### "Clusters"

Considere a seguinte arquitectura para a concretização de um "cluster":




**Questão 16** *(*2 valores*) Diga como é que uma base de dados poderia explorar esta arquitectura para oferecer um serviço de elevada disponibilidade.*

## 4.2 Época de 2000-2001

### Conceitos

**Questão 1** *(*2 valores*) Defina os seguintes conceitos: fiabilidade (*reliability*),* maintainability*, disponibilidade (*availability*) e segurança (*safety*).*

**Questão 2** *(*2 valores*) Qual é a diferença entre o processamento de erros baseado em detecção e recuperação e o processamento de erros baseado em mascaramento de erros.*

### Acordo distribuído

Chandra e Toueg definiram uma hieraquia de detectores de falhas baseados em propriedades de exactidão e plenitude. Demonstraram também que o detector de falhas mais fraco que permite resolver o problema do consenso distribuído num sistema assíncrono é o detector $\diamond S$ com as seguintes propriedades:

- Exactidão fraca alguma-vez: existe um momento a partir do qual um processo correcto não é marcado como falhado por nenhum processo correcto.
- Plenitude fraca: Uma falha é detectada por pelo menos um processo correcto.

**Questão 3** *(*1 valor*) Defina o problema do acordo distribuído.*

**Questão 4** *(*1 valor*) Dê uma explicação intuitiva porque é que o detector $\diamond S$ permite resolver o acordo em sistemas assíncronos.*

### Comunicação em grupo

Considere que concretiza um servidor replicado usando uma estratégia de replicação activa.

Os clientes interagem com o servidor do seguinte modo: obtêm de um servidor de nomes uma lista de réplicas; enviam o pedido através de uma mensagem ponto-a-ponto para uma réplica e esperam uma resposta durante um período de tempo pré-definido; caso a resposta não chegue, voltam a fazer o pedido para outra réplica (e assim sucessivamente até obterem a resposta). Cada cliente só faz um pedido de cada vez.

Do lado do servidor, os pedidos dos clientes são interceptados por uma camada de coordenação que funciona do seguinte modo: o pedido do cliente é interceptado; o pedido é reencaminhado para todos os membros do grupo usando uma primitiva de comunicação em grupo com qualidade de serviço $X$; quando o pedido é recebido através do sistema de comunicação em grupo é entregue à aplicação ficando registado que o seu processamento está em curso; quando a aplicação responde, a resposta é memorizada num registo e enviada ao cliente; se um pedido é recebido uma segunda vez (quer directamente de um cliente, quer através do grupo, este é descartado, sendo a resposta, caso já exista, retransmitida para o cliente).

Considere as seguintes qualidades de serviço possíveis: causal, total e causal-total.

**Questão 5** *(*1.5 valores*) Descreva de um modo breve o serviço prestado por cada uma das qualidades de serviço atrás referidas.*

**Questão 6** *(*1 valor*) Qual das qualidades de serviço usaria no exemplo anterior (X)? Justifique.*

Considere agora que fazia a seguinte optimização: os pedidos podiam-se distinguir entre pedidos de escrita e de leitura; neste caso os pedidos de leitura são executados apenas pela réplica que recebe o pedido vindo do cliente, não ficando o seu processamento registado.

**Questão 7** *(*1 valor*) Caso não tivesse assegurada a ordem causal em todo o sistema, descreva através de uma execução concreta um exemplo de um resultado incorrecto que poderia ser fornecido ao cliente.*

**Questão 8** *(*1 valor*) Como poderia alterar o protocolo para poder executar a optimização e garantir um resultado correcto mesmo sem ter ordem causal oferecida ao nível do protocolo de comunicação?*

### Sincronização de relógios

Considere o seguinte algoritmo de sincronização de relógios: cada processo envia para todos os outros o valor do seu relógio; depois de receber todos os valores cada processo descarta os $f$ valores mais altos e os $f$ valores mais baixos e escolhe o processo que possui o valor mediano como o relógio de referência; cada processo ajusta o seu relógio de modo a ficar com um valor próximo ao do relógio de referência escolhido no passo anterior.

Assuma que possui $3f + 1$ processos e que $f$ processos podem falhar, avançando a uma taxa superior ou inferior à taxa correcta mas não têm um comportamento Bizantino.

