

Sistemas Distribuídos

Comunicação

Sincronização

Prof. Marcelo de Paiva Guimarães

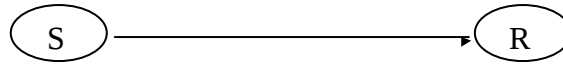
1

Comunicação de Grupo

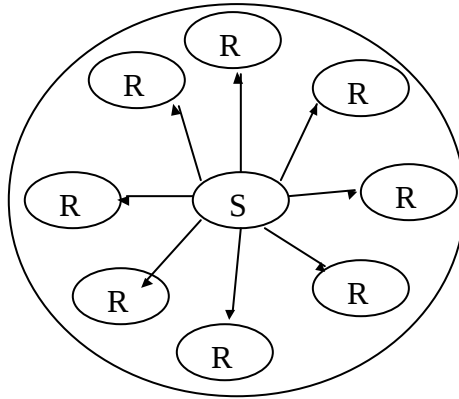
- Grupo
 - É uma coleção de processos que agem juntos em um sistema, de tal forma que quando uma mensagem é enviada para o grupo, todos os membros do grupo recebem

2

Comunicação de Grupo



Comunicação ponto-a-ponto



Comunicação um-para-vários

3

Comunicação de Grupo

- Grupos são dinâmicos
 - Novos grupos podem ser criados e grupos existentes podem ser eliminados
 - Um processo pode entrar para um grupo ou deixá-lo
 - Um processo pode ser membro de diversos grupos simultaneamente
 - São necessários mecanismos para gerenciar o conceito de grupo
- O propósito do conceito de grupos
 - É permitir que processos tratem conjuntos de processos como uma simples abstração
- Um processo pode enviar uma mensagem para um grupo sem ter de saber
 - Quantos são estes servidores
 - Ou onde eles estão localizado
 - Sendo que estas informações podem mudar de uma chamada para outra

4

Comunicação de Grupo

- **Multicast**
 - Endereço especial que múltiplas máquinas podem receber
 - Implementação direta
 - Basta atribuir a cada grupo um endereço multicasting diferente
- **Broadcasting**
 - Pacotes contendo certos endereços são enviados para todas as máquinas
 - Menos eficiente que multicasting
 - Todas as máquinas recebem as mensagens enviadas por broadcasting
 - O software precisa verificar se o pacote é dele
 - Necessita somente um pacote para atingir todos os membros do grupo
- **Unicasting**
 - Transmissão separadas de pacote para cada membro do grupo
 - N membros, n pacotes necessários

5

Comunicação de Grupo

- **Grupos Fechados versus Abertos**
 - As vezes é devido a detalhes de implementação
 - Grupos fechados
 - Somente os membros do grupo podem enviar mensagens para o grupo
 - Grupos abertos
 - Qualquer processo do sistema pode enviar mensagens para qualquer grupo
- **Grupos igualitários e Grupos Hierárquicos**
 - Igualitários
 - Todos os processos são iguais
 - Decisões é mais complicada (depende de todos)
 - Hierárquicos
 - Existe um processo coordenador
 - Possui um ponto único de falha

6

Comunicação de Grupo

- Controle dos Membros de um grupo
 - Deve existir um método para:
 - criar e deletar grupos
 - Incluir e excluir processos
 - Servidor de Grupo
 - Mantém todas as informações sobre os grupos
 - Ponto de falha
- Endereçamento de Grupo
 - Cada grupo deve ter um endereço único

7

Comunicação de Grupo

- Propriedades
 - Atomicidade
 - Ordenação das mensagens

8

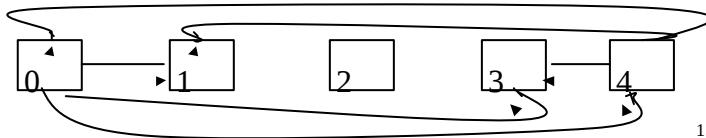
Comunicação de Grupo

- Atomicidade
 - Quando uma mensagem é enviada para um grupo
 - Ela deve chegar corretamente para todos os membros do grupo ou não chegar para nenhum membro
 - Uma maneira de ter certeza que todos os destinatários receberam a mensagem
 - É implementar o envio do ACK para cada mensagem recebida

9

Ordenando mensagens

- Sistema centralizado
 - É possível determinar a ordem na qual dois eventos ocorreram,
 - Existe memória e clock comuns
- É importante o sistema ser capaz de determinar a ordem
 - A melhor garantia
 - Fazer com que as msg sejam expedidas na mesma ordem em que foram enviadas
 - Um recurso só pode ser utilizado depois de concedido



