URI | UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES

30-769

Sistemas Distribuídos

MSc. Fernando Schubert

TOLERÂNCIA A FALHAS

O QUE VAMOS APRENDER?

- INTRODUÇÃO A TOLERÂNCIA A FALHAS
- TIPOS DE FALHAS
- MODELOS DE FALHA
- MASCARAMENTO DE FALHAS
- ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS
- RECUPERAÇÃO DE FALHAS

INTRODUÇÃO

- Uma característica de sistemas distribuídos que os distingue de sistemas de uma máquina é a noção de falha parcial
- Uma falha parcial pode acontecer quando um componente de um SD falha
- Esta falha pode afetar a operação de alguns componentes e ao mesmo tempo deixar outros completamente ilesos
- Uma falha em um sistema de uma máquina só é quase sempre total se afeta todos os componentes

INTRODUÇÃO

- Um objetivo importante no projeto de um sistema distribuído:
  - Construir o sistema de modo que ele possa se recuperar automaticamente de falhas parciais sem afetar seriamente o desempenho global.
  - Sempre que ocorrer uma falha o SD deve continuar a funcionar de maneira aceitável enquanto o sistema se recupera
    - O sistema deve ser tolerante a falhas

INTRODUÇÃO

- Tolerância a falhas está fortemente relacionada a sistemas confiáveis.
- Um sistema confiável abrange uma série de requisitos:
    - Disponibilidade
    - Confiabilidade
    - Segurança
    - Capacidade de Manutenção

INTRODUÇÃO

- Disponibilidade:
  - É a propriedade de um sistema estar pronto para ser usado imediatamente
  - Um sistema de alta disponibilidade é aquele que mais provavelmente estará funcionando em dado instante
- Confiabilidade:
  - É a propriedade do sistema funcionar continuamente sem falhas
  - Um sistema de alta confiabilidade continuará a executar sem interrupção durante um período de tempo longo
  - Tempo médio entre falhas.

INTRODUÇÃO

- Disponibilidade x Confiabilidade:
  - Disponibilidade é definida em termos de instante de tempo e confiabilidade em termos de intervalo de tempo
  - Um sistema muito confiável é aquele que continua a trabalhar sem interrupção durante um período relativamente longo de tempo
  - Se um sistema ficar fora do ar por um milissegundo a cada hora, terá uma disponibilidade de mais de 99.99999%, mas sua confiabilidade ainda será muito baixa
  - Por outro lado, um sistema que nunca cai mas é desligado por duas semanas no ano, tem alta confiabilidade, mas somente 96% de disponibilidade

INTRODUÇÃO

- Segurança:
  - Refere-se a situação que se o sistema deixar de funcionar corretamente por um certo tempo nada catastrófico acontecerá
- Capacidade de Manutenção:
  - Refere-se a facilidade com que um sistema que falhou pode ser consertado
  - Sistemas de alta capacidade de manutenção também podem mostrar alto grau de disponibilidade, em especial se as falhas puderem ser detectadas e reparadas automaticamente

INTRODUÇÃO

- Falhas x Erros x Defeitos:
  - **Defeito**: se um sistema não pode cumprir suas promessas, apresenta defeito
  - **Erro**: parte do estado de um sistema que pode levar a um defeito
    - Exemplo: Pacotes danificados transmitidos
  - **Falha**: é a causa de um erro
    - Um meio de transmissão errado ou ruim pode danificar pacotes, neste caso fácil reconhecer a falha
    - Alguns erros de transmissão podem ser causados por más condições atmosféricas difícil remover a falha
  - **Exemplo**:
    - Um programa que trava porque executou código com bug do programador
    - O programador é a falha do erro que levou ao defeito do programa

TIPOS DE FALHAS

- Falha transiente:
  - Ocorre uma vez e desaparece
  - Se a operação for repetida a falha não acontecerá novamente
    - Pássaro voando na frente de um feixe de micro-ondas interrompe a transmissão

TIPOS DE FALHAS

- Falha intermitente:
  - Ocorre, para por um período indeterminado, reaparece, e assim por diante
  - São difíceis de diagnosticar
  - Mau contato

TIPOS DE FALHAS

- Falha permanente:
  - Continua a existir até que o componente faltoso seja substituído
  - Chip queimado
  - Mais fácil de ser diagnosticada

MODELOS DE FALHAS

- Um sistema que apresenta defeito não fornece seus serviços adequadamente → Como encontrar o problema?
  - Nem sempre o servidor que está funcionando mal é a falha que está se procurando
  - Se o servidor depende de outros servidores, por exemplo, pode ser que a falha esteja em outro lugar
- Tais relações de dependência acontecem muito em sistemas distribuídos
  - Um disco defeituoso em um servidor de arquivos que faz parte de um banco de dados distribuído → pode comprometer o funcionamento adequado de todo o banco

MODELOS DE FALHAS

- Para melhor identificar as falhas, foram desenvolvidos diversos esquemas de classificação:

Tipo de falha	Descrição
Falha por queda	O servidor pára de funcionar, mas estava funcionando corretamente até parar.
Falha por omissão	O servidor não consegue responder a requisições que chegam
Omissão de recebimento	O servidor não consegue receber mensagens que chegam
Omissão de envio	O servidor não consegue enviar mensagens
Falha de temporização	A resposta do servidor se encontra fora do intervalo de tempo
Falha de resposta	A resposta do servidor está incorreta
Falha de valor	O valor da resposta está errado
Falha de transição de estado	O servidor se desvia do fluxo de controle correto
Falha arbitrária	Um servidor pode produzir respostas arbitrárias em momentos arbitrários

Tabela 8.1 Diferentes tipos de falhas.