**Questão 9** *(*1 valor*) Este algoritmo permite sincronizar os relógios? Justifique.*

**Questão 10** *(*1 valor*) Neste algoritmo, quais os factores que impedem que os relógios fiquem com os relógios completamente sincronizados? Indique o impacto destes factores na precisão obtida pelo algoritmo.*

**Questão 11** *(*1 valor*) Se os processos falhados pudessem ter um comportamento Bizantino, enviando um valor diferente para cada um dos restantes processos, que alterações seria necessário fazer ao algoritmo?*

### Replicação de dados/votação

Considere que possui 25 réplicas e que pretende executar um algoritmo de votação baseado em quorums.

**Questão 12** *(*0.5 valor*) Se o quorum de escrita for* 20 *qual o quorum mínimo de leitura?*

**Questão 13** *(*0.5 valor*) Qual o quorum de escrita mínimo?*

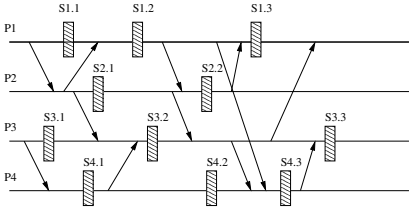
Considere agora que organizava as réplicas numa estrutura lógica em forma de um quadrado (5 por 5).

**Questão 14** *(*1 valor*) Diga como é possível explorar esta estrutura lógica para definir um sistema de quorums que permita ter um quorum de escrita inferior ao indicado anteriormente.*

**Questão 15** *(*0.5 valor*) O que é que se perdeu para conseguir diminuir o tamanho dos quorums?*

### Salvaguardas

A figura seguinte ilustra interacções entre quatro processos (P1,P2; P3 e P4) e diversas salvaguardas locais a cada processo.

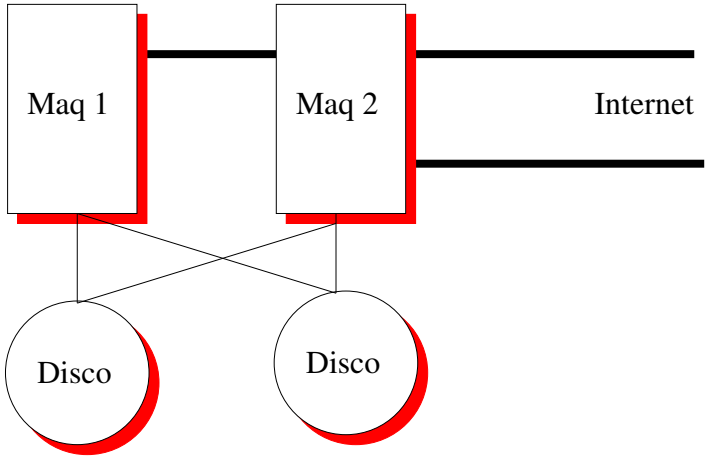


**Questão 16** (1 valor) *Após a falha do processo P2, diga qual o conjunto de salvaguardas coerentes para o qual seria possível retroceder (rollback).*

**Questão 17** (1 valor) *Diga quais as diferenças entre um sistema de salvaguarda distribuído coordenado e não coordenado.*

**"Clusters"**

Considere a seguinte arquitectura para a concretização de um "cluster":




**Questão 18** (2 valores) *Diga como esta arquitectura poderia ser utilizada para concretizar um servidor WWW de elevado desempenho (pode acrescentar os componentes que considere necessários à arquitectura).*

## 4.3 Época de 2001-02

**Conceitos**

**Questão 1** (1.5 valores) *Diga que entende por* falta, erro *e* falha

**Questão 2** (1.5 valores) *Defina os seguintes conceitos: fiabilidade* (reliability), maintainability, *disponibilidade* (availability) *e segurança* (safety).

**Acordo distribuído**

**Questão 3** (1 valor) *Defina o problema do acordo distribuído.*

**Questão 4** (1 valor) *Dê uma explicação intuitiva porque é que o detector* $\diamond S$ *permite resolver o acordo em sistemas assíncronos.*

### Difusão em grupo fiável

**Questão 5** *(*1 valor*) Quando se fala de difusão fiável, distingue-se geralmente a difusão fiável uniforme da difusão fiável não-uniforme. Qual é a diferença entre estas duas primitivas?*

Considere um sistema assíncrono. Considere também que o sistema possui $N$ processos interligados por canais ponto-a-ponto fiáveis. Finalmente considere que no máximo $f < N/2$ processos podem falhar.