10

Sincronização em SD

- A comunicação é muito importante em SD e está ligada com
 - Como os processos se cooperam
 - Como os processos se sincronizam
- Os SD utilizam
 - Algoritmos distribuídos que:
 - Espalham as informações entre diversas máquinas
 - Os processos toma decisões baseadas nas informações locais
 - Deve ser evitado a existência de um único ponto de falha
 - Não existe nenhuma fonte de clock comum, ou qualquer fonte precisa de tempo globam

11

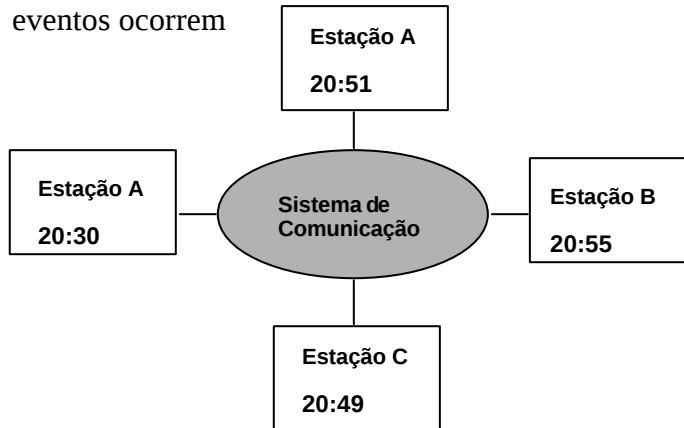
Sincronização em SD

- Make local
 - Verifica os instantes de tempo nos quais os arquivos fontes e seus correspondentes arquivos-objeto foram modificados pela última vez
 - Se o arquivo-fonte foi alterado então deve ser recompilado
- Make distribuído
 - Output.o tem o valor associado 2144
 - Output.c (outra máquina) foi modificado e tem o valor associado 2143
 - Neste caso o make não irá compilar o arquivo modificado

12

Sincronização em SD

- Clock Lógico
 - O que realmente importa são os tempos relativos e não absolutos
 - Todos devem estar certos em relação a ordem em que os eventos ocorrem



13

Ordenação de Eventos em Sistemas Distribuídos (Lamport)

- Um sistema é visto como uma coleção de processos, cada um deles uma sequência de eventos
- A relação "acontece antes" ($a \rightarrow b$)
 - Sejam a e b eventos em um mesmo processo; se a ocorre antes de b , então $a \rightarrow b$ é verdadeira
 - Se a é o evento "envio de msg. para um processo" e b o evento "recepção de msg. por outro processo", então a relação $a \rightarrow b$ é tb. Verdadeira. Uma mensagem não pode ser recebida antes de ser enviada, ou no mesmo tempo em que foi enviada

14

Ordenação de Eventos em Sistemas Distribuídos (Lamport)

- "acontece antes" é uma relação transitiva
 - Se $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$
- Se dois eventos x e y acontecem em diferentes processos que não trocam msgs
 - Então $x \rightarrow y$ e $y \rightarrow x$ não são verdadeiros
 - Estes eventos são ditos concorrentes
 - Significa que nada pode ser dito sobre quando eles acontecem
- Deve ter um modo de medir o tempo
 - De tal forma que para todo evento a ,
 - Nós podemos assumir que ele aconteceu em um tempo $C(a)$ no qual todo processo concorda
 - Se $a \rightarrow b$, então $C(a) < C(b)$

15

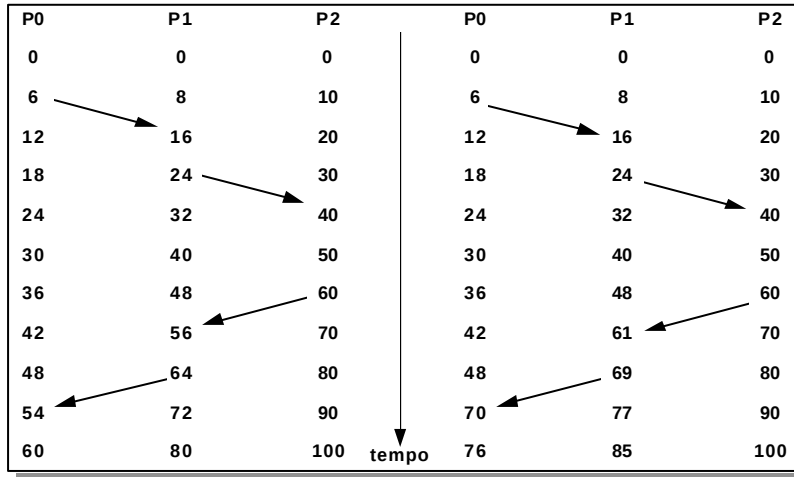
Algoritmo de Lamport para corrigir os clocks