MODELOS DE FALHAS

- Falha por queda (fail-stop):
  - Um servidor para de funcionar, mas estava funcionando corretamente até sua parada.
  - Exemplo: Um SO que para em um estado que somente um reboot possa fazer ele voltar a funcionar

MODELOS DE FALHAS

- Falha por omissão:
  - Isso ocorre quando o servidor falha em responder a solicitações dos clientes ou falha em receber mensagens ou em enviar mensagens
  - Razões:
    - Omissão-recebimento: A conexão entre cliente e servidor foi estabelecida corretamente, mas não tem thread para receber as mensagens.
    - Omissão-envio: Um buffer de envio estoura e a mensagem não é enviada. O servidor tem que estar preparado porque um cliente pode solicitar
    - Um loop infinito onde cada iteração cria um novo processo causando que o processo pare em algum momento.

MODELOS DE FALHAS

- Falha por temporização
  - Uma resposta do servidor está fora de um intervalo de tempo específico
  - Um site de e-commerce site pode definir que uma resposta ao usuário não deve ser dada em mais de 5 segundos
  - Em uma aplicação de vídeo por demanda, um cliente tem que receber os frames em uma determinada frequência
  - Difíceis de gerenciar

MODELOS DE FALHAS

- Falhas arbitrárias (bizantinas):
  - Servidor está realizando respostas incorretas, mas que não podem ser detectadas como incorretas
  - Um servidor faltoso pode estar trabalhando maliciosamente com outros servidores para produzir respostas erradas

MASCARAMENTO DE FALHAS

- Para o sistema ser tolerante a falhas, as ocorrências das falhas devem ser ocultas de outros processos e usuários
  - A técnica fundamental para mascarar falhas é usar redundância:
    - Redundância de informação – bits extras podem ser adicionados para recuperação de bits deteriorados. Código de Hamming

MASCARAMENTO DE FALHAS

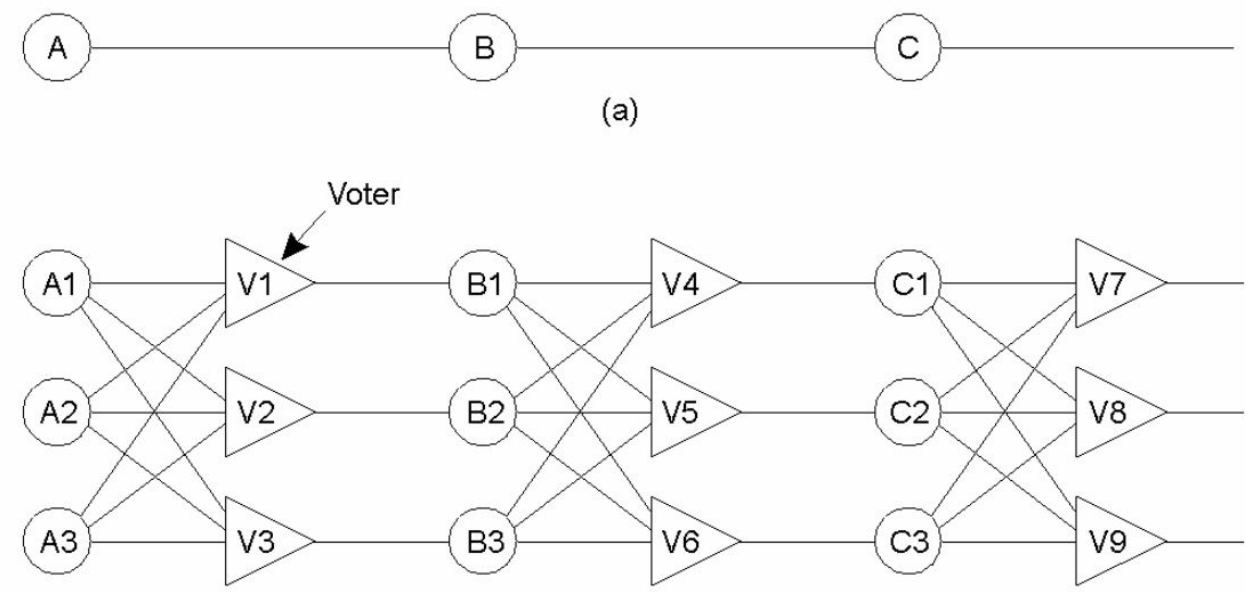
- REDUNDÂNCIA DE TEMPO
  - Uma ação é realizada e se necessário é realizada novamente
  - Exemplo: Se uma transação aborta, ela pode ser refeita sem prejuízo.
  - Especialmente útil quando as falhas são transientes ou intermitentes

MASCARAMENTO DE FALHAS

- REDUNDÂNCIA FÍSICA
  - Processos ou equipamentos extras são adicionados para possibilitar que o sistema possa como um todo tolerar a perda ou mau funcionamento de alguns componentes
  - Redundância física pode ser feita em hardware ou em software
  - Exemplos in hardware:
    - Aeronave: 747 tem 4 motores mas voa com 3.
    - Aeronave espacial: Tem 5 computadores
    - Circuitos eletrônicos