**Questão 6** *(*2 valores*) Apresente em pseudo-código, um algoritmo que concretize a difusão fiável uniforme.*

**Questão 7** *(*1 valor*) Caracterize o serviço geralmente designado por sincronia virtual.*

### Ordenação de mensagens

Considere que utiliza um relógio lógico para ordenar de modo causal as mensagens trocadas em difusão num grupo de processos.

**Questão 8** *(*2 valores*) Apresente em pseudo-código, um algoritmo que concretize ordenação causal com base num relógio lógico de Lamport.*

**Questão 9** *(*1 valor*) O que teria de alterar para passar a utilizar relógios vectoriais em vez de um simples relógio lógico?*

### Utilização da comunicação em grupo

Considere a seguinte aplicação que funciona segundo um modelo cliente servidor, usando comunicação ponto-a-ponto (pap):

```
Cliente:
  operacao iniciar
    s = escolhe-contacto ();
  fim iniciar;

  operacao ler (int item)
    envia-pap (s, LER, item);
    recebe-pap (valor);
    retorna valor;
  fim ler;

  operacao escrever (int item, int valor)
    envia-pap (s, ESC, item, valor);
```

```
  fim escrever;
fim cliente;

Servidor:
  operacao iniciar
    para i := 1 ate MAXITEM
      dados[i] := 0;
  fim iniciar;

  operacao ler (upcall)
    recebe-pap (LER, item);
    envia-pap (dados[i])
  fim ler;

  operacao escrever (upcall)
    recebe-pap (ESC, item, valor);
    dados[i] := valor;
  fim escrever;
fim servidor;
```

Considere que replicava este serviço usando comunicação em grupo. Considere que o pacote de comunicação em grupo lhe oferecia as seguintes funções:

```
envia-grupo (ordem, mensagem);
recebe-grupo (mensagem); (upcall)
recebe-vista (lista-de-membros); (upcall)
```

O parâmetros ordem na função envia-grupo pode assumir as qualidades de serviço: causal e causal-total.

**Questão 10** *(*2 valores*) Considerando que pode alterar tanto os servidores como os clientes (fazendo com que estes se juntem a um grupo), utilize o serviço de grupos para que a aplicação seja tolerante a faltas.*

**Questão 11** *(*1 valor*) Proponha um algoritmo que permita acrescentar mais elementos ao grupo sem parar os clientes.*

### Sincronização de relógios

Considere o seguinte algoritmo de sincronização de relógios: cada processo envia para todos os outros o valor do seu relógio; depois de receber todos os valores cada processo descarta os $f$ valores mais altos e os $f$ valores mais baixos e escolhe o processo que possui o valor mediano como

o relógio de referência; cada processo ajusta o seu relógio de modo a ficar com um valor próximo ao do relógio de referência escolhido no passo anterior.

Assuma que possui $3f + 1$ processos e que $f$ processos podem falhar, avançando a uma taxa superior ou inferior à taxa correcta mas não têm um comportamento Bizantino.

**Questão 12** *(*1 valor*) Se os processos falhados pudessem ter um comportamento Bizantino, enviando um valor diferente para cada um dos restantes processos, que alterações seria necessário fazer ao algoritmo?*

### Replicação de dados/votação

Considere que possui 20 réplicas e que pretende executar um algoritmo de votação baseado em quorums.

**Questão 13** *(*0.5 valor*) Se o quorum de escrita for* 15 *qual o quorum mínimo de leitura?*

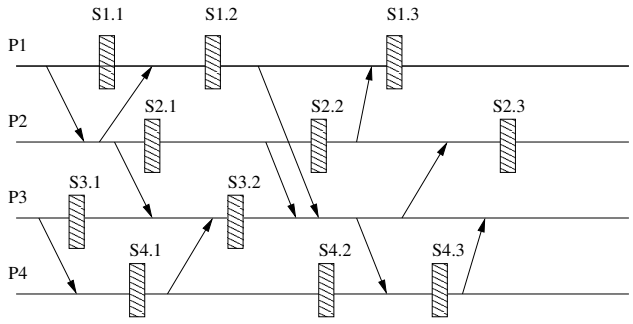
**Questão 14** *(*0.5 valor*) Qual o quorum de escrita mínimo?*

Considere agora que organizava as réplicas numa estrutura lógica em forma de uma matriz de 4 por 5.

