SDIS

Exame 23-11-2017

1.

1.1. Duas vantagens de distribuição na arquitetura de aplicações informáticas passar pelo aumento do poder computacional, pela possibilidade de paralelização do processamento e pela melhor availability e reliability, ou seja, melhor tolerância a falhas pela não existência de um single point of failure.

1.2. Virtualização de recursos consiste na imitação do comportamento de um sistema através da substituição de uma interface.

1.3. Um relógio de Lamport é um relógio lógico, ou seja, é um relógio que apenas tem em conta ordem e não tempo verdadeiro. Assim, um relógio de Lamport conta os eventos ocorridos num processo, incrementando um contador antes de executar cada evento local. Quando um processo recebe uma mensagem, mensagem essa que contém o timestamp do emissor, sincroniza o seu relógio, atribuindo-lhe o valor $L_j = 1 + \max(L_j, t)$, para que seja imposta uma relação de happened-before.

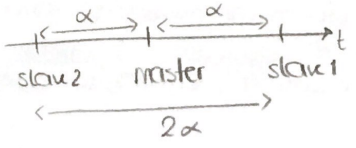
2. $\delta = 3\mu s$ → precisão das medições de correntes e tensões e da atuação das proteções

2.1. O Protocolo PTP, Precision Time Protocol, baseia-se em mensagens de Follow-Up para estimar offset entre um slave e o master e para estimar o atraso introduzido pela rede. Baseia-se, também, em mensagens de Sincronização. Deste modo, apenas as mensagens de Follow-Up transportam timestamp. Com estas mensagens é possível corrigir os relógios dos slaves, a partir das estimativas do delay e do offset em relação ao master.

2.2. $\alpha = 1\mu s$ → exatidão dos relógios dos slaves face ao do master

$$\max_{i,t} |C_i(t) - t| \leq \alpha \quad \text{(exatidão)}$$

$$\max_{i,j,t} |C_i(t) - C_j(t)| \leq \delta \quad \text{(precisão)}$$



A exatidão corresponde à máxima diferença, em valor absoluto, entre o relógio de um slave e o do master, num determinado instante de tempo t.

A precisão corresponde à diferença máxima, em valor absoluto, entre o relógio de um slave e de outro, pelo que está relacionada com o desvio em relação à média, ou seja, desvio padrão.

Assim, como se mostra no esquema, se a exatidão é de α, então é possível concluir-se que, no máximo, a precisão é de 2α, pois, no pior caso, os relógios dos 2 slaves estão afastados do master em α, mas em sentidos contrários, isto é, um dos relógios está atrasado α unidades de tempo face ao master e o relógio do outro slave está adiantado α

unidades de tempo face ao master, pelo que, no máximo estão 2α unidades de tempo afastados entre si, daí a precisão ser de 2α, no máximo.

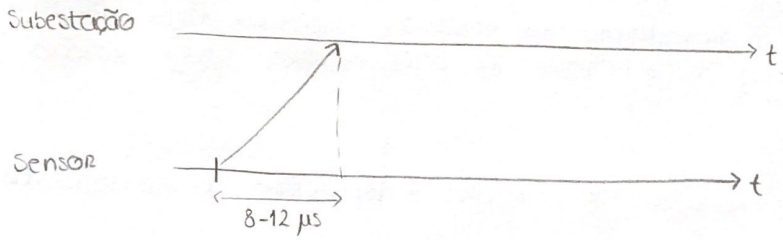
2.3. $T = 200\,\mu s$

O protocolo de transporte mais adequado é TCP, pois garante fiabilidade, isto é, garante que não há perda de mensagens e nem duplicados, o que é essencial nesta aplicação, pois a subestação precisa de receber informação de todos os sensores para que possa atuar nas linhas e evitar sobrecargas assimétricas.

O modelo de interação subjacente é cliente servidor, pois as ações são despoletadas pelos clientes, os sensores, que enviam os seus dados ao servidor, o controlo da subestação, numa comunicação unicast.

2.4. $8\,\mu s < delay < 12\,\mu s$

$\delta = 1\,\mu s$ → precisão dos relógios (offset máximo)



2.5. Consideraria uma arquitetura multi-thread ou event-driven, com uma thread por pedido ou uma FSM (Finite State Machine) por pedido, respetivamente. Assim, é possível tratar atempadamente vários pedidos concorrentes.

2.6. Uma rede overlay, como o próprio nome indica, é uma rede lógica criada por cima de outra já existente (rede física). Essa rede poderia ser implementada com uma spanning tree, cujos nós são membros do grupo multicast e cujas arestas são ligações entre os nós. O nó que quer enviar multicast para o grupo, envia a mensagem para a raiz da rede e a mensagem é propagada usando comunicação unicast entre os ramos, da raiz até às folhas. Assim, a subestação conseguiria comunicar em multicast com as outras subestações.