MASCARAMENTO DE FALHAS

- REDUNDÂNCIA FÍSICA



MASCARAMENTO DE FALHAS

- REDUNDÂNCIA FÍSICA
  - No circuito eletrônico anterior, cada dispositivo é replicado três vezes.
  - Após cada estágio de dispositivo existe um votante triplicado
  - Cada votante é um circuito que possui 3 entradas e uma saída
  - Se duas ou três entradas são as mesmas, a saída é igual a esse valor de entrada
  - Se todas as entradas são diferentes, a saída é indefinida
  - Este tipo de projeto é conhecido como TMR (Triple Modular Redundancy)
    - TMR pode ser aplicado a qualquer unidade de hardware
    - TMR pode mascarar completamente a falha de uma unidade de hardware
    - Não há necessidade de se executarem ações específicas para detecção de erro, recuperação, etc
    - Particularmente adequado para falhas transientes

MASCARAMENTO DE FALHAS

- REDUNDÂNCIA FÍSICA
  - Este esquema não consegue tratar a falha de duas unidades
  - Caso uma unidade falhe, é essencial que as outras duas continuem a trabalhar corretamente
  - O esquema TMR depende criticamente do elemento votante, que é tipicamente um circuito simples que é fácil de ser muito confiável
  - A falha de um único votante não é tolerada

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- Resiliência de Processos
  - Replicação de processos em grupos
  - Grupos Simples ou Hierárquicos
- Comunicação Confiável Cliente-Servidor
  - Falhas de Comunicação
  - Canal de Comunicação pode exibir falhas por queda, por omissão, arbitrárias
- Comunicação Confiável de Grupo
  - Implementar entrega confiável de mensagens a todos processos
- Comprometimento Distribuído
  - Envolve a realização de uma operação por cada membro de um grupo de processos ou por absolutamente nenhum

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - GRUPOS
  - A abordagem fundamental para tolerar um processo faltoso é organizar vários processos idênticos em um grupo
    - Quando uma mensagem é enviada a um grupo, todos membros do grupo a recebem
    - Se um processo falhar, espera-se que algum outro se encarregue da mensagem em seu lugar
    - Grupos podem ser dinâmicos
    - A finalidade de introduzir grupos é permitir que processos tratem conjuntos de processos como uma única abstração, como um único processo

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - GRUPOS
  - Processos podem ser organizados em grupos simples ou grupos hierárquicos

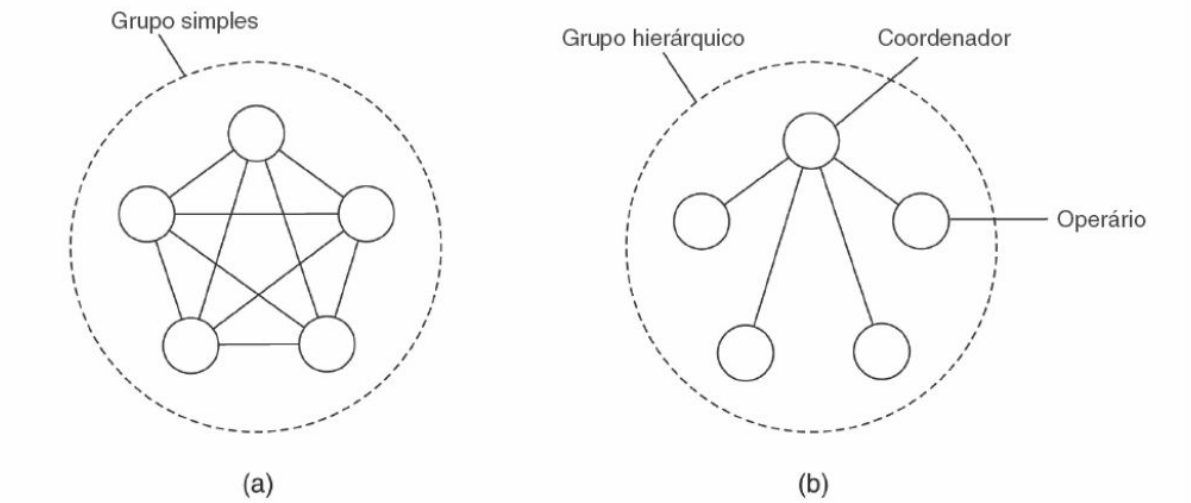


Figura 8.2 (a) Comunicação em um grupo simples. (b) Comunicação em um grupo hierárquico simples.

