Validação de Ações Atômicas Distribuídas

Thierson Couto Rosa

Validação de Ações Atômicas Distribuídas

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Thierson Couto Rosa e aprovada pela Comissão Julgadora.

Campinas, 21 de julho de 1994.

Prof. Ricardo O. Anido
Orientador

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Validação de Ações Atômicas Distribuídas[1]

Thierson Couto Rosa[2]

Departamento de Ciência da Computação
IMECC - UNICAMP

Banca Examinadora:

- Ricardo O. Anido (Orientador)[3]
- Orlando Loques Filho[4]
- Eliane Martins[5]
- Maria Beatriz Felgar de Toledo (Suplente)[5]
- Maurício Ferreira Magalhães (Suplente)[6]

---

[1]Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação da UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

[2]O autor é Bacharel em Ciência da Computação pela Universidade Federal de Goiás.

[3]Professor do Departamento de Ciência da Computação - IMECC - UNICAMP.

[4]Professor da UFF-RJ

[5]Professora do Departamento de Ciência da Computação - IMECC - UNICAMP.

[6]Professor do Departamento de Computação e Automação - FEE - UNICAMP.

A Tiettre Couto Roza - um exemplo verdadeiro de inteligência, coragem, honestidade, dedicação apaixonada à conquista de objetivos.

À Ana Maria C. Couto Roza - amor, carinho e abnegação.

Agradecimentos

Agradeço aos meus pais Tiettre e Ana Maria, à minha tia Tilly e aos meus irmãos Marcello e Mirtes, que apesar de todos os problemas e dificuldades jamais pouparam carinho e apoio que me foram essenciais durante a realização deste trabalho.

Ao meu orientador Dr. Ricardo O. Anido, por ter sugerido o tema da dissertação e pelas importantes sugestões e correções propostas.

Em particular, a Fábio Nogueira de Lucena pela amizade, colaboração nas mais diversas situações e por ter ter lido pacientemente esta dissertação, sugerindo importantes correções.

A Mário Massato Harada, por ter lido e aconselhado correções a este trabalho, e pelo apoio recebido.

Aos meus colegas de mestrado da turma de 1991 pelo conforto nos momentos de dificuldade e pelos momentos de descontração.

A Marcus Vinícius, Maurício Fernandez e Atta, pela companhia constante e alegre.

Aos professores e funcionários do Departamento de Ciência da Computação da UNICAMP.

Aos meus colegas do Departamento de Estatística e Informática, pela compreensão e incentivo.

À Universidade Federal de Goiás por ter me dispensado das atividades didáticas durante o curso de mestrado e à Comissão de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro a este trabalho.

Resumo

Neste trabalho estudamos o problema de validação de ações atômicas distribuídas e de protocolos de validação que visam resolvê-lo. Além do estudo de protocolos de validação existentes na literatura, propomos uma variação do protocolos de duas fases, denominada protocolo semibloqueante.

No Capítulo 1 são apresentados o conceito de ação atômica distribuída e uma motivação para sua utilização. Também são descritas as funções básicas de um sistema gerenciador genérico de ações atômicas.

No Capítulo 2 é definido um modelo de sistema distribuído. Com base neste modelo, é definido o problema de validação de ações atômicas distribuídas e são apresentados formalismos para descrição do funcionamento de protocolos de validação em geral. São também estudadas as condições necessárias para a existência de protocolos não-bloqueantes e de recuperação independente de processos falhos.

No Capítulo 3 são estudados protocolos de validação de duas fases que visam a eficiência quanto à finalização da ação. Estes protocolos são descritos com base em um conjunto de parâmetros definidos no início do capítulo.

No Capítulo 4 são estudados protocolos de validação de duas fases que visam ser tolerantes a falhas. No início do capítulo é adotado um padrão para descrição desses protocolos.

O Capítulo 5 descreve um conjunto de protocolos que não se enquadram na categoria de protocolos de duas fases.

No Capítulo 6 apresentamos o protocolo semibloqueante, um protocolo intermediário entre os protocolos de duas fases e os protocolos não-bloqueantes. Durante uma execução sem falhas, o protocolo comporta-se como um protocolo bloqueante. Quando surgem falhas no sistema, o protocolo passa a funcionar como um protocolo de três fases não-bloqueante.

A Capítulo 7 finaliza o texto com alguns comentários e sugestões de trabalhos futuros.

Abstract

In this work we study the atomic actions commit problem and a set of commit protocols which aim to solve this problem. We also propose a new two-phase commit protocol, the *semiblocking* commit protocol.

In Chapter, 1 we give the definition of distributed atomic actions. We also present the basic definitions of a generic atomic action management system.

In Chapter 2, we define a model of distributed system which is used to define the atomic action commit problem and the formalism used to describe the operation of generic commit protocols.

In Chapter 3, we describe several efficient two-phase commit protocols from the literature. The description of these protocols are based on a set of parameters which are defined at the beginning of the chapter.

In Chapter 4, we describe a series of published fault-tolerant commit protocols. At the beginning of the chapter, we define a set of parameters which are used for the descriptions of these protocols.

Chapter 5 contains a set of protocols which do not belong to the two-phase category of commit protocols.

In Chapter 6, we introduce the semiblocking commit protocol which stands between the class of two-phase commit protocols and that of nonblocking commit protocols. During a failure-free execution, the protocol behaves like a blocking commit protocol. Whenever a fault occurs in the system, the semiblocking commit protocol begins to act like a nonblocking three-phase commit protocol.

Chapter 7 closes the text with some comments and suggestions for future work.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Sistemas de computadores distribuídos têm-se tornado comuns em diversas áreas de trabalho. O baixo custo e a popularidade dos microcomputadores, aliados ao desenvolvimento da tecnologia de redes locais e das telecomunicações têm permitido a utilização desses sistemas em larga escala.

Um dos principais motivos que dificultam o desenvolvimento de aplicações distribuídas é a ocorrência de falhas aleatórias nos computadores e canais de comunicação do sistema. Espera-se que as aplicações continuem executando apesar de ocorrência de falhas parciais. Mecanismos de programação têm sido propostos e utilizados com o objetivo de tornar menos complexa a programação de aplicações tolerantes a falhas. Dentre esses mecanismos destaca-se a *ação atômica* ou *transação atômica*. Este mecanismo possibilita que a execução de um conjunto de operações definido como uma ação atômica tenha a propriedade de que as alterações causadas no sistema pelas operações de uma ação atômica são todas mantidas ou são todas desfeitas. Esta propriedade permite que a situação existente no sistema antes da execução de uma ação atômica volte a existir caso uma falha ocorra durante sua execução. Além disso, o mecanismo garante também que as alterações causadas por uma ação atômica não são desfeitas após sua execução caso ocorram falhas futuras.

Uma ação atômica pode ser distribuída, isto é, as operações que a constituem podem ser executadas em diferentes computadores do sistema. Esta possibilidade dá origem ao seguinte problema: como determinar se a ação atômica executou com sucesso, ou se todas as operações devem ser desfeitas devido a uma falha em uma operação? Esse problema é denominado *problema de validação de ações atômicas distribuídas*. As soluções deste problema devem incluir um mecanismo de troca de mensagens entre computadores que executam operações de uma mesma ação atômica. O controle de envio dessas mensagens é feito através de um protocolo que é denominado *protocolo de validação*.

Este trabalho tem como objetivos o estudo do problema de validação e de protocolos que visam resolvê-lo, bem como a apresentação de um novo protocolo de validação de ações atômicas distribuídas.

1.1 Sistemas Distribuídos e Tolerância a Falhas

Um sistema distribuído é formado por um conjunto de computadores que comunicam entre si apenas por troca de mensagens através de um sistema de comunicação. Considera-se que cada computador é constituído por um processador e por dois tipos de memória: *memória estável* e *memória volátil*. A *memória volátil* é caracterizada por perder os dados que armazena após a ocorrência de falha ou após o computador ser desligado. A *memória estável* continua a reter os dados mesmo após uma falha ou quando cessa o fornecimento de energia ao computador. Geralmente, o acesso à memória estável é mais lento que o acesso à memória volátil.

Os componentes de um sistema distribuído, com exceção da memória estável, estão sujeitos aos seguintes tipos de falha:

- **Falha de parada** - o componente apenas para de funcionar. Ele pode não se recuperar, pode se recuperar e voltar ao estado em que estava no momento da falha, ou pode retornar a um estado inicial predefinido. Há, ainda, uma variação de falha de parada em que, após a recuperação do componente, parte do seu estado é o estado inicial e parte é o estado existente no momento da falha.

- **Falha de temporização** - o componente continua funcionando e gera resultados corretos, porém não no prazo esperado. As falhas de temporização podem ser ainda classificadas em *falhas de atraso* e *falhas por antecipação*.

- **Falha de valor ou falha bizantina** - o componente continua funcionando após a falha, porém produz resultados errados. Por exemplo, a execução de um programa que gera como saída valores errados ou inesperados, ou canais de comunicação que alteram o conteúdo das mensagens ou que criam mensagens na rede.

É possível, ainda, ocorrer combinações dos tipos de falhas acima. Um nó, por exemplo, pode atrasar a produção de uma determinada saída e o valor produzido pode estar errado. Uma combinação de falhas de nós e dos canais de comunicação pode causar um tipo de falha denominado *falha de partição*. Quando esta falha ocorre, os nós ficam isolados em grupos ou partições de nós. Enquanto persistir a falha, cada nó consegue comunicar-se apenas com os nós que estão na partição a qual ele pertence. As falhas de partição podem ser classificadas em *falha de partição simples* e *falha de partição múltipla*. A primeira gera somente duas partições, a segunda gera mais de duas partições. Uma falha de partição é *otimista* se as mensagens enviadas são devolvidas aos nós emissores no momento da falha. A falha é *pessimista* se as mensagens são perdidas no momento em que ocorre a falha.

Um sistema é considerado tolerante a falhas se possuir pelo menos uma das seguintes propriedades:

- ter um comportamento bem definido após a falha de seus componentes, ou,

- continuar fornecendo seus serviços mesmo após as falhas de alguns componentes. Neste caso, diz-se que o sistema mascara as falhas ocorridas.

Existem diversos mecanismos de software e de hardware que podem ser utilizados com o objetivo de tornar um sistema tolerante a falhas. Este texto trata particularmente de ações atômicas – uma ferramenta de programação utilizada para permitir que um sistema tenha um comportamento bem definido após a ocorrência de uma falha.

1.2 Motivação Para o Uso de Ações Atômicas

Os dados de um sistema de computação podem ser classificados quanto ao seu tempo de duração em duas classes: *dados temporários* e *dados de longa duração*. Os primeiros existem apenas durante a execução do programa que lhes deu origem. As variáveis auxiliares e os arquivos temporários são exemplos dessa categoria de dados. Em contrapartida, um dado de longa duração pode persistir no sistema, mesmo após o término do programa que o criou, ou dos programas que dele fazem uso. Como exemplo dessa classe de dados podemos citar os bancos de dados.

Em sistemas de computação distribuída há sempre a preocupação em manter a consistência dos dados, especialmente dos dados de longa duração. Há determinadas aplicações que devido aos seus requisitos de segurança exigem que a execução do programa ou trecho de programa que as executam seja indivisível com relação a falhas e a interferências de outros programas. O exemplo tradicional desse tipo de aplicação é a operação de transferência de valores entre contas bancárias. O programa que executa essa transferência deve ser indivisível com relação aos seguintes requisitos:

1. a subtração do valor de transferência da primeira conta e a adição desse valor na segunda conta são duas operações tais que ambas são executadas, ou nenhuma delas é executada. Se ocorre uma falha enquanto o valor não for adicionado na segunda conta, deve haver uma forma de desfazer a subtração realizada, restaurando-se o valor inicial da primeira conta. Nesse caso, a operação de transferência é abortada e pode ser tentada posteriormente;

2. uma operação concorrente não deve ter permissão de leitura ou escrita a nenhuma das contas bancárias durante a operação de transferência. Como contra-exemplo, considere que uma operação de impressão de saldo das duas contas esteja sendo executada concorrentemente à operação de transferência. Se a operação de impressão tiver acesso às contas durante a transferência, a seguinte seqüência de eventos poderia ocorrer:

   a. a operação de transferência realiza a subtração do valor na conta 1;

   b. a operação de impressão imprime o valor da conta 1;

   c. a operação de impressão imprime o valor da conta 2;

   d. a operação de transferência adiciona o valor à conta 2.

É claro que os dois resultados impressos deixariam os proprietários das duas contas surpresos e o banco em uma situação delicada.

Há, ainda, um terceiro requisito importante que está implícito nesse exemplo:

3. os resultados da subtração e da adição realizadas devem permanecer no sistema após o término da operação de transferência.

A terceira exigência tem uma implicação quanto aos dispositivos de armazenamento dos dados que representam as contas: eles devem ser armazenados em memória estável.

O programa de transferência bancária do exemplo acima pode ser implementado por meio do mecanismo de *ação atômica* o qual garante a satisfação dos três requisitos anteriores. As operações de subtração e de adição são definidas como sendo parte de uma ação atômica. A Figura 1.1 ilustra uma possível codificação do programa de transferência.

1.3 Definição de Ação Atômica

Dado o exemplo acima, pode-se definir o conceito de ação atômica como sendo uma seqüência de operações que possui duas propriedades: *indivisibilidade* e *permanência de resultado.* A *indivisibilidade* abrange dois significados: primeiro, que não é possível a uma ação atômica A_2 ter acesso aos resultados gerados por uma operação de uma ação atômica A_1 enquanto A_1 não se finalizar. Segundo, todos ou nenhum dos resultados produzidos pelas operações de uma ação atômica são visíveis por outras ações atômicas (mesmo quando ocorre falha durante a execução da ação). Quando todas as operações de uma ação atômica são executadas e os seus resultados podem ser visíveis, dize-se que a ação *foi validada.* Se os resultados das operações são desfeitos para que não sejam visíveis por outras ações, diz-se que a ação *foi abortada.*

A *permanência de resultados* é a propriedade que garante que os resultados produzidos por uma ação sobre os dados de longa duração não sejam desfeitos por falhas que possam ocorrer futuramente no sistema.

Neste trabalho serão consideradas ações atômicas distribuídas, que são ações cujas operações podem estar espalhadas em diversos nós da rede. Um programa cria uma ação atômica em um nó e operações locais podem ativar a execução remota de uma ou mais operações. Cada operação remota pode, por sua vez, ativar novas operações locais ao nó onde executa, ou remotamente em outros nós.

As propriedades das ações atômicas são geralmente implementadas pelos seguintes módulos lógicos: módulo de controle de concorrência, módulo de gerenciamento dos dados e módulo de validação. As próximas seções desse capítulo tratam de modo resumido cada um desses módulos.

```
algoritmo transferência (int conta_fonte; int conta_destino; real valor);
var
    ia:              identificador-de-ação;
    cr:              cóodigo-de-retorno;
    Pont-fonte:      ↑ conta;
    Pont-destino:    ↑ conta;
início
   {início da transação}
   ia ← InícioTransação;

   {Subtrair da primeira conta}
   cr ← verifique(ia, conta_fonte, Pont-fonte);
   se (cr <> sucesso) então
   início
      escreva("conta inválida", conta-fonte);
      AborteTransação;
   fim
   subtrair(ia,Pont-fonte,valor);

   {adicionar à segunda conta}
   cr ← verifique(ia, conta_destino, Pont-destino);
   se (cr <> sucesso) então
   início
      escreva("conta inválida", conta-destino);
      AborteTransação;
   fim
   somar(ia,Pont-destino,valor);

   {validar a transferencia}
   cr ← ValideTransação;
   se (cr <> sucesso) então
   escreva("Transferência abortada");
   senão
   escreva("Transferência validada");
fim
```

Figura 1.1: Procedimento para transferência de valores entre contas bancárias.

1.4 Módulo de Controle de Concorrência

Na seção anterior, foi apresentado o exemplo em que as operações da tarefa de impressão de saldos interferia na execução da tarefa de transferência de fundos, produzindo resultados finais errados. Existem inúmeros exemplos em que a interferência de uma ação na execução de outra produz resultados inconsistentes. Esse problema se deve à intercalação de duas ou mais operações de ações diferentes. Se forem consideradas as duas tarefas da seção anterior como duas ações, pode-se observar que se uma ação fosse executada completamente antes do início da execução da outra, não haveria problema algum de interferência. A execução em série das ações atômicas, entretanto, não é desejável, pois faz com que os recursos do sistema fiquem sub-utilizados.

A solução consiste, portanto, em exigir que uma execução concorrente de várias ações seja equivalente a alguma execução em série dessas ações. Uma execução concorrente com essa característica é denominada *execução seriável.*

É necessário, portanto, que o módulo de controle de concorrência - MCC ordene as operações das ações atômicas de tal modo que a execução concorrente resultante dessa ordenação seja seriável.

O método de controle de concorrência mais utilizado é o *controle de trancas em duas fases ("Two Phase Lock"* ou *2PL).* Neste método, cada item dado possui uma tranca associada. Quando uma transação vai fazer acesso ao dado, ela faz o pedido de acesso ao MCC. Se nenhuma transação está com a posse da tranca, o MCC libera a tranca para a ação requerente. Uma vez de posse da tranca, a ação pode fazer uso do dado correspondente. Se entretanto, outra ação estiver com a posse da tranca, a ação que a requisitou por último permanece esperando a liberação dessa tranca para utilizar o item dado correspondente.

Há dois tipos de trancas associadas a cada dado x, conforme o tipo de operação realizada sobre esse dado. Estes tipos são: *tranca para leitura* $tl[x]$ e *tranca para gravação* $tg[x]$. Duas trancas são conflitantes entre si se fazem acesso ao mesmo dado e uma delas requer tranca para gravação.

Define-se dois tipos de operações sobre as trancas: *posse da tranca* $tg_i[x]$ *pela ação* A_i - $P(tg_i[x])$ e *liberação da tranca* $tg_i[x]$ *pela ação* A_i - $L(tg_i[x])$. A execução das operações P e L são gerenciadas pelo MCC que segue o método de controle de tranca em duas fases - 2PL.

O controle de tranca em duas fases baseia-se em três regras:

Regra 1.4.1 *Quando o MCC recebe um pedido de posse* $P(tp_i[x])$ *de uma ação* A_i, *ele verifica se existe um pedido de posse* $P(tq_j[x])$ *anterior feito por uma ação*A_j *sobre o mesmo dado* x. *Se houver tal pedido, o MCC concede a posse da tranca* $tp_i[x]$ *à ação* A_i, *somente se* $tp_i[x]$ *e* $tq_j[x]$ *não forem conflitantes. Caso contrário, o pedido* $P(tp_i[x])$ *é mantido em uma fila de espera até que o dado* x *possa ser utilizado pela ação* A_i.

Regra 1.4.2 *Após ceder a posse de uma tranca $tp_i[x]$ a uma ação A_i, o MCC não pode liberar uma tranca sobre x para outra ação conflitante, enquanto A_i não realizar a operação $L(tp_i[x])$.*

Regra 1.4.3 *Se o MCC executa uma operação $L(tq_i[x])$ para uma ação A_i, ele não pode atender a mais nenhum pedido de posse de tranca sobre nenhum dado para ação A_i.*

A Regra 1.4.1 garante que as operações são escalonadas na ordem em que as trancas são cedidas pelo MCC. A Regra 1.4.2 garante que a execução das operações é feita na ordem em que são escalonadas.

A Regra 1.4.3 define as duas fases do 2PL. Existe uma fase que corresponde à aquisição de trancas por uma determinada ação. Essa fase inicia-se no instante em que a ação executa o primeiro pedido de posse de alguma tranca e termina no momento em que a ação faz seu primeiro pedido de liberação de alguma tranca. A segunda fase corresponde à liberação das trancas obtidas por uma ação. O início dessa fase ocorre quando a ação faz o primeiro pedido de liberação de tranca e termina após a liberação de todas as trancas adquiridas pela ação. Quando a ação entra na segunda fase, ela não pode mais fazer pedidos de posse de tranca. Todas as trancas utilizadas por uma ação são automaticamente liberadas quando a ação finaliza (aborta ou valida).

Apesar de garantir execuções seriáveis, os MCCs que têm como base o método 2PL estão sujeitos à ocorrência de *espera mútua* ou *"deadlock"*. É necessário, portanto, que o MCC possua também mecanismos de detecção e resolução de "deadlock". A resolução de "deadlock" tem como conseqüência o aborto de uma das ações para que as demais possam obter as trancas requisitadas, interrompendo a situação de espera mútua.

O método de controle de concorrência 2PL é o método utilizado na maioria dos sistemas. Nesta seção foi apresentada a versão básica desse método, embora haja variações. Existem outras técnicas de controle de concorrência que não utilizam o conceito de trancas, como por exemplo, o método de controle pelo tempo em que a ação foi criada ("time-stamp") e o método certificador. Mais informações sobre o assunto podem ser obtidas em [BVHNG87].

1.5 Módulo Gerenciador de Dados

Os principais objetivos do módulo gerenciador de dados são os seguintes:

1. efetuar operações de leitura e gravação sobre dados de longa duração;
2. executar a validação de uma ação, efetivando os valores por ela gravados;
3. executar o aborto de uma ação, garantindo que os valores dos dados utilizados pela ação sejam os mesmos existentes antes da ação se iniciar;

4. restaurar o último *estado validado* do sistema após o nó se recuperar de uma falha. Entende-se por *último estado validado* a versão mais recente do conjunto de dados do sistema formada apenas por valores gravados por ações validadas.

A memória estável está conceitualmente dividida em duas regiões de dados. Em uma dessas regiões estão armazenados os itens de dados de longa duração. Denomina-se essa região de área de dados do sistema. A outra região é utilizada pelo gerenciador de dados para armazenar os dados necessários para execução de suas operações básicas (gravação, leitura, aborto, validação e recuperação). Essa região é denominada *log*. O log representa a história das execuções das ações no sistema e pode ser considerado, conceitualmente, como um arquivo seqüencial que contém entradas do tipo $[T, x, v]$, onde T é o identificador da ação que gravou o valor v no dado x. As entradas são gravadas no log na ordem correspondente à execução das operações; portanto, o log corresponde a um retrato do escalonamento feito pelo MCC. Além dessas entradas, o log possui três listas: a lista de ações validadas, a lista de ações abortadas e a lista de ações em execução.

O modo como o módulo gerenciador de dados executa as operações de gravação em memória estável, operações de aborto, validação e recuperação, depende da estratégia utilizada por ele para movimentar os dados da memória volátil para a memória estável. Há basicamente duas estratégias. A primeira consiste em permitir que os novos valores sejam atualizados diretamente nos itens de dados situados na área de dados do sistema. Esses valores podem ter sido gravados por ações que estavam em execução no momento em que a movimentação dos dados foi realizada. Com isso, se uma ação aborta, o algoritmo de recuperação deve remover os valores alterados pela ação abortada e restaurar os valores que existiam anteriormente em cada item. Essa técnica consiste, portanto, em "desfazer" as alterações feitas na área de dados do sistema. Para tanto, o módulo gerenciador de dados deve copiar o valor do item existente na área de dados do sistema para o log antes de mover o valor da memória volátil para a área de dados do sistema.

Outra estratégia consiste em copiar os dados da memória volátil inicialmente para o log. Se a ação aborta, não é necessário recuperar os dados na área de dados do sistema, uma vez que eles não tiveram os seus valores alterados. Porém, essa estratégia permite que uma ação seja validada sem que os itens na área de dados do sistema sejam atualizados. Se uma falha ocorre depois da validação da ação, quando o nó se recupera, o módulo gerenciador de dados correspondente deve "refazer" os resultados gerados pela ação validada, copiando-os do log para a área de dados do sistema.

Quando uma ação é abortada ou validada, o módulo gerenciador de dados inclui o identificador da ação na lista correspondente e o remove da lista de ações em execução. Durante a operação de recuperação, o módulo gerenciador de dados consulta essas listas e, de acordo com a estratégia utilizada, restaura o último estado validado do sistema.

1.6 Validação de Ações Atômicas

Para garantir a atomicidade de ações atômicas cujas operações estão distribuídas em vários nós não são suficientes apenas o módulo de controle de concorrência e o módulo gerenciador de dados. É necessário que o módulo gerenciador de ações atômicas do nó onde a ação foi iniciada se comunique com os módulos gerenciadores de dados dos demais nós onde a ação executa operações remotas. Essa comunicação se faz necessária para que haja um consenso entre todas as operações remotas sobre o aborto ou validação da ação. Mesmo que uma operação remota tenha sido executada com sucesso, ela deve ser abortada, caso outra operação remota da mesma ação tenha abortado. A ação como um todo só pode ser validada se todas operações, remotas ou não, forem executadas com sucesso.