4.2. A mensagem de HALT faz com que um processo passe para o estado de ELECTION.

Se o processo 5 não recebesse a mensagem de HALT, não iria passar para o estado de ELECTION nem guardar o id do candidato. Assim, manter-se-ia em estado NORMAL e com o nó 2 como líder (5(5).c = 2), proveniente da eleição anterior ao nó 2 falhar. Porém, ao receber as mensagens HALT e LEADER o nó 8 passa ao estado ELECTION e, posteriormente, NORMAL com 5(8).c = 4. Neste caso, a Assertion 1 foi violada, pois existem dois nós no estado NORMAL que NÃO concordam no coordenador.

O mesmo aconteceria se fosse o nó 8 a não receber a mensagem de HALT.

NOTA: a msg de HALT garante que os nós de id + alto (- fortes) estão todos num estado conhecido (de "simulated failure"), para que quando a msg LEADER chega todos os nós mudem para NORMAL e concordem no coordenador, garantindo a não violação da Assertion 1

5.

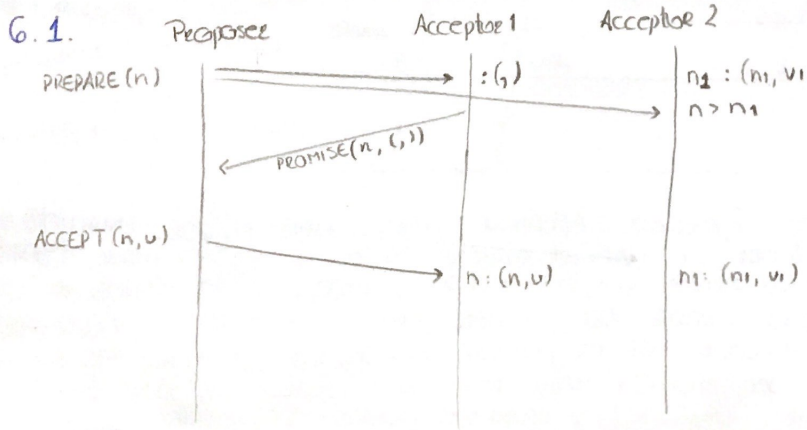
AC3 - Imagine-se o caso em que um processo decidiu commit, que no contexto duma transação distribuída pode, por exemplo, significar que uma das réplicas de uma base de dados executou uma operação, enquanto que as outras réplicas, que não decidiram commit, não a executaram. Esta situação leva à existência de divergência do estado das diferentes réplicas, que se deseja que tenham sempre todas o mesmo estado. Assim, a omissão de AC3 da especificação produziria resultados indesejáveis.

AC4 - Se todos os processos votaram commit, significa que estão todos disponíveis e disponíveis a executar determinada transação. Na ausência de falhas, tendo todos votado commit, o coordenador irá enviar ordem para que seja decidido commit em todos os processos. Não havendo falhas, todos os processos deveriam decidir commit. Não existindo AC4, se um processo, por alguma razão, votasse commit e, posteriormente, não decidisse commit, iria haver inconsistência de estado, pois todos os outros processos iriam decidir commit e executar a transação.

↓
por exemplo, maliciosa

6. Paxos

6.1.



Imagine-se que os acceptors não podiam aceitar mais do que um valor e que A2 já tinha, anteriormente, aceite a proposta n_1, com $n > n_1$. Ao chegar o pedido PREPARE(n), A1 irá aceitá-lo e prometer não aceitar propostas com id menor que n. Apesar de $n_1 < n$, A2 não pode aceitar + propostas, pelo que irá ignorar PREPARE(n). Tendo maioria, P envia ACCEPT(n, v), pelo que A1 escolhe v, mas A2 continua a ter escolhido v_1, não se chegando a um acordo.

6.2. A proposta é válida, pois se uma proposta foi aceite por uma maioria, então, eventualmente, todos os processos irão escolher esse valor, isto é, se um proposer vai enviar um ACCEPT, então é porque uma maioria de processos respondeu ao seu PREPARE, aceitando a sua proposta. Como tal, um valor foi escolhido e será divulgado aos acceptors. Na solução original apenas os acceptors que responderam ao PREPARE com um PROMISE ficam com esta informação. Enviando o ACCEPT a todos os acceptors, independentemente de terem enviado PROMISE ou não, permite que todos sejam informados do valor escolhido, sem comprometer a sua escolha, pelo que a proposta continua a garantir a correção do algoritmo.

Esta proposta tem a vantagem de que um acceptor que tenha enviado PROMISE, mas cuja mensagem se tenha perdido, aceita na mesma o ACCEPT correspondente.

Uma desvantagem passa pelo aumento do número de mensagens trocadas entre processos, pois os acceptors que não enviaram PROMISE, na proposta original, iriam na mesma descobrir o valor escolhido sem ser necessário esta mensagem adicional de ACCEPT(, embora esta mensagem permita que os acceptors apanhem + depressa que um valor foi escolhido e qual esse valor).