- 3 processos
 - Em máquinas diferentes
 - Com oscilações de frequência diferentes
- Primeiro intervalo
 - 6 interrupções p_0
 - 8 interrupções p_1
 - 10 interrupções p_2
- No tempo 6 de p_0
 - Envia uma mensagem para p_1
 - O clock do p_1 estará marcando 16
- A mensagem C
 - Deixa p_2 com 60 e chega em p_1 com 56
- Solução
 - A mensagem C deve chegar ao seu destino no tempo 61 ou mais tarde
 - Cada msg carrega consigo o instante de transmissão de acordo com o clock do transmissor
 - Quando chega
 - O receptor adianta o seu clock conforme o tempo indicado na mensagem

16

Algoritmo de Lamport para corrigir os clocks

- Para fazer as correções necessárias o tempo de relógio C precisa ser incrementado (nunca decrementado)



Sincronização Externa

- Em alguns sistemas (tempo real por exemplo) um clock real é importante
 - É necessário clocks físicos externos

Algoritmos para Sincronização de clocks

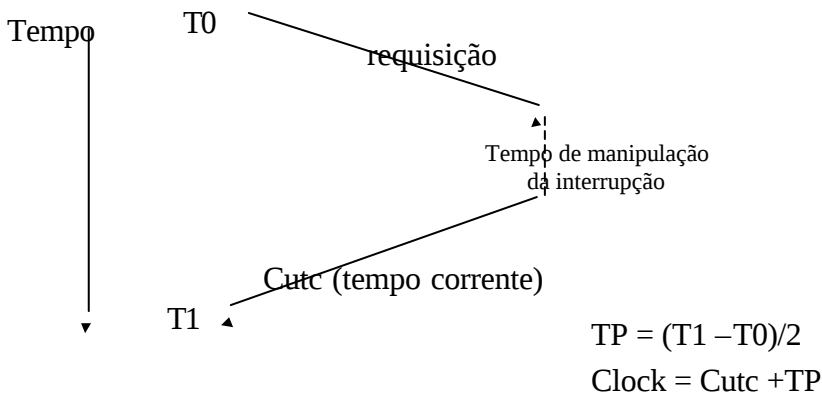
- Algoritmo Cristian

- Periodicamente, cada máquina envia uma mensagem para o Servidor de Tempo perguntado pelo tempo corrente (atual)
- Esta máquina responde o mais rápido possível com uma mensagem contendo o tempo corrente (Cutc)
- Problemas
 - O tempo nunca pode andar para trás
 - Pode-se reduzir o fator de incremento do clock até acertar
 - Ocorre um tempo não-nulo para resposta por parte do servidor

19

Algoritmo Cristian

Adquirindo o tempo corrente do Servidor de Tempo



T0 e T1 são medidos com o mesmo clock

20

Algoritmo de Berkeley

- Servidor de Tempo
 - É ativo, e requer, periodicamente de cada máquina, o tempo do seu relógio
 - O Servidor de Tempo calcula a média dos tempos e diz para cada máquina como ajustar seu relógio

21

Algoritmos Baseados em Média

- Trabalha dividindo o tempo em intervalos fixos de resincronização
- No início de cada intervalo
 - Cada uma das máquinas envia em broadcast uma mensagem contendo seu tempo corrente
 - Após determinada máquina ter enviado sua mensagem
 - Ela deve inicializar um temporizador local para coletar todas as outras mensagens em broadcast que chegarem durante um intervalo de tempo S
 - Quando todas as mensagens forem recebidas, ela deve rodar um algoritmo para calcular novo tempo corrente
 - Baseado nos tempos informados pelas demais máquinas
 - Tal algoritmo pode simplesmente calcular a média dos valores
 - Uma pequena variação descarta os m valores mais altos e m mais baixos (protege-se contra até m falhas no envio dos clocks)

22

Exclusão mútua

- A maneira mais direta de conseguir exclusão mútua em um sistema distribuído é imitar o que é feito no sistema centralizado
- Um processo é eleito como Coordenar
 - Quando um processo quer entrar na região crítica, ele envia uma msg requisitando ao Coordenador
- Se nenhum outro processo está na região crítica, o Coordenador envia uma resposta dando permissão
- Quando a msg resposta chega, o processo entra na região crítica