O módulo de validação tem como finalidade controlar a comunicação entre operações remotas de tal modo a obter um consenso de todas elas sobre o aborto ou validação da ação atômica à qual pertencem. O controle desta comunicação é feito com base em um *protocolo de validação*. Os protocolos de validação constituem o tema central desta dissertação.

1.7 Contribuições da Dissertação

Este trabalho constitui-se de um estudo sobre o problema de validação de ações atômicas distribuídas em uma rede de computadores. É tratada a questão da possibilidade de existirem soluções tolerantes a falhas para o problema. É descrito e analisado um conjunto de protocolos de validação de ações atômicas propostos na literatura. Dentre esses protocolos, são considerados: o protocolo de duas fases e suas variações, protocolos com mais de duas fases e protocolos *bloqueantes* e *não-bloqueantes*.

O critério adotado para descrever os protocolos baseia-se nos seguintes itens: 1) funcionamento do Protocolo em execuções livres de falhas; 2) comportamento do protocolo após ocorrência de falhas; 3) demonstração de correção; 4) tempo de execução. 5) número de gravações em log[1].

Apenas uma pequena parte dos textos onde os protocolos foram originalmente apresentados adota a padronização acima. Portanto, como contribuições desta dissertação pode-se citar a demonstração da correção e o estudo do número de mensagens enviadas e do tempo de execução de alguns dos protocolos tratados.

Ainda como contribuição da dissertação, destaca-se a proposta de uma nova variação do protocolo de duas fases. A característica principal do protocolo apresentado é a sua habilidade em transformar-se em um protocolo *não-bloqueante* após a ocorrência de uma falha.

Para melhor compreensão dessa característica, convém ressaltar que os protocolos de validação pertencem a duas categorias: protocolos *bloqueantes* e protocolos *não-bloqueantes*.

[1] Este parâmetro não é utilizado na descrição dos protocolos do Capítulo 5.

Nos protocolos bloqueantes, os processos operacionais não conseguem finalizar a ação enquanto os processos que estão falhos não se recuperarem. Nos protocolos não-bloqueantes, os processos operacionais finalizam de modo independente dos processos falhos. Os protocolos não-bloqueantes são mais tolerantes a falhas, porém o número de fases de envio de mensagem é maior, o que diminui a eficiência dos mesmos.

O protocolo apresentado no Capítulo 6 funciona como um protocolo bloqueante de duas fases durante execuções sem falhas. Tal fato permite ao protocolo eficiência comparável aos protocolos de duas fases tradicionais em execuções normais.

Ao ser detectada uma falha, o protocolo passa a funcionar como um protocolo não-bloqueante utilizando-se de três fases de envio de mensagens, aumentando a tolerância a falhas em detrimento da eficiência. Devido a esse comportamento versátil, o protocolo proposto foi denominado protocolo semibloqueante.

Capítulo 2

O Problema de Validação de Ações Atômicas

A validação de ações atômicas cujas operações estão distribuídas em uma rede é um problema complexo. Não basta que o programa criador da ação envie ordens de validação às operações envolvidas. Alguma operação participante pode não conseguir executar as tarefas que lhe foram atribuídas. Nesse caso, todas operações devem abortar as tarefas realizadas, mesmo que as demais tenham executado corretamente as suas tarefas.

O problema agrava-se quando é considerada a possibilidade de ocorrência de falhas durante o procedimento de validação. As operações e os nós onde elas executam e os canais de comunicação podem falhar de modo independente, o que dificulta a solução do problema. É necessário ter mecanismos de detecção de falhas, mecanismos que permitam às operações isentas de falhas tomarem uma decisão quanto ao aborto ou validação da ação e mecanismos para garantir que todas as operações finalizem a ação segundo a mesma decisão, após as falhas terem sido resolvidas.

Um modelo genérico para estudo de validação de ações atômicas é apresentado na próxima seção. A Seção 2.2 apresenta o problema de validação de ações atômicas de uma forma geral. Um modelo formal para descrição de protocolos de validação, bem como questões relacionadas com tolerância a falhas e eficiência desses protocolos, são fornecidos na Seção 2.3.

2.1 Um Modelo Genérico de Sistema

Nesta seção, é apresentado um modelo genérico do sistema subjacente onde as ações atômicas são executadas. Este modelo serve aos seguintes propósitos:

1. fornecer subsídios para a apresentação do problema de validação;
2. servir como padrão para descrição dos protocolos descritos nos próximos capítulos.

Será utilizado um sistema distribuído formado por uma rede de nós onde cada nó possui um processador e dois tipos de memória: memória estável e memória volátil. Visando simplificar a descrição dos protocolos estudados neste texto, será considerado que cada operação é implementada por um processo. Os processos de uma mesma ação podem estar em um mesmo nó, ou distribuídos em vários nós da rede. A comunicação entre os processos se faz exclusivamente por trocas de mensagens.

A entrega de uma mensagem ao processo ao qual ela se destina é garantida pela rede[1]. Uma mensagem pode sofrer um atraso máximo entre dois processos quaisquer da rede. Um processo aguarda a chegada de uma mensagem durante o tempo máximo, o qual é denominado *tempo de espera*[2]. Se uma mensagem não chega durante o tempo de espera o processo destino tem a garantia de que não receberá mais aquela mensagem.

Em cada nó existe um sistema gerenciador de ações atômicas que possui um módulo de controle de concorrência e um módulo gerenciador de dados. O controle da validação de uma ação é feito pelos próprios processos que a constituem.

Quando um nó se recupera de uma falha, o módulo gerenciador de dados recupera o último estado validado daquele nó. Em seguida, o sistema gerenciador de ações atômicas instala novamente os processos que executavam no nó antes da falha.

O modelo de sistema apresentado acima é o modelo-base para os protocolos discutidos nos próximos capítulos. Alguns protocolos exigem alguns requisitos que não fazem parte do modelo apresentado nesta seção. Nesse caso, extensões ao modelo são acrescentadas apenas na descrição do protocolo que faz uso de um requisito específico.

Convém ressaltar que as características do sistema que são descritas nesta seção não são sempre as mesmas consideradas pelos autores dos protocolos estudados neste trabalho. Por exemplo, alguns autores não consideram cada operação de uma ação atômica correspondendo a um processo. Entretanto, a padronização adotada acima não afeta a lógica de funcionamento dos protocolos descritos.

2.2 O Problema da Validação

De acordo com a descrição do sistema dada na seção anterior, uma ação é executada por vários processos. Cada processo pode executar com sucesso, pode falhar, ou pode ser forçado a abortar pelo módulo de controle de concorrência do nó onde ele executa, conforme visto na Seção 1.4. No primeiro caso, diz-se que o *processo concorda em validar a ação.* Nos dois últimos casos, diz-se que o processo *não concorda em validar a ação*, ou *concorda com o aborto da ação.* Um processo que concorda em validar a ação vota SIM pela validação da ação; caso contrário, o seu voto é NÃO. O problema de validação de ações atômicas consiste em obter-se uma decisão quanto ao aborto ou validação da ação com base nos votos dos

[1] A entrega de mensagens não é garantida apenas em caso de falhas de partição da rede.
[2] Do inglês: *"timeout"*.

processos que dela participam. Uma decisão deve satisfazer as seguintes condições:

c1. todos os processos que decidem tomam a mesma decisão;

c2. um processo não pode mudar sua decisão após ter decidido;

c3. a decisão de validar a ação só ocorre se todos os processos possuem voto SIM;

c4. se não ocorrem falhas e todos os processos possuem voto SIM, então a decisão é a de validar a ação;

c5. se todas as falhas existentes forem reparadas e não ocorrerem novas falhas por um longo período de tempo, todos os processos tomam a mesma decisão.

A condição (c1) garante o término consistente da ação em todos os processos que decidem. Não é exigido que todos os processos tomem uma decisão, devido a casos em que um nó pode falhar e não se recuperar jamais.

A condição (c2) declara que a decisão de um processo é irrevogável.

A condição (c3) indica que a única possibilidade para que um processo valide a ação é que todos os processos concordem com essa validação.

A condição (c4) garante que a menos que ocorram falhas, se todos os valores iniciais são SIM, a ação deve ser validada. A condição evita a solução trivial mas pouco útil de decidir sempre abortar. A condição (c4) não necessita ser satisfeita quando ocorrem falhas, isto é, pode ocorrer o caso em que todos os processos possuem voto SIM, mas devido à ocorrência de falhas, a decisão seja a de abortar a ação.

A condição (c5) requer que um processo ao se recuperar de uma falha termine a ação, ou seja, soluções em que os processos ficam bloqueados mesmo após todas as falhas serem sanadas, não são aceitas.

2.3 Protocolos de Validação

Os protocolos utilizados para resolver o problema de validação especificado na Seção 2.2 são denominados *protocolos de validação.*

O funcionamento de um protocolo de validação em cada processo envolvido em uma ação pode ser modelado por um autômato finito. A rede de comunicação entre os processos serve como fita de entrada e saída. Uma transição do autômato envolve: a leitura (recebimento) de um conjunto não vazio de mensagens, algum processamento local, o envio de um conjunto (possivelmente vazio) de mensagens e uma mudança de estado.

Existem algumas características comuns a todos os protocolos de validação, que são identificadas a seguir:

a. o conjunto de estados finais está dividido em dois conjuntos: *conjunto de estado de aborto A* e *conjunto de estados de validação V*;

b. não há transição de estados do conjunto A para estados não pertencentes a A. De modo análogo, não há transição de estados pertencentes a V para estados não pertencentes a esse conjunto. Essa característica é que permite aos protocolos de validação satisfazerem a condição (c2);

c. o diagrama de estado do autômato é acíclico, o que significa que o número de mensagens enviadas é finito.

Os estados de um protocolo de validação são classificados quanto à sua capacidade de definir a finalização do protocolo em:

a. *Estados validáveis.*

b. *Estados não-validáveis;*

Um estado é dito *validável* se quando a ocupação desse estado por um processo significa que todos os processos concordam em validar a ação. Os estados que não possuem essa característica são denominados *estados não-validáveis.*

A seguir, é apresentado como exemplo de protocolo de validação, o protocolo de duas fases.

2.3.1 Protocolo de Validação de Duas Fases

No protocolo de validação de duas fases há um processo que centraliza o controle do protocolo. Esse processo é denominado *coordenador.* Os demais processos que participam do protocolo são denominados *subordinados.* No início do protocolo todos os processos se encontram no estado e_0, que é o estado inicial. A primeira fase corresponde à obtenção, pelo coordenador, de uma decisão com base nos votos dos subordinados. O coordenador envia mensagens PREPARE a todos os subordinados e entra em estado de espera e_1.

Cada subordinado, ao receber uma mensagem PREPARE, pode:

a. enviar uma mensagem com o voto NÃO ao coordenador e entrar em estado de aborto a; ou

b. enviar o voto SIM ao coordenador e mudar para o estado PREPARADO p.

Se o coordenador recebe pelo menos um voto NÃO, ele decide abortar a ação e entra em estado de aborto a. Se todos os votos recebidos forem SIM, o coordenador decide validar a ação e entra em estado de validação v.

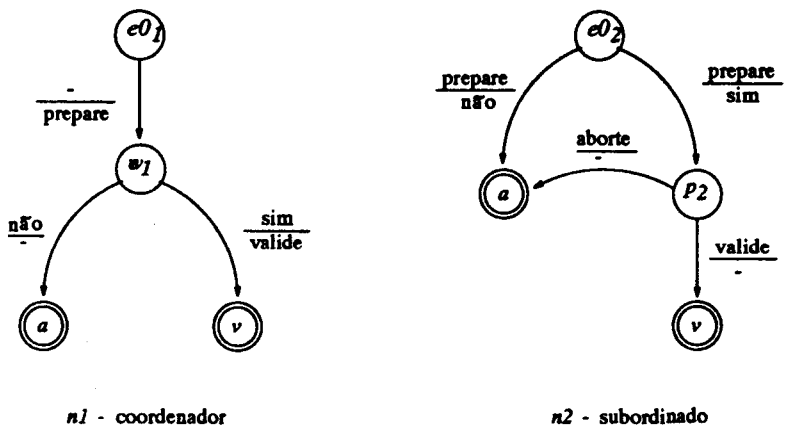

Figura 2.1: Autômatos finitos correspondentes ao protocolo de duas fases executado por apenas dois processos $n1$ e $n2$.

Na segunda fase do protocolo, o coordenador envia a decisão aos subordinados. Se a decisão for a de validar a ação, ele envia mensagens VALIDE a todos os subordinados. Se a decisão for a de abortar a ação, o coordenador envia a decisão ABORTE a todos os processos cujos votos foram *sim*[3].

Cada subordinado em estado PREPARADO muda para o estado final v ao receber uma mensagem com a decisão VALIDE, ou muda para o estado final a ao receber uma mensagem com a decisão ABORTE. A Figura 2.1 mostra os autômatos finitos correspondentes ao coordenador e a cada subordinado. O conjunto de aborto A do protocolo de duas fases é formado pelo estado a e o conjunto V, pelo estado v.

O protocolo de duas fases descrito acima é classificado como *protocolo centralizado*. A comunicação se faz somente entre coordenador - subordinado. Dois processos subordinados não comunicam entre si. A existência de dois autômatos finitos diferentes, um para descrever o funcionamento do coordenador e outro para descrever o funcionamento dos subordinados é uma características dos protocolos centralizados.

Há protocolos que são distribuídos, ou seja, o controle do protocolo não está centralizado em nenhum processo, mas está igualmente distribuído em todos os processos. Os protocolos distribuídos são caracterizados por possuírem um único autômato que é executado por todos os processos.

A próxima seção descreve um modelo genérico de funcionamento dos protocolos de validação quando não ocorrem falhas no sistema.

[3]O coordenador não envia mensagem ABORTE aos processos que votam NÃO pois eles abortam a ação antes de emitirem seus votos.

2.3.2 Modelo de Funcionamento Sem Falhas dos Protocolos de Validação

A seguir, é considerado o funcionamento dos protocolos de validação de um modo geral.

Uma *configuração* ou *estado global* de um protocolo de validação é constituído por dois conjuntos:

1. o conjunto formado pelos estados locais ocupados por cada processo;
2. o conjunto das mensagens em trânsito na rede, isto é, mensagens emitidas e que ainda não foram recebidas.

A *transição de uma configuração* se dá quando ocorre uma transição de um estado local em algum processo. Exceto em caso de falha, essa é a única situação em que uma transição de configuração ocorre.

Uma configuração é dita *configuração final* se todos os seus estados são estados finais.

A *configuração inicial* é aquela em que todos os processos estão em estado inicial e ainda não receberam nenhuma mensagem.

Uma configuração *inconsistente* é aquela que possui um estado de aborto e um estado de validação. Um protocolo que possui uma configuração inconsistente não é um protocolo de validação, pois falha em satisfazer a condição (c1).

Pode-se obter o conjunto de configurações possíveis de um determinado protocolo, aplicando sucessivas transições de configuração a partir da configuração inicial. A seqüência em que este conjunto é obtido corresponde ao *grafo de configuração do protocolo*. A Figura 2.2 mostra o grafo de configuração do protocolo de duas fases para uma execução envolvendo dois processos: $s1$ e $s2$. Cada elipse da figura corresponde a uma configuração e portanto, a um vértice do grafo. As elipses 1 e 2 correspondem a configurações finais do protocolo.

Uma *execução* E de um protocolo de validação é uma seqüência de configurações C_0, C_1, ..., C_n, onde a configuração C_0 é a configuração inicial e cada configuração C_i, $i > 0$ é resultado de uma transição da configuração C_{i-1}.

Uma *configuração C é alcançável*, se existe uma execução E em que C é a última configuração de E e C_0 é a configuração inicial.

Dadas as definições acima, apresenta-se a seguir duas propriedades comuns a todos os protocolos de validação:

Propriedade 2.3.1 *Seja C uma configuração alcançável em uma execução E de um protocolo de validação P em que o estado $q_i \in C$ é um estado de validação correspondente ao processo i. Para cada processo j envolvido na ação, j possui o voto* SIM *e existe uma seqüência de uma ou mais mensagens, com origem no processo j e término no processo i.*

Essa propriedade é conseqüência da definição do problema de validação, pois de acordo com a condição (c3) um processo valida a ação somente se todos os demais processos tiverem

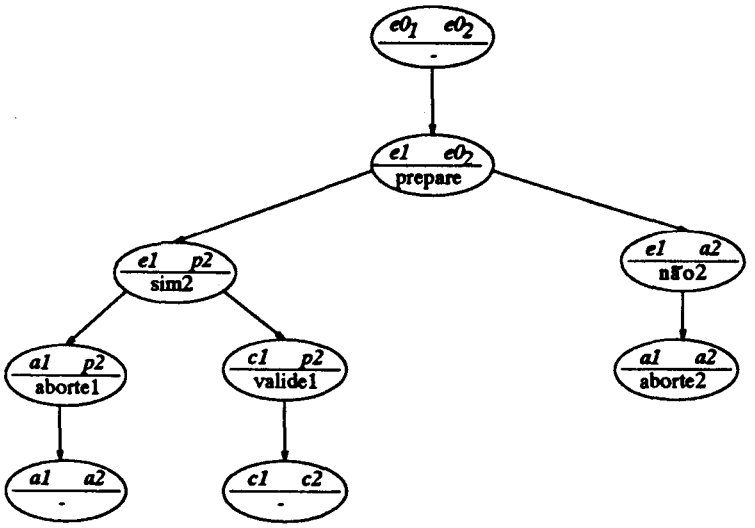

Figura 2.2: Grafo de configuração correspondente ao protocolo de duas fases executado em dois processos.

voto SIM. Obviamente, um processo i toma conhecimento do voto do processo j por meio de mensagens. Essa informação pode ser passada por j diretamente a i (seqüência unitária) ou j pode enviar mensagens contendo o seu voto a outros processos e estes por sua vez transmitem essa informação a i por meio de outras mensagens.

Propriedade 2.3.2 *Seja C uma configuração alcançável em uma execução E de um protocolo de validação P, em que o estado $q_i \in C$ é um estado de aborto correspondente ao processo i. Então, existe um nó j tal que o voto de j é* NÃO *e a menos que $i = j$, existe uma seqüência de mensagens tal que o emissor da primeira mensagem é o processo j e o receptor da última mensagem da seqüência é o processo i.*

Deve ser observado que a Propriedade 2.3.2 é valida somente em execuções normais do protocolo. Se ocorrer alguma falha que impeça um processo de obter o conhecimento dos votos dos demais processos, esse processo pode entrar em estado de aborto mesmo que todos os votos sejam SIM.

De acordo com a Propriedade 2.3.1, o conhecimento completo dos votos de cada processo é uma condição necessária mas não suficiente[4] para que um processo entre em estado de validação. Essa condição não é necessária e nem suficiente para um processo entrar em estado de aborto. Essa simetria se deve ao fato de que as condições para um processo entrar em estado de aborto devem ser mutuamente exclusiva às condições para se entrar em estado validado.

[4] A suficiência implica no conhecimento de que todos os votos são iguais a SIM.

2.3.3 Ocorrência de Falhas em Protocolos de Validação

Até o momento, foram consideradas apenas execuções sem falhas dos protocolos de validação. É necessário adaptar o modelo de autômatos finitos para que ele possa expressar o comportamento de cada processo após sofrer uma falha. A falha de um processo, quando ele se encontra em um determinado estado, pode ser modelada através de uma *transição de falha*. De modo diferente ao que ocorre com as transições normais, a transição de falha não envolve o recebimento de um conjunto de mensagens e nenhuma mensagem é enviada nessa transição. Estão sendo consideradas apenas falhas em que o processo pára de funcionar. Portanto, o estado que o processo ocupa após a transição é o mesmo em que ele se encontra ao recuperar-se da falha.

É necessário que o modelo de descrição de protocolos possua mecanismos para representar a detecção de falhas por processos operacionais. Considerando-se que um processo detecta a falha de outro por tempo de espera (ver Seção 2.1), pode-se representar esse evento também por meio de uma transição de estados, a qual é denominada *transição de detecção de falha*. A transição de detecção de falha é disparada pelo mecanismo de detecção de término do tempo de espera quando uma mensagem não é recebida durante o intervalo de tempo esperado.

Quando ocorre uma falha durante a execução de um protocolo de validação, o problema da validação é agravado e pode ser subdividido em dois problemas que consistem em garantir:

a. que os processos operacionais finalizem a ação satisfazendo as condições (c1) a (c5) de modo independente dos processos falhos ou dos processos operacionais com os quais não podem se comunicar (caso a falha seja de comunicação);

b. que os processos falhos ao se recuperarem finalizem a ação de modo consistente com os demais processos que permaneceram operacionais (condição (c5)).

O problema citado no item (a) é denominado *problema de finalização* de ações atômicas. Os protocolos destinados a solucioná-lo são denominados *protocolos de finalização*. Os processos operacionais interrompem a execução do protocolos de validação ao detectarem a falha e iniciam a execução de um protocolo de finalização.

O problema do item (b) consiste no *problema de recuperação de processos falhos*.

As próximas duas seções tratam dos dois problemas citados acima e das condições necessárias para que hajam protocolos que possam solucioná-los.

2.3.4 Finalização de Processos Operacionais

Conforme visto na Seção 1.7, os protocolos de validação podem ser classificados em protocolos *bloqueantes* e *não-bloqueantes* de acordo com sua capacidade de permitir que os processos operacionais finalizem a ação de modo independente da recuperação das falhas.

O bloqueio dos processos operacionais não causa finalizações inconsistentes, mas é indesejável, pois os processos operacionais bloqueados devem reter as trancas dos dados por elas utilizadas enquanto não obtiverem uma decisão quanto ao fim da ação.

O protocolo de duas fases apresentado na Seção 2.3.1 é um exemplo de protocolo bloqueante: se todos os processos subordinados votam SIM, mas o coordenador falha antes de emitir sua decisão, os subordinados ficam bloqueados. Não há como os subordinados inferirem a decisão do coordenador, mesmo se eles interagirem entre si. Ou seja, mesmo que cada subordinado tenha a informação de que ele e os demais têm voto SIM, ele não pode validar a ação. Isto porque, o coordenador pode não ter recebido o voto de um dos subordinados e ter decidido abortar a ação antes de falhar.

A existência de protocolos de validação não-bloqueantes depende dos possíveis tipos de falhas (falhas de processos e falhas dos canais de comunicação) e das características do sistema distribuído considerado. Inicialmente não serão considerados esses problemas dependentes do sistema e se concentrará em definir as necessidades de informação que cada processo operacional deve satisfazer para terminar a ação de modo consistente.

Um conceito importante à caracterização de protocolos não-bloqueantes é o de *estados concorrentes*. Dois estados são ditos *concorrentes* se eles pertencem a uma mesma configuração alcançável em alguma execução do protocolo. Ou seja, se eles pertencem à mesma configuração do grafo de configurações correspondente ao protocolo. O *conjunto concorrente* de um estado q é o conjunto formado por todos os estados concorrentes a q. No exemplo da Figura 2.2, o conjunto concorrente de $e1$ é $\{e0_2, a2, p2\}$.

Um protocolo pode ser identificado como não-bloqueante com base nos conjuntos concorrentes de cada processo. Considere o caso em que apenas um processo permanece operacional. Seja X esse processo e seja p um estado não final em que ele se encontra ao detectar uma falha. X pode abortar seguramente a ação somente se p não possuir um estado de validação em seu conjunto concorrente. De modo análogo, X pode validar seguramente a ação somente se o estado p não possuir um estado de aborto em seu conjunto concorrente. X fica impossibilitado de finalizar a ação se p é um estado não-validável e o conjunto concorrente de p possui um estado de validação. Nesse caso, X não pode validar, pois ele não sabe se os demais processos possuem o voto SIM, e não pode abortar, porque algum processo pode estar no estado de validação concorrente a p.

Essas restrições servem de base para o Teorema Fundamental dos protocolos não-bloqueantes proposto por Skeen, enunciado a seguir. A demonstração deste Teorema pode ser encontrada em [Ske81].

Teorema 2.3.1 *Um protocolo é não-bloqueante se e somente se ele satisfaz ambas condições seguintes em cada processo participante:*

1. *não há nenhum estado cujo conjunto concorrente possua um estado de validação e um estado de aborto;*

2. *não há nenhum estado não-validável cujo conjunto concorrente contenha um estado de validação.*

O caso em que apenas um processo permanece operacional, mostrado acima, demonstra a necessidade das condições do teorema. Estas condições entretanto, não são suficientes para que o bloqueio seja evitado em qualquer situação de falha. A seguir, são discutidas as situações em que é possível projetar um protocolo não-bloqueante.