**Questão 15** *(*1 valor*) Diga como é possível explorar esta estrutura lógica para definir um sistema de quorums que permita ter um quorum de escrita inferior ao indicado anteriormente.*

### Salvaguardas

A figura seguinte ilustra interacções entre quatro processos (P1,P2; P3 e P4) e diversas salvaguardas locais a cada processo.



**Questão 16** *(*1 valor*) Após a falha do processo P3, diga qual o conjunto de salvaguardas coerentes para o qual seria possível retroceder (rollback).*

**Questão 17** *(*1 valor*) Diga quais as diferenças entre um sistema de salvaguarda distribuído coordenado e não coordenado.*

## 4.4 Época de 2002-03

### Conceitos

**Questão 1** *(*1.5 valores*) Diga qual a diferença entre fiabilidade (*reliability*) e disponibilidade (*availability*) e como estas métricas se relacionam.*

**Questão 2** *(*1.5 valores*) Diga qual a diferença entre recuperação para trás (*backward recovery*) e para a frente (*forward recovery*).*

### Acordo distribuído

**Questão 3** *(*2 valores*) Existem duas definições possíveis para o problema do acordo distribuído: o acordo regular e o acordo uniforme. Qual a diferença entre estas duas definições?*

**Questão 4** *(*3 valores*) Considere que possui um sistema que oferece três serviços: um serviço de detecção de falhas não fiável (através de uma primitiva* suspeita*), um serviço de difusão fiável uniforme (através de primitivas* envia *e* entrega*) e um serviço de acordo uniforme (através de primitivas* propõe *e* decide*). Através de pseudo-código, indique como poderia resolver o problema da confirmação atómica distribuída usando estes serviços.*

### Replicação usando difusão em grupo fiável

Considere um grupo de processos que necessita de aceder a um recurso partilhado. Para sincronizarem o acesso a este recurso, devem comunicar entre si de modo a concretizar um trinco. Este trinco possui uma interface com as seguintes duas primitivas: *lock* e *unlock*. A primeira primitiva é bloqueante e só retorna quando o recurso é entregue ao processo.

Considere que, para tolerância a faltas, o serviço de trincos é concretizado fazendo com que todos os processos tenham uma cópia local do estado do trinco. Pressuponha também que é possível concretizar um detector de falhas perfeito e, consequentemente, libertar o trinco quando o processo que detêm o trinco falha.

Considere que para resolver este problema possui um serviço de comunicação em grupo que oferece sincronia virtual. A interface deste serviço baseia-se nas seguintes primitivas: *joinrequest* (para se juntar ao grupo), *view* (entrega uma vista de grupo), *data-send* (difusão fiável) e *data-deliver* (entrega de mensagens). Para além destas primitivas, existe também um serviço de difusão atómica (*atomic-send* e *atomic-deliver*).

**Questão 5** *(*1 valor*) Caracterize o serviço geralmente designado por sincronia virtual.*

**Questão 6** *(*4 valores*) Através de pseudo-código, diga como poderia replicar o serviço de trincos, usando para esse efeito as primitivas do serviço de comunicação em grupo.*

### Replicação de dados/votação

Considere que possui 30 réplicas e que pretende executar um algoritmo de votação baseado em quorums.

**Questão 7** *(*1 valor*) Se o quorum de escrita for* 20 *qual o quorum mínimo de leitura?*

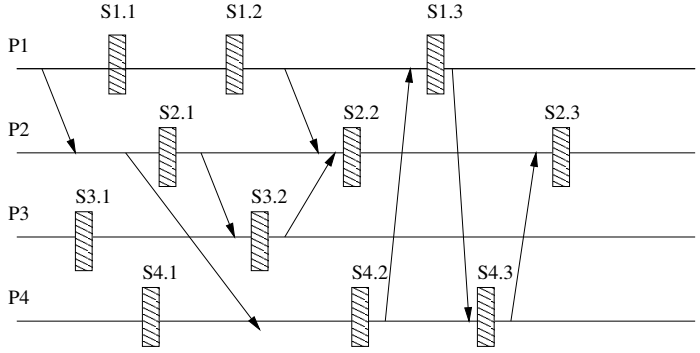
**Questão 8** *(*1 valor*) Qual o quorum de escrita mínimo?*

Considere agora que organizava as réplicas numa estrutura lógica em forma de uma matriz.

