8. Tolerância a Falha

Técnica fundamental → Redundância

Um sistema distribuído tolerante a falhas deve apresentar as seguintes características…

- **Disponibilidade:** propriedade relacionada a probabilidade do sistema estar funcionando em um instante qualquer do tempo.

Disponibilidade	Downtime/ano		
99%	3 dias 15:36:00		
99,9%	08:45:36		
99,99%	00:52:34	→	Alta disponibilidade
99,999%	00:05:15	→	Alta disponibilidade

- **Confiabilidade:** refere-se a propriedade de um sistema distribuído poder funcionar continuamente sem falhas. Um sistema de alta confiabilidade é aquele que mais provavelmente continuará a funcionar sem interrupção durante um período de tempo relativamente longo.

	Disponibilidade	Confiabilidade
Sistema que fica offline por 1ms a cada hora	99,9999%	Baixa
Sistema que nunca cai mas sempre é desligado nas duas primeiras semanas do ano	96%	Alta

- **Segurança:** em caso de falha do sistema nada de catastrófico deve ocorrer, como a perda de dados ou vida. A segurança também está relacionada a integridade e proteção dos dados.

  Exemplos: Sistema de gerenciamento de usina nuclear e o sistema de controle da Apolo 11.

- **Capacidade de manutenção:** propriedade relacionada a facilidade com que eventuais falhas são corrigidas.

Tipos de falha…

- Falha por queda (crash): hardware ou software.
- Falha por omissão: buffer de envio/recebimento de mensagens cheio ou erro no canal de comunicação.
- Falha de temporização: tempo máximo de resposta do servidor (threshold) foi atingido.
- Falha de resposta: resposta do servidor incorreta.
- Falha arbitrária (Falha Bizantina): respostas/mensagens que não podem ser detectadas como incorretas e que não deveriam estar sendo produzidas.

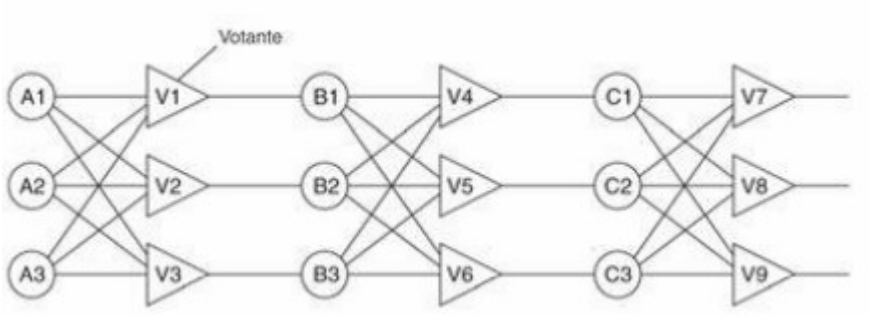
Mascaramento de falha por redundância…

- Redundância de informação: adiciona-se bits extras para permitir a recuperação de dados corrompidos. Exemplo: Sistemas RAID.
- Redundância de tempo: uma ação é executada e, se necessário for, a ação é executada novamente. Exemplo: Uma transação anteriormente abortada.
- Redundância física: ocorre com replicação de componentes de software ou hardware.

A redundância física ocorre…

- Biologia: dois olhos, dois ouvidos.
- Aviões: Um 747 com quatro motores pode voar com apenas três.
- Esportes: Mais de um juiz.
- Circuitos eletrônicos…