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - GRUPOS
  - Grupos simples
    - Todos processos são iguais dentro de um grupo
    - Decisões são tomadas coletivamente
  - Vantagem
    - Não tem ponto de falha único → Mesmo que um processo caia, o grupo continua a oferecer o serviço
    - Desvantagem Tomada de decisão pode ser complicada, com necessidade de uma votação → retardo

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - GRUPOS
  - Grupos hierárquicos
    - Existe um processo coordenador e demais são denominados operários
    - Sempre que uma requisição e gerada, é enviada ao coordenador → O coordenador decide qual e o operário mais adequado para executá-la
    - Vantagem
      - Decisões são centralizadas
    - Desvantagem
      - Caso o coordenador falhe, o serviço falhará

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - GRUPOS
  - Com a existência da comunicação em grupos, se torna necessário mecanismos para:
    - Criar e eliminar grupos
    - Permitir a entrada e saída de processos em um grupo
  - Existem duas abordagens possíveis para este gerenciamento:
    - Servidor de grupo
    - Gerenciamento distribuído

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - GRUPOS
  - Servidor de Grupo
    - Recebe todas as requisições e mantém banco de dados completo sobre todos grupos e seus membros
    - Método direto, eficiente e razoavelmente fácil de implementar
    - Desvantagem
      - Por ser uma abordagem centralizada → Um único ponto de falha
      - Se o servidor de grupo cair, o gerenciamento deixa de existir

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - GRUPOS
  - Gerenciamento Distribuído
    - Se existe multicast confiável, um processo pode enviar uma mensagem a todos os membros do grupo anunciando que deseja se juntar ao grupo
    - Para sair de um grupo, o processo deveria mandar uma mensagem de adeus a todos
    - Dificuldades:
      - Difícil de detectar quedas
      - Entrar/Sair de um grupo devem ser síncronos com as mensagens enviadas/recebidas

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - MASCARAMENTO E REPLICAÇÃO
  - Para construir sistemas tolerantes a falhas, podemos replicar processos e organizá-los em um grupo para substituir um único processo (vulnerável) por um grupo (tolerante a falha).
  - A replicação pode ser abordada de duas maneiras:
    - Protocolos baseados em primários: Um grupo de processos é organizado de modo hierárquico no qual um servidor primário coordena todas operações de escrita. Os servidores de backup executam um algoritmo de eleição caso o primário caia.
    - Protocolos de escrita replicada: organiza um conjunto de processos idênticos em um grupo simples.

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO - MASCARAMENTO E REPLICAÇÃO
  - Quanto de replicação é necessária ao se criar um grupo de processos tolerante a falhas?
    - Quando um grupo de processos deseja mascarar k falhas simultâneas, é classificado como um grupo k-tolerante a falha 1.
      - Se k processos falharem silenciosamente, sem propagar informações erradas, basta ter k+1 processos
      - Se os processos exibirem falhas e continuarem a enviar respostas erradas às requisições, é preciso um mínimo de 2k+1 processadores para conseguir k-tolerância, porque teremos k enviando respostas erradas mas k+1 enviando respostas certas e pode-se acreditar na maioria

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

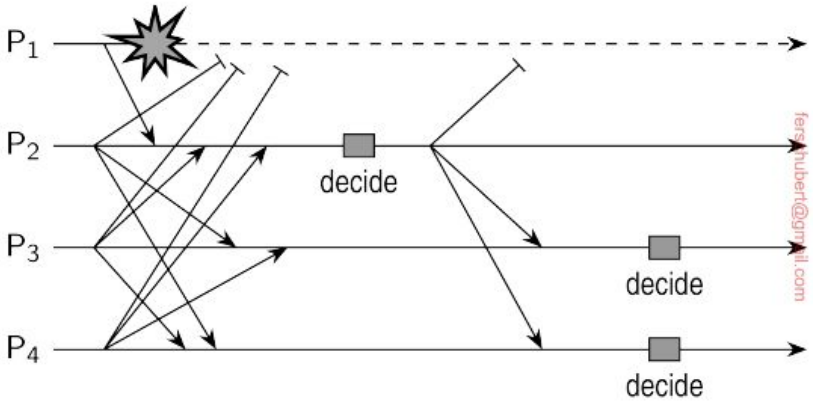
- RESILIÊNCIA DE PROCESSO – CONSENSO
  - Organizar processos replicados em um grupo ajuda a aumentar a tolerância a falha
  - Em muitos casos, um grupo de processos deve chegar a algum tipo de acordo: eleger um coordenador, decidir a validação de uma transação, repartir tarefas entre operários
  - Objetivo é que todos os processos que não apresentam falhas cheguem a um consenso sobre alguma questão, dentro de um número finito de etapas
  - O problema é complicado pelo fato de que premissas diferentes sobre o sistema requerem soluções diferentes

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO – CONSENSO
  - Existem várias situações:
    - Sistemas síncronos X assíncronos
    - Atraso de comunicação limitado ou não
    - Entrega de mensagens ordenada ou não
    - Transmissão de mensagens unicast ou multicast

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO – CONSENSO POR INUNDAÇÃO
  - suposições: *n* processos em um grafo não direcionado completamente conectado, processos sabem o tamanho do grupo, síncrono, ocorrem falhas do tipo queda com detecção perfeita da falha,não há perda de mensagens, o conjunto de valores possíveis *{V}* é composto de todos os valores propostos, cada processo tem exatamente um valor proposto, qualquer função de decisão determinística pode ser aplicada.



ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO – CONSENSO POR INUNDAÇÃO
  - Processos executam em rounds
  - Cada processo mantém um conjunto de propostas, e esse conjunto é aumentado quando o processo passa de um round para o próximo
  - Em cada round, todos os processos disseminam seu conjunto para todos os outros através do uso de broadcast Um processo decide por um valor específico do seu conjunto quando ele sabe que obteve todas as propostas que serão vistas por qualquer um dos processos corretos, ou ele detectou que não houve falhas em dois rounds sucessivos Um processo então decide enviar sua decisão para todos no próximo round; todos os processos corretos que ainda não decidiram até este momento, irão decidir quando receberem a mensagem de decisão do processo

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO – CONSENSO POR INUNDAÇÃO
  - O acordo não é estritamente violado: mas processos corretos devem decidir um valor que seja consistente com valores decididos pelos processos que podem ter decidido antes de falhar
  - Suponha um processo que recebe as mensagens de todos e decide o valor mas cai imediatamente antes de conseguir enviar o valor para todos
  - Os processos que restaram se movem para um próximo round detectando a falha, e depois para o próximo round, onde não ocorrem mais falhas e aí eles podem decidir por um valor diferente
  - O problema pode ser minimizado empregando um mecanismo confiável de broadcast

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO – DETECÇÃO DE FALHAS
  - Para mascarar falhas é preciso primeiramente detectá-las
  - Detecção de falhas pode ser resumida da seguinte forma:
    - Dado um grupo de processos, membros não faltosos devem ser capazes de decidir quem ainda é um membro e quem não é
    - Ou seja, detectar quando um membro falhou
  - Existem dois mecanismos para detecção:
    - Processos enviam ativamente mensagens uns aos outros perguntando: Você está vivo?
    - Ou esperam passivamente pela entrada de mensagens de processos diferentes

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- RESILIÊNCIA DE PROCESSO – DETECÇÃO DE FALHAS
  - Em essência, falhas são detectadas através de mecanismos de temporizadores
  - Definir temporizadores de maneira eficiente é muito difícil e depende da aplicação
  - É difícil distinguir falhas dos processos das falhas da rede
  - Possível solução
    - Considerar possíveis notificações de falhas entre os membros do sistema Algoritmos de Gossiping, anunciam regularmente que estão vivos
    - Ou esperam passivamente pela entrada de mensagens de processos diferentes

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR
  - Ao se tratar falhas, também devem ser consideradas as falhas de comunicação
  - Canais de comunicação podem exibir falhas por queda, por omissão, de temporização e arbitrárias
  - Na construção de canais de comunicação o objetivo principal é mascarar falhas por queda e omissão
  - Falhas arbitrárias podem ocorrer sob a forma de mensagens duplicadas

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO
  - Em sistemas distribuídos, a comunicação confiável ponto a ponto é estabelecida pelo uso de um protocolo de transporte confiável como TCP
  - TCP mascara falhas por omissão, que ocorrem sobre a forma de mensagens perdidas, usando reconhecimento e retransmissão
  - TCP não mascara falhas por queda conexão interrompida abruptamente
    - O único modo de mascarar tais falhas é permitindo que o SD reestabeleça a conexão através do reenvio de um pedido de conexão, que implica que o outro lado se recuperou

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Chamadas de procedimento remoto (RPC) têm como objetivo ocultar comunicação
  - Uma RPC faz com que chamadas de procedimentos remotos pareçam exatamente como locais
  - Se não existem erros, o RPC desempenha bem seu papel
  - O problema é quando surgem erros:
    - Neste caso, as diferenças entre chamadas locais e remotas não são fáceis de se mascarar

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Cinco classes diferentes de falhas podem ocorrer em um sistema com RPC:
    - 1. O cliente não consegue localizar o servidor
    - 2. A mensagem de requisição do cliente para o servidor se perde
    - 3. O servidor cai após receber uma requisição
    - 4. A mensagem de resposta do servidor para o cliente se perde
    - 5. O cliente cai após enviar uma requisição
  - Cada uma dessas falhas apresenta problemas diferentes e requer soluções diferentes

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Cliente não consegue localizar servidor:
    - O cliente pode não localizar um servidor adequado
      - Por exemplo, na atualização de apêndices, o apêndice de cliente não é atualizado
    - O erro pode ativar uma exceção para permitir seu tratamento e emitir uma resposta programador programa a exceção
    - Nem todas linguagens suportam exceções
    - A transparência é destruída pois o programador tem que tratar a falha de chamar a um procedimento remoto

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Mensagem de requisição do cliente para o servidor se perde:
    - É o erro mais fácil de se tratar: basta fazer um temporizador ao enviar a requisição
    - Se o temporizador expirar antes do recebimento da resposta de reconhecimento, a mensagem é reenviada
    - Se a mensagem foi realmente perdida, o servidor não conseguirá perceber a diferença entre a retransmissão e a mensagem original tudo funcionará bem
    - Se a requisição não foi perdida permitir que o servidor detecte que está recebendo uma retransmissão