Proposição 2.3.1 *Existe um protocolo não-bloqueante se o sistema distribuído dispõe de um mecanismo de broadcast atômico.*

O *broadcast atômico* é um tipo de transmissão em que uma mensagem é enviada e, após um intervalo de tempo limitado, garante-se que todos os processos do sistema recebem a mesma mensagem, ou nenhum processo a recebe.

Prova: Seja o seguinte protocolo: *cada processo envia, pelo mecanismo de broadcast atômico, o seu voto a todos os demais processos envolvidos em uma ação. Um processo muda para o estado de aborto se o seu voto ou um dos votos recebidos for* NÃO *e muda para o estado de validação ao receber todos os votos iguais a* SIM.

É fácil verificar que o protocolo está correto quando não ocorre nenhuma falha no sistema. É descrito, a seguir, um protocolo de finalização correspondente: *se um processo não recebe um dos votos ele entra em estado de aborto.*

O broadcast atômico garante que uma mensagem enviada é recebida por todos processos ou que nenhum processo recebe a mensagem. Portanto, em cada processo operacional o conhecimento dos votos dos demais processos envolvidos na ação é o mesmo. Por esse motivo, todos os processos tomam a mesma decisão especificada pelo protocolo de finalização. ■

Pode ser observado que a existência de broadcast atômico implica que uma transição de uma configuração C_i para uma configuração C_j durante uma execução E do protocolo corresponde a uma transição de estado local de cada processo envolvido na ação.

Na prática, contudo, é muito custoso e difícil implementar mecanismo de broadcast atômico, pois há sempre a possibilidade de apenas uma parte dos processos receberem uma mensagem que deveria ser entregue a todos. Por outro lado, a não disponibilidade desse mecanismo implica que falhas de comunicação podem ocorrer durante a execução de um protocolo de validação, levando a falha de partição da rede.

Skeen [SS83] demonstrou que não existe protocolo não-bloqueante quando ocorre falha de partição múltipla na rede. Demonstrou, ainda, que somente no caso otimista de falha de partição simples, em que as mensagens são devolvidas aos emissores é possível obter-se um protocolo não-bloqueante. Entretanto, apesar de não apropriado, a maioria dos autores utiliza o termo não-bloqueante para se referir a protocolos que obedecem as duas condições do Teorema 2.3.1. O protocolos que não satisfazem aquelas condições são classificados como protocolos bloqueantes. Neste texto também será utilizada essa classificação.

É possível, contudo, obter protocolos não-bloqueantes quando ocorrem apenas falhas de processos que não impliquem em falha de partição da rede [SS83]. A suficiência das condições do teorema fundamental para se obter protocolos não-bloqueantes nessas circunstâncias, pode ser demonstrada através de um protocolo de finalização que permita aos processos operacionais finalizarem consistentemente a ação. Na Seção 5.1 é apresentado um protocolo com essa finalidade.

2.3.5 Recuperação de Processos Falhos

Quando um processo se recupera de uma falha, ele deve realizar uma transição, mudando do estado em que se encontra imediatamente após a recuperação, para um estado final que seja consistente com a decisão tomada pelos processos operacionais. Há duas estratégias possíveis para recuperação de processos falhos:

1. *Recuperação Independente*;
2. *Recuperação Cooperativa*;

Na *recuperação independente*, o processo muda diretamente para um estado final, sem se comunicar com os demais processos. Somente as informações locais disponíveis são utilizadas na recuperação. Na *recuperação cooperativa*, como o próprio nome indica, um processo que se recupera necessita de informações dos processos operacionais para finalizar a ação corretamente.

A recuperação independente é interessante porque o protocolo executado por um processo ao se recuperar é simples e eficiente, pois não há necessidade de troca de mensagens entre os processos. A recuperação cooperativa é menos eficiente: envolve comunicação entre os processos e exige que processos operacionais mantenham informações sobre ações já finalizadas enquanto houver processos falhos no sistema. Além disso, a recuperação cooperativa pode causar bloqueio do processo em recuperação, caso este, por algum motivo, não consiga obter as informações que necessita para finalizar a ação[5]. Obviamente, a recuperação independente é não-bloqueante. Infelizmente, não é possível desenvolver protocolos que utilizem apenas recuperação independente. A seguir, são discutidos os casos de possibilidade desse tipo de recuperação.

Restrição 2.3.1 *A recuperação independente pode ser utilizada em caso de falhas de um único processo em protocolos de validação em que para qualquer estado e_1 do protocolo, o conjunto concorrente de e_1 não contenha simultaneamente um estado de aborto e um estado de validação.*

Essa restrição pode ser explicada através da definição de conjunto concorrente. Seja um estado local e_1, cujo conjunto concorrente contenha um estado de aborto e um estado de

[5]Os demais processos podem estar falhos ou podem estar em partições diferentes.

validação. Um processo que falha no estado e_1 não pode realizar a recuperação independente, pois, ao se recuperar, se ele mudar para o estado final de validação, um outro processo pode estar no estado de aborto e portanto o término da ação fica inconsistente. O mesmo acontece caso o processo falho mude de e_1 para o estado de aborto e o outro processo esteja no estado de validação. Pode-se observar que os protocolos não-bloqueantes obedecem à Restrição 2.3.1.

Se um protocolo obedece à restrição acima, então a recuperação independente é implementada com base na seguinte regra:

Regra 2.3.1 *Para todo estado local e não final do protocolo, se o conjunto concorrente de e contém um estado de validação, então a recuperação independente é feita aplicando-se uma transição de e ao estado final de validação. Caso contrário, aplica-se uma transição de e ao estado final de aborto.*

Deve ser notado que a Regra 2.3.1 se aplica quando apenas um processo falha. Se for considerado que mais de um processo pode falhar ao mesmo tempo, então é impossível o emprego de recuperação independente em cada um dos processos falhos sem gerar términos locais inconsistentes da ação atômica. Essa observação nos leva à seguinte afirmação quanto ao uso da recuperação independente:

Proposição 2.3.2 *Não existe nenhum protocolo de validação para o qual a recuperação independente possa ser utilizada quando dois ou mais processos falham arbitrariamente.*

As falhas que não são toleradas são duas ou mais falhas de processos quase simultâneas, que ocorrem próximas umas das outras o suficiente para que nenhum dos processos em recuperação consiga detectar as falhas dos outros, antes de falhar.

A explicação para a Proposição 2.3.2 é a seguinte: suponha que **P** seja um protocolo que sempre preserva a consistência em ausência de falhas. Sejam dois processos i e j. Pode ser mostrado que para toda execução de **P** sem falha, há um ponto nessa execução em que a ocorrência de falha em i seguida de uma outra falha em j, leva a uma inconsistência.

Uma execução sem falhas E corresponde a um caminho $C_0, ..., C_m$ no grafo de configurações, onde C_0 é a configuração inicial e C_m é a configuração final. Seja C_m uma configuração em que todos os estados são estados de validação. Convém lembrar que uma configuração consiste de um conjunto de estados locais e do conjunto de mensagens em trânsito na rede. Entretanto, serão levados em consideração apenas o conjunto de estados locais. Deve-se lembrar, também, que a transição de uma configuração para outra é causada por uma única transição entre dois estados locais de um dos processos.

Pode haver dois tipos de transição de configuração, conforme o tipo de mecanismo de transmissão utilizado seja ponto-a-ponto ou broadcast (não atômico)[6]. No primeiro caso,

[6] Um mecanismo de broadcast não atômico envia uma mensagem a todos os processos, mas não garante a entrega desta mensagem a todos os processos.

uma transição de configuração corresponde a uma única transição de estado local de algum processo. No segundo caso, numa transição livre de falhas de comunicação, todos os processos realizam uma transição local mudando de estado. Como esse mecanismo não é atômico, durante uma transmissão em que ocorrem falhas de comunicação, parte dos processos recebe as mensagens e realiza a transição normal esperada e a outra parte dos processos detecta a ocorrência de falhas de comunicação e realiza uma transição de detecção de falhas.

Seja C_k uma configuração da execução E, constituída pelos estado locais: $< e_{k1}, e_{k2}, ..., e_{kN} >$, onde e_{ki} é o estado local do processo i e N é o número de processos envolvidos. Seja $f(e)$ o estado final após a recuperação independente de um processo que estava no estado e quando uma falha ocorreu. Seja F_k a configuração resultante após a ocorrência de falhas nos processos i e j, quando o protocolo está na configuração C_k. F_k é portanto semelhante a C_k, exceto que os estados e_{ki} e e_{kj} são substituídos pelos estados $f(e_{ki})$ e $f(e_{kj})$.

Seja a seqüência: $F_0, ..., F_m$ e os estados locais de i e j nessa seqüência. Pela aplicação da Regra 2.3.1, o par $(f(e_{0i}), f(e_{0j}))$ deve ser igual ao par (a_i, a_j), onde a_t é o estado de aborto do processo $t, i \leq t \leq j$. Também pela aplicação da Regra 2.3.1, o par $(f(e_{mi}), f(e_{mj}))$ deve ser igual ao par (c_i, c_j) na configuração final C_m da ação, uma vez que está sendo considerado que a ação é validada. Seja k o menor inteiro tal que se o mecanismo de transmissão utilizado pelo protocolo for ponto-a-ponto, $f(e_{ki}) = c_i$ ou $f(e_{kj}) = c_j$ e se o mecanismo de broadcast for utilizado, $f(e_{kj}) = c_j$ e $f(e_{ki}) = c_i$.

No caso ponto-a-ponto ocorre a seguinte situação:

Seqüência de validação $C_0, ..., C_m$	Configurações($F_0, ..., F_m$) resultantes da ocorrência de falhas em i e j
$C_0 < ..., e_{0i}, ..., e_{0j}, ... >$	$F_0 < ..., a_i, ..., a_j, ... >$
$\vdots$	$\vdots$
$C_{k-1} < ..., e_{k-1,i}, ..., e_{k-1,j}, ... >$	$F_{k-1} < ..., a_i, ..., a_j, ... >$
$C_k < ..., e_{ki}, ..., e_{kj}, ... >$	F_k onde $f(e_{ki})$ ou $f(e_{kj})$, exclusivamente é um estado de validação.
$\vdots$	$\vdots$
$C_m < ..., e_{mi}, ..., e_{mj}, ... >$	$F_m < ..., c_i, ..., c_j, ... >$

Caso o protocolo faça uso de broadcast, uma falha de comunicação pode ocorrer durante a transição $C_{k-1} \rightarrow C_k$, de tal modo que o processo i e o processo j fiquem em partições diferentes, ou seja, apenas um dos dois recebe a mensagem enviada durante aquela transição. Essa falha de comunicação pode ocorrer próximo o suficiente das falhas dos dois processos, de tal modo que um deles não perceba a falha do outro e nem a falha de comunicação. Essa situação é semelhante àquela que ocorre quando o mecanismo ponto-a-ponto é utilizado. A seqüência abaixo ilustra essa situação:

Seqüência de validação $C_0, ..., C_m$	Configurações($F_0, ..., F_m$) resultantes da ocorrência quase simultânea de falhas de comunicação e falhas em i e j
$C_0 < ..., e_{0i}, ..., e_{0j}, ... >$	$F_0 < ..., a_i, ..., a_j, ... >$
$\vdots$	$\vdots$
$C_{k-1} < ..., e_{k-1,i}, ..., e_{k-1,j}, ... >$	$F_{k-1} < ..., a_i, ..., a_j, ... >$
$C_k < ..., e_{ki}, ..., e_{kj}, ... >$	F_k onde $f(e_{ki})$ ou $f(e_{kj})$, exclusivamente é um estado de validação.
$\vdots$	$\vdots$
$C_m < ..., e_{mi}, ..., e_{mj}, ... >$	$F_m < ..., c_i, ..., c_j, ... >$

Em ambas as situações, $e_{k-1,i} = e_{ki}$, ou exclusivamente, $e_{k-1,j} = e_{kj}$ e $f(e_{k-1,i}) = f(e_{ki})$ ou $f(e_{k-1}, j) = f(e_{kj})$. Conclui-se então que ($e_{ki} = c_i$ e $e_{kj} = a_j$) ou ($e_{ki} = a_i$ e $e_{kj} = c_j$) e que portanto F_k é inconsistente.

Uma vez que um processo não pode detectar falhas concorrentes em outros processos, ele não pode determinar com segurança quando usar a recuperação independente pois corre o risco de gerar inconsistência. Em outras palavras, a Proposição 2.3.2 nos permite chegar à seguinte conclusão:

> *Nenhum protocolo de validação pode garantir recuperação independente de processos falhos que tentem se reintegrar no sistema.*

Ou seja, sempre quando um processo falhar, um tipo de recuperação *dependente* deve ser utilizada e com isso, há necessidade de armazenar informações sobre ações já finalizadas, nos processos operacionais e corre-se o risco de haver bloqueio do processo em recuperação.

2.3.6 Medidas de Eficiência dos Protocolos de Validação

Nas seções anteriores, foram consideradas características de tolerância a falhas dos protocolos de validação. Outro aspecto importante na avaliação desses protocolos é a sua eficiência. Tolerância a falhas e eficiência são objetivos conflitantes entre si em sistemas distribuídos de modo geral. O problema de validação de ações atômicas distribuídas não constitui exceção. A maioria dos autores baseia-se nos três seguintes parâmetros para medida de eficiência dos protocolos de validação:

1. tempo de execução;

2. número de mensagens;

3. número de gravações no log.

Estes parâmetros são avaliados durante uma execução sem falhas do protocolo, correspondente a uma ação validada, pois essa é a situação que se espera que ocorra com mais freqüência no sistema. Cada um destes parâmetros é apresentado a seguir.

Tempo de Execução

O tempo de execução dos protocolos de validação é uma medida que depende das características do sistema distribuído em que ele executa, tais como: taxa de transmissão da rede, tempo de execução dos processos e tipo de comunicação entre os processos (broadcast ou ponto-a-ponto). Torna-se, por esse motivo, difícil avaliar o tempo de execução dos protocolos com base nas características de sistemas reais. Para efeito de comparação entre os protocolos, a maioria dos autores considera modelos abstratos de sistemas distribuídos. Ramarao [Ram89] considera uma rede em que o tempo gasto pelos processos para processarem uma mensagem recebida é considerado nulo, e uma mensagem enviada em um instante t é lida no instante $t + 1$ pelo processo destinatário.

Abaixo é dada a definição de *tempo de execução* utilizada por Ramarao [Ram89]:

Definição 2.3.1 Tempo de execução *corresponde ao intervalo de tempo que compreende o instante de início do protocolo, ao instante em todos os processos validam a ação.*

Utilizando a definição de tempo de execução dada acima, Ramarao demonstrou, formalmente, os limites de tempo de execução para protocolos não-bloqueantes que podem ser obtidos em redes ponto-a-ponto, redes broadcast e rede tipo anel. Os resultados são mostrados na Tabela 2.1. O conceito de tempo de execução dado acima, apesar de utilizado por muitos autores, não considera o tempo que os processos operacionais devem aguardar até que tenham garantia de que todos os processos finalizaram a ação. Essa certificação tem uma implicação prática: se um processo não é informado sobre a finalização dos demais processos, ele é obrigado a reter informações sobre a ação indefinidamente, pois a qualquer instante ele pode receber uma mensagem de um processo em recuperação, perguntando sobre a decisão tomada.

Nos protocolos de duas fases, por exemplo, o coordenador, após enviar a decisão de validar ou de abortar a ação, aguarda uma mensagem especial RECEBI de cada subordinado. Enquanto não receber de todos os subordinados essa mensagem, o coordenador mantém na memória volátil do nó onde reside um registro indicando a decisão tomada.

Uma outra definição de tempo de execução leva em conta também o aspecto de liberação de dados da memória após a validação da ação:

Definição 2.3.2 Tempo de execução *de um protocolo de validação é o intervalo de tempo que compreende desde o tempo de início de execução do protocolo até o instante em que o processo que iniciou a ação pode remover de sua memória as informações sobre a ação.*

Topologia	Nº de Mensagens	Tempo
Ponto-a-ponto	$2n-1$	$2\log_2 n$
Broadcast	n	n
Rede em anel	$2n-1$	$5n/4$

Tabela 2.1: Limites inferiores de tempo e de mensagens para protocolos não-bloqueantes. n corresponde ao número de processos envolvidos na ação.

Será adotada a Definição 2.3.2 no estudo de tempo de execução de todos os protocolos apresentados neste texto.

Número de Mensagens

O número de mensagens enviadas durante a execução do protocolo é um importante parâmetro para avaliar a sua eficiência. De modo semelhante ao que ocorre com a avaliação de tempo, a contagem do número de mensagens é feita durante uma execução sem falhas. O tamanho da mensagem não é levado em consideração como medida de eficiência, pois na maioria dos protocolos as mensagens são consideradas pequenas. Obviamente, se esse não fosse o caso, apenas o número de mensagens enviadas não seria um parâmetro suficiente.

Neste texto, serão considerados o tempo de execução e o número de mensagens em uma rede ponto a ponto em que cada nó pode comunicar-se diretamente com os demais e os custos são os mesmos em qualquer comunicação.

Número de Gravações no Log

Durante a execução de um protocolo de validação, sempre que um processo muda de estado, ele necessita gravar em memória estável informações que lhe permita iniciar um procedimento de recuperação, caso o processo onde executa sofra uma falha durante o período em que esse processo ocupa o novo estado. Essas informações são gravadas no log.

O número de gravações no log é um parâmetro muito utilizado como medida de eficiência em protocolos que são utilizados na prática, como é o caso dos protocolos de duas fases. Isto se deve obviamente ao custo de gravação e recuperação de dados no log, que não pode ser desconsiderado em implementações de protocolos em sistemas reais. Entretanto, a maioria

dos textos que propõe ou analisa protocolos que não são de duas fases não considera esse parâmetro. Esses protocolos são mais de interesse teórico e são geralmente protocolos em que o número de mensagens e o tempo de execução são grandes; portanto, a preocupação maior é em reduzir esses parâmetros. No Capítulo 4, que trata de protocolos que não são de duas fases, o número de gravações no log não será utilizado.

2.3.7 Modelo de Comunicação Entre Processos ou Topologia Lógica de Comunicação

O modo como os processos comunicam-se entre si durante a execução de um protocolo de validação define uma topologia virtual entre eles. Esta topologia não depende da topologia da rede física que os conecta. Por exemplo, a topologia virtual de comunicação entre os processos que executam o protocolo de duas fases descrito na Seção 2.3.1 corresponde a uma estrela. O centro dessa estrela corresponde ao processo coordenador e na periferia encontram-se os processos subordinados. No Capítulo 3 serão apresentados protocolos derivados do protocolo original de duas fases que apresentam topologia virtual linear e topologia virtual em forma de árvore.

Alguns protocolos descentralizados possuem uma topologia virtual que corresponde a um grafo fortemente conexo. O protocolo descentralizado apresentado na Seção 5.3 possui uma topologia que corresponde a um hipercubo de dimensão k, $k = log_2 N$, onde N é o número de processos envolvidos no protocolo.

É interessante notar que a topologia virtual indica apenas a topologia física de rede mais apropriada para o protocolo correspondente. Por exemplo, a rede apropriada para os protocolos de validação de duas fases centralizado e descentralizado é uma rede de broadcast[7]. No protocolo linear, a disponibilidade de broadcast na rede não influência a eficiência do protocolo, uma vez que a comunicação do protocolo é inerentemente ponto-a-ponto.

[7] No protocolo de duas fases centralizado, o recurso de broadcast só é realmente interessante na primeira fase em que a comunicação dá-se no sentido do centro para a periferia da estrela.

Capítulo 3

Protocolos Eficientes de Duas Fases

Os protocolos de duas fases são os protocolos de validação mais utilizados em sistemas de ações atômicas distribuídas. São, também, os mais comumente encontrados na literatura. A popularidade desses protocolos deve-se principalmente aos seguintes fatores:

- eficiência quanto ao número de mensagens trocadas entre os processos que participam da ação;
- simplicidade, quando comparados com outros tipos de protocolos.

Este capítulo trata de protocolos de duas fases que visam eficiência quanto ao tempo de execução, quanto ao número de mensagens enviadas ou quanto ao número de registros gravados no log.

Nas próximas três seções são descritas as características comuns à maioria dos protocolos de duas fases. Na Seção 3.5 é apresentada a padronização a ser adotada na discussão sobre cada protocolo.

3.1 Funcionamento Genérico dos Protocolos de Duas Fases

Os processos que executam as operações de uma ação atômica são classificados em *processo coordenador* e *processos subordinados*. O processo coordenador possui o controle da execução do protocolo. É ele o responsável por obter uma decisão quanto à finalização da ação com base nos votos dos demais processos que são os subordinados.

Na maioria dos protocolos estudados neste capítulo, o processo coordenador é o processo

que iniciou a execução da ação[1]. Ele geralmente é criado no mesmo nó onde executa o programa de aplicação que criou a ação.

Um protocolo inicia-se quando o programa de aplicação executa o comando VALIDE-AÇÃO. Um processo ao receber esse comando, torna-se coordenador e inicia a primeira fase do protocolo.

Na primeira fase [Gra78], o coordenador coleta os votos dos subordinados e toma a decisão quanto ao fim da ação. Esta fase inicia-se quando o coordenador ordena aos subordinados que emitam seus votos sobre a validação da ação. Cada subordinado, ao receber a ordem, emite o voto SIM se for favorável em validar a ação, ou o voto NÃO em caso contrário. Ao receber todos os votos dos subordinados, o coordenador decide validar a ação se receber apenas votos SIM. Se receber pelo menos um voto NÃO, o coordenador decide abortar a ação. Ao se decidir, o coordenador finaliza a ação localmente de acordo com a decisão, isto é, aborta ou valida a ação.

A segunda fase corresponde à fase em que o coordenador comunica aos subordinados a decisão obtida. Cada subordinado finaliza a ação de acordo com essa decisão.

3.2 Tipos de Mensagens Utilizadas

Os protocolos que são descritos nas próximas seções utilizam os tipos de mensagens mostrados na Tabela 3.1. Alguns protocolos utilizam, além desses, outros tipos de mensagens que serão tratados especificamente durante as seções que descrevem esses protocolos.

MENSAGENS	
TIPO	VALORES POSSÍVEIS
ordem	PREPARE
voto	SIM, NÃO
decisão	VALIDE, ABORTE
indagação	PERGUNTANDO
sinal	RECEBI

Tabela 3.1: Tipos de mensagens utilizadas pelos protocolos de duas fases.

Mensagens com a ordem PREPARE são enviadas pelo coordenador aos subordinados para que estes emitam os seus votos.

Os votos SIM e NÃO emitidos por um subordinado expressam a intenção em, respectivamente, validar e abortar a ação.

[1] Há exceções. O protocolo O2PC mostrado na Seção 4.1 é um exemplo em que o processo que inicia a execução da ação é diferente do processo coordenador.

As mensagens VALIDE e ABORTE são emitidas para comunicar a todos os processos a decisão obtida durante a primeira fase.

A mensagem PERGUNTANDO é enviada com o objetivo de se conhecer a decisão. Essa mensagem só é utilizada em situações anormais, por exemplo, quando nó onde executava um processo subordinado se recupera de uma falha. Dependendo do estado em que o subordinado estava no momento da falha, esse nó envia a mensagem PERGUNTANDO ao processo coordenador com o objetivo de obter a decisão sobre a finalização da ação.

O último tipo de mensagem, RECEBI, é enviado por um processo subordinado para indicar que ele recebeu a decisão e que já finalizou a ação localmente.

3.3 Estados dos Processos

Durante a execução do protocolo para uma dada ação, cada processo pode estar em um dos tipos de estado relacionados abaixo:

a. estado DESCONHECIDO;

b. estado PREPARADO;

c. estado VALIDADO;

d. estado ABORTADO;

O estado DESCONHECIDO é o estado em que cada processo se encontra no início do protocolo e após a ação se finalizar em todos os processos.

Um subordinado muda do estado DESCONHECIDO para o estado PREPARADO somente se ele recebe a ordem PREPARE e concorda em validar a ação. Entrar em estado PREPARADO significa que o processo concorda em validar a ação, mas ainda não sabe se há concordância dos demais processos em fazer o mesmo. Quando o processo entra em estado PREPARADO ele armazena no log um registro PREPARADO. O registro PREPARADO contém dados para a recuperação em caso de falha. Entre esses dados estão: as trancas obtidas pelo subordinado para executar a ação, o identificador da ação.

A transição do estado DESCONHECIDO para o estado ABORTADO em um subordinado ocorre se esse processo:

a. receber uma decisão ABORTE;

b. receber uma ordem PREPARE, mas não concordar com a validação da ação;

c. não receber nenhuma mensagem PREPARE durante o tempo de espera.