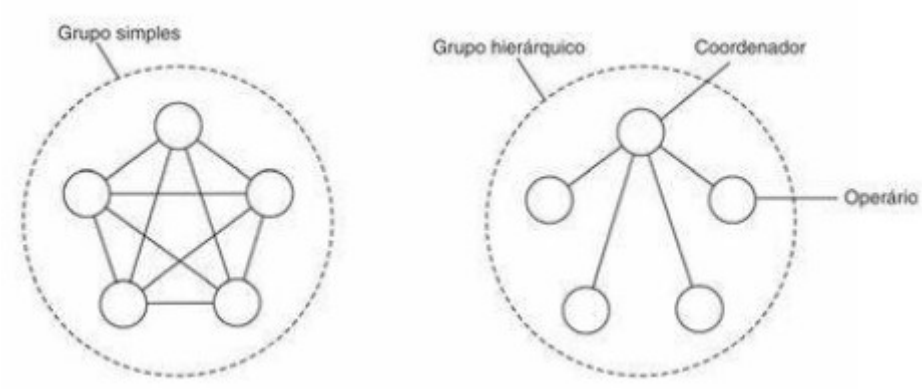
Componentes eletrônicos não redundantes



Redundância modular tripla

Resiliência de processo

Como? Replicação de processos em grupos.



Grupo simples:

- Todos os processos são iguais.
- Não há ponto único de falha.
- A resposta à uma requisição requer votação.

Grupo hierárquico:

- Processos coordenador e operários.
- Ponto único de falha.
- A resposta à uma requisição não requer quórum.

Como detectar a falha de um processo?

- Esgotamento de temporização: enviar uma mensagem do tipo “Hello”. Se o processo destinatário não responder dentro de um limite máximo de tempo, considera-se o processo como não ativo → sujeito a falsos-positivos.
- Troca periódica de mensagens entre todos os processos. Se o processo A recebe uma mensagem de “Hello” de um processo B, o qual recentemente recebeu uma mensagem de “Hello” do processo C, então A sabe que B e C estão ativos mesmo não recebendo uma mensagem de “Hello” diretamente de C.

Comunicação confiável em sistemas distribuídos

Processos podem falhar, mas as vezes o que falha é a comunicação entre processos.

Como obter tolerância a falha de comunicação?

Redundância! TCP é um protocolo de transporte confiável que detecta perdas de pacotes por meio de mensagens de reconhecimento de transmissão.

Mas e se houver uma queda do enlace de comunicação? Redundância física.

Um exemplo com comunicação RPC…

Problemas que podem ocorrer:

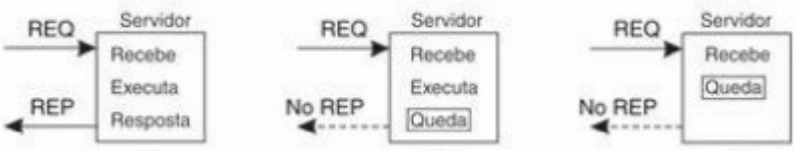
1. Cliente não consegue se conectar ao servidor.

As razões são inúmeras: o servidor pode simplesmente estar *offline*, o enlace de comunicação pode ter caído, a rede pode estar congestionada, a “ligação” (stub) no lado cliente não é compatível com a API atual do servidor, etc.

2. Mensagem de requisição perdida.

O protocolo TCP garante a retransmissão e o correto sequenciamento da mensagem, além do descarte da mesma caso já tenha sido recebida pelo servidor, situação que ocorre quando a mensagem é transmitida mais de uma vez.

3. Servidor cai após receber requisição.



Como saber se a ação foi ou não executada no servidor?

Problemas:

- Se continuarmos tentando enviar novas requisições até obtermos uma resposta, corre-se o risco de executarmos a ação mais de uma vez.

- Se desistirmos logo de início, corre-se o risco de não executarmos a ação nenhuma vez.

4. Mensagem de resposta perdida.

Em caso de não recebimento de uma resposta, pode-se simplesmente refazer a requição para operações do tipo idempotente, como consultas. Se a requisição for uma atualização não idempotente, então talvez seja necessário fazer uma consulta antes para verificar o estado atual do dado.

5. O cliente cai.

Em havendo queda do cliente, a situação mais cômoda é simplesmente o servidor fazer uma desconecção, sem qualquer armazenamento do estado atual do cliente (stateless). O gerenciamento do estado atual do cliente (statefull) pelo servidor leva a sistemas não escaláveis.