**Questão 9** *(*1 valor*) Seria possível explorar esta estrutura para ter um quorum de leitura de* 6 *e um quorum de escrita de* 10*?*

### Salvaguardas e recuperação

A figura seguinte ilustra interacções entre quatro processos (P1,P2; P3 e P4) e diversas salvaguardas locais a cada processo.



**Questão 10** *(*1 valor*) Após a falha do processo P1, diga qual o conjunto de salvaguardas coerentes para o qual seria possível retroceder (rollback).*

**Questão 11** *(*1 valor*) Diga quais as diferenças entre um sistema de salvaguarda distribuído coordenado e não coordenado.*

**Questão 12** *(*2 valores*) Uma das técnicas de tolerância a faltas mais comuns em sistemas tipo* cluster *é designada por* failover*. Explique em que medida a utilização de discos partilhados por diferentes nós de um cluster facilita a concretização desta técnica.*

## 4.5 Época de 2003-04

### Conceitos

**Questão 1** *(*1 valor*) Diga qual a diferença entre fiabilidade (*reliability*) e disponibilidade (*availability*) e como estas métricas se relacionam.*

**Questão 2** *(*1 valor*) O que entende por "confiança no funcionamento"?*

### Difusão fiável

Considere o seguinte algoritmo:

---

**Uses:**
BestEffortBroadcast (beb).
PerfectFailureDetector ($\mathcal{P}$).

**upon event** ⟨ *Init* ⟩ **do**
delivered := ∅; correct := Π; $\forall_{p_i \in \Pi}$ : from[$p_i$] := ∅;

**upon event** ⟨ *rbBroadcast*, $m$ ⟩ **do**
**trigger** ⟨ *bebBroadcast*, [DATA, self, $m$] ⟩;

**upon event** ⟨ *bebDeliver*, $p_i$, [DATA, $s_m$, $m$] ⟩ **do**
**if** $m \notin$ delivered **then**
delivered := delivered ∪ {$m$}
**trigger** ⟨ *rbDeliver*, $s_m$, $m$ ⟩;
from[$p_i$] := from[$p_i$] ∪ {[$s_m$, $m$]}
**if** $p_i \notin correct$ **then trigger** ⟨ *bebBroadcast*, [DATA, $s_m$, $m$] ⟩;

**upon event** ⟨ *crash*, $p_i$ ⟩ **do**
correct := correct \ {$p_i$}
**forall** $[s_m, m] \in from[p_i]$: **do**
**trigger** ⟨ *bebBroadcast*, [DATA, $s_m$, $m$] ⟩;

---

**Questão 3** *(*1 valor*) Diga, justificando, se este algoritmo concretiza a difusão fiável regular ou uniforme.*

**Questão 4** *(*2 valores*) Se respondeu regular, altere o algoritmo para concretizar a difusão fiável uniforme. Se respondeu uniforme, altere o algoritmo para concretizar a difusão fiável regular.*

### Acordo distribuído

**Questão 5** *(*2 valores*) Diga o que entende por um detector de falhas não fiável, e qual o seu papel na resolução do problema do acordo em sistemas assíncronos.*

**Questão 6** *(*3 valores*) Considere que possui um sistema que oferece três serviços: um serviço de detecção de falhas não fiável (através de uma primitiva* suspeita*), um serviço de difusão fiável uniforme (através de primitivas* envia *e* entrega*) e um serviço de acordo uniforme (através de primitivas* propõe *e* decide*). Através de pseudo-código, indique como poderia resolver o problema da ordenação total de mensagens usando estes serviços.*

### Replicação usando difusão em grupo fiável

**Questão 7** *(*1 valor*) Caracterize o serviço geralmente designado por sincronia virtual.*

**Questão 8** *(*1 valor*) Explique porque é que muitos serviços de comunicação em grupo obrigam a parar a troca de mensagens durante a instalação de uma nova vista.*

Considere que possui um servidor de um jogo de combate. O jogo suporta vários clientes. Cada cliente envia comandos ao servidor de modo a controlar o seu avatar. O servidor, periodicamente, envia actualizações de estado aos clientes. Para além disto, o servidor também anima personagens que se movem autonomamente.

Pretende replicar o servidor para obter tolerância a faltas.

Considere que lhe são dados os seguintes serviços de comunicação em grupo:

- Um serviço de sincronia virtual, para gerir a filiação e a comunicação entre os servidores.
- Um protocolo de difusão em grupo com ordem total, que pode ser usado para comunicar entre réplicas do servidor e entre os clientes e os servidores (para disseminação dos comandos).