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Quedas do servidor

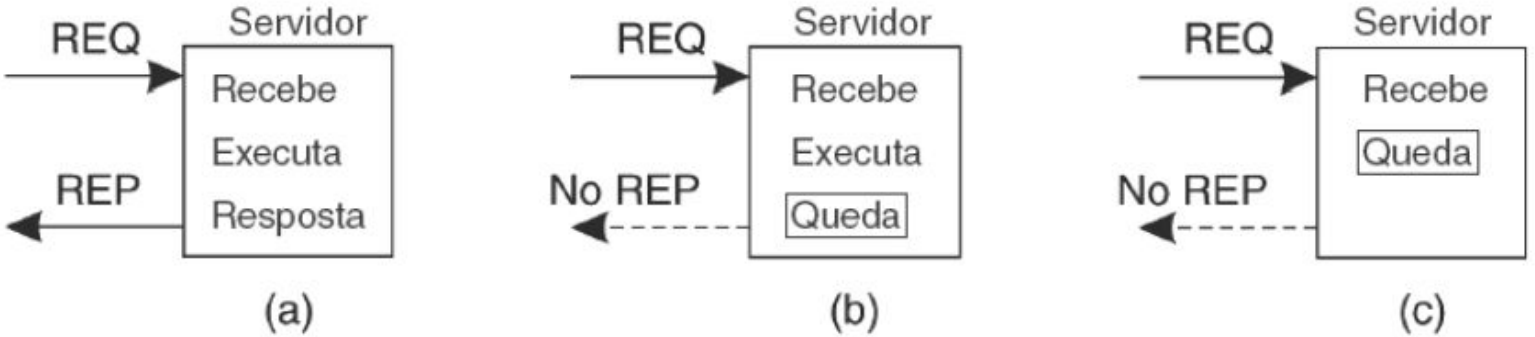


Figura 8.6 Servidor em comunicação cliente–servidor. (a) Caso normal. (b) Queda após a execução. (c) Queda antes da execução.

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - O temporizador do cliente não possui discernimento entre os 2 casos de falha
    - Em (b) o sistema deve informar a falha ao cliente (exceção)
    - Em (c) o cliente pode apenas retransmitir a requisição

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Quedas do servidor
    - Há três linhas de pensamento sobre o que fazer
      - 1. Semântica ao menos uma vez
        - Garante que a RPC seja executada ao menos uma vez, mas possivelmente mais de uma
      - 2. Semântica no máximo uma vez
        - Garante que a RPC seja executada no máximo uma vez, mas possivelmente nenhuma
      - 3. Não garantir nada
    - Infelizmente não é possível fazer uma semântica exatamente uma vez

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Situações possíveis na ocorrência de uma queda do servidor
  - Exemplo: O cliente tenta imprimir um documento
    - Cliente envia requisição, recebe uma mensagem de reconhecimento do servidor
    - Servidor pode enviar uma mensagem de conclusão ao cliente
      - Antes de acionar a impressora ou depois do texto ter sido impresso
    - Considere que 3 eventos podem ocorrer no servidor
      - M → enviar mensagem de conclusão
      - P → imprimir o texto
      - C → cair

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Com duas estratégias para o servidor e quatro para o cliente, existem oito possíveis combinações a considerar
    - Nenhuma satisfatória!

Cliente	Servidor					
	Estratégia M→P			Estratégia P→M		
Estratégia de reemissão	*MPC*	*MC(P)*	*C(MP)*	*PMC*	*PC(M)*	*C(PM)*
Sempre	DUP	OK	OK	DUP	DUP	OK
Nunca	OK	ZERO	ZERO	OK	OK	ZERO
Somente quando recebe ACK	DUP	OK	ZERO	DUP	OK	ZERO
Somente quando não recebe ACK	OK	ZERO	OK	OK	DUP	OK

OK = Texto é impresso uma vez
DUP = Texto é impresso duas vezes
ZERO = Texto não é impresso

Figura 8.7 Diferentes combinações de estratégias de cliente e servidor na presença de quedas de servidor.

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Mensagens de resposta perdidas:
    - A mensagem é entregue, mas a resposta não
    - É feito como nas mensagens de requisição perdidas: é feito um temporizador ajustado pelo SO do cliente
    - Se o temporizador expirar antes do recebimento da resposta de reconhecimento, a mensagem é reenviada
    - O que torna mais difícil é que o cliente não sabe com certeza o motivo da ausência de resposta: a requisição ou a resposta se perderam, ou é lentidão do servidor?

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Mensagens de resposta perdidas:
    - Pedidos idempotentes podem ser executados mais de uma vez sem problemas, mas nem todos o são
    - Solução:
      - Cliente gera um número de sequência a cada requisição
      - Se o servidor monitorar o número de sequência mais recentemente recebido de cada cliente, poderá distinguir entre uma requisição original e uma retransmissão
      - O servidor poderá se recusar a executar qualquer requisição uma segunda vez

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Quedas de cliente:
    - O que acontece se um cliente enviar uma requisição de serviço a um servidor e cair antes de o servidor responder?
    - Neste caso, existe uma computação ativa e nenhum pai está esperando o resultado → processo órfão
    - Processos órfãos podem interferir com a operação normal do sistema
      - Desperdício de CPU, travar arquivos e amarrar recursos valiosos
      - Além disso, se o RPC cliente for reiniciado e executar novamente a RPC, mas a resposta voltar logo em seguida, pode haver confusão

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL CLIENTE-SERVIDOR - COMUNICAÇÃO PONTO A PONTO - RPC
  - Quedas de cliente: O que fazer com órfãos?
    - 1. Extermínio de órfão
      - Manter registro da RPC no disco → Após uma reinicialização o registro é verificado → Se existe órfão, estes são eliminados
        - Vários acessos a disco deterioram o desempenho
        - Órfãos podem gerar novas RPCs, gerando netos órfãos, etc.
    - 2. Reencarnação
      - O tempo é dividido em épocas sequenciais numeradas.
      - Ao ser reiniciado, o cliente envia broadcast declarando início de nova época. TODAS as computações remotas em nome dele são removidas