Commit distribuído

Em um commit local, o servidor garante que todas as operações pertencentes a uma transação sejam efetivadas com sucesso: a transação se comporta coma uma unidade atômica.

Em um commit distribuído, os servidores devem cooperar e garantir que as operações efetuadas por todos eles devem ser comprometidas com sucesso: a transação distribuída também deve se comportar como uma unidade atômica.

Tipicamente emprega-se um processo coordenador em um commit distribuído.

Protocolo de commit de uma fase (1PC, 1-Phase Commit)

- O coordenador envia uma mensagem a todos os processos participantes da transação distribuída para que as operações sejam efetivadas.
- Se por ventura um dos participantes não puder realizar o commit (por exemplo, por uma violação de integridade ou questões de concorrência), então não é possível notificar o coordenador ou os outros participantes e, neste caso, o comprometimento da transação ocorre parcialmente, o que viola por si só a semântica da própria transação.

Protocolo de commit de duas fases (2PC, 2-Phase Commit)

- Fase 1 – Fase de votação:
  - O coordenador envia uma mensagem VOTE_REQUEST a todos os participantes da transação distribuída.
  - Ao receber a mensagem VOTE_REQUEST, cada participante avalia o estado local das alterações e responde com uma mensagem VOTE_COMMIT ou VOTE_ABORT.
- Fase 2 – Fase de decisão:
  - Se todos os participantes responderem com uma mensagem VOTE_COMMIT, então o coordenador compromete a transação e envia uma mensagem GLOBAL_COMMIT a todos os participantes informando do comprometimento distribuído. Caso um dos participantes tenha respondido com uma mensagem VOTE_ABORT, então o coordenador envia uma mensagem GLOBAL_ABORT a todos os participantes que votaram VOTE_COMMIT informando da necessidade de abortar a transação.
  - Aqueles participantes que votaram VOTE_COMMIT aguardam pela decisão do coordenador: GLOBAL_COMMIT ou GLOBAL_ABORT.

Problema: Em caso de falha…

- Se um dos processos envolvidos falhar, então o protocolo como um todo para, pois eles ficam na espera pela decisão e/ou resposta dos participantes. Neste caso, o coordenador deve optar por, após algum tempo de espera, abortar a transação.
- Se o coordenador parar, então todos os participantes devem parar também. Isso ocorre porque eles não podem tomar uma decisão unilateral de abortar as operações locais pertinentes a transação devido ao fato de que o coordenador pode tê-la comprometido mas falhado logo em seguida.

Protocolo de commit de três fases (3PC, 3-Phase Commit)

- Fase 1 – Fase de votação:
  - O coordenador envia uma mensagem VOTE_REQUEST a todos os participantes da transação distribuída.
  - Ao receber a mensagem VOTE_REQUEST, cada participante avalia o estado local das alterações e responde com uma mensagem VOTE_COMMIT ou VOTE_ABORT.
- Fase 2 – Fase de decisão:
  - Se todos os participantes responderem com uma mensagem VOTE_COMMIT, então o coordenador envia uma mensagem PREPARE_COMMIT a todos os participantes notificando que um commit distribuído será iniciado. Caso um dos participantes tenha respondido com uma mensagem VOTE_ABORT, então o coordenador envia uma mensagem GLOBAL_ABORT a todos os participantes que votaram VOTE_COMMIT informando da necessidade de abortar a transação.
  - Aqueles participantes que votaram VOTE_COMMIT aguardam pela decisão do coordenador: PREPARE_COMMIT ou GLOBAL_ABORT.
- Fase 3 – Fase de comprometimento final:
  - Se todos os participantes responderem a mensagem PREPARE_COMMIT, então o coordenador compromete a transação e envia uma mensagem GLOBAL_COMMIT a todos os participantes informando do comprometimento distribuído.