O coordenador muda do estado DESCONHECIDO para o estado ABORTADO se ele:

a. não concordar com a validação da ação; ou

b. receber o voto NÃO de pelo menos um dos subordinados; ou

c. não receber o voto de um dos subordinados durante o tempo de espera.

Ao entrar em estado ABORTADO, um processo grava um *registro* ABORTADO no log. Esse registro contém: o identificador da ação, o identificador do processo, e dados para que o protocolo possa ter continuidade após o nó se recuperar de uma falha.

Um processo subordinado em estado PREPARADO muda para o estado VALIDADO somente quando ele sabe que os demais processos também concordam com a validação da ação. Esta informação é obtida quando o subordinado recebe a decisão VALIDE.

O coordenador muda do estado DESCONHECIDO para o estado VALIDADO se recebe o voto SIM de todos os seus filhos.

Um processo ao entrar em estado VALIDADO grava um *registro* VALIDADO no log. Esse registro contém também informações para a recuperação do processo caso ocorra uma falha.

O estado DESCONHECIDO não possui um registro correspondente. Os dados utilizados para continuidade do protocolo, gravados nos registros descritos acima, dependem de um protocolo em particular e serão tratados durante as seções que descrevem os protocolos discutidos neste capítulo.

A gravação dos registros no log pode ser feita de dois modos: gravação *síncrona* ou *forçada* e gravação *assíncrona* ou *não forçada*. Na gravação síncrona de um registro, todos os registros que estão na memória volátil do nó são imediatamente gravados no log. Quando um registro é gravado de modo síncrono por um subordinado ou coordenador, o protocolo fica impedido de prosseguir nesse processo enquanto a gravação no log não terminar. Isto significa que se o nó falha após a gravação síncrona ocorrer, o registro cuja gravação foi forçada e todos os demais anteriores a ele estão seguramente armazenados no log. Posteriormente, quando o nó se recupera, esses registros estão disponíveis no log e podem ser trazidos de volta à memória volátil.

Na gravação assíncrona ou não forçada, a escrita do registro no log é adiada para quando uma página do buffer do log (na memória volátil) ficar cheia, ou quando ocorrer uma gravação síncrona posterior de outro registro. O protocolo pode prosseguir nos processos responsáveis pela gravação assíncrona mesmo antes de ocorrer a migração dos dados para o log. Se o nó falha após a gravação assíncrona, o registro pode se perder se não tiver ocorrido nenhuma gravação síncrona entre o momento da gravação e o momento da falha.

A gravação síncrona proporciona mais segurança aos dados, porém ela causa um aumento no tempo de resposta de execução da ação quando comparada com a gravação assíncrona. Na gravação síncrona, sempre que um registro é gravado, é necessário fazer uma acesso à memória estável somente para gravá-lo. Na gravação assíncrona, o número de acessos à memória estável é menor, pois vários registros são gravados em cada acesso ao log. Portanto,

uma forma de medir a eficiência de um protocolo de duas fases é verificar o número de gravações assíncronas que ele utiliza.

3.4 Modelo de Comunicação Entre Processos

Cada protocolo de duas fases apresentado neste capítulo e nos capítulos 4 e 6 baseia-se em um dos seguintes modelos de comunicação entre processos: hierárquico, estrela, linear e barramento.

Modelo Hierárquico

O protocolo aproveita a estrutura hierárquica criada entre os processos que participam da execução da ação. O primeiro processo criado para executar a ação é denominado *processo raiz*. Durante a execução da ação cada processo pode criar novos processos no mesmo nó ou em outros nós. Desse modo, a execução de uma ação organiza os processos participantes segundo uma *árvore de processos*.

As referências aos componentes da árvore de processos, são feitas utilizando os termos empregados na descrição de estruturas do tipo árvore, tais como: *processo folha*, *processo filho*, *processo pai*, *processos descendentes* e *processos ancestrais*. *Processos descendentes* de um processo I são todos os processos que constituem a subárvore com raiz no processo I, excluindo o processo I. *Processo ancestral* de um processo I é qualquer processo pertencente ao caminho que liga I ao processo raiz, excluindo o processo I. Um processo que não é raiz nem folha é denominado *processo intermediário*. Um *caminho* na árvore é uma seqüência $P_1, P_2, P_3, \cdots, P_n$ de processos da árvore tal que o processo P_{i+1} é filho do processo P_i.

Considera-se que o sistema de comunicação permite que um processo P_2, ativado remotamente por um operação P_1, retorne a P_1 o identificador de todos os processos onde P_2 ativou operações remotas. Desse modo, as informações sobre os identificadores dos processos fluem em direção ao processo raiz. O processo raiz consegue então obter a estrutura correspondente a árvore de processos que participam da ação. A obtenção dessa estrutura não requer custos adicionais, pois é obtida através das comunicações normais entre as operações.

O programa de aplicação que cria a ação executa no mesmo processo onde reside o processo raiz da árvore de processos que executam a ação. Quando o usuário deseja validar a ação criada, ele executa através do programa de aplicação um comando do tipo VALIDEAÇÃO. Esse comando faz com que o processo raiz torne-se o coordenador do protocolo. Os processos folhas funcionam como SUBORDINADOS. Os processos intermediários funcionam como subordinados em relação a seus pais e como coordenadores em relação a seus processos filhos. Durante a execução do protocolo cada processo comunica-se apenas com o seu processo pai e os seus processos filhos na árvore de processos.

Modelo Estrela

O modelo de comunicação tipo estrela pode ser visto como uma especialização do modelo hierárquico, em que a árvore de processos tem apenas um nível. Não há processos intermediários. Um processo desempenha o papel de coordenador e os demais desempenham apenas a função de subordinados. O protocolo tradicional de duas fases proposto por Gray [Gra78] baseia-se neste modelo de comunicação.

Modelo Linear

O modelo linear de comunicação entre os processos também pode ser visto como uma especialização do modelo hierárquico, onde há apenas um processo folha. O coordenador e cada processo intermediário possui apenas um processo filho. O protocolo de duas fases linear apresentado na Seção 3.11 utiliza esse modelo de comunicação.

Modelo de Barramento

Neste modelo, cada processo se comunica com todos os demais durante a execução do protocolo. Na Seção 4.2 é apresentado um protocolo que utiliza esse modelo de comunicação entre os processos.

3.5 Modelo para Apresentação dos Protocolos de Duas Fases

Com o objetivo de facilitar e padronizar a apresentação de cada protocolo tratado neste capítulo, utiliza-se um modelo de apresentação baseado nos seguintes itens:

1. **Descrição do Protocolo**

   Neste item é feita uma descrição do funcionamento do protocolo em uma execução livre de falhas.

2. **Tratamento de Falhas**

   É feita uma descrição das atividades executadas pelo protocolo em um nó ao se recuperar de uma falha e do comportamento de um nó operacional ao detectar falha no sistema.

3. **Correção do Protocolo**

   Neste item, a correção do protocolo é demonstrada informalmente. Para cada protocolo, é demonstrada apenas a satisfação das condições (c1) e (c5) da Seção 2.2. Ou seja, se todos os processos finalizam, todos a abortam ou todos a validam e após todas

as falhas se recuperarem e não ocorrem mais falhas por certo período de tempo, todos os processos finalizam a ação de modo consistente. Isto é, todos a validam ou todos a abortam. O atendimento às condições (c2), (c3) e (c4) é óbvia em cada protocolo estudado e não será demonstrado.

4. **Tempo de Execução**

   É apresentado, neste item, o tempo de execução do protocolo, segundo a Definição 2.3.2.

5. **Número de Mensagens**

   Neste item, é estudado o número máximo de mensagens enviadas pelos processos envolvidos no protocolo para uma dada ação. Esse número é calculado com base em uma execução sem falha do protocolo, em que a ação é validada.

6. **Número de Gravações no Log**

   O número de gravações de registros no log é um fator importante para avaliação de protocolos que são implementados em algum sistema gerenciador de ações atômicas. Neste item é avaliado o número de gravações forçadas e não-forçadas feitas por cada processo no protocolo em estudo.

Nas seções seguintes são descritos alguns protocolos de duas fases, segundo a padronização apresentada a cima. Os protocolos apresentados neste capítulo visam eficiência em detrimento de tolerância a falhas. Protocolos que visam ser tolerantes a falhas são apresentados no próximo capítulo.

3.6 Protocolo de Duas Fases Hierárquico

O protocolo de duas fases foi proposto por Gray [Gra78]. Trata-se do protocolo de validação mais utilizado na maioria dos sistemas distribuídos.

O protocolo descrito nesta seção é uma extensão do protocolo original proposto por Gray, aplicado a sistemas em que as ações são executadas segundo o modelo de árvore de processos apresentado na Seção 3.4. Este protocolo é apresentado em [ML83].

Descrição do Protocolo

Primeira Fase

O coordenador inicia a primeira fase enviando mensagens PREPARE aos seus processos filhos. O significado da mensagem PREPARE é o de ordenar aos subordinados que emitam os seus votos a respeito da finalização da ação. Após enviar as mensagens PREPARE, o coordenador permanece aguardando os votos de seus filhos.

Ao receber as mensagens PREPARE, os subordinados filhos do coordenador enviam mensagens PREPARE aos seus filhos e permanecem aguardando os votos deles. As mensagens PREPARE propagam-se até os processos folhas. Se um processo folha concorda com a validação da ação, ele força a gravação de um registro PREPARADO no log. Diz-se que o processo entra em estado PREPARADO após a gravação desse registro.

Após entrar em estado PREPARADO, um processo folha envia o voto SIM a seu pai e permanece aguardando a decisão quanto à finalização da ação. Se o processo folha não concorda com a validação da ação, ele força a gravação de um registro ABORTADO no log e envia o voto NÃO a seu pai.

Cada processo intermediário I da árvore de processos obtém o seu voto a partir do momento em que recebe os votos de seus filhos. Se I concorda em validar a ação e recebe todos os votos de seus filhos e eles são SIM, I força a gravação de um registro PREPARADO no log e envia o voto SIM a seu pai. Se I recebe um voto NÃO ou não concorda em validar a ação ou não recebe o voto de um de seus filhos durante o tempo de espera, I decide abortar a ação. Para tanto, ele força a gravação de um registro ABORTADO no log e envia o voto NÃO a seu pai. Os votos propagam-se na árvore de processos no sentido oposto ao das mensagens PREPARE, isto é, no sentido das folhas para raiz.

Quando o coordenador recebe os votos de seus, filhos ele decide sobre a finalização da ação. Se todos os votos forem recebidos e eles forem apenas votos SIM, o coordenador entra em estado VALIDADO, forçando a gravação de um registro VALIDADO no log. O coordenador decide abortar a ação se não recebe o voto de um de seus filhos durante o tempo de espera, ou se um dos votos recebidos for NÃO. Ao decidir abortar a ação, o coordenador força a gravação de um registro ABORTADO, após o que ele entra em estado ABORTADO. Uma vez gravado no log o registro correspondente à decisão tomada, o coordenador inicia a execução da segunda fase do protocolo que corresponde ao envio da decisão a todos os subordinados.

Segunda Fase

Se a decisão tomada na primeira fase for a de validar a ação, o coordenador envia mensagens com a decisão VALIDE a todos os seus processos filhos. Se a decisão for a de abortar a ação, o coordenador envia mensagens com a decisão ABORTE apenas aos processos filhos que votaram SIM na primeira fase[2]. Após enviar mensagens com a decisão, o coordenador permanece aguardando a chegada das mensagens RECEBI de seus filhos.

Cada processo intermediário, ao receber uma mensagem valide, entra em estado VALIDADO através da gravação forçada de um registro VALIDADO no log do computador onde ele executa. A seguir, o processo intermediário envia mensagens com o sinal RECEBI a seu pai e mensagens com a decisão valide a todos os seus filhos e permanece aguardando desses

[2]Os filhos que votaram NÃO abortam a ação por si mesmos. Esta é uma otimização em relação ao protocolo proposto por Gray [Gra78], onde um processo ao votar NÃO adiava o aborto para quando recebesse a decisão ABORTE de seu superior. Assim, a liberação das trancas retidas pelo processo ficava adiada.

últimos as mensagens com o sinal RECEBI. Um processo intermediário em estado PREPARADO ao receber uma mensagem ABORTE entra em estado ABORTADO, forçando a gravação de um registro ABORTADO. Portanto, um processo intermediário pode entrar em estado ABORTADO na primeira fase ao decidir abortar, ou na segunda fase, ao receber a decisão ABORTE de seu pai. Em ambas as situações, após entrar em estado ABORTADO um processo intermediário envia mensagens com a decisão ABORTE a seus filhos cujos votos foram SIM e permanece aguardando deles os sinais RECEBI.

Ao receber uma mensagem com a decisão, um processo folha força a gravação de um registro correspondente no log, envia mensagem com o sinal RECEBI a seu processo pai e "esquece" a ação, apagando de sua memória volátil qualquer informação sobre ela.

Cada processo não-folha (intermediário ou coordenador) grava de modo assíncrono um registro FIM após obter os sinais RECEBI de todos os seus filhos e esquece a ação, apagando de sua memória volátil as informações a ela relacionadas. A gravação do registro FIM significa que o processo não-folha sabe que todos os seus filhos finalizaram a ação de acordo com a decisão e que, portanto, não é mais necessário manter informação sobre a ação.

Tratamento de Falhas

Após um processo recuperar-se de uma falha, ele verifica o estado em que estava no momento em que ocorreu a falha. Caso a falha do processo se deve à falha do nó onde ele executava, o sistema gerenciador de ações atômicas copia as informações referentes a ação do log para a memória volátil. Quando o processo se recupera, ele encontra na memória volátil os dados sobre a ação que ele executava. Se for encontrado apenas um registro PREPARADO para uma dada ação significa que o processo que o gravou era um subordinado e estava em estado PREPARADO aguardando a decisão de seu pai quando ocorreu a falha. O processo envia periodicamente mensagem PERGUNTANDO ao seu processo pai. O identificador do processo pai é obtido do registro PREPARADO. Ao receber uma decisão, o processo em recuperação finaliza a ação conforme explicado na seção anterior.

Se for encontrado um registro ABORTADO ou VALIDADO, o comportamento do processo em recuperação dependerá da sua posição na hierarquia da árvore de processos. Se o processo for um processo folha, ele envia o sinal RECEBI a seu pai. Essa exigência é necessária, pois um processo subordinado envia o sinal RECEBI somente após ter gravado um registro VALIDADO ou ABORTADO no log. O processo em recuperação pode ter falhado antes de enviar esse sinal ao seu pai. Nesse caso, o processo pai não grava o registro FIM enquanto não receber esse sinal, o que o obriga a continuar mantendo informações sobre a ação em sua memória volátil.

Se o processo em recuperação é o coordenador, seu comportamento dependerá da existência de um registro FIM correspondente ao registro ABORTADO ou VALIDADO encontrado. Se for encontrado um registro FIM, não há nada a fazer, o processo em recuperação simplesmente esquece a ação. Se, entretanto, não for encontrado tal registro, é sinal que o coordenador falhou quando aguardava os sinais RECEBI de seus filhos. Nesse caso, o processo

em recuperação envia aos processos filhos do coordenador mensagens com a decisão correspondente ao registro encontrado. Ao obter os sinais RECEBI desses processos, o coordenador em recuperação grava de modo assíncrono um registro de FIM e apaga de sua memória volátil as informações sobre a ação.

Caso o processo em recuperação seja um processo intermediário, ele apresenta os dois comportamentos descritos anteriormente relativos a de um processo folha e do coordenador. Portanto, ao encontrar o registro ABORTADO ou VALIDADO, um processo intermediário envia o sinal RECEBI ao seu pai e se não encontrar um registro FIM correspondente, ele envia mensagens com a decisão aos seus filhos e, neste caso, grava o registro de FIM ao obter deles os sinais RECEBI de seus processos filhos.

Se um processo recebe uma mensagem PERGUNTANDO, ele verifica a existência de algum registro correspondente ao identificador da ação incluído na mensagem recebida. Se for encontrado um registro ABORTADO ou VALIDADO, o processo envia a decisão correspondente (ABORTE ou VALIDE) como resposta e aguarda o sinal RECEBI do processo que enviou a indagação. Se for encontrado um registro PREPARADO, o processo adia a emissão da resposta para quando ele receber uma decisão. Se nenhum registro é encontrado para aquela ação, é sinal de que o processo que recebe a mensagem PERGUNTANDO havia enviado pedido de votos aos seus filhos, mas falhou antes de obtê-los e, portanto, antes de gravar qualquer registro no log. Nesse caso, o processo era um processo intermediário ou o coordenador da ação, porém todas as informações se perderam no momento da falha. Como conseqüência, o processo decide abortar a ação. Ele força a gravação de um registro ABORTADO e envia a decisão ABORTE em resposta.

Há duas situações em que o coordenador ou um intermediário podem detectar término do tempo de espera por mensagens dos processos filhos:

1. quando ele aguarda os votos dos processos filhos;
2. quando ele aguarda os sinais RECEBI dos processos filhos.

Na situação (1), o processo que aguarda a mensagem aborta a ação. Ele força a gravação de um registro ABORTADO no log e envia mensagens com a decisão ABORTE aos seus filhos cujos votos foram SIM. Se o processo for um intermediário, ele envia ainda o voto NÃO a seu pai.

Na situação (2), o processo deve permanecer aguardando os sinais RECEBI de seus filhos para que possa gravar um registro FIM e apagar de sua memória volátil as informações referentes à ação.

Há também duas situações em que um processo subordinado (intermediário ou folha) detecta término do tempo de espera por mensagens do seu processo pai:

1. quando o subordinado aguarda a ordem PREPARE de seu pai;
2. quando o subordinado está em estado PREPARADO e aguarda o voto de seu pai.

Na situação (1), o subordinado aborta a ação. Ele comporta-se como se tivesse recebido a decisão ABORTE, isto é, força a gravação de um registro ABORTADO e envia o voto não ao seu pai. Se o subordinado for também um processo intermediário, ele envia a decisão ABORTE a seus filhos e após receber todos os sinais RECEBI, grava assíncrono o registro FIM no log e apaga de sua memória volátil as informações sobre a ação.

Quando ocorre a situação (2), o subordinado envia mensagens PERGUNTANDO periodicamente ao seu pai enquanto não obtiver dele uma decisão. Ao obter a decisão, o subordinado comporta-se normalmente como em uma execução sem falhas do protocolo.

Devido à ocorrência de falhas, um processo operacional pode receber mensagens repetidas ou que, por demorarem a chegar, não afetam mais a finalização de ações nesse processo. Assim, se um processo recebe uma mensagem ABORTE ou VALIDE para uma ação sobre a qual ele não possui informações, ele envia mensagens RECEBI ao remetente da mensagem. Essa situação ocorre quando o processo já havia sinalizado o recebimento da decisão anteriormente, mas o seu pai falhou antes de receber o sinal. Nesse caso, conforme descrito anteriormente, o processo em recuperação envia novamente a decisão.

Se um processo recebe um voto correspondente a uma ação já finalizada, ou que ele não conhece, ele simplesmente ignora a mensagem de voto. Da mesma forma, se a mensagem PREPARE é recebida contendo o identificador de uma ação abortada ou sobre a qual não há nenhuma informação na memória volátil, o voto NÃO é enviado como resposta.

Correção do Protocolo

O protocolo de duas fases é um protocolo bloqueante. Se o coordenador falha antes de emitir o seu voto, todos os seus descendentes que votam SIM ficam bloqueados, ou seja, sem poder finalizar a ação. Isto ocorre pois quando um processo subordinado está em estado PREPARADO, ele concorda em validar a ação, mas não conhece os votos dos demais processos. Esse processo não pode finalizar a ação enquanto não receber de seu pai a decisão. A impossibilidade de um processo em estado PREPARADO decidir por si só abortar a ação fica demonstrada com os seguintes contra-exemplos:

1. considere que o processo PREPARADO decide por conta própria validar a ação após detectar a falha do nó onde executava o seu processo pai. Algum outro processo pode ter decidido abortá-la, logo os dois processos finalizam a ação de modo inconsistente e o protocolo fica obviamente incorreto;

2. se um subordinado X em estado PREPARADO decide abortar a ação por conta própria, ao detectar a falha do nó onde reside seu processo pai, pode ocorrer que seu processo pai tenha validado a ação e falhado antes de emitir a decisão VALIDE. Deve-se lembrar que de acordo com o protocolo, um processo ao entrar em estado PREPARADO envia o voto SIM a seu pai. Se todos os processos ancestrais de X e todos os seus irmãos concordam em validar a ação e não ocorre falha alguma pelo menos até o coordenador

tomar a decisão, o filho do coordenador que é o ancestral de X envia o voto SIM ao coordenador. Se os demais filhos do coordenador também votarem SIM, de acordo com o protocolo, o coordenador decide validar a ação, logo X e o coordenador finalizam a ação de modo inconsistente entre si.

Conclui-se, portanto, que os processos em estado PREPARADO devem permanecer bloqueados enquanto não receberem a decisão vinda de seus processos pais. Isto significa que o protocolo de duas fases pode não terminar, isto é, há casos em que um ou mais processos não conseguem finalizar a ação. Essa situação ocorre quando há processos em estado PREPARADO e o coordenador falha antes de emitir sua decisão e não se recupera mais. Entretanto, é fácil ver que as atividades relacionadas com a recuperação de processos falhos, descritas na seção anterior, uma vez eliminadas todas as falhas, permitem que todos processos finalizem a ação. Portanto, a condição (c5) é atendida.

É necessário demonstrar que a condição (c1) também é satisfeita, isto é, todos os processos que finalizam a ação o fazem de modo consistente.

Observando o protocolo, pode-se notar que há certa assimetria entre os modos como as decisões de validar e abortar a ação podem ser obtidas. Qualquer processo pode decidir abortar a ação durante a primeira fase, porém, somente o coordenador pode decidir validar. Portanto, para demonstrar a correção do protocolo, é suficiente demonstrar a validade das seguintes afirmações:

A1. se um processo aborta a ação, todos os processos que a finalizam também a abortam;

A2. se o coordenador valida a ação, então todos os processos que a finalizam também a validam.

A afirmação (A1) pode ser demonstrada analisando as situações em que um processo X pode decidir abortar a ação. Essas situações são as seguintes:

S1. X é um subordinado que não concorda em validar a ação;

S2. X é um subordinado que detecta término do tempo de espera pela mensagem PREPARE;

S3. X é um processo intermediário ou coordenador e detecta término do tempo de espera pelo voto de um de seus filhos.

S4. X é um processo que está se recuperando de uma falha ou que recebe uma mensagem PERGUNTANDO e X encontra um registro ABORTADO ou não encontra registro algum correspondente à ação.

Nas três primeiras situações, X envia o voto NÃO a seu pai. O pai de X também decidirá abortar a ação por receber o voto não de X ou por não receber voto algum durante o término do tempo de espera pelo voto de X. Pelos mesmos motivos, todos os ancestrais de X também

abortarão a ação. Logo o voto do processo filho do coordenador que é ancestral de X, se existir, será também NÃO. Se o coordenador estiver operacional, obrigatoriamente ele aborta a ação por receber o voto NÃO desse seu filho ou por não receber o voto dele durante o tempo de espera. Conforme visto anteriormente, a decisão de validar a ação é tomada apenas pelo coordenador. Além disso, durante a segunda fase, a decisão ABORTE é propagada a todos os nós em que há registros PREPARADO correspondentes a ação. Esses nós ao receberem a decisão também abortam a ação.

Para concluir a demonstração da veracidade da afirmação (A1), deve-se garantir que a situação (S4) apresentada acima não causa finalizações inconsistentes. Se nenhuma informação com relação a uma determinada ação é encontrada no log, é porque o processo falhou antes de emitir seu voto, caso ele seja um subordinado, ou falhou antes que ele emitisse a decisão, caso ele seja o coordenador[3]. O processo decide abortar a ação. Se for encontrado um registro ABORTADO, é porque o processo já possuía a decisão. Em ambos os casos, os ancestrais do processo abortarão a ação por receberem o voto NÃO desse processo ou por detectarem término do tempo de espera por esse voto. Pelas mesmas razões, o coordenador decidirá abortar a ação. Portanto, também na situação (S4), nenhum outro processo validará ação, pois o único processo que pode decidir validar é o coordenador e ele aborta a ação neste caso. Os processos que estão em estado PREPARADO e que finalizam a ação só poderão fazê-lo se receberem uma mensagem com a decisão, mas essa só pode ser a decisão ABORTE.

