TOLERÂNCIA A FALTAS DISTRIBUÍDA

2004-2005
13 de Janeiro de 2005

Nota prévia

- O exame tem duração de duas horas e 30 minutos.
- Não se esqueça de identificar *todas* as folhas com o seu nome e número.

Conceitos

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

Questão 2 *(*1.5 valores*) Qual a diferença fundamental entre as técnicas que se designam por "recuperação para trás" (backward error recovery) e "recuperação para a frente" (forward error recovery)?*

Difusão fiável

Considere o seguinte algoritmo que concretiza difusão fiável não uniforme usando um detector de falhas perfeito:

```
01 Uses:
02   BestEffortBroadcast (beb).
03   PerfectFailureDetector (P).

04 upon event ⟨ Init ⟩ do
05   delivered := ∅; correct := Π; ∀_{p_i∈Π} : from[p_i] := ∅;

06 upon event ⟨ rbBroadcast, m ⟩ do
07   trigger ⟨ bebBroadcast, [DATA, self, m] ⟩;

08 upon event ⟨ bebDeliver, p_i, [DATA, s_m, m] ⟩ do
09   if m ∉ delivered then
10        delivered := delivered ∪ {m}
11        trigger ⟨ rbDeliver, s_m, m ⟩;
12        from[p_i] := from[p_i] ∪ {[s_m, m]}
13        if p_i ∉ correct then trigger ⟨ bebBroadcast, [DATA, s_m, m] ⟩;

14 upon event ⟨ crash, p_i ⟩ do
15   correct := correct \ {p_i}
16   forall [s_m, m] ∈ from[p_i]: do
17        trigger ⟨ bebBroadcast, [DATA, s_m, m] ⟩;
```

Questão 3 *(*1 valor*) Ilustre, através de um exemplo concreto, porque é que o algoritmo estaria incorrecto sem a linha 13.*

Questão 4 *(*2 valores*) Altere o algoritmo para concretizar a difusão fiável uniforme sem recorrer a um detector de falhas perfeito. Indique que hipóteses adicionais é necessário fazer acerca do sistema para este problema ter solução.*

Acordo distribuído

O problema da "difusão fiável com terminação" pode ser descrito do seguinte modo. Um processo é o emissor de uma mensagem e todos os processos (incluindo o emissor) são receptores dessa mesma mensagem. Todos os processos invocam o serviço através da primitivas *dft-envia (e, m)*", em que e é o emissor da mensagem (conhecido por todos) e o conteúdo de m pode variar: inclui os dados no emissor ou um valor por omissão ($\perp$) nos restantes processos. Como resultado da invocação o sistema retorna *necessariamente* um resultado através da primitiva *dft-entrega (e, m)*" em que m inclui os dados quando o emissor permance correcto e os dados ou o valor por omissão caso o emissor falhe.

Questão 5 *(*2 valores*) Considere que possui um sistema que oferece três serviços: um serviço de detecção de falhas perfeito (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 difusão fiável com terminação.*

Considere o algoritmo descrito abaixo que concretiza o acordo não uniforme. O algoritmo funciona do seguinte modo: existe uma hieraquia pré-definida entre os processos. Cada processo só decide depois de receber o valor dos seus superiores hierárquicos correctos e, posteriormente, informa todos os seus subalternos da sua decisão.

Implements:
Consensus (c);

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

upon event $\langle$ *Init* $\rangle$ **do**
suspected := $\emptyset$; round := 1;
proposal := *nil*; prop-round :=0;
for i = 1 **to** N **do**
delivered[i] := broadcast[i] := false;

upon event $\langle$ *crash*, p_i $\rangle$ **do**
suspected := suspected $\cup\{$ *rank*(p_i) $\}$;

upon event $\langle$ *cPropose*, v $\rangle$ **do**
proposal := v;

upon (round = *rank* (self)) $\wedge$ (proposal $\neq$ *nil*) $\wedge$ (broadcast[round] = false) **do**
broadcast[round] := true;
trigger $\langle$ *cDecide*, proposal $\rangle$;
trigger $\langle$ *bebBroadcast*, [DECIDED, round, proposal] $\rangle$;

upon (round $\in$ suspected) $\vee$ (delivered[round] = true) **do**
round := round + 1;

upon event $\langle$ *bebDeliver*, p_i, [DECIDED, r,v] $\rangle$ **do**
if (r < *rank* (self)) $\wedge$ (r > prop-round) **then**
proposal := v; prop-round := r;
delivered[r] := true;

Questão 6 *(*2 valores*) Através de um exemplo concreto, ilustre porque é que o algoritmo falharia se um processo decidisse mal recebesse o valor do seu superior mais graduado.*

Replicação usando difusão em grupo fiável

Questão 7 *(*2 valores*) Caracterize o serviço geralmente designado por sincronia virtual.*

Considere que pretende concretizar um servidor que mantém o estado de uma sala de reuniões partilhada. Os clientes podem fazer duas operações sobre o objecto:

- "estado=ver-estado(intervalo)", que permite ver quais as reservas efectuadas num dado intervalo de tempo;
- "resultado=marcar(utilizador, intervalo)" que permite marcar a sala para um determinado intervalo. Caso dois clientes tentem marcar a sala concorrentemente para o mesmo intervalo, apenas um dos clientes deve obter a reserva.

Como as consultas são mais frequentes que as reservas, o sistema é concretizado mantendo em cada cliente uma réplica do servidor. Para manter o estado coerente, a gestão da replicação pode usar serviços de comunicação em grupo.

Questão 8 *(*3 valores*) Através de pseudo-código, diga como poderia replicar este servidor.*

Replicação de dados/votação

Considere que possui 100 réplicas e que pretende executar um algoritmo de votação baseado em quorums (cada réplica possui exactamente um voto).

Questão 9 *(*1 valor*) Se o quorum de escrita for* 80 *qual o quorum mínimo de leitura?*

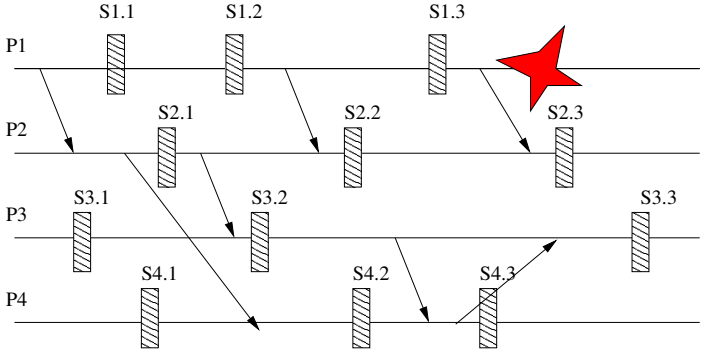
Questão 10 *(*1 valor*) Qual o quorum de escrita mínimo?*

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

Questão 11 *(*1 valor*) Seria possível explorar esta estrutura para ter um quorum de leitura de* 10 *e um quorum de escrita de* 19*?*

Salvaguardas e recuperação

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



Questão 12 *(*1 valor*) Após a falha do processo P1, diga qual o conjunto de salvaguardas fortemente coerentes para o qual seria possível retroceder (rollback).*

Aplicações

Questão 13 *(*1 valor*) O Ethernal é um sistema de que suporta a replicação de objectos CORBA. Este sistema suporta dois tipos de replicação passiva, designados por "warm"e "cold". Quais as diferenças entre estes dois métodos?*