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL DE GRUPO
  - Tem como objetivo garantir entrega a todos os membros em um grupo de processos
    - Multicast através de unicasts confiáveis requer que cada processo estabeleça conexão ponto-a-ponto com os que se quer trabalhar
    - Desperdício de largura de banda
    - Entretanto se o número de processos for pequeno esta é uma solução simples

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL DE GRUPO
  - Para implementar o multicast confiável, deve-se definir:
    - Se um processo que acabou de entrar em um grupo deve receber uma mensagem enviada antes de sua entrada?
    - Se um processo que acabou de sair de um grupo deve receber uma mensagem enviada antes de sua saída?
    - Distinguir entre comunicação confiável que:
      - Considera presença de processos faltosos
      - Considera que processos funcionem corretamente

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL DE GRUPO
  - Para implementar o multicast confiável, deve-se definir:
    - Se um processo que acabou de entrar em um grupo deve receber uma mensagem enviada antes de sua entrada?
    - Se um processo que acabou de sair de um grupo deve receber uma mensagem enviada antes de sua saída?
    - Distinguir entre comunicação confiável que:
      - Considera presença de processos faltosos
      - Considera que processos funcionem corretamente

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

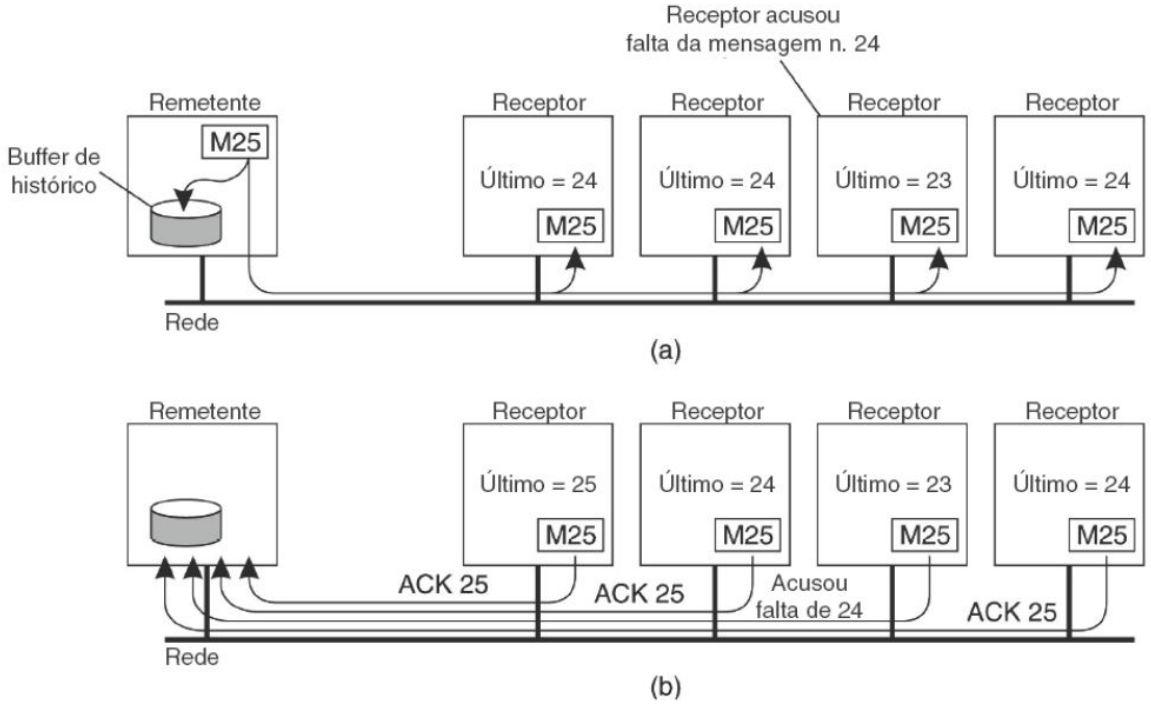


Figura 8.8 Uma solução simples para multicast confiável quando todos os receptores são conhecidos; a premissa é que nenhum falhe. (a) Transmissão de mensagem. (b) Realimentação de relatório.

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMUNICAÇÃO CONFIÁVEL DE GRUPO
  - Um problema apresentado é a implosão de retorno, que pode lotar o remetente com mensagens de confirmação de recebimento
  - Solução: Não confirmar recebimento, mas apenas devolver respostas quando detectar faltas
  - Neste caso, o remetente será forçado a manter uma mensagem em seu buffer de histórico para sempre!
  - Mensagens podem ser apagadas para evitar que o buffer transborde mas pode resultar que uma retransmissão não ser honrada

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- MULTICAST ATÔMICO
  - Para multicast confiável na presença de falhas de processos, deseja-se:
    - Que a mensagem seja entregue a todos os processos ou nenhum deles
    - Mensagens são entregues na mesma ordem em todos os processos
  - Este problema é conhecido como Multicast Atômico
    - Útil para garantir consistência em réplicas de bancos de dados

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- MULTICAST ATÔMICO
  - Exemplo: Banco de Dados replicado
    - Banco de dados é construído como um grupo de processos, um processo para cada réplica
    - Operações de atualização são enviadas em multicast a todas as réplicas
    - Suponha que durante a execução de uma das atualizações de uma sequência, uma réplica caia
    - Quando a réplica se recupera é essencial que seja atualizada em relação as outras réplicas