- Um protocolo de difusão fiável sem ordem total que pode ser usado para comunicação entre os servidores e também para os servidores enviarem actualizações de estado para os clientes.
- Um serviço de temporização local, não sincronizado entre as réplicas, que pode ser usado por cada servidor para activar a animação dos personagens não-jogadores.

**Questão 9** *(*3 valores*) Através de pseudo-código, diga como poderia replicar o servidor do jogo.*

### Replicação de dados/votação

Considere que possui 50 réplicas e que pretende executar um algoritmo de votação baseado em quorums (cada réplica possui exactamente um voto).

**Questão 10** *(*1 valor*) Se o quorum de escrita for* 30 *qual o quorum mínimo de leitura?*

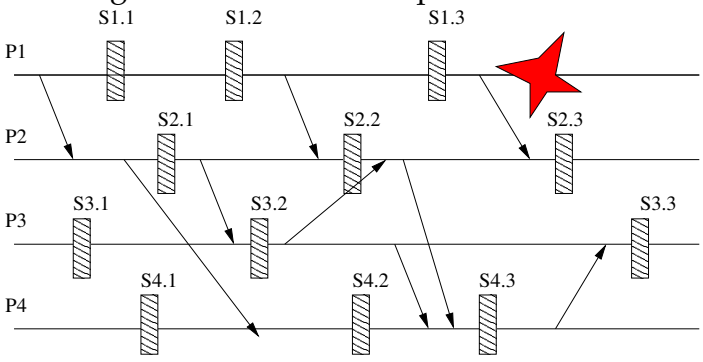
**Questão 11** *(*1 valor*) Qual o quorum de escrita mínimo?*

Considere agora que organizava as réplicas numa estrutura lógica em forma de uma matriz.

**Questão 12** *(*1 valor*) Seria possível explorar esta estrutura para ter um quorum de leitura de* 5 *e um quorum de escrita de* 14*?*

### Salvaguardas e recuperação

A figura seguinte ilustra interacções entre quatro processos (P1,P2; P3 e P4) e diversas salvaguardas locais a cada processo.




**Questão 13** *(*1 valor*) Após a falha do processo P1, diga qual o conjunto de salvaguardas coerentes para o qual seria possível retroceder (rollback).*

**Aplicações**

Considere o seguinte algoritmo para replicação de bases de dados.

- processamento-local ($t$):
  1. Obter a lista $l$ de objectos acedidos pela transacção.
  2. Obter um trinco de leitura para cada objecto em $l$. Se algum desses objectos está trancado para escrita, a transacção é posta em espera até que esse objecto seja libertado.
- confirmar($t$):
  1. Obter a lista de objectos escritos ($\mathrm{WS}_t$) por $t$.
  2. Comunicar $< t, \mathrm{WS}_t >$ usando a **primitiva X**.
  3. Quando a transacção for recebida através da **primitiva X**:
     (a) Tentar obter um trinco de escrita para todos os objectos $o$ constantes em $\mathrm{WS}_t$:
        i. Se existir um trinco de leitura em $o$, a transacção que detém o trinco de leitura é abortada, e o trinco de escrita é concedido a $t$.
        ii. Se existir um trinco de escrita em $o$, ou todos os trincos de leitura em $o$ são de transacções $u$ cuja mensagem $< u, \mathrm{WS}_u >$ já tenha sido recebida, $t$ fica em espera neste objecto até que esses trincos sejam libertos.
        iii. Se não existir mais nenhum trinco em $o$, conceder o trinco a $t$.
     (b) Apenas no nó $N$ onde a transacção foi executadaa localmente $t$, enviar $\textsc{Confirmar}_t$ usando a **primitiva Y**.
  4. Quando é entregue uma mensagem $\textsc{Confirmar}_t$: Confirmar a transacção, escrevendo todas as alterações e libertando todos os trincos adquiridos por $t$. Todas as transacções que estavam à espera de obter um trinco de escrita em algum objecto alterado por $t$ são abortadas (se a transacção for local, enviar $\textsc{Abortar}_t$ usando a **primitiva Z**).
  5. Quando é entregue uma mensagem $\textsc{Abortar}_t$: Se a transacção é local a mensagem é ignorada, senão aborta $t$, libertando todos os seus trincos.

**Questão 14** *(*1 valor*) Diga quais as abstracções que devem ser concretizadas pelas primitivas **X**, **Y**, e **Z**.*