Unidade 4

Programação Distribuída

- Sistemas Distribuídos
- Tempo Global
- Estado Global
- Transações Distribuídas
- Controle de Concorrência

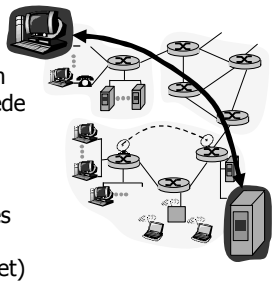
Sistemas Distribuídos

- O que são?
  - São sistemas compostos por diversas partes cooperantes que são executadas em máquinas diferentes interconectadas por uma rede
- Exemplos de aplicações distribuídas
  - WWW, ICQ, IRC, Morpheus/Kazza, etc.
  - Sistemas bancários
  - Sistemas de gerenciamento de redes de telecomunicações, transmissão de energia, etc.
  - Sistemas de informação de grandes empresas
  - ...

2

Sistemas Distribuídos

- Estrutura Física
  - Máquinas (*hosts*) são conectadas a um provedor (ISP) ou rede local, metropolitana, sem fio, etc.
  - Estas redes são interligadas por redes de longa distância públicas (ex.: Internet) ou privadas



© Kurose & Ross

3

Sistemas Distribuídos

- Vantagens
  - Usam melhor o poder de processamento por distribuir a carga entre as máquinas
  - Podem apresentar melhor desempenho, maior confiabilidade, e suportar um maior número de usuários
  - Permitem compartilhar dados e recursos e reutilizar serviços já disponíveis
  - ...

4

Sistemas Distribuídos

- Dificuldades
  - Acoplamento fraco
    - Máquinas trocam dados pela rede
    - Tempo de comunicação ilimitado → pode comprometer o funcionamento do sistema
    - Como reduzir o tráfego na rede?
  - Não-deterministas
    - Comportamento pouco previsível
    - Como evitar congestionamento/sobrecarga?

5

Sistemas Distribuídos

- Dificuldades
  - Sistemas heterogêneos
    - Máquinas, linguagens e S.O.´s diferentes
    - Como tratar estas diferenças?
  - Não há uma base de tempo global
    - Em geral os relógios não estão sincronizados
    - Como ordenar os eventos no sistema?
  - Difícil ter uma visão global
    - Transações envolvem várias máquinas
    - Como manter o estado global do sistema?

6

Sistemas Distribuídos

- Dificuldades
  - Acesso concorrente a dados e recursos
    - Dados/recursos em diferentes máquinas
    - Como controlar o acesso concorrente?
    - Como evitar deadlocks?
  - Difícil gerenciar e manter o sistema
    - Máquinas dispersas pela rede
    - Administração pode ser independente

7

Sistemas Distribuídos

- Dificuldades
  - Sujeito a falhas
    - As máquinas e a rede podem falhar
    - Como evitar que estas falhas comprometam o funcionamento do sistema?
  - Segurança
    - Mais difícil controlar o acesso a dados e recursos em sistemas distribuídos
    - Como garantir a segurança do sistema e o sigilo dos dados trocados pela rede?

8

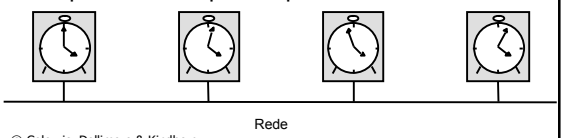
Tempo Global

- É necessário determinar com exatidão o tempo no qual os eventos ocorrem nos sistemas computacionais
  - Registro de acesso ao sistema (login/logout)
  - Determinação do tempo de uso (ex.: telefonia, acesso à rede, vídeo sob demanda, videoconferência, etc.)
  - Registro de transações bancárias
  - Registro de compra/venda de produtos
  - Auditoria e segurança
  - ...

9

Tempo Global

- Os relógios das máquinas não são sincronizados
  - Os relógios não são precisos: há um desvio entre o tempo real e o marcado pelo relógio
  - Mesmo que se acerte os relógios, com o passar do tempo eles perdem a sincronia



10

Tempo Global

- Possíveis soluções:
  - Relógios atômicos colocados em todas as máquinas, marcando o tempo universal (UTC)
    - Inviável!
  - Receptores de satélite em todas as máquinas (satélites difundem o sinal de tempo UTC)
    - Sairia caro demais!
  - Sincronização pela rede
    - Servidores de tempo, com relógios precisos, difundem um sinal de tempo pela rede

11

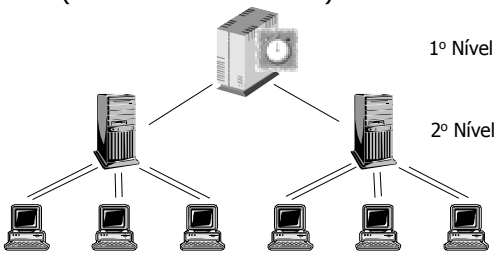
Tempo Global

- NTP (*Network Time Protocol*)
  - Hieraquia de Servidores
    - Servidores de 1º nível (ou primários): possuem relógios precisos
    - Servidores de 2º, 3º, ..., Nº nível: recebem o tempo do nível anterior
  - Clientes (máquinas dos usuários finais): perguntam o tempo a um servidor e atualizam seus relógios com base na resposta recebida