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- MULTICAST ATÔMICO
  - Exemplo: Banco de Dados replicado
    - A operação de atualização que foi enviada a todas as réplicas um pouco antes de uma delas cair ou é executada em todas as réplicas não faltosas ou em nenhuma
    - A atualização é realizada se as réplicas restantes concordarem que a réplica que caiu não pertence mais ao grupo
    - Após a recuperação, a réplica é validada como sendo do grupo e recebe as atualizações
    - O registro de uma réplica no grupo só é aceito se a réplica se atualizar com o grupo

ESTRATÉGIAS E ALGORITMOS DE TOLERÂNCIA A FALHAS

- COMPROMETIMENTO DISTRIBUÍDO
  - Protocolo de comprometimento de 1 fase
    - Se um dos participantes não puder executar a operação, não tem como avisar ao coordenador
  - Protocolo de comprometimento 2 fases
    - Este esquema é o mais comum, solucionando o problema anterior
    - Não manipula com total eficiência a falha do coordenador
  - Protocolo de comprometimento 3 fases
    - Um esquema mais complicado que surgiu para lidar de maneira mais eficiente com as falhas do coordenador

RECUPERAÇÃO DE FALHAS

- Ocorrida uma falha, é necessário não apenas identificá-la, mas recuperar-se da mesma e voltar para um estado correto
- Essencialmente existem 2 maneiras de se recuperar:
  - Recuperação retroativa (Backward Recovery)
    - Volta para um estado anterior à falha
  - Recuperação para frente (Forward Recovery)
    - Tenta levar o sistema para um novo estado correto para que possa continuar a executar

RECUPERAÇÃO DE FALHAS

- Recuperação retroativa (Backward Recovery)
  - Retorna o sistema a algum estado que antes estava correto, continuando a execução após a recuperação.
  - É necessário gravar o estado do sistema de tempos em tempos
    - Toda vez que o estado do sistema é registrado, diz-se que foi feito um ponto de verificação checkpointing
  - Sempre que ocorre uma falha o estado do sistema é restaurado para o momento do último ponto de verificação
  - Checkpoints podem ser mecanismos caros
  - Em alguns casos não é possível retroagir a um estado sem erros

RECUPERAÇÃO DE FALHAS

- Recuperação para frente (Forward Recovery)
  - Quando o sistema falha ao invés de retroagir para um estado anterior, é feita uma tentativa para levar o sistema para um novo estado correto
  - É preciso saber de antemão quais erros podem ocorrer
  - Tendo conhecimento de todos os erros e como levar o sistema para um estado correto, é possível recuperar totalmente o sistema

RECUPERAÇÃO DE FALHAS

- Muitos sistemas combinam pontos de verificação (checkpointing) com registro de mensagens
  - No caso do uso isolado de pontos de verificação, os processos são restaurados para o ponto antes da falha e o comportamento pode ser diferente após a recuperação
    - Mensagens podem ser entregues em ordenação diferente
  - No caso do registro de mensagens, o comportamento é reproduzido do mesmo modo entre o ponto de recuperação e o ponto em que ocorreu a falha

RECUPERAÇÃO DE FALHAS

- Armazenamento Estável
  - Para recuperar um estado anterior, é necessário que informações necessárias para permitir a recuperação estejam seguramente armazenadas
  - Segurança neste contexto significa que informações de recuperação sobrevivam a quedas de processos e falhas de sites, e possivelmente também a falhas de mídia de armazenamento

RECUPERAÇÃO DE FALHAS

- Armazenamento Estável
  - Para recuperar um estado anterior, é necessário que informações necessárias para permitir a recuperação estejam seguramente armazenadas
  - Segurança neste contexto significa que informações de recuperação sobrevivam a quedas de processos e falhas de sites, e possivelmente também a falhas de mídia de armazenamento

RESUMO

- Tolerância a falha é uma questão importante no projeto de sistemas distribuídos
- Característica pela qual um sistema pode mascarar a ocorrência e a recuperação de falhas
- Vários tipos de falha podem ocorrer em um SD:
  - Falha por queda, por omissão, temporização, falha de resposta, arbitrárias
  - Redundância é a técnica fundamentalmente necessária para conseguir tolerância a falhas

RESUMO

- Estratégias para Tolerância a Falhas
  - Replicação de processos em grupos
    - Grupos podem ser simples (todos processos tomam decisões da mesma maneira) ou hierárquicos (coordenador gerencia)
    - Grupos de processos muitas vezes precisam chegar a um acordo Muitas vezes conseguir um acordo é difícil
  - Comunicação Confiável Cliente-Servidor
  - Comunicação Confiável de Grupo
  - Comprometimento Distribuído
    - Envolve a realização de uma operação por cada membro de um grupo de processos ou por absolutamente nenhum

RESUMO

- Recuperação em sistemas tolerantes a falhas é alcançada por pontos de verificação periódicos do estado do sistema
- A verificação é completamente distribuída, sendo esta uma operação cara, principalmente no caso de pontos de verificação independentes
- Para melhorar o desempenho, muitos sistemas distribuídos combinam pontos de verificação com registro de mensagens
- Registrando a comunicação entre processos, torna-se possível reproduzir a execução do sistema após a ocorrência de uma queda