23

Exclusão mútua

- Supondo que outro processo solicite permissão para entrar na região crítica, o Coordenador sabendo que outro processo está na região, não envia resposta bloqueando este processo até que possa entrar na região
- Quando o processo deixa a região, ele envia para o Coordenador uma msg liberando a região
- O algoritmo centralizado tem o problema de uma falha no Coordenador inviabilizar o mecanismo
- Em um sistema grande o Coordenador pode se tornar o gargalo

24

Exclusão mútua

- Algoritmo Distribuído
 - Quando um processo quer entrar na região crítica, ele constrói uma msg contendo o nome da região, o seu número (processo) e o tempo corrente
 - Ele envia a msg para todos os outros processos

25

Exclusão mútua

- Algoritmo Distribuído
 - Quando um processo recebe uma msg de requisição de outro processo
 - Se o receptor não está na região crítica e não quer entrar, ele envia de volta uma msg OK
 - Se o receptor já está na região crítica, ele não responde e coloca a requisição na fila
 - Se o receptor quer entrar na região crítica mas ainda não fez, de compara o tempo da msg que chegou com o tempo da msg que ele enviou para os outros processos
 - O menor tempo vence
 - Se a msg que chegou é menor ele envia de volta uma msg OK.
 - Se a sua msg tem o menor tempo ele coloca na fila a requisição que chegou e não envia nada
 - Após pedir permissão, um processo espera até que todos tenham dado a sua permissão.
 - Quando todas as permissões chegam o processo pode entrar na região crítica
 - Quando ele sai da região crítica, ele envia uma msg OK para todos os processos na sua fila

26

Exclusão Mútua

- Algoritmo “Token Ring”
 - É construído um anel lógico por software no qual a cada processo é atribuído uma posição no anel
 - Quando o anel é inicializado, o processo 0 ganha o “token”
 - O “token” circula no anel
 - Quando o processo ganha o “token” ele verifica se ele quer entrar na região crítica
 - Caso positivo, ele entra na região, realiza o seu trabalho e ao deixar a região passa o “token” para o elemento seguinte do anel.
 - Não é permitindo entrar em uma segunda região crítica com o mesmo “token”
 - Se o processo não quer entrar na região crítica, ele simplesmente passa o “token”.
 - Como consequência quando nenhum processo quer entrar na região crítica o “token” fica circulando pelo anel

27

Exclusão Mútua

- Algoritmo “Token Ring”
 - Problemas:
 - Se o “token” é perdido ele precisa ser regenerado
 - A detecção de um “token” perdido é difícil
 - Se um processo falha também ocorre problemas
 - A solução é fazer com que o processo que recebe o “token” confirmar o recebimento.
 - O processo que falhou pode ser retirado do anel, e o “token” enviado para o processo seguinte
 - Essa solução requer que todos os processos conheçam a configuração do anel

28

Comparação dos Algoritmos

Algoritmo	Mensagens por entrada/saída da região crítica	Retardo antes da entrada (em tempos de mensagens)	Problemas
Centralizado	3	2	Falha do Coordenador
Distribuído	$2(n-1)$	$2(n-1)$	Falha de qualquer processo
Token Ring	1 a infinito	0 a $n-1$	Perda do Token, queda de um processo

29

Algoritmo de Eleição

- Muitos algoritmos distribuídos requerem um processo como coordenador
- O objetivo é assegurar que, quando for iniciada uma eleição, ela termine com todos os processos do sistema sabendo quem é o novo coordenador

30

Algoritmo de Eleição

- Algoritmo “Bully” (Ditador)
 - Quando um processo nota que o coordenador não está respondendo a uma requisição, ele inicia uma eleição
 - P envia uma mensagem de ELEIÇÃO para todos os processos com números maiores que o seu
 - Se nenhum responde, P ganha a eleição e se torna coordenador
 - Se um processo com número maior responde, ele assume a coordenação
 - Quando um processo recebe uma msg de ELEIÇÃO de um processo de menor número, ele envia uma msg OK de volta indicando que ele vai tomar o comando
 - Depois disto ele inicia uma eleição
 - Eventualmente todos os processos abandonam a disputa, com exceção de um que é o novo coordenador
 - Ele envi uma msg a todos os processos avisando que é o novo coordenador
 - Se o processo que estava “fora do ar” voltar, ele inicia uma eleição

31

Algoritmo de Eleição

- Algoritmo de Anel
 - Baseado no uso do anel mas sem o “token”
 - Quando um processo nota que o coordendor não está funcionando, ele envia uma msg de ELEIÇÃO para o seu sucessor contendo o seu número de processo
 - A msg segue e a cada passo é incluído o número do processo na lista de msg
 - Quando a msg dá a volta no anel o novo coordenador é determinado (o processo com mais alto número da lista) e uma msg circula novamente informando quem é o novo coordenador
 - Quando a msg dá a volta no anel ela é retirada