12

Tempo Global

- NTP (*Network Time Protocol*)



Tempo Global

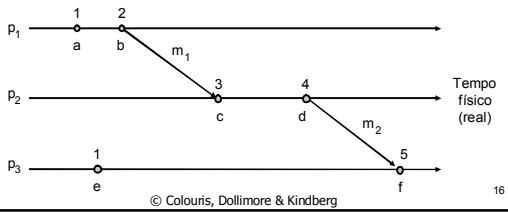
- Características do NTP
  - Barato por não exigir relógios precisos (atômicos ou com GPS) em todas as máquinas
  - Impreciso devido ao tempo de propagação das mensagens pela rede não ser constante
    - O algoritmo de sincronização dos relógios leva em conta uma estimativa do tempo de propagação das mensagens pela rede
    - Mesmo assim, existe um erro implícito que pode chegar a dezenas de ms, e que deve ser levado em conta pelos sistemas

Tempo Global

- Relógios Lógicos (Lamport, 1978)
  - Usados para definir a ordem dos eventos produzidos por um grupo de processos
  - Cada processo mantém um contador, e atribui marcas de tempo (*timestamp*) aos eventos (ex.: envio ou recepção de mensagens)
  - Cada mensagem trocada pela rede carrega consigo o *timestamp* atribuído pelo remetente
  - Os eventos são ordenados pelos *timestamps*

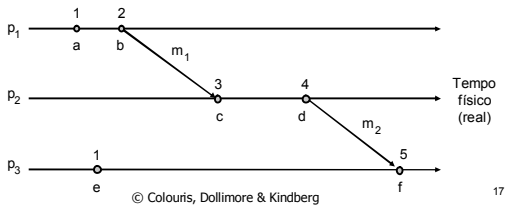
Tempo Global

- Relógios Lógicos – Exemplo:
  - Analisando os *timestamps* temos que: a→b→c→d→f ; a||e ; e→f



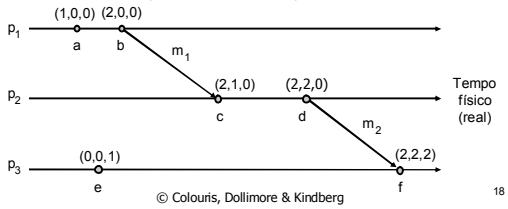
Tempo Global

- Relógios Lógicos com Ordenação Total
  - O identificador do processo é usado para definir a ordem de eventos concorrentes
  - Assim podemos assumir que a→e



Tempo Global

- Relógios Vetorias (Mattern, 89; Fidge, 91)
  - Usa vetores de *timestamps* com o valor conhecido do relógio lógico de cada processo
  - Se $V_x[i] \leq V_y[i]$ $\forall i$ e $V_x \neq V_y$, então x→y



Estado Global

- Estado Global de um Sistema Distribuído
  - É o conjunto dos estados locais dos processos
  - Como o estado local de cada processo muda continuamente, é muito difícil conhecer o estado global do sistema
  - Tentar obter o estado global consultando cada processo pode levar a inconsistências
- Exemplos:
  - Crianças trocando figurinhas
  - Saldo das contas de correntistas de um banco

19

Estado Global

- Quando é preciso obter o Estado Global?
  - *Garbage Collection* : verificar se existem clientes de objetos acessados remotamente
  - *Debugging* Distribuído: verificar os valores das variáveis de um programa distribuído
  - Controle de *Membership* : listar os membros de um grupo de processos ou usuários
  - ...
- Como obter um Estado Global consistente?
  - É preciso considerar o efeito das mensagens trafegando na rede para obter o estado global

20

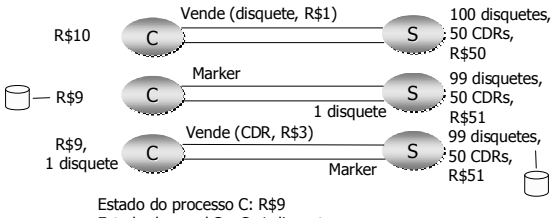
Estado Global

- Protocolo de *Snapshot* (Chandra & Lamport,1985)
  - O processo que inicia o protocolo salva seu estado e envia uma mensagem *marker* por todos os seus canais de saída
  - Os demais processos salvam seu estado ao receber o 1º *marker* e enviam *markers* por todos os seus canais de saída
  - O estado de um canal consiste nas mensagens recebidas pelo canal até chegar um *marker*
  - Assim obtém o estado de todos os processos e canais de comunicação do sistema

21

Estado Global

- Protocolo de *Snapshot* – Exemplo:



Estado do processo C: R$9
Estado do canal S->C: 1 disquete
Estado do processo S: 99 disquetes, 50 CDRs, R$51
Estado do canal C->S: vazio

22

Estado Global

- Replicação de Dados
  - Os dados do sistema podem estar replicados
  - É necessário garantir a consistência de réplicas → réplicas com o mesmo valor!
  - Alterações nos dados devem ser propagadas para as máquinas com réplicas do dado
  - Replicação dificulta a obtenção do estado global do sistema
  - Cada réplica deve atualizar seu estado ao reintegrar-se ao sistema; qualquer inconsistência nos dados deve ser resolvida

23

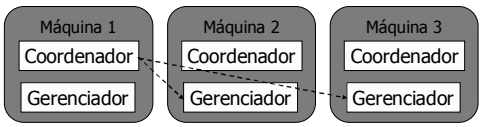
Transações Distribuídas

- Transações distribuídas alteram o estado de vários processos em várias máquinas
  - Devem ser garantidos a atomicidade, a consistência, o isolamento e a durabilidade
  - Falhas podem impedir a execução de uma transação
    - Falha de software
    - Falha da máquina
    - Falha na comunicação
  - Nem sempre é possível saber quem falhou!

24

Transações Distribuídas

- Execução de Transações Distribuídas
  - Coordenadores de transações coordenam a execução das transações iniciadas localmente
  - Gerenciadores de transações administram a execução da parte local da transação



25

Transações Distribuídas

- Coordenador de Transações
  - Inicia a execução da transação
  - Divide a transação em sub-transações e as distribui no sistema para execução
  - Efetiva ou faz o *rollback* em todos os *sites*
- Gerenciador de Transações
  - Controla a execução da sub-transação local
  - Matém um *log* das operações executadas para permitir a recuperação da transação

26

Controle de Concorrência

- Controle de Concorrência em Sist. Distrib.
  - Protocolos para controlar o acesso a dados e recursos compartilhados
  - Mecanismos de detecção de *deadlock*
- Protocolo de Bloqueio Único (Centralizado)
  - Uma máquina funciona como gerenciador de bloqueios, que administra todos os pedidos de bloqueio de dados ou recursos no sistema
  - É simples, pois a detecção de *deadlocks* é local, mas é sujeito a falhas e pouco escalável

27

Controle de Concorrência

- Protocolo de Bloqueio Múltiplo (Descentralizado)
  - São usados vários gerenciadores em diferentes máquinas
  - Cada gerenciador administra os bloqueios de um conjunto de dados/recursos
  - Evita o 'gargalo' do protocolo centralizado, mas dificulta a detecção de *deadlocks*
  - Impede o acesso ao dado/recurso se o gerenciador falhar, mesmo que a máquina com o dado/recurso esteja funcionando

28

Controle de Concorrência

- Protocolo de Bloqueio Distribuído
  - Cada máquina do sistema tem o seu gerenciador de bloqueios
  - Cada gerenciador administra os bloqueios de dados/recursos locais
  - Assim como o anterior, evita o 'gargalo', mas dificulta a detecção de *deadlocks*
  - Impede o acesso somente quando a máquina com o dado/recurso falhar

29

Controle de Concorrência

- Protocolo com Cópia Primária
  - Usado no acesso a dados replicados
  - Cada dado replicado possui uma cópia primária em um máquina
  - O bloqueio é solicitado à máquina com a cópia primária daquele dado
  - Simples e escalável, mas impede o acesso às réplicas se o primário falhar, e dificulta a detecção de *deadlocks*

30

Controle de Concorrência

- Protocolo da Maioria
  - Usado no acesso a dados replicados
  - Cada máquina possui um gerenciador para bloqueio de réplicas de dados locais
  - O bloqueio de dados com réplicas é concedido se a maioria dos *sites* permitir
  - Mais complexo para implementar que os anteriores, e difícil detectar *deadlocks*

31

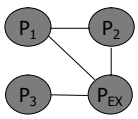
Controle de Concorrência

- Protocolo Parcial
  - Cada máquina tem o seu gerenciador de bloqueio das réplicas locais de dados
  - Bloqueios compartilhados são obtidos contatando o gerenciador de somente uma réplica do dado
  - Bloqueios exclusivos devem ser solicitados em todos os gerenciadores de bloqueio de todas as máquinas com réplicas
  - Bem escalável, mas difícil detectar *deadlocks*

32

Controle de Concorrência

- Detecção de *Deadlocks* Distribuídos
  - Cada máquina monta um gráfico de espera com os bloqueios mantidos localmente
  - Um nó P_{EX} é adicionado ao gráfico para representar uma espera por recursos externos bloqueados
  - Um ciclo passando apenas por nós locais representa um *deadlock*
  - Um ciclo envolvendo P_{EX} indica um possível *deadlock*, que deve ser verificado junto às máquinas envolvidas



33