A afirmação (A2) também está correta, pois o coordenador só decide validar uma ação após receber os votos SIM de todos os seus processos filhos. Um processo intermediário vota SIM somente se concordar em validar a ação e receber votos SIM de todos os seus filhos. Portanto, para que a ação tenha sido validada pelo coordenador, todos os subordinados devem ter votado SIM. Se todos os subordinados votaram SIM, então todos forçaram a gravação do registro PREPARADO antes de votar. Logo, não há risco dos nós onde executam esses processos virem a abortar a ação por não encontrarem informação correspondente a ela durante a recuperação de uma falha. Os processos em estado PREPARADO (subordinados e processos de recuperação) finalizarão a ação somente após receberem a decisão de seus processos superiores, propagada a partir do processo coordenador. Esta decisão é a de validar a ação. Desse modo, fica demonstrada a correção do Protocolo de Duas Fases Hierárquico.

Tempo de Execução

No modelo de cálculo do tempo de execução descrito na Seção 2.3.6 o tempo gasto para emitir mensagens em paralelo é o mesmo para emitir uma única mensagem. No protocolo de duas fases hierárquico, há duas situações em que as mensagens podem ser enviadas em paralelo:

[3]Esse argumento é verdadeiro, pois de acordo com o protocolo, os subordinados gravam o registro PREPARADO antes de emitir o voto SIM e o registro ABORTADO antes de emitir o voto NÃO. O coordenador, por sua vez, grava o registro correspondente à decisão antes de enviá-la.

- quando um processo não folha emite mensagens aos seus filhos e
- quando os processos filhos emitem mensagens do mesmo tipo a seus pais.

O cálculo do tempo total de execução do protocolo de duas fases hierárquico é obtido com base na altura h da árvore de processos. Ao todo, são gastas $2h+2$ unidades de tempo: h para o envio de mensagens PREPARE, h para o envio dos votos, 1 unidade de tempo para o coordenador enviar mensagens com a decisão aos seus processos filhos e 1 uma unidade de tempo para que os filhos do coordenador enviem os sinais RECEBI ao coordenador.

Número de Mensagens

O número de mensagens enviadas pelo protocolo é medido como sendo o número de mensagens enviados pelos N processos que estão envolvidos na ação que é validada. Portanto, o número de mensagens do protocolo de duas fases hierárquico é $4(N-1)$ que corresponde a $(N-1)$ mensagens PREPARE, $(N-1)$ mensagens de votos, $(N-1)$ mensagens de decisão e $(N-1)$ mensagens RECEBI.

Número de Gravações no Log

Cada processo folha força a gravação de no mínimo um registro no log que é o registro ABORTADO, e força a gravação de no máximo dois registros (registros PREPARADO e ABORTADO ou registros PREPARADO e VALIDADO).

Cada processo intermediário grava no mínimo dois registros: um registro ABORTADO que tem a gravação forçada no log e um registro FIM cuja a gravação é assíncrona. No máximo, cada processo intermediário força a gravação de um registro PREPARADO e de um registro ABORTADO ou VALIDADO e grava de modo assíncrono um registro FIM.

O coordenador força a gravação de um dos registros ABORTADO ou VALIDADO e grava de modo assíncrono um registro FIM.

3.7 Protocolo PA ("Presumed Abort")

Durante a descrição do protocolo hierárquico, foi demonstrado que quando um processo recebe uma mensagem PERGUNTANDO contendo o identificador de uma ação sobre a qual ele não possui informação alguma, ele aborta a ação. Logo, um novo protocolo pode ser derivado do protocolo hierárquico de tal modo que quando uma ação seja abortada, o registro ABORTADO não necessita ter sua gravação forçada. Além disso, o coordenador e os intermediários que abortam a ação não necessitam aguardar os sinais RECEBI de seus filhos. Se posteriormente receberem uma mensagem PERGUNTANDO, eles enviam a decisão ABORTE

como resposta. Esse protocolo, proposto por Mohan e Lindsay[ML83], é denominado protocolo PA ("Presumed Abort"). Como o próprio nome indica, o protocolo PA presume que a ação tenha abortado quando não há nenhuma informação relacionada a ela.

O protocolo PA obtém uma redução ainda maior do número de mensagens enviadas e do número de gravações no log, por tratar de modo especial as ações de *leitura apenas*. Ações de *leitura apenas* não causam alteração nos dados estáveis dos nós onde atuam. São ainda classificadas em *ações parcialmente de leitura*, se apenas os dados de alguns nós não são alterados, e *ações totalmente de leitura*, se os dados de todos os nós são apenas lidos. Os processos que apenas lêem os dados estáveis locais são denominados *processos de leitura apenas*.

Descrição do Protocolo

O funcionamento do protocolo é idêntico ao do protocolo hierárquico para ações validadas que causam alterações dos dados estáveis dos nós onde atuam. As economias ocorrem somente em caso de aborto ou em ações de leitura apenas. Portanto, o protocolo será descrito somente para o caso de aborto e de ações de leitura apenas.

Primeira Fase

Um processo folha que não concorda em validar a ação entra em estado ABORTADO e após receber a mensagem PREPARE de seu pai, envia a ele o voto NÃO, conforme ocorre no protocolo hierárquico. Porém, ao entrar em estado ABORTADO, um processo folha executa uma gravação não-forçada do registro ABORTADO. Se um processo folha é um processo de leitura apenas, ao receber uma mensagem PREPARE, ele envia o voto LEITURA como resposta a seu processo pai. Nenhum registro PREPARADO é gravado em processos de leitura apenas. Para um processo de leitura apenas a decisão final de abortar ou validar a ação é indiferente, pois ele não altera dado algum do sistema. Um processo folha comporta-se como no protocolo hierárquico se o seu voto é sim.

Um processo intermediário I não grava registro PREPARADO se ele apenas lê os dados e recebe somente votos LEITURA de seus filhos. Neste caso, I envia o voto LEITURA a seu pai. Nenhum registro é gravado no log. I permanece no estado DESCONHECIDO. Por outro lado, I entra em estado PREPARADO e envia o voto SIM ao seu pai se seguintes fatos ocorrerem:

- um ou mais processos da subárvore com raiz em I alteram os dados locais;
- nenhum dos processos subordinados a I votam NÃO;
- o processo I concorda em validar a ação ou é um processo de leitura apenas.

Se I recebe um voto NÃO, ou não concorda em validar a ação, ou ainda, não recebe um dos votos durante o tempo de espera, I envia o voto NÃO a seu pai e entra em estado

ABORTADO, porém a gravação do registro ABORTADO é não-forçada.

O coordenador não toma decisão alguma se for um processo de leitura apenas e receber somente votos LEITURA de seus filhos. Nesse caso, ele simplesmente apaga de sua memória volátil as informações sobre a ação. Não há segunda fase do protocolo neste caso.

Ao decidir abortar a ação, o coordenador entra em estado ABORTADO e executa a gravação não-forçada do registro ABORTADO.

Segunda Fase

Se a decisão for a de ABORTAR a ação, cada processo não-folha envia a decisão somente aos filhos que votaram SIM. Neste caso, o processo não-folha não aguarda nenhum sinal de seus filhos. Ele simplesmente remove de sua memória volátil as informações sobre a ação[4].

Se a decisão é a de validar a ação, mas os votos de todos os filhos de um processo não-folha forem LEITURA, após entrar em estado VALIDADO esse processo não-folha não grava nenhum registro de FIM. Ele simplesmente apaga de sua memória volátil as informações sobre a ação.

Cada subordinado (intermediário ou folha) ao receber a decisão ABORTE entra em estado ABORTADO e executa uma gravação não-forçada do registro ABORTADO.

Tratamento de Falhas

Se um processo encontra um registro PREPARADO ao se recuperar de uma falha, ele envia mensagens PERGUNTANDO ao processo pai até que uma decisão seja recebida. A identidade do processo pai é obtida a partir do registro PREPARADO. Ao receber uma decisão, o processo comporta-se como em uma execução sem falha do protocolo.

Se for encontrado um registro VALIDADO no log para uma determinada ação, então se o processo que o gravou é um processo folha, não há nada a fazer; o processo simplesmente apaga da memória volátil as informações sobre a ação. Se o processo que gravou o registro era um processo não-folha, então se há um registro FIM correspondente, é porque o protocolo já havia terminado para aquela ação. Senão, o processo envia mensagens valide para todos os subordinados e aguarda a chegada dos sinais RECEBI. Ao receber todos os sinais dos subordinados, o processo grava de modo assíncrono um registro FIM e apaga de sua memória volátil as informações sobre a ação.

Ao receber uma mensagem PERGUNTANDO, um processo verifica se há informações sobre a ação na sua memória volátil ou no log. Se não houver nenhuma informação ou se um registro de ABORTADO (gravado de modo assíncrono) for encontrado , ele envia uma mensagem ABORTE ao processo que enviou a mensagem PERGUNTANDO. Se for encontrado um registro VALIDADO, o processo envia mensagem VALIDE como resposta e aguarda o recebimento dos

[4]O coordenador grava de modo assíncrono um registro FIM em ações parcialmente de leitura.

sinais RECEBI de todos subordinados. Ao receber esses sinais, o processo grava o registro FIM.

Correção do Protocolo

Uma execução sem falha do protocolo é semelhante a uma execução correspondente do protocolo hierárquico de duas fases para ações que não são de leitura apenas. A diferença se deve ao fato de que os registros ABORTADO não têm sua gravação forçada e os subordinados não enviam mensagens com os sinais RECEBI aos seus pais caso a decisão recebida seja ABORTE. Portanto, a possibilidade do protocolo estar incorreto poderia ocorrer durante a recuperação de processos falhos. Será demonstrado que essa possibilidade não ocorre.

O protocolo estaria incorreto se um processo em recuperação abortasse a ação enquanto os demais processos a tivessem validado, ou vice-versa. Se algum subordinado aborta a ação e vota NÃO durante a primeira fase, os seus ancestrais (incluindo o coordenador) também decidirão abortar a ação. Esses ancestrais abortam a ação por receberem o voto NÃO do subordinado, ou por detectarem término de espera pelo voto desse subordinado (em caso de falha de comunicação). Como conseqüência, os demais subordinados em estado PREPARADO que estão operacionais recebem a decisão ABORTE durante a segunda fase e também abortam a ação. Ao se recuperar de uma falha, um subordinado em estado PREPARADO envia mensagem PERGUNTANDO ao seu pai, o qual irá enviar a decisão ABORTE como resposta, quer por encontrar um registro ABORTADO, ou por não encontrar nenhuma informação sobre a ação (aborto presumido). Logo, se um processo aborta a ação, todos os processos que a finalizam também a abortam.

Um processo não-folha que valida a ação, grava o registro FIM e apaga as informações sobre a ação somente após forçar a gravação do registro VALIDADO e após receber os sinais RECEBI de todos os seus processos filhos. Cada subordinado, por sua vez, envia o sinal RECEBI somente após ter forçado a gravação do registro de VALIDADO. Logo, se um processo envia mensagem PERGUNTANDO ao coordenador ele irá obter a decisão VALIDE como resposta e, além disso, não há o risco do processo coordenador receber uma mensagem PERGUNTANDO após gravar o registro FIM. Portanto, se o coordenador valida a ação, todos os subordinados que finalizam a ação também a validam.

Tempo de Execução

O tempo de execução do protocolo PA é o mesmo do protocolo de duas fases hierárquico para uma ação em que todos os processos alteram dados. Se a ação é totalmente de leitura, o tempo de execução do protocolo é $2h$ unidades de tempo que corresponde a h unidades de tempo para emissão de mensagens PREPARE e h unidades de tempo para emissão dos votos. Se a ação é parcialmente de leitura, então o tempo de execução corresponde ao intervalo $(2h, 2h+2)$.

Número de Mensagens

O número de mensagens enviadas é o mesmo do protocolo de duas fases hierárquico, se a ação altera os dados em todos os processos e ela é validada. Nesse caso, o número de mensagens enviadas é $4N - 4$, onde N é o número de processos envolvidos.

Se a ação é completamente de leitura, então o total de mensagens é $2N-2$ que corresponde a $(N - 1)$ mensagens PREPARE e $(N - 1)$ votos LEITURA.

Se a ação é validada mas ela é parcialmente de leitura, então se há k subordinados onde a ação apenas lê os dados, o número de mensagens é: $2(2N - 2 - k)$, isto é, $N - 1$ mensagens PREPARE, $N - 1$ votos, $N - 1 - k$ mensagens com a decisão, $N - 1 - k$ sinais RECEBI, com $0 \leq k \leq N - 1$.

Número de Gravações no Log

Nenhum registro é gravado se a ação é totalmente de leitura.

Em ações parcialmente de leitura, cada processo intermediário força a gravação dos registros PREPARADO e VALIDADO e grava de modo assíncrono o registro FIM se algum de seus subordinados altera dados. Se, entretanto, apenas o processo intermediário altera dados, ele grava apenas dois registros: PREPARADO e VALIDADO (ambos têm a sua gravação forçada). Se um processo intermediário e todos os seus subordinados apenas lêem os dados, o processo intermediário não grava registro algum em nenhuma das duas fases. O coordenador força a gravação do registro VALIDADO e grava de modo assíncrono o registro FIM se tiver recebido pelo menos um voto SIM. Se somente o coordenador altera dados, então ele grava apenas o registro VALIDADO, que tem a sua gravação forçada no log. Um processo folha força a gravação dos registros PREPARADO e VALIDADO, se ele altera os dados locais e não grava registro algum se for um processo de leitura apenas.

3.8 Protocolo PC ("Presumed Commit")

Espera-se que a maioria das ações atômicas executadas sejam validadas. Portanto é interessante que as reduções no número de gravações forçadas e no número de mensagens enviadas sejam feitas no caso de validação da ação. O protocolo PC, apresentado em [ML83], tem esse objetivo. Como o protocolo PA, o PC é resultante de algumas alterações do protocolo de duas fases hierárquico.

Descrição do Protocolo

A primeira fase do protocolo PC é semelhante à do protocolo hierárquico quando a ação altera os dados em todos os nós e é abortada. As economias ocorrem quando a ação é

validada e em ações de leitura apenas. Portanto, de modo análogo à descrição do protocolo PA, será descrito somente o funcionamento do protocolo em caso de validação e de ações de leitura apenas.

No protocolo PC as gravações do registro VALIDADO não são forçadas, não há envio de sinais RECEBI e não ocorre a gravação do registro de FIM em caso de validação. O protocolo PC se vale também das ações de leitura apenas para reduzir o número de mensagens e de gravações durante a execução do protocolo para este tipo de ação.

Pode-se tentar derivar o protocolo PC a partir do protocolo de duas fases hierárquico, efetuando apenas as alterações citadas acima, de modo análogo ao que ocorreu com o protocolo PA. Entretanto, essa derivação simplista apresenta um problema, o qual é descrito a seguir.

No protocolo de duas fases hierárquico, um processo não-folha grava um registro para recuperação somente após receber os votos de seus filhos. Se esse mesmo procedimento fosse adotado no protocolo PC, poderia haver finalizações inconsistentes. O processo não-folha abortaria a ação ao se recuperar, mas responderia VALIDE após receber uma mensagem PERGUNTANDO, pois não há informação no log. Logo, o processo não-folha e os seus filhos finalizam a ação de modo contraditório: o primeiro aborta a ação e os demais a validam.

A solução apresentada pelo protocolo PC para esse problema consiste na gravação forçada de um registro especial denominado *registro* COLETA. Esse registro é gravado por cada processo não-folha antes de iniciar o envio das mensagens PREPARE aos seus filhos. O registro COLETA contém apenas o identificador da ação e os identificadores dos processos filhos do processo não-folha que o grava. Ao se recuperar de uma falha, um processo não-folha verifica a existência do registro de COLETA no seu log. Se existir tal registro, a ação deve ser abortada e a mensagem ABORTE é enviada aos processos cujos identificadores estão no registro COLETA.

Um processo folha que altera os dados locais, ao receber uma mensagem PREPARE, entra em estado ABORTADO se não concorda em validar a ação, o que corresponde a uma gravação forçada do registro ABORTADO. A seguir, envia o voto NÃO ao seu pai. Se concorda em validar a ação, entra em estado PREPARADO, forçando a gravação de um registro PREPARADO. Se o processo folha é um processo de leitura apenas, ele envia o voto LEITURA em resposta à mensagem PREPARE e apaga de sua memória volátil as informações sobre a ação.

Um processo intermediário que não é de leitura apenas, ao receber uma mensagem PREPARE, força a gravação de um registro COLETA e envia mensagens PREPARE aos seus filhos. Os motivos que levam um processo intermediário a entrar em estado PREPARADO ou ABORTADO são os mesmos descritos na primeira fase do protocolo de duas fases hierárquico. Os processos intermediários forçam a gravação dos registros PREPARADO e ABORTADO. Após entrar no estado PREPARADO ou ABORTADO, o processo intermediário envia a seu pai o voto correspondente ao novo estado (SIM ou NÃO). Se um intermediário é um processo de leitura apenas e recebe apenas votos LEITURA de seus filhos, ele envia o voto LEITURA a seu pai e executa a gravação não-forçada de um registro VALIDADO.

O coordenador força a gravação do registro COLETA no início da primeira fase, antes de enviar as mensagens PREPARE aos seus filhos. O coordenador decide abortar a ação se pelo menos um dos votos recebidos for NÃO ou se não receber os votos de um de seus filhos durante o tempo de espera. A decisão é a de validar a ação se o coordenador recebe todos os votos de seus filhos, sendo que nenhum deles é o voto NÃO e a ação não é totalmente de leitura. Nesse caso, o coordenador entra em estado VALIDADO, forçando a gravação de um registro VALIDADO e apaga de sua memória volátil as informações sobre a ação. O coordenador não aguarda sinais RECEBI e não grava registros FIM para ações validadas. Se a ação é de leitura apenas, o coordenador executa uma gravação não-forçada do registro VALIDADO.

Segunda Fase

Quando a decisão é de validar a ação, cada processo não-folha envia a decisão VALIDE aos seus processos filhos que não são de leitura apenas e apaga de sua memória volátil as informações sobre a ação. Nenhum sinal RECEBI é aguardado ou enviado e nenhum registro FIM é gravado. Um processo folha ao receber a decisão VALIDE executa a gravação não-forçada do registro VALIDADO. Em caso de aborto de uma ação, o funcionamento do protocolo é semelhante ao do protocolo de duas fases hierárquico.

Tratamento de Falhas

Ao se recuperar de uma falha, um processo verifica as informações que dispõe sobre determinada ação. Se for encontrado apenas um registro COLETA no log, o processo em recuperação força a gravação de um registro ABORTADO e envia mensagens ABORTE a todos os seus processos filhos. Ao receber todas as mensagens RECEBI, ele grava de modo assíncrono o registro FIM.

Se um processo não-folha em recuperação encontrar também um registro ABORTADO, mas não for encontrado um registro FIM, ele envia mensagens ABORTE a todos os processos subordinados e aguarda o recebimento dos sinais RECEBI. Ao recebê-los, o processo em recuperação comporta-se como descrito no parágrafo anterior.

Se um processo não-folha recebe uma mensagem PERGUNTANDO referente a uma ação sobre a qual não possui nenhuma informação, ele envia a mensagem VALIDE como resposta.

Se o processo folha em recuperação encontrar um registro ABORTADO, ele envia uma mensagem RECEBI ao seu pai e apaga de sua memória volátil as informações sobre a ação.

Se ao se recuperar um processo encontra um registro PREPARADO, ele envia mensagem PERGUNTANDO ao seu processo pai e aguarda uma resposta contendo a decisão final.

Correção do Protocolo

Durante uma execução sem falhas, o protocolo PC, assim como o protocolo PA, difere do protocolo hierárquico apenas pelo número e o modo com que os registros são gravados em log e pelo número de mensagens RECEBI enviadas. Logo, é necessário demonstrar apenas que os processos em recuperação que finalizam a ação o fazem de modo consistente com os processos operacionais.

O processo o coordenador de uma determinada ação pode encontrar os seguintes seguintes registros ao se recuperar:

a. encontrar um registro COLETA apenas;

b. encontrar um registro COLETA e um registro ABORTADO;

c. encontrar um registro COLETA e um registro VALIDADO;

No caso do item (a), o coordenador falha antes de obter uma decisão. Nenhum outro processo pode ter validado a ação, pois somente o coordenador pode decidir validar. O coordenador decide abortar a ação e envia mensagens ABORTE aos seus filhos. Os processos em estado PREPARADO que finalizarem a ação a abortarão por receberem a decisão ABORTE de seu pai. logo, todos abortam a ação.

No item (b), o coordenador falha após decidir abortar a ação. Nesse caso, se não houver um registro FIM o coordenador envia mensagens ABORTE aos filhos do coordenador e como ocorre no item (a), todos os processos que ainda não finalizaram a ação a abortam ao receberem a mensagem ABORTE dos seus processos pais. Se houver um registro FIM, então todos os subordinados também abortaram a ação.

No item (c), o coordenador não envia nenhuma mensagem. Ele não copia nenhuma informação do log referente à ação para sua memória volátil. Se posteriormente receber uma mensagem PERGUNTANDO para a ação, o coordenador enviará a mensagem valide como resposta. Todos os processos que estão PREPARADOS permanecem enviando mensagens PERGUNTANDO a seus pais enquanto não receberem a decisão. Em conseqüência, após a recuperação, todos os processos em estado PREPARADO que finalizarem a ação a validam.

Um processo subordinado pode encontrar as seguintes aos se recuperar:

a. encontrar um registro COLETA apenas;

b. encontrar um registro COLETA e um registro PREPARADO;

c. encontrar um registro COLETA e um registro ABORTADO;

d. encontrar um registro COLETA, um registro PREPARADO e um registro ABORTADO;

e. encontrar um registro COLETA, um registro PREPARADO e um registro VALIDADO.

No item (a), o processo subordinado ao se recuperar envia o voto NÃO ao seu pai e a decisão ABORTE aos seus filhos. Logo, todos os processos que finalizam a ação a abortam por receberem mensagens do processo subordinado ou, ainda, por detectarem término do tempo de espera quando esse subordinado falha.

No item (b), o processo subordinado (intermediário) não pode tomar nenhuma decisão, pois ele está em estado PREPARADO. Ele permanece aguardando a chegada de uma decisão. Portanto, ele não finaliza a ação de modo inconsistente ao dos outros processos.

Nos itens (c), (d) e (e), o subordinado já havia finalizado a ação antes da falha e, portanto, não toma nenhuma nova decisão inconsistente com a dos demais processos.

Tempo de Execução

Dada a altura h da árvore de processos, o cálculo do tempo total de execução do protocolo PC é $2h$ unidades de tempo: h unidades de tempo para o envio de mensagens PREPARE e h unidades de tempo para as mensagens com os votos. O tempo total é o mesmo para ações que alteram dados ou para ações de leitura apenas. Ao obter os votos de todos os subordinados e decidir validar a ação, o coordenador pode esquecer a ação. Não há o tempo adicional para o envio das mensagens com a decisão e das mensagens com os sinais RECEBI.

Número de Mensagens

O número total de mensagens enviadas para uma ação validada em que todos os processos alteram os dados é $3(N-1)$ que corresponde a $N-1$ mensagens PREPARE, $N-1$ mensagens de votos e $N-1$ mensagens com a decisão VALIDE. Onde N é o número de processos envolvidos na ação.

Se a ação é totalmente de leitura, então o total de mensagens é $2N-2$ que corresponde a $N-1$ mensagens PREPARE e $N-1$ votos LEITURA).

Se a ação é validada, então o número de mensagens é $3(N-1)-k$ mensagens com a decisão, onde $k, 0 \leq k \leq N-1$ é o número de processos cujos votos foram LEITURA.

Número de Gravações no Log

Durante uma ação totalmente de leitura, cada processo não-folha grava dois registros: o registro COLETA cuja gravação é forçada e o registro VALIDADO que é gravado de modo assíncrono. Os processos folhas não gravam registro algum.

Em ações que não são totalmente de leitura e que são validadas, o coordenador grava dois registros: COLETA e VALIDADO. A gravação desses registros é forçada. Cada processo intermediário que altera dados (ou possui subordinados que alteram dados) força a gravação dos registros COLETA e PREPARADO e grava de modo assíncrono o registro VALIDADO. Se

o processo intermediário e todos os seus descendentes apenas lêem os dados, O número de registros gravados pelo processo intermediário é o mesmo quando a ação é totalmente de leitura. O mesmo ocorre no caso de processo folha que apenas lê os dados. Cada processo folha que altera dados grava dois registros: o registro COLETA cuja gravação é forçada e o registro VALIDADO que é gravado de modo assíncrono.