32

Transações Atômicas

- Um processo anuncia que quer começar uma transação com um ou mais processos
- Ele executa várias operações por um certo período de tempo
- Então o processo iniciador anuncia que quer que todos confirmem (commit) o trabalho feito
- Se todos concordam com o trabalho feito se torna permanente
- Se um ou mais processos se recusam (ou falham) a situação é revertida para o estado em que todos estavam antes da transação começar
- Este procedimento é chamado de propriedade do tudo-ou-nada e facilita o trabalho do programador

33

Transações Atômicas

- Exemplo
 - Retirada de dinheiro de uma conta bancária e depósito em outra usando um PC com um modem
 - Operação em dois passos
 - Retirar (quantia, conta1)
 - Depositar (quantia, conta2)
 - Se a conexão cair depois do primeiro passo, mas antes do segundo
 - Teremos o desaparecimento do dinheiro
 - O problema é resolvido agrupando os dois passos em uma transação atômica
 - Ou os dois passos são completados ou nenhum deles serão executados
 - Se a transação falhar o modelo restaura o estado inicial

34

Propriedades da Transações Atômicas

- ACID
 - Atomicidade
 - Para o ambiente externo, a transação acontece de forma indivisível
 - Consistência
 - A transação não viola nenhuma invariante do sistema
 - Isolamento
 - Transações concorrentes não interferem uma nas outras
 - Durabilidade
 - Uma vez que a transação é efetivada (commit), a mudança é permanente

35

Implementação

- Espaço de trabalho privado
 - Quando um processo começa uma transação, ele recebe um espaço privado contendo todos os arquivos que ele tem que acessar
 - O problema com este procedimento é que o custo de copiar tudo para a área privada é geralmente proibitivo
 - Entretanto, algumas otimizações são possíveis
 - Quando um processo só lê um arquivo não é necessário fazer uma cópia
 - Em vez de copiar o arquivo todo, somente os índices do arquivo são copiados na área privada

36

Implementação

- Escrevendo um “log” (lista de intenções)
 - Os arquivos são relamente modificados, mas antes de qualquer bloco ser modificado é registrada, em um armazenamento estável, a transação que está fazendo mudança
 - Que arquivo e boco estão sendo modificados
 - Os valores velhos e os novos
 - Se a transação abortar, o “log” pode ser usado para voltar o sistema ao estado original
 - Começando do fim para o início, cada registro do “log” é lido e a mudança é feita
 - O “log” também pode ser usado para recuperar o sistema de uma falha

37

Protocolo de Aceitação de Duas Fases

- A confirmação (commit) da transação precisa ser feita atomicamente, isto é, de forma instantânea e indivisível
 - Isto requer a cooperação de todos os processos participantes da transação
- Um dos processos é o coordenador, usualmente o que executou a transação
- O protocolo começa quando o coordenador escreve em uma entrada no arquivo “log” dizendo que ele começou o protocolo, enviado em seguida uma msg para cada processo participante dizendo para se prepararem para a confirmação
- Cada um dos subordinados irá escrever no “log” e envia a sua decisão para o coordenador
- Ao receber todas as respostas o coordenador saberá se a transação teve sucesso ou não
- O coordenador comunica a todos os outros participantes a decisão final

38

Controle de Concorrência

- Locking – Travamento
 - Quando um processo precisa ler ou escrever em um arquivo (ou qualquer outro recurso) como parte da transação, ele primeiro trava o arquivo
 - O gerente do “lock” mantém uma lista dos arquivos travados, e rejeita qualquer tentativa de acesso por outros processos
 - Esta operação normalmente é feita pelo sistema e não requer a interferência do programador
 - Adquirir e liberar “locks”, precisamente no momento em que eles são necessários, pode levar a deadlocks
 - Por causa disto a maioria das implementações usam o “two-phase locking”
 - Trava todos os recursos
 - Utiliza
 - Libera todos os recursos

39

Controle de Concorrência

- Otimista
 - A idéia desta idéia é simples
 - execute tudo (sem prestar atenção no que os outros estão fazendo)
 - Se tiver algum problema, se preocupe depois
 - Na confirmação são verificadas todas as outras transações para ver se algum dos arquivos foram modificados desde que a transação começou
 - Se houve modificação, a transação é abandonada, caso contrário é confirmada
 - Permite o máximo de paralelismo
 - Em condições de muita carga no sistema a probabilidade de falha é bem maior

40