3.9 NPC - "New Presumed Commit"

O protocolo PC, conforme a Seção 3.8, objetiva reduzir o número de mensagens quando uma ação é validada. Entretanto, o número de gravações em log feitas pelo coordenador e pelos processos intermediários para ações em validação é maior que no protocolo hierárquico de duas fases. Esse acréscimo no número de gravações se deve à necessidade de se gravar o registro de coleta, o que não ocorre no protocolo PA e no protocolo hierárquico.

Descrição do Protocolo

Nesta seção, é descrita uma nova versão do protocolo PC que visa eliminar o registro COLETA utilizado no protocolo PC[5].

Os objetivos do registro COLETA são:

- identificar ações abortadas devido a falhas de um nó;
- permitir informar sobre o aborto da ação aos subordinados de algum processo não-folha que poderia existir nesse nó.

O protocolo NPC [LL93] elimina o registro de coleta por meio da obtenção de um conjunto de identificadores de ações que podem ser abortadas devido à falha de um nó onde executa um processo não-folha. Esse conjunto é denominado conjunto A e corresponde a um intervalo estimado de identificadores de ações (*ias*) dentro do qual encontram-se as ações que estavam em execução durante a ocorrência da falha. No protocolo NPC, quando um processo não-folha decide abortar ao se recuperar de uma falha, ele não envia mensagens ABORTE aos seus filhos, pois não possui registro COLETA com os identificadores dos seus processos filhos. Esses processos filhos (em estado PREPARADO) são notificados sobre o aborto, após receberem as respostas às mensagens PERGUNTANDO que eles enviam periodicamente ao processo pai. O processo em recuperação responde ABORTE às mensagens PERGUNTANDO contendo o identificador de uma ação pertencente ao conjunto A. O conjunto A é obtido pelo gerenciador de ações atômicas do nó onde reside o processo em recuperação. Para obtenção do conjunto A é necessário que os *ias* sejam atribuídos em ordem crescente e de modo monotônico. O conjunto A pode ser obtido com base em dois outros conjuntos a saber:

[5]Será mostrado posteriormente que essa eliminação não é sempre possível ou recomendável para todas as ações.

- *REC* - O conjunto de ações *recentes* no sistema no momento da falha. É formado por todos os *ias* que se encontram no intervalo delimitado por ia_{inf} e ia_{sup}. O limite inferior ia_{inf} indica que abaixo desse limite todas as ações já haviam finalizado antes da ocorrência da falha. Logo, os identificadores das ações que estavam em execução devem ser maiores que ia_{inf}. O limite superior corresponde ao identificador da última ação que pode ter sido executada antes da falha;

$$REC = \{ia \mid ia_{inf} < ia < ia_{sup}\} \tag{3.1}$$

- *VAL* - Conjunto dos identificadores das ações validadas até o momento da falha. Ou seja, ações para as quais há registros VALIDADO correspondentes no log.

Dada as definições dos conjuntos acima, pode-se definir o conjunto A do seguinte modo:

$$A = REC - VAL = REC - (VAL \cap REC) \tag{3.2}$$

$(VAL \cap REC)$ é o conjunto de ações recentes que foram validadas. É importante notar que ia_{inf} e ia_{sup} não necessitam ser identificadores de ações reais, eles são apenas limites cujos valores podem ser estimados. Além disso, como o intervalo por eles delimitado é um intervalo contínuo, nem todos os *ias* pertencentes a este intervalo correspondem a ações que realmente existiram no sistema. Logo, o que se garante é que todas as ações que existiram e são recentes estão em *REC*.

Quando um nó se recupera de uma falha, é necessário que o sistema gerenciador de ações atômicas obtenha o conjunto A para que possa responder ABORTE às mensagens PERGUNTANDO que possam ser recebidas indagando sobre uma ação cujo identificador se encontre nesse conjunto.

O conjunto A pode ser representado pela seguinte estrutura:

$$< ia_{inf}, ia_{sup}, VAL \cap REC > \tag{3.3}$$

São descritos a seguir os modos como os limites podem ser obtidos e como o conjunto $(VAL \cap REC)$ pode ser representado.

Determinando ia_{sup}

O ia_{sup} pode ser obtido de dois modos:

1. utilizando um Δ: Denomina-se ia_{lmax} o maior *ia* encontrado no log após o coordenador se recuperar de uma falha. Ou seja, ia_{lmax} representa o maior identificador de ação cuja finalização foi registrada em log antes da ocorrência da falha. Escolhe-se um Δ fixo, por exemplo, e obtém-se $ia_{sup} = ia_{lmax} + \Delta$. Obviamente, Δ deve ser grande o suficiente para garantir que para todo *ia* de ações que executavam antes da falha sejam tais que $ia < ia_{lmax} + \Delta$; ou

2. gravando ia_{sup}: O ia_{sup} pode ser obtido durante a recuperação do coordenador lendo-se o seu valor diretamente do log. Isto requer que o valor de ia_{sup} seja periodicamente gravado de modo síncrono no log. Obviamente, essa solução é bem mais ineficiente que a anterior.

Quando o coordenador se recupera de uma falha, independente do método utilizado para obter ia_{sup}, ele deve utilizar *ias* maiores que o valor de ia_{sup} para novas ações, evitando a reutilização de *ias*, o que poderia gerar finalizações inconsistentes nos diversos processos.

Determinando ia_{inf}

Quando o sistema está operando normalmente é possível determinar qual ação mais antiga e que ainda não foi finalizada. O limite inferior ia_{inf} é gravado no log quando essa ação mais antiga finalizar. Se a ação mais antiga é validada, o ia_{inf} é gravado no log no momento da gravação do registro VALIDADO da ação. Se a ação mais antiga está sendo abortada, somente após o coordenador receber todos os sinais RECEBI é que ele grava (de modo assíncrono) o novo ia_{inf} no log, no mesmo instante em que é gravado o registro de FIM. O ia_{inf} gravado não corresponde necessariamente ao *ia* dessa ação, mas é geralmente um *ia* maior de uma ação também finalizada. O objetivo é obter um ia_{inf}, de modo que o intervalo $[ia_{inf}, ia_{sup}]$ seja o menor possível e, portanto, reduzindo ao máximo o tamanho do conjunto A. O próximo ia_{inf} será gravado no log quando a próxima ação mais antiga finalizar.

Se um nó onde executa um processo não-folha falhar antes que o valor de ia_{inf} tenha ultrapassado o valor de *ia* de uma ação validada, esse *ia* se encontra no intervalo (ia_{inf}, ia_{sup}), mas como o seu registro VALIDADO está no log, *ia* pertence a $(REC \cap VAL)$ e, portanto, não pertence ao conjunto A. Se o coordenador receber uma mensagem PERGUNTANDO contendo *ia*, ele enviará a mensagem VALIDE como resposta. Se, entretanto, o coordenador falhar após o valor de ia_{inf} ter ultrapassado o valor *ia* de alguma ação validada, então presume-se que a ação foi validada, pois o seu *ia* não está em REC. Do mesmo modo, a resposta é a decisão VALIDE.

Se um nó onde executa um processo não-folha falhar antes que o valor de ia_{inf} tenha ultrapassado o valor de *ia* de uma ação abortada, esse *ia* se encontra no intervalo (ia_{inf}, ia_{sup}), mas não pertence ao conjunto $(REC \cap VAL)$ e, portanto, *ia* pertence ao conjunto A. Se, entretanto, um nó onde executa um processo não-folha falhar após o valor de ia_{inf} ter ultrapassado o valor *ia* de alguma ação abortada os sinais RECEBI dessa ação já devem ter sido recebidos e, portanto, não haverá nenhuma mensagem PERGUNTANDO para essa ação.

Representação do Conjunto $VAL \cap REC$

Após a ocorrência de uma falha, o conjunto VAL pode ser obtido coletando-se através da leitura seqüencial do log os identificadores das ações para os quais são encontrados registros VALIDADO. Uma vez obtido o conjunto VAL, o conjunto $VAL \cap REC$ é obtido trivialmente.

É importante que o conjunto $VAL \cap REC$ seja representado de um modo bastante compacto, pois ele é mantido por tempo indeterminado na memória volátil de um nó em recuperação. A seguir, são apresentadas duas formas para representação desse conjunto:

1. se os *ias* são atribuídos consecutivamente, o conjunto pode ser representado através de um vetor de bits. O ia_{inf} é a origem do vetor V. O tamanho do vetor V é: $ia_{lmax} - ia_{inf}$, pois não há nenhuma ação validada com *ia* maior que ia_{lmax}. O conjunto $VAL \cap REC$ é representado pelo vetor do seguinte modo:

$$VAL \cap REC = \{ia \mid V[ia - ia_{inf}] = 1\} \tag{3.4}$$

2. se os *ias* não são atribuídos consecutivamente[6], uma representação esparsa como lista deve ser utilizada.

Funcionamento do Protocolo NPC

A seguir é descrito o funcionamento do protocolo para ações em que a gravação do registro COLETA não é necessária. Na próxima seção são tratados os casos especiais em que a gravação desse registro é requerida.

Na primeira fase, o protocolo é semelhante ao protocolo PC, exceto que não são gravados registros de COLETA. As mensagens se propagam pela árvore de processos a partir do coordenador e em direção aos processos folhas.

O coordenador decide abortar a ação se não receber o voto de um de seus filhos durante o tempo de espera ou se pelo menos um dos votos for NÃO. Se o coordenador receber o voto de todos os seus filhos e nenhum deles for o voto NÃO e pelo menos um deles for o voto SIM, o coordenador decide validar a ação. Se os votos recebidos forem todos LEITURA-APENAS, o coordenador não toma decisão alguma; ele simplesmente apaga de sua memória volátil as informações sobre a ação. Nenhum registro é gravado no log para ações totalmente de leitura.

Quando a decisão é de validar a ação, o coordenador força a gravação de um registro VALIDADO no log. O registro VALIDADO não necessita conter os identificadores dos subordinados. O coordenador envia mensagens com a decisão VALIDE a todos os processos que são seus filhos. Nenhum sinal RECEBI é aguardado e nenhum registro FIM é gravado. As informações sobre a ação são apagadas da memória volátil. Se a ação validada é a ação mais antiga ainda não finalizada, um novo ia_{inf} é gravado no logo no momento da gravação do registro VALIDADO. Cada processo intermediário, ao receber a decisão VALIDE, grava de modo assíncrono um registro VALIDADO, envia mensagens com a decisão VALIDE a seus filhos e apaga de sua memória volátil as informações sobre a ação. Se a ação a ser gravada for a mais antiga ainda não finalizada, o processo intermediário grava junto com o registro VALIDADO

[6] Por exemplo quando "timestamps" fazem parte da composição do *ia*.

um novo ia_{inf}. Cada processo folha, ao receber a decisão VALIDE, grava de modo assíncrono um registro VALIDADO e apaga de sua memória volátil as informações sobre a ação.

Se a decisão é a de abortar a ação, o coordenador força a gravação de um registro ABORTADO, envia mensagens ABORTE a seus filhos que votaram SIM e aguarda deles o sinal RECEBI. Cada processo intermediário em estado PREPARADO, ao receber a decisão ABORTE, força a gravação do registro ABORTADO no log, envia o sinal RECEBI a seu pai, mensagens com a decisão ABORTE a seus filhos e permanece aguardando destes o sinal RECEBI. Ao obter todos os sinais RECEBI, o coordenador e cada processo intermediário grava de modo assíncrono um registro de FIM. Se a ação abortada for a ação mais antiga ainda não finalizada, um novo ia_{inf} é gravado no momento da gravação do registro de FIM. Cada processo folha, ao receber a mensagem ABORTE, força a gravação de um registro VALIDADO e envia a mensagem com sinal RECEBI ao seu pai.

Ações Especiais

Há determinadas ações atômicas para as quais é necessário que o coordenador e os processos intermediários forcem a gravação do registro de coleta no log. Nesses casos, o protocolo deve funcionar como o protocolo PC.

Abaixo estão relacionadas algumas situações em que a utilização do registro de COLETA se faz desejável ou mesmo necessária:

1. quando a ação é abortada devido à detecção da falha de um dos processos subordinados. Nesse caso, haverá uma demora (indeterminada) no envio do sinal RECEBI pelo subordinado falho. Enquanto o coordenador não obtém todos os sinais RECEBI para uma determinada ação abortada cujo identificador é ia, o valor de ia_{inf} não pode ser atualizado para um valor maior que ia;

2. uma ação é muito longa. Enquanto não finalizar, ela impede que o ia_{inf} ultrapasse o valor de seu ia;

3. um mesmo nó pode conter um processo raiz para determinada ação e somente processos subordinados para outras ações. Quando contém apenas processos subordinados, o nó não tem controle de novos ias. Como conseqüência, um nó pode receber um valor de ia menor que o valor de seu ia_{inf}.

Nos primeiros dois casos, o uso do registro de COLETA é desejável, pois a impossibilidade de avançar o valor de ia_{inf} faz com que o intervalo (ia_{inf}, ia_{sup}) seja muito grande, o que torna o conjunto $VAL \cap REC$ grande, consumindo maior espaço de armazenamento.

No terceiro caso, o uso do registro de COLETA é realmente necessário, pois um nó recebe um ia menor que o seu ia_{inf}. Essa situação pode gerar finalizações inconsistentes nos vários nós envolvidos na ação. Por exemplo, seja um nó N que possui um processo P_{Nia1} que participa de uma ação cujo identificador é $ia1$ é menor que o limite ia_{inf} de N. Seja P_{Nia_1}

um processo não-folha que tenha pelo menos um processo filho executando em outro nó da rede. Se o nó N falha quando os processos filhos estão em estado PREPARADO, ao se recuperar, o processo P_{Nia_1} receberá uma mensagem PERGUNTANDO do processo filho executado remotamente. Essa mensagem possui o identificador ia_1. Mas o ia_{inf} obtido por P_{Nia_1} ao se recuperar é maior que ia_1. Logo N presume que a_1 validou, o que pode não ser verdade.

A solução utilizada nos três casos relacionados a cima é a mesma e corresponde à gravação do registro de COLETA pelo coordenador antes de iniciar o envio das mensagens PREPARE. Quando um processo não-folha recebe um identificador ia menor que o seu ia_{inf}, ele grava um registro COLETA para a ação correspondente ao ia recebido. Após a gravação do registro coleta, o processo não-folha pode incrementar o valor de ia_{inf} mesmo que a ação ia ainda não tenha sido finalizada.

Ao se recuperar de uma falha, o processo não-folha copia para sua memória volátil as informações contidas no registro COLETA obtidos do log.

Tratamento de Falhas

Se um processo em recuperação estava em estado PREPARADO, ele envia periodicamente mensagens PERGUNTANDO ao seu pai. Ao obter uma resposta, o processo em recuperação comporta-se como um subordinado PREPARADO ao receber a decisão em uma execução sem falha do protocolo. Se for encontrado um registro ABORTADO e não for encontrado um registro FIM correspondente, o processo em recuperação envia mensagens ABORTE a todos os seus filhos caso ele seja um processo não-folha. Ao obter todos os sinais RECEBI, o processo em recuperação grava um registro FIM e apaga de sua memória volátil as informações sobre a ação.

Quando um nó se recupera de uma falha, o sistema gerenciador de ações atômicas deve obter os intervalos ia_{inf} e ia_{sup} e o conjunto A de ações que devem ser abortadas.

Quando um nó em recuperação recebe uma mensagem PERGUNTANDO contendo o identificador de uma ação, ele inicialmente verifica a existência de registros correspondentes a essa ação. Se for encontrado um registro ABORTADO ou um registro COLETA, o sistema gerenciador de ações atômicas desse nó recupera o processo que gravou esse registro. O processo em recuperação comporta-se como no protocolo PC (ver *Recuperação de Erro* na Seção 3.8). Se não for encontrado nenhum registro, é porque a ação não é considerada uma ação especial. Nesse caso, o sistema gerenciador de ações atômicas envia mensagem ABORTE se o ia correspondente à ação pertencer ao conjunto A. Em caso contrário, a decisão VALIDE é enviada.

Correção do Protocolo

Durante uma execução sem falhas, o protocolo é semelhante ao protocolo de duas fases hierárquico, diferindo apenas quanto ao número e modo como os registros são gravados no log. Portanto, durante execuções sem falha, todos os nós que finalizam a ação o fazem de modo consistente. É necessário demonstrar que, após a ocorrência de uma falha, os processos operacionais e os processos em recuperação finalizam a ação de modo consistente entre si.

Quando ocorre a falha de um nó após a decisão de um processo em abortar a ação, ou antes do processo entrar em estado PREPARADO, os demais processos também abortam a ação. Os ancestrais desse processo (caso ele não seja o coordenador) abortam por receberem o voto NÃO ou por não receberem voto algum. Os descendentes abortam por receberem a mensagem ABORTE desse processo, ou por não receberem a mensagem PREPARE durante o tempo de espera.

Quando um subordinado está em estado PREPARADO, ele envia periodicamente mensagens PERGUNTANDO a seu pai, enquanto não obtiver dele uma decisão. Quando um processo recebe uma mensagem PERGUNTANDO, relacionada com uma dada ação, o protocolo garante que a resposta desse processo seja consistente com o modo como ele finalizou a ação. Essa garantia se deve ao registro COLETA, ao conjunto A e ao princípio de validação presumida, os quais permitem detectar se a ação foi abortada ou validada nesse nó. Como conseqüência, a resposta enviada é coerente com a finalização da ação. Logo todos os processos subordinados em estado PREPARADO que recebem uma decisão finalizam a ação de modo consistente com os demais nós que a finalizaram.

Tempo de Execução

Idêntica ao do protocolo PC.

Número de Mensagens

Idêntica ao do protocolo PC.

Número de Gravações no Log

É abordado nesta seção apenas o número de gravações em log de ações que não são consideradas pelo protocolo como ações especiais. Para as ações especiais o protocolo funciona como o protocolo PC e, portanto, o número de gravações é idêntico ao desse protocolo.

Em uma ação validada totalmente de leitura, nenhum processo grava registros.

Em ações que não são totalmente de leitura, o coordenador força a gravação do registro VALIDADO. Um processo intermediário que é raiz de uma subárvore que altera dados força a gravação dos registros PREPARADO e VALIDADO. Se todos os processos da subárvore apenas

lêem os dados, o processo intermediário raiz dessa subárvore comporta-se como no caso de ação de leitura apenas. Ou seja, não grava registro algum. Um processo folha que altera dados força as gravações de dois registros: PREPARADO e VALIDADO. Nenhum registro é gravado se o processo folha for de leitura apenas.

3.10 Protocolo Flexível de Validação de Duas Fases

O protocolo Flexível [Bue89] é um protocolo derivado do protocolo hierárquico de duas fases em que cada processo envolvido em uma ação tem a liberdade de iniciar o protocolo assim que terminar a sua execução em favor da ação. Os processos subordinados não necessitam aguardar a mensagem PREPARE para emitirem os seus votos. Assim que um processo obtém o seu voto ele o envia a seu pai.

Descrição do Protocolo

O protocolo flexível de validação de duas fases permite a independência entre a tarefa de iniciar e a tarefa de coordenar o protocolo. Qualquer processo subordinado que terminar a execução da ação pode dar início a execução do protocolo. Para isso, basta que esse processo envie o seu voto a seu nó pai assim que as operações relacionadas forem concluídas. Um subordinado, portanto, não necessita aguardar a mensagem PREPARE de seu nó pai para enviar seu voto.

O protocolo flexível se baseia na árvore de processos para a emissão das mensagens de modo parecido ao que ocorre no protocolo hierárquico de duas fases. O protocolo se difere desse último no que diz respeito à primeira fase. A seguir, é apresentada a primeira fase do protocolo Flexível.

Primeira Fase

Cada processo subordinado intermediário N executa os seguintes passos:

1. ao terminar de executar a ação localmente, um processo intermediário N envia a mensagem PREPARE aos seus processos filhos, mesmo que ainda não tenha recebido a mensagem PREPARE de seu pai;

2. N aguarda a chegada das mensagens contendo os votos de seus filhos;

3. se todos os votos recebidos forem SIM e N concorda em validar a ação, ele entra em estado PREPARADO, forçando a gravação de um registro PREPARADO no log do nó onde N executa e envia a mensagem com o voto SIM a seu processo pai. Se N não concorda em validar a ação, ou recebe um voto NÃO de um de seus filhos, ou não recebe um ou

mais votos durante o tempo de espera, ele força a gravação de um registro ABORTADO e envia mensagem com o voto NÃO a seu pai.

Um processo folha envia o seu voto ao nó pai ao terminar a sua execução local da ação.

O processo raiz, o coordenador, executa os passos (2) e (3) descritos para os processos intermediários, exceto que no passo (3) ele não envia voto a nenhum processo.

O protocolo flexível possui as seguintes vantagens:

1. adaptação ao tipo de aplicação - em algumas aplicações distribuídas, os processos folhas terminam primeiro; logo, a ativação do protocolo se dará a partir dos nós folhas em direção à raiz, podendo-se ter dois ou mais nós folhas iniciando o protocolo ao mesmo tempo;

2. número de mensagens - pode-se obter uma redução do número de mensagens enviadas durante a execução do protocolo, evitando o envio de mensagens PREPARE. Um processo não-folha N não necessita enviar mensagem PREPARE a um processo filho F, se já tiver recebido de F o seu voto. A eliminação por completo das mensagens PREPARE pode ser obtida fixando-se os processos-folhas como iniciadores. Desse modo, o protocolo inicia-se diretamente com o envio das mensagens de voto.

O protocolo flexível funciona como uma generalização do protocolo de validação de duas fases, com relação ao início do protocolo. O protocolo Flexível pode ser ainda otimizado, através da técnica de aborto presumido ou validação presumida e através de tratamento diferenciado de ações de leitura apenas. Assim, pode-se derivar um protocolo PA Flexível ou PC Flexível a partir desse protocolo, objetivando diminuir o número de mensagens e o número de gravações no log.

Tratamento de Falhas

A recuperação de nós falhos e as medidas executadas por cada processo ao detectar falha de outro processo são semelhantes às do protocolo hierárquico de duas fases, exceto que um subordinado aguarda a mensagem PREPARE de seu pai para emitir o seu voto. Portanto, o subordinado não aborta a ação por não ter recebido essa mensagem, como ocorre no protocolo hierárquico de duas fases.

Correção do Protocolo

A lógica de execução do protocolo Flexível é semelhante a do protocolo hierárquico de duas fases, portanto, argumentos semelhantes podem ser utilizados para demonstrar sua correção.

Tempo de Execução

O melhor caso ocorre quando todos os subordinados terminam a execução antes que qualquer mensagem PREPARE seja enviada. Nesse caso, considerando h a altura da árvore de processos, o tempo de execução é h unidades de tempo a menos que o tempo de execução do protocolo hierárquico de duas fases. Ou seja, $4h - h$.

O pior caso é semelhante ao tempo de execução do protocolo hierárquico de duas fases e corresponde a uma execução em que todos os nós não-folhas emitem mensagens PREPARE.

Número de Mensagens

No melhor caso, são enviadas $3(N - 1)$ mensagens. No pior caso, o número de mensagens é o mesmo do protocolo hierárquico de duas fases, ou seja, $4(N - 1)$ mensagens.

Número de Gravações em Log

O número de gravações em log é o mesmo do protocolo hierárquico, para cada processo.

3.11 Protocolo Linear

Descrição do Protocolo

Primeira Fase

No protocolo linear os processos são enumerados em ordem crescente. Seja K o menor número dentre os processos envolvidos na ação, e seja M o maior número. O processo de número K inicia o protocolo enviando o seu voto ao processo $K+1$. O voto de cada processo de número $I, K < I < M$ é uma combinação do voto que o processo I recebe do processo $I - 1$ com sua concordância ou não em validar a ação. O processo I vota SIM, se receber o voto SIM do processo $I - 1$ e ele (processo I) concordar em validar a ação. Nesse caso, I força a gravação do registro PREPARADO e envia o voto SIM ao processo $I + 1$. Se o processo I não receber o voto do processo $I - 1$ ou receber e ele for NÃO ou se o processo I não concordar em validar a ação, I força a gravação do registro ABORTADO e envia o voto NÃO ao processo $I + 1$.

O processo de número M decide validar a ação se concordar em validá-la e receber do processo $M - 1$ o voto SIM. Nesse caso, o processo M força a gravação de um registro VALIDADO. O processo M aborta a ação se receber o voto NÃO do processo $M - 1$, ou não receber dele nenhum voto durante o tempo de espera, ou se M não concorda em validar a ação. Ao decidir abortar a ação o processo M força a gravação de um registro ABORTADO.

Segunda Fase

O processo M inicia a segunda fase enviando mensagens com a decisão ao processo $M-1$ e aguarda dele o sinal RECEBI. Ao receber o sinal RECEBI, o processo M grava de modo assíncrono o registro FIM e apaga de sua memória volátil as informações sobre a ação. Cada processo $I, K < I \leq M-1$ ao receber a decisão, executa os seguintes passos:

1. força a gravação de um registro correspondente à decisão (i.e., ABORTADO ou VALIDADO);
2. envia mensagem com a decisão processo $I-1$;
3. envia mensagem RECEBI ao processo $I+1$;
4. aguarda mensagem RECEBI do processo $I-1$.
5. ao receber o sinal RECEBI, o processo $I-1$ grava de modo assíncrono um registro FIM e apaga de sua memória volátil as informações sobre a ação.

O processo K, ao receber a decisão de $K+1$, executa os passos 1 e 3 e apaga de sua memória volátil as informações sobre a ação.

Cada processo $I, K < I \leq M$, ao receber um sinal RECEBI, grava de modo assíncrono, um registro FIM.

Tratamento de Falhas

Um processo X pode detectar a falha de outro processo nas seguintes situações:

1. X aguarda o voto de $X-1$;
2. X aguarda a decisão de $X+1$;
3. X aguarda o sinal RECEBI de $X-1$.

Na situação (1), X comporta-se como se tivesse recebido o voto NÃO de $X-1$, ou seja, aborta a ação e envia o voto NÃO a $X+1$.

Na situação (2), X está em estado PREPARADO. Ele não pode sair desse estado enquanto não receber a decisão. X envia periodicamente a mensagem PERGUNTANDO a $X+1$, enquanto uma decisão não for obtida. Ao obter a decisão, X comporta-se conforme descrito na segunda fase do protocolo.

Na situação (3), X aguarda o sinal RECEBI de $X-1$. Enquanto não receber esse sinal X deve manter informações sobre a ação em sua memória volátil. X envia, periodicamente,

mensagens com a decisão a $X - 1$, enquanto não obter o sinal RECEBI. Quando o sinal é recebido, X grava de modo assíncrono um registro FIM.

Quando um processo Y se recupera de uma falha, se ele encontrar um registro VALIDADO ou ABORTADO, mas não for encontrado um registro FIM correspondente, então, X envia mensagem com a decisão ao processo $X - 1$, se $X > K$ e envia o sinal RECEBI a $X + 1$, se $X < M$ e grava um registro de FIM. Se for encontrado um registro FIM, o processo apaga de sua memória volátil as informações sobre a ação. Se for encontrado um registro PREPARADO gravado por um processo X, esse processo envia periodicamente mensagens PERGUNTANDO ao processo $X + 1$. Ao obter uma resposta, X age como o processo X agiria, na segunda fase do protocolo.

Se um processo uma mensagem PERGUNTANDO referente a uma ação sobre a qual ele não possui nenhuma informação, ele envia a mensagem ABORTE como resposta.

Se um processo envolvido no protocolo (coordenador, subordinado) recebe uma das mensagens: RECEBI, SIM, NÃO, ABORTE, ou VALIDE, correspondente a uma ação sobre a qual ele não possui nenhuma informação, ele apenas despreza a mensagem.

Correção do Protocolo

O protocolo linear pode ser visto como um caso particular do protocolo Flexível mostrado na Seção 3.10, em que o processo raiz da árvore corresponde ao processo M e há apenas um processo folha que é o processo P. Cada processo processo I da árvore tem apenas um processo filho que é o processo $I - 1$. Portanto a demonstração da correção do protocolo é semelhante à do protocolo Flexível quando a árvore de processos possui a forma linear descrita acima.

Tempo de Execução

Cada processo envia uma mensagem após ter recebido uma mensagem do processo anterior (ou posterior, dependendo da fase considerada). Logo, o tempo gasto é $3N - 3$, onde N é o número de processos.

Número de Mensagens

O número de mensagens enviadas durante a execução do protocolo linear é $3N - 3$, onde N é o número de processos envolvidos na ação.

Número de Gravações em Log

O processo K força a gravação dos registros PREPARADO e VALIDADO.

O processo M força a gravação do registro VALIDADO e grava de modo assíncrono o registro FIM.

Cada processo $I, K < I < M$ força a gravação dos registros PREPARADO e VALIDADO e grava de modo assíncrono o registro FIM.

Capítulo 4

Protocolos de Duas Fases Tolerantes a Falhas

Neste capítulo são descritas algumas variações de protocolos de duas fases que visam continuar sua execução mesmo em presença de falhas no sistema.

É adotado o mesmo modelo utilizado para a apresentação dos protocolos descritos no capítulo anterior.

4.1 Protocolo O2PC ("Open Two-Phase Commit")

O protocolo apresentado nesta seção foi proposto por Rothermel e Pappe [RP90] para sistemas distribuídos cujos nós podem ser nós *confiáveis* ou nós *não-confiáveis*. Os nós *confiáveis* são aqueles que em algum instante se recuperam de uma falha. Os nós *não-confiáveis* são aqueles que ao falharem podem não voltar a fazer parte do sistema. Um exemplo de sistema com esta característica é um sistema em que alguns nós da rede são permanentes, como os nós servidores. Estes nós são considerados *confiáveis*, pois espera-se que eles se recuperem de falhas e voltem a participar da rede em um intervalo de tempo considerado pequeno. Nesse sistema, entretanto, computadores pessoais podem se conectar e se desconectar da rede imprevisivelmente. Esses computadores são considerados *não-confiáveis*, pois ao falharem eles podem ser retirados definitivamente da rede.

O protocolo O2PC utiliza o modelo hierárquico de comunicação entre processos. Os processos que executam em favor de uma ação são denominados *processos confiáveis* se eles executam em nós *confiáveis*. Em caso contrário, são denominados *processos não-confiáveis*. Processos não-confiáveis podem ativar processos confiáveis e não-confiáveis, mas processos confiáveis ativam apenas processos confiáveis. Desse modo, todos os descendentes de um processo confiável na árvore de execução de uma dada ação são também confiáveis. Um processo confiável é denominado *processo de entrada* se o seu superior na árvore de execução

for um processo não-confiável.

Descrição do Protocolo

Após o término da execução da ação, o nó que a iniciou (que é o nó raiz da árvore de execução) transfere o controle da finalização ao processo coordenador que reside em um nó confiável. Essa transferência é necessária apenas se o processo iniciador da ação residir em um nó não-confiável. Consumada a transferência, o coordenador passa a executar o protocolo de validação. O protocolo O2PC está, portanto, dividido (logicamente) em duas etapas: a etapa de transferência do controle e a etapa de validação da ação que é executada através do protocol genérico (O2PCg). Inicialmente, é descrito o protocolo genérico de validação e, a seguir, são descritos os métodos para transferir o controle do protocolo do processo raiz da árvore de execução para o processo coordenador.

O Protocolo Genérico O2PCg

O protocolo genérico O2PCg opera com base na *árvore de terminação* que é a árvore de processos obtida a partir da árvore de execução após o controle ter sido transferido do iniciador da ação para o coordenador. A árvore de terminação é basicamente uma versão reestruturada da árvore de execução em que o processo raiz não é necessariamente o iniciador da ação, mas sim o coordenador do protocolo. Todos os processos que participam da árvore de execução também participam da árvore de terminação. O mesmo ocorre com cada aresta que liga dois processos, embora essas arestas possam ter sentidos opostos na árvore de terminação. O formato da árvore de terminação depende do método de transferência utilizado.

O protocolo O2PCg é obtido a partir do protocolo PA (Seção 3.7). No protocolo O2PCg todas as mensagens com a ordem PREPARE contém o identificador do coordenador. Cada nó subordinado ao entrar em estado PREPARADO, inclui no registro PREPARADO o valor do identificador do nó onde reside o processo coordenador. Essa alteração é necessária pela seguinte razão: sejam A e B dois processos da árvore de terminação. O processo A não confiável é pai do processo confiável B (processo de entrada). Considere que B esteja PREPARADO. Se A falha definitivamente[1] antes de propagar a decisão ao processo B, esse último fica bloqueado para sempre. Essa situação pode ser evitada, fazendo-se com que B, ao detectar a falha de A, envie mensagens PERGUNTANDO diretamente ao coordenador. O coordenador envia a decisão ao processo B que pode finalizar a ação.

A alteração apresentada acima causa a necessidade de se alterar novamente o protocolo PA no que diz respeito ao envio de mensagens RECEBI. No protocolo PA, um processo intermediário I, ao receber uma mensagem VALIDE, entra em estado VALIDADO, envia a mensagem VALIDE aos seus filhos e o sinal RECEBI a seu pai. Por questão de simplicidade será considerado que o processo I seja filho do coordenador e que I executa em um nó não-confiável

[1] A falha definitiva de um processo não confiável significa que ele não mais se recupera após a falha.

N. Se N falhar após I enviar o sinal RECEBI a seu pai, os processos filhos de I podem, por algum motivo, não receber a decisão. Esses processos em estado PREPARADO, de acordo com a alteração proposta no parágrafo anterior, irão enviar mensagens PERGUNTANDO diretamente ao coordenador. O coordenador já deve ter esquecido a ação após obter as mensagens RECEBI de todos os seus filhos, inclusive de I. Como conseqüência, o coordenador presumirá a ocorrência de aborto e sua resposta será ABORTE. Logo, os filhos de I abortarão a ação, sendo que os demais processos a validaram. Para evitar essas finalizações inconsistentes, no protocolo O2PCg é exigido que um processo intermediário adie o envio do sinal RECEBI para quando obtiver os sinais RECEBI de todos os seus filhos. Desse modo, quando o coordenador recebe os sinais RECEBI de todos os seus filhos, tem-se a certeza de que todos os descendentes de cada filho do coordenador já haviam recebido a decisão VALIDE e haviam validado a ação.

No protocolo PA, o coordenador, ao receber os sinais RECEBI de seus filhos, grava um registro de FIM e apaga de sua memória volátil as informações sobre a ação. No protocolo O2PCg se um dos filhos do coordenador for um processo não-confiável, pode ocorrer que o coordenador nunca receba um sinal desse processo filho. Nesse caso, ele ficaria armazenando informações sobre a ação indefinidamente. Esse armazenamento não é conveniente na prática, pois torna a execução dos processos mais lenta. A seguir, é apresentada uma solução para o problema do armazenamento indefinido dessas informações.

Após entrar em estado VALIDADO ou ao encontrar um registro VALIDADO ao se recuperar, o coordenador executa o seguinte procedimento:

1. envia a decisão VALIDE a todos os seus processos filhos e aos processos entrada;
2. aguarda a chegada dos sinais RECEBI dos processos mencionados no item anterior;
3. enquanto não receber os sinais RECEBI dos processos de entrada, envia novamente a decisão a esses processos sempre que detectar término do tempo de espera.

O procedimento descrito acima envia mensagens com a decisão apenas uma vez aos filhos não-confiáveis do coordenador. Se o coordenador não obtém deles o sinal RECEBI, ele assume que esses filhos falharam definitivamente e continua exigindo os sinais RECEBI apenas dos processos de entrada. Uma vez que cada processo de entrada sinaliza o recebimento da decisão, o coordenador tem a garantia de que todos os processos confiáveis irão validar a ação[2].

É necessário que o coordenador conheça o identificador de todos os processos de entrada envolvidos na ação, para que o procedimento descrito anteriormente possa ser executado. Essa informação pode ser obtida durante a primeira fase. Cada processo de entrada inclui o seu identificador na mensagem que contém seu voto. Cada processo confiável ou não, ao emitir seu voto, coleta os identificadores que recebem com os votos de seus filhos e os

[2]Essa garantia é devida ao fato de todos os processos descendentes dos processos de entrada serem também confiáveis. Logo, se um desses descendentes falha, em algum instante após se recuperar ele receberá decisão do seu pai.

inclui no voto que irá enviar ao seu processo pai. Desse modo, no final da primeira fase o coordenador conhece os identificadores de todos os processos de entrada. O coordenador deve incluir no registro de VALIDAÇÃO esses identificadores. Ao receber os sinais RECEBI de todos os processos de entrada, o coordenador grava um registro de FIM e apaga de sua memória volátil as informações sobre a ação.

Embora o procedimento do processo de recuperação descrito acima garanta a finalização consistente dos processos confiáveis, o mesmo não é verdade para os processos não-confiáveis.

Ao se recuperar de uma falha, um processo não-confiável pode vir a abortar de modo inconsistente uma ação. Esse fato ocorre, pois o coordenador após receber os sinais RECEBI dos processos de entrada grava o registro de FIM e apaga de sua memória virtual as informações sobre a ação. Ao receber uma mensagem PERGUNTANDO posteriormente, para esta ação, o coordenador responde ABORTE uma vez que não encontra informações sobre ela em sua memória volátil.

A solução para esse problema, apresentada em [RP90], consiste em utilizar um *processo arquivo*. O coordenador cria um processo arquivo ao detectar término do tempo de espera pelo sinal de um de seus filhos não-confiáveis. O processo arquivo tem como finalidade armazenar em memória estável uma indicação que a ação validou e foi esquecida pelo processo coordenador. Esse processo é utilizado apenas para atender a mensagens PERGUNTANDO vindas dos nós não-confiáveis. Após receber os sinais dos processos de entrada, o processo coordenador grava um registro de FIM e apaga de sua memória volátil as informações sobre a ação, como descrito anteriormente.

Ao receber uma mensagem PERGUNTANDO sobre a qual não possui informação, o coordenador verifica inicialmente se a mensagem vem de um nó confiável. Se vier, ele envia a mensagem ABORTE como resposta. Se a mensagem PERGUNTANDO vem de um nó não-confiável, o coordenador ativa o processo arquivo para verificar se a ação foi esquecida. Se o processo arquivo encontrar um registro para a ação, o coordenador envia a mensagem VALIDE como resposta. Em caso contrário, a mensagem ABORTE é enviada.

Resumindo, o protocolo O2PCg pode ser obtido a partir do protocolo PA pelas seguintes alterações:

A1. as mensagens PREPARE incluem o identificador do coordenador.

A2. os registros PREPARADO devem incluir o identificador do coordenador.

A3. as mensagens com os votos de cada processo devem incluir os identificadores dos processos de entrada que o processo conhece.

A4. um processo de entrada em estado PREPARADO envia mensagens PERGUNTANDO diretamente ao coordenador ao se recuperar de uma falha ou ao detectar término do tempo de espera por uma decisão.

A5. um processo intermediário, ao receber a decisão VALIDE de seu processo pai, força a

gravação de um registro de VALIDAÇÃO no log, envia a decisão aos seus filhos e aguarda deles as mensagens com os sinais RECEBI. Somente após receber os sinais RECEBI de seus filhos, o processo intermediário envia esse sinal ao seu pai, grava um registro de FIM e apaga da memória volátil as informações sobre a ação.

A6. Para uma dada ação validada, mas para a qual ainda não há um registro FIM correspondente, o processo coordenador envia , periodicamente, mensagens com a decisão aos processos de entrada e envia uma única mensagem com a decisão aos seus filhos que são processos não-confiáveis. Ao obter o sinal RECEBI de todos os processos de entrada, o coordenador grava de modo assíncrono um registro FIM. Se detectar término do tempo de espera por um sinal RECEBI de um processo filho não-confiável, o coordenador ativa o processo arquivo.

A7. Ao receber uma mensagem PERGUNTANDO para uma ação sobre a qual não há informação na memória volátil, o coordenador pode:

1. enviar mensagem ABORTE se o nó que pergunta for confiável;
2. ativar o processo de arquivo para verificar se a ação foi esquecida. Se houver uma indicação da validação, o coordenador envia a mensagem VALIDE, caso contrário, a decisão ABORTE é enviada.

Métodos de Transferência do Controle do Protocolo

Quando o processo iniciador da ação recebe um pedido da aplicação para validar a ação, ele deve transferir o controle do protocolo para um processo de entrada caso ele seja um processo não confiável. Essa transferência se dá de processo a processo através do *caminho de transferência* que é definido como o caminho da árvore de execução que inicia no processo raiz e termina em um processo de entrada. Para cada ação pode haver mais de um caminho de transferência. Com exceção dos processos de entrada, os processos que constituem o caminho de transferência são não-confiáveis. Técnicas de otimização podem ser aplicadas na escolha do melhor caminho. Não serão considerados estas técnicas na descrição do protocolo. Assume-se que o processo raiz da árvore de execução transfere o controle através de um caminho de transferência previamente escolhido.

Método de Transferência Explícita

No método de transferência explícita, a função de coordenador é transferida explicitamente em uma fase separada que precede a execução do protocolo genérico O2PCg. O nó raiz não confiável, ao receber um pedido da aplicação para validar a ação, envia a mensagem especial TRANSFERÊNCIA para o seu filho que faz parte do caminho de transferência. Quando um processo do caminho de transferência envia a mensagem transferindo a função do coordenador para o seu sucessor, a aresta que os liga tem o seu sentido invertido, ou seja, o processo

que envia a mensagem TRANSFERÊNCIA passa a ser o filho do processo que a recebe. Um processo, ao receber a mensagem TRANSFERÊNCIA, verifica se a ação não abortou no nó onde ele reside. Se a ação tiver sido abortada, o processo envia uma mensagem ABORTE ao remetente da mensagem TRANSFERÊNCIA. Desse modo, a cada transferência entre dois processos a árvore de execução vai se transformando na árvore de terminação. Quando a mensagem TRANSFERÊNCIA atinge o processo de entrada, este se torna a raiz da árvore de terminação e o coordenador que irá iniciar a execução do protocolo O2PCg. Obtém-se, assim, a árvore de terminação.

Método de Transferência Implícita

No método que é descrito a seguir, a transferência de função do coordenador é feita implicitamente, durante a primeira fase do protocolo genérico O2PCg.

Quando o programa de aplicação decide validar a ação, o processo iniciador envia mensagens PREPARE a todos os seus filhos que não estejam no caminho de transferência escolhido e aguarda pela chegada de seus votos. Se os votos chegarem e forem todos SIM, o processo iniciador envia mensagem PREPARE$_t$ ao seu filho que pertence ao caminho de transferência. Se, ao contrário, o processo iniciador não receber um dos votos durante o tempo de espera, ou receber o voto NÃO, envia a decisão ABORTE a todos os seus filhos cujos votos foram diferentes de NÃO (inclusive ao filho pertencente ao caminho de transferência) e esquece a ação.

Ao receber a mensagem PREPARE$_t$, um processo P do caminho de transferência envia mensagens PREPARE a todos os seus filhos que não pertencem ao caminho de transferência e aguarda os votos deles. Se todos os votos forem SIM, P decide validar a ação se ele for um processo de entrada, ou envia mensagem PREPARE$_t$ ao seu filho pertencente ao caminho de transferência, caso P seja um processo não confiável. Se, ao contrário, P recebe um voto NÃO, ou não recebe um dos votos, ele envia a decisão ABORTE a todos os seus filhos cujos votos foram diferentes de NÃO e apaga de sua memória volátil as informações sobre a ação.

A alteração da primeira fase do O2PCg descrita acima não considera os votos de leitura apenas. A seguir, são descritas extensões necessárias no protocolo para torná-lo mais eficiente em ações de leitura apenas.

Se o processo iniciador apenas lê os dados estáveis locais e todos os votos recebidos forem votos LEITURA, o processo iniciador envia a mensagem LEITURA-PREPARE$_t$ ao seu filho que pertence ao caminho de transferência. Essa mensagem indica ao destinatário que os descendentes do remetente apenas lêem dados e, portanto, aqueles processos não necessitam participar da segunda fase do protocolo.

Um processo não confiável P do caminho de transferência envia a mensagem LEITURA-PREPARE$_t$ ao seu sucessor no caminho de transferência, somente se P for um processo que apenas lê dados e recebe somente votos de leitura de seus filhos. Se um processo de entrada recebe uma mensagem LEITURA-APENAS$_t$ e ele também é um processo de leitura apenas, e os

votos de seus descendentes forem todos LEITURA, o processo de entrada simplesmente apaga de sua memória volátil as informações sobre a ação. Não há segunda fase do protocolo nesse caso.

No protocolo O2PCg, a mensagem PREPARE contém o identificador do coordenador do protocolo. Logo, o processo iniciador deve saber quem será o coordenador e qual o caminho de transferência antes de enviar mensagens PREPARE aos seus filhos. É necessário, portanto, que o processo iniciador da ação conheça o identificador de cada processo de entrada e os caminhos que incluem cada um deles. De posse dessa informação, o processo iniciador pode decidir qual processo de entrada será o coordenador. Essa informação pode ser obtida de diversos modos. Uma alternativa seria obter essa informação durante a fase de execução da ação. Os identificadores dos processos podem ser enviados junto com os processos envolvidos na árvore de execução. Outra alternativa seria criar uma nova fase após a execução da ação. Antes de iniciar o protocolo O2PCg, o processo iniciador envia mensagens do tipo QUEM-PROCESSO-ENTRADA aos seus filhos. Essas mensagens se propagam por todos os nós intermediários até os nós folhas. As respostas contendo os identificadores dos processos de entrada viriam em sentido contrário. Cada processo intermediário acumularia as respostas recebidas de seus filhos e enviaria um conjunto de identificadores de processos de entrada ao seu pai. No final dessa nova fase, o processo iniciador teria o conhecimento de todos os caminhos de transferência possíveis.

Método de Transferência a um Processo Dedicado

No método de transferência a um processo dedicado, um processo é criado em um nó confiável para ser o coordenador do protocolo O2PCg. O mesmo processo pode ser o coordenador de mais de uma ação.

Em [RP90] são citados dois métodos distintos em que a transferência pode ser feita a um processo dedicado. O primeiro deles, menos eficiente, consiste no seguinte: ao receber um pedido do programa de aplicação para validar a ação, o processo iniciador transfere explicitamente a função de coordenador ao processo dedicado. Essa transferência é feita explicitamente antes da execução do protocolo O2PCg. O processo iniciador envia uma mensagem TRANSFERÊNCIA ao processo dedicado e esse passa a executar o protocolo O2PCg como coordenador. A árvore de terminação é formada pela árvore de execução acrescida do processo dedicado que se torna a raiz.

No segundo método, os autores sugerem que a transferência seja feita implicitamente durante a execução do O2PCg. O processo iniciador da execução também é o processo iniciador do protocolo. Ele envia mensagens PREPARE aos seus filhos e aguarda deles os seus votos. Ao receber todos os votos, o processo iniciador obtém o seu próprio voto como ocorre no protocolo PA. A transferência é feita através do envio do voto do processo iniciador ao processo dedicado, porém essa transferência só ocorre se o voto for SIM. Se o voto for NÃO, o processo iniciador envia uma mensagem ABORTE a seus filhos e esquece a ação. Se o voto for LEITURA, o processo iniciador simplesmente esquece a ação. Esse método de transferência

implícita é mais eficiente que o anterior, pois economiza mensagens entre o processo iniciador e o processo dedicado.

Correção do Protocolo

Pode-se verificar a correção do protocolo O2PC em duas etapas. Inicialmente, é verificada a correção do protocolo genérico O2PCg. Posteriormente, analisa-se a correção dos métodos de transferência da função de coordenador.

Durante uma execução sem falhas do protocolo O2PCg, seu funcionamento se mantém semelhante a do protocolo PA, exceto pelo envio de mais informações junto com os votos e com a ordem PREPARE, e pelo fato de que o envio dos sinais RECEBI serem enviados por um nó intermediário somente após ele ter também recebido esse sinal de seus filhos. Logo, durante uma execução sem falhas, a justificativa de correção do protocolo são as mesmas aplicadas ao protocolo PA.

São analisadas a seguir as diversas situações em que um nó detecta a falha de outro:

1. o coordenador falha antes de emitir as mensagens PREPARE;
2. um nó subordinado falha antes de emitir o seu voto;
3. um subordinado em estado PREPARADO falha antes de receber a decisão;
4. o coordenador falha antes de emitir a decisão.

Na situação (1), o procedimento adotado pelos filhos do coordenador ao detectarem a sua falha é semelhante ao adotado pelo protocolo PA e consiste em abortar a ação. O coordenador, ao se recuperar, também a aborta. Portanto, nesse caso, todos abortam a ação.

Na situação (2) o nó superior, ao detectar a falha do nó filho, aborta a ação e envia o voto NÃO ao seu pai (se for um subordinado) e a decisão aborte a todos os seus filhos. Logo, todos os nós que finalizam a ação a abortam.

Na situação (3), os nós em estado PREPARADO ficam bloqueados enquanto o coordenador não se recuperar, porém, como o nó coordenador é um nó confiável, espera-se que em algum instante ele se recupere e finalize a ação.

Na situação (4), de acordo com a alteração (A6) do protocolo PA, o coordenador, ao encontrar um registro VALIDAÇÃO para o qual não há registro FIM correspondente, permanece enviando mensagens VALIDE aos processos de entrada enquanto estes não enviarem os sinais RECEBI (o processo coordenador conhece esses identificadores graças a alteração (A3)). Ao obter todos os sinais RECEBI dos processos de entrada, o coordenador ativa o processo arquivo caso não tenha recebido o sinal de um de seus processos filhos não-confiáveis. Uma vez que cada processo subordinado envia o sinal RECEBI somente após forçar a gravação

do registro de validação e após receber os sinais RECEBI de seus filhos, tem-se a garantia que o coordenador não receberá mais nenhuma mensagem PERGUNTANDO de um processo de entrada. Além disso, o processo arquivo é ativado sempre que o coordenador receber uma mensagem PERGUNTANDO de um processo não confiável e não encontrar nenhuma informação na memória volátil para a ação identificada na mensagem. Logo, não há risco do processo coordenador responder ABORTE a uma mensagem PERGUNTANDO correspondente a uma ação que tenha sido validada. Os processos subordinados em estado PREPARADO permanecem enviando mensagens PERGUNTANDO diretamente ao coordenador. Eles possuem o identificador do coordenador que é recebido com a ordem PREPARE e incluído no registro PREPARADO. Conforme demonstrado anteriormente, a resposta vinda do coordenador é coerente com as finalizações anteriores. Logo, ao receber a decisão do coordenador, o processo subordinado finaliza a ação de modo consistente.

Fica, portanto, demonstrado que todos os nós que finalizam a ação o fazem de modo consistente. Além disso, como o coordenador é um nó confiável e como todos os nós em recuperação consultam o coordenador, nenhum nó fica em bloqueio infinito.

Foi demonstrada a correção do protocolo genérico O2PCg. A demonstração completa da correção do protocolo O2PC inclui, ainda, a demonstração da correção dos métodos de transferência.

O objetivo dos métodos de transferência da função de coordenador é o de garantir que o coordenador seja um nó confiável. Para que esses métodos estejam corretos, é necessário que eles garantam que se ocorrer uma falha de comunicação ou de nó ao longo do caminho de transferência, os subordinados em estado PREPARADO não fiquem em situação de bloqueio infinito. A seguir, é demonstrado que essa situação não ocorre em nenhum dos três métodos apresentados.

Correção do Método de Transferência Explícita

Se o protocolo O2PC for obtido a partir da aplicação desse método, pode-se garantir que os subordinados em estado PREPARADO não ficam em bloqueio infinito. Essa afirmação se deve ao fato de que a transferência da função de coordenador é feita em uma fase extra que antecede a fase de execução do protocolo O2PCg em que os processos entram em estado PREPARADO. Portanto, durante essa fase não há nenhum processo em estado PREPARADO. Se ocorre uma falha durante a transferência que impeça que a função de coordenador seja entregue ao processo de entrada, todos os nós irão abortar a ação, pois irão detectar término do tempo de espera pela ordem PREPARE. Logo, o protocolo O2PC que utiliza o método de transferência explicita está totalmente correto.

Correção do Método de Transferência Implícita

Nesse método, o processo iniciador da ação ou um processo intermediário não confiável entra em estado PREPARADO após receber os votos SIM de seus filhos não-confiáveis que não pertencem ao caminho de transferência. Após entrar em estado PREPARADO, esse processo envia a mensagem PREPARE_i ao seu filho pertencente ao caminho de transferência.

O envio da mensagem PREPARE_i corresponde à transferência da função de coordenador. Quando um processo de entrada recebe a mensagem ele se torna o novo coordenador. A mensagem PREPARE, enviada a todos os subordinados não pertencentes ao caminho de transferência, inclui o identificador do processo de entrada que deve ser o novo coordenador. Se a função de coordenador chegar a ser transferida a esse processo de entrada, ele se torna o novo coordenador e passa a executar o protocolo O2PCg.

Se ocorrer uma falha durante a transferência e as mensagens não chegarem ao processo de entrada, os processos em estado PREPARADO não ficam em bloqueio definitivo, pois possuem o identificador do processo de entrada que deveria receber a função de coordenador. Esses processos em estado PREPARADO enviam mensagens PERGUNTANDO ao referido processo de entrada que aborta a ação e envia mensagens ABORTE como resposta. Desse modo fica demonstrado que não há risco dos processos subordinados em estado PREPARADO ficarem em bloqueio infinito.

Correção do Método de Transferência a Um Processo Subordinado

Conforme descrito anteriormente, a transferência da função de coordenador a um processo dedicado pode ser feita de dois modos: modo implícito e modo explícito. No primeiro, o processo iniciador do protocolo transfere explicitamente a função de coordenador através da mensagem TRANSFERÊNCIA. Logo, a demonstração da correção nesse caso é semelhante a do método de transferência explícita.

No modo implícito, a função de coordenador é transferida juntamente com o voto SIM do processo iniciador. Se não ocorrer nenhuma falha durante a transferência, o processo dedicado passa a executar o protocolo genérico O2PCg. Se entretanto, ocorrer uma falha durante a transferência, os processos subordinados, ao detectarem término do tempo de espera, enviam periodicamente mensagens PERGUNTANDO ao processo dedicado que seria o coordenador (os processos subordinados são informados do identificador do processo dedicado através da mensagem PREPARE). O processo dedicado, ao receber uma mensagem PERGUNTANDO, envia a decisão ABORTE como resposta por não receber a função de coordenador.

Logo, em ambos os modos de transferência, garante-se que não ocorre o bloqueio infinito dos subordinados em estado PREPARADO. Fica demonstrada, assim, a correção do protocolo O2PC.

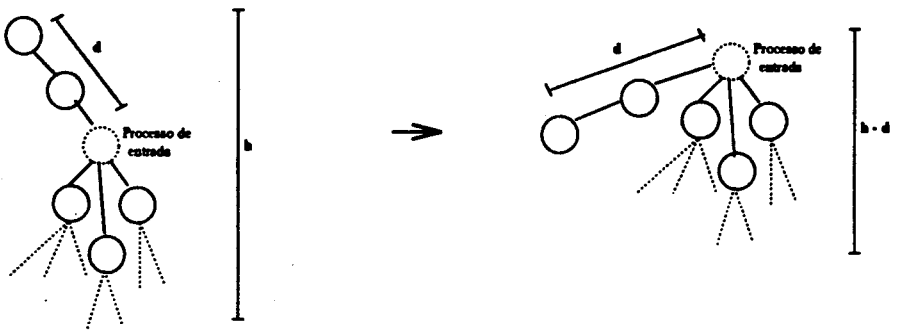

Figura 4.1: Estrutura da árvore de processos correspondente ao tempo mínimo de execução do O2PCe.

Tempo de Execução

O tempo de execução do protocolo O2PC depende do método de transferência utilizado. Serão adotadas as siglas: O2PCe, O2PCi e O2PCd, para significar execuções do protocolo que utilizam respectivamente os métodos de transferência externa, interna e a um processo dedicado.

O cálculo do tempo de execução do protocolo O2PC depende, ainda, de duas variáveis: h e d, onde d corresponde ao número de arestas do caminho de transferência da árvore de execução e h corresponde à altura da árvore de execução. Nas variações O2PCe e O2PCi, o tempo de execução depende também da árvore de execução. Para esses protocolos é calculado o tempo mínimo e o tempo máximo de execução.

Convém lembrar que, de acordo com a Definição 2.3.2, o tempo de execução corresponde ao intervalo entre o início de execução do protocolo até o instante em que o processo que o iniciou possa remover de sua memória volátil as informações sobre a ação. Nos protocolos descritos anteriormente, tal processo era o coordenador do protocolo. Conforme visto nesta seção, no protocolo O2PC, o iniciador da ação e o coordenador do protocolo não necessitam ser o mesmo processo.

A seguir, são descritos os cálculos dos tempos de execução desses protocolos.

O2PCe

Para o protocolo O2PCe, o tempo de execução mínimo é obtido se o valor da altura (h_t) da árvore de terminação é o máximo entre d e $h - d$. O tempo mínimo é obtido em árvores de execução em que o caminho de transferência pertence ao maior caminho que liga o processo iniciador da ação (processo raiz) a um processo folha, conforme mostra a Figura 4.1.

O tempo mínimo de execução para uma ação que não é de leitura apenas, constitui-se

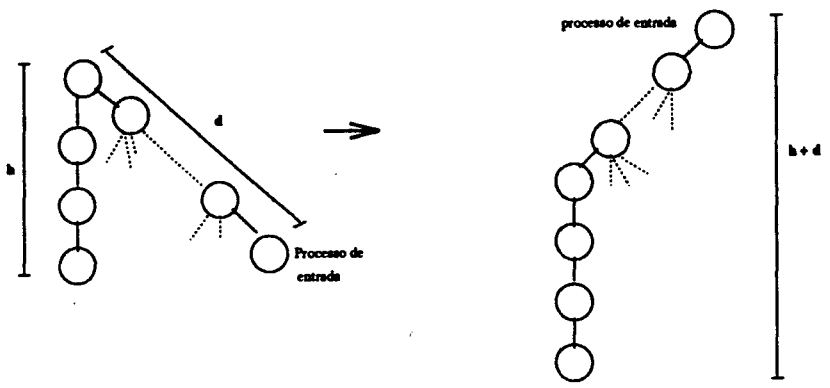

Figura 4.2: Estrutura da árvore de processos correspondente ao tempo máximo de execução do O2PCe.

de d unidades de tempo para o envio de mensagens de transferência, $max(d, h-d)$ unidades de tempo para o envio de mensagens PREPARE, $max(d, h-d)$ para o envio de mensagens de voto, $max(d, h-d)$ para envio de mensagens com a decisão e $max(d, h-d)$ com o envio de mensagens RECEBI. O tempo mínimo de execução do protocolo O2PCe é, portanto, $d + 4max(d, h-d)$ unidades de tempo.

Em uma ação de leitura apenas o cálculo do tempo mínimo é parcelado do seguinte modo: d unidades de tempo para o envio de mensagens de transferência, $max(d, h-d)$ mensagens PREPARE e $max(d, h-d)$ mensagens de voto, perfazendo um total de $d + max(d, h-d)$ unidades de tempo.

O tempo é máximo nos casos em que a altura (h_t) da árvore de terminação corresponde a $h + d$ e cada processo do caminho de transferência é raiz de uma subárvore a qual possui algum processo cuja distância ao processo raiz é h. A Figura 4.2 exemplifica esta situação. Nestas circunstâncias, para uma ação que altera dados, são gastas d unidades de tempo com mensagens de transferência, $h + d$ com mensagens PREPARE, $h + d$ com mensagens de voto, $h + d$ com mensagens contendo a decisão e h unidades de tempo com os sinais RECEBI. Ao todo, o tempo máximo gasto pelo protocolo O2PCe é $4d + 4h$.

Em ações de leitura apenas, o tempo máximo execução do protocolo é dado por d unidades de tempo com mensagens de transferência, $h + d$ com mensagens PREPARE e $h + d$ com mensagens de voto, perfazendo um total de $2h + 3d$ unidades de tempo.

O2PCi

No protocolo O2PCi, o tempo mínimo de execução ocorre quando cada processo não confiável do caminho de transferência possui apenas um descendente. Esta situação é mostrada na Figura 4.3.

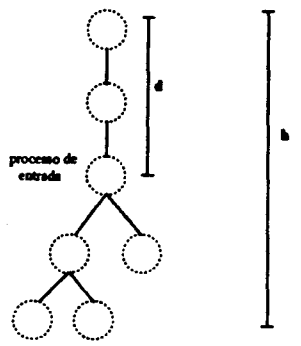

Figura 4.3: Estrutura da árvore de processos correspondente ao tempo mínimo de execução do O2PCi.

Em uma ação que altera dados, o tempo mínimo de execução do protocolo é calculado do seguinte modo: h unidades de tempo para o envio de mensagens PREPARE$_t$, $h - d$ para o envio de mensagens PREPARE, $h - d$ para mensagens com os votos e d unidades de tempo para as mensagens com a decisão chegarem ao processo iniciador da ação. Portanto, o tempo máximo corresponde a $2h$ unidades de tempo.

Em uma ação de leitura apenas, o tempo mínimo do protocolo é calculado de forma semelhante ao caso de uma ação que altera dados, exceto que não há o envio de mensagens com a decisão. Portanto, o tempo mínimo de execução para ações de leitura apenas é $2h - d$.

O tempo máximo de execução do protocolo ocorre quando a altura (h_t) da árvore de terminação é $h + d$ e cada processo do caminho de transferência é raiz de uma subárvore que possui algum processo cuja distância ao processo raiz da árvore de execução é h. Esta situação é idêntica àquela ilustrada na Figura 4.2.

No caso de uma ação que não é de leitura apenas, o tempo máximo de execução do protocolo é calculado do seguinte modo: o envio de mensagens PREPARE leva $h + (h-1) + (h-2) + \cdots + (h-d)$ unidades de tempo. O envio de mensagens contendo os votos consome $h + (h-1) + (h-2) + \cdots (h-d)$ unidades de tempo. o envio de mensagens PREPARE$_t$ ao longo do caminho de transferência consome d unidades de tempo. O envio de mensagens com a decisão consome $h + d$ unidades de tempo. O tempo gasto para o envio de mensagens RECEBI é h unidades. O tempo máximo total de execução do protocolo é $2hd + 4h - d^2 + d$ unidades de tempo.

Em uma ação de leitura apenas, o cálculo do tempo máximo é semelhante ao descrito anteriormente, exceto que não há o envio de mensagens com a decisão e de mensagens RECEBI. Nesse caso, o tempo máximo total é de $2hd + 2h - d^2$ unidades de tempo.

O2PCd

O tempo de execução do protocolo O2PCd para uma ação que altera dados é $4h + 2$ que corresponde a h unidades de tempo para o envio de mensagens PREPARE, h unidades de tempo para o envio dos votos, 1 unidade de tempo para o envio da mensagem com o voto do processo iniciador, 1 unidade de tempo para o envio da decisão ao processo iniciador, h unidades de tempo para o envio de mensagens com a decisão pelo processo dedicado aos subordinados, e h unidades de tempo para o envio de mensagens recebi ao processo dedicado.

No caso de ações de leitura apenas, o tempo de execução do protocolo é $2h$ que corresponde a h unidades de tempo para o envio de mensagens PREPARE pelo processo iniciador do protocolo e h unidades de tempo para o envio dos votos pelos processos subordinados.

Número de Mensagens

O número de mensagens enviadas durante o protocolo O2PC depende do método de transferência utilizado. A seguir é descrito para cada método de transferência, o número de mensagens em termos do número N de processos envolvidos na ação e em termos do número d de processos que constituem o caminho de transferência. São utilizados os termos: O2PCe, O2PCi e O2PCd para significar protocolos O2PC que utilizam respectivamente os métodos de transferência: explícita, implícita e a um processo dedicado. Será considerado apenas o modo implícito do protocolo O2PCd, por ser mais eficiente que o modo explícito.

O2PCe

O número máximo de mensagens do protocolo O2PCe é $4N - 4 + d$, que corresponde a d mensagens TRANSFERÊNCIA, $(N - 1)$ mensagens PREPARE, $(N - 1)$ mensagens de voto, $(N - 1)$ mensagens com a decisão e $(N - 1)$ sinais RECEBI.

O2PCi

O número máximo do protocolo O2PCi é $4N - 4 - d$ que corresponde a $(N - 1)$ mensagens PREPARE e PREPARE$_t$, $(N - 1 - d)$ mensagens de voto, pois os votos dos processos do caminho de transferência estão implícitos nas mensagens PREPARE$_t$, $(N - 1)$ mensagens com a decisão e $(N - 1)$ sinais RECEBI.

O2PCd

O número máximo do protocolo O2PCd é $4N - 1$ que corresponde a $(N - 1)$ mensagens PREPARE, $(N - 1)$ mensagens de voto 1 mensagem de voto do processo iniciador, 1 uma mensagem com a decisão enviada ao processo iniciador, $(N - 1)$ mensagens com a decisão

enviadas aos subordinados, 1 mensagem RECEBI enviada ao processo iniciador e $(N-1)$ mensagens RECEBI enviadas aos processos subordinados.

Número de Gravações no Log

O protocolo O2PC executa o mesmo número de gravações que o protocolo PA em execuções normais. Há, entretanto, uma gravação adicional executada quando o processo arquivo é ativado.

4.2 Protocolo Distribuído de Duas Fases

No Protocolo Distribuído de Duas Fases, cada processo envolvido no protocolo executa o mesmo algoritmo, pois o controle do protocolo fica igualmente distribuído por todos os processos.

Para a utilização do protocolo considera-se que o sistema gerenciador de ações atômicas mantém duas tabelas que são utilizadas pelo protocolo. Uma das tabela, a *tabela volátil de ação* (TVA) é mantida na memória volátil do nó. Esta tabela esta dividida em duas regiões. Na primeira, são armazenadas informações sobre os processos locais envolvidos na ação. A segunda região contém informações sobre os processos remotos participantes da ação, conhecidos pelo gerenciador local. Cada entrada da primeira região da TVA contém os seguintes campos referentes a um processo local: identificador do processo, seu estado (DESCONHECIDO, PREPARADO) e um vetor de identificadores que contém o identificador do processo que o ativou e dos processos por ele ativado, ou seja, os processos diretamente ligados a ele.

Uma entrada da segunda região da TVA contém para cada processo remoto os seguintes campos: o identificador do processo e o estado do processo.

A outra tabela mantida pelo subsistema gerenciador de ações atômicas contém os mesmos dados da primeira região da TVA. Esta tabela é denominada *Tabela Estável de Ação* (TEA) e é mantida na memória estável do nó.

Descrição do Protocolo

Ao receber um pedido do processo de aplicação para validar a ação, o sistema gerenciador de ações atômicas cria um identificador para a transação e aloca espaço para as tabelas TVA e TEA, insere em TVA o identificador do processo de aplicação e modifica o campo de estado para ativo nessa tabela.

Ao receber um pedido de execução de uma operação, um nó cria um processo P que irá executar a operação, aloca espaço para as tabelas TVA e TEA, se já não tiver sido alocado anteriormente e insere a entrada: $< P$,DESCONHECIDO,$- >$ na TVA local.

Ao terminar a execução da operação o processo criado para executá-la entra em estado PREPARADO através da execução da primitiva *prepare-validação* do gerenciador local de ações. Esta primitiva tem como parâmetros de entrada o identificador da ação, o identificador do processo e a lista de processos participantes da ação diretamente ligados a ele.

A primitiva prepare-validação quando ativada por um processo P executa as seguintes tarefas:

a. atualiza em TVA o estado de P para PREPARADO;

b. se todos os processos locais na TVA estiverem em estado PREPARADO, copia as entradas da TVA correspondentes a esses processos na TEA;

c. cria uma entrada $< P_i$, DESCONHECIDO,– $>$ na segunda região da TVA para todos os processos participantes remotos P_i ligados a P e não conhecidos anteriormente (pai, ou filhos de P que executam em outros nós);

d. se todas as entradas em TVA estão em estado PREPARADO então a primitiva:

   d1. valida a ação localmente;

   d2. envia mensagens PREPARADO(ia, processos-ligados-a-P) todos os processos envolvidos na ação.

e. se ainda houver alguma entrada cujo processo está em estado ativo, então, a primitiva executa o seguinte:

   e1. copia a entrada da TVA correspondente a P em TEA;

   e2. envia mensagens PREPARADO(ia, P, lista-de-processos-ligados-a-P) a todos os nós conhecidos e que estão envolvidos na ação; e

   e3. para cada processo local P_i, em estado PREPARADO, envia mensagens PREPARADO(ia, P_i,lista-de-processos-ligados-a-P_i) aos nós que ainda não eram conhecidos.

O envio das mensagens PREPARADO é utilizado para informar aos demais nós conhecidos pela primitiva os processos por ela conhecidos (locais ou não) que estão preparados para validar. O conhecimento desses processos varia ao longo do tempo à medida que as mensagens são enviadas entre os nós. Por esse motivo, sempre que um processo entra em estado PREPARADO, em um nó N, é necessário informar aos nós envolvidos na ação que ainda não eram conhecidos por N sobre todos os processos que N conhecia e que estão PREPARADOS.

Nesse protocolo, seguindo a otimização proposta por Anciloti, quando um processo entra estado PREPARADO, ele não necessita gravar nenhum registro PREPARADO correspondente no log. Na verdade, apenas o campo de estado da TVA correspondente a esse processo é modificado para PREPARADO. A indicação de que os processos de um nó estão em estado

PREPARADO só é gravada em memória estável quando todos os processos locais criados no nó entram em estado PREPARADO. Portanto, essa gravação pode se dar mais de uma vez para uma dada ação, pois um novo processo pode ser criado localmente em favor da ação. Esse processo, está inicialmente em estado DESCONHECIDO. Quando ele entrar em estado PREPARADO e como os demais processos já se encontram nesse estado, uma nova gravação da TEA é feita no log.

Um processo que pretende abortar a ação executa a primitiva ABORTE que faz com que o subsistema gerenciador de ações atômicas locais execute os seguintes passos:

1. desfaz as alterações causadas pela ação nos dados locais;
2. envia mensagens ABORTE contendo o identificador da ação a todos os nós;
3. apaga de sua memória volátil as informações sobre a ação.

Quando o gerenciador de ações atômicas de um nó envolvido na ação recebe uma mensagem PREPARADO, ele executa os seguintes passos:

1. modifica em TVA o estado do processo cujo identificador está incluso na mensagem para PREPARADO;
2. cria nova entrada na tabela TVA para cada processo participante não conhecido anteriormente e cujo identificador está na incluso na mensagem PREPARADO recebida. Atribui o estado DESCONHECIDO a cada um destes processos incluídos na TVA;
3. se todos os processos da TVA estiverem com estado PREPARADO, efetiva as alterações feitas pela ação nos dados estáveis locais e grava um registro VALIDADO;
4. se nem todos os processos estiverem em estado PREPARADO, para cada processo P_i em estado PREPARADO envia mensagens PREPARADO(ia,P_i,lista-de-processos-ligados-a-P_i) aos nós envolvidos na ação e não conhecidos anteriormente.

Ao receber uma mensagem ABORTE contendo o identificador de uma ação, um nó aborta a ação localmente e apaga de sua memória volátil as informações sobre ela, inclusive a TVA.

Tratamento de Falhas

Quando um nó se recupera de uma falha, um *processo recuperador* é criado para recuperar informações do log e as tabelas TEA das ações em execução no momento em que ocorreu a falha.

Se um nó em recuperação recebe uma mensagem referente a uma ação sobre a qual ele não possui nenhuma informação, ele aborta a ação.

Quando o processo recuperador encontra uma TEA para uma determinada ação, ele verifica para cada entrada correspondente a um processo local a P a existência de processos locais em estado DESCONHECIDO ligados a P. O processo recuperador descobre que cada processo P_1 local estava executando quando o identificador de P_1 aparece na lista de processos ligados a um processo P_2 em estado PREPARADO, mas não há entrada na TEA correspondente a P_1. Se existir algum processo que estava em estado desconhecido no momento da falha, o processo recuperador aborta a ação.

Se uma TEA é encontrada contendo todos os processos locais registrados em estado PREPARADO, o processo recuperador cria uma nova TVA, copia os dados da TEA para a TVA e insere na TVA uma nova entrada para cada processo remoto obtido da TEA. Os estados destes processos remotos são iniciados com o valor DESCONHECIDO na TVA. A seguir, o processo recuperador coloca em funcionamento um relógio e envia uma mensagem especial: TEMPO-ESPERA a todos os nós conhecidos. Essa mensagem contém o identificador da ação, e uma lista contendo os identificadores dos processos remotos que estão ligados diretamente a processos locais ao nó em recuperação.

O processo recuperador permanece aguardando mensagens dos nós aos quais ele enviou a mensagem TEMPO-ESPERA. Se não obtiver uma resposta de um dos nós durante o tempo de espera, o processo recuperador reinicia o relógio e envia novamente a mensagem TEMPO-ESPERA a esse nó. O relógio também é reiniciado sempre que o processo recuperador recebe uma mensagem PREPARADO.

Ao receber uma mensagem TEMPO-ESPERA, um nó comporta-se do seguinte modo:

1. envia uma mensagem VALIDE como resposta, se a ação tiver sido validada ou;
2. envia uma mensagem ABORTE como resposta, se a ação tive sido abortada ou;
3. envia uma mensagem PREPARADO para cada processo local envolvido na ação que esteja em estado PREPARADO e cujos identificadores estejam na mensagem TEMPO-ESPERA recebida.

Quando um nó se recupera de uma falha e encontra todos os processos locais em estado PREPARADO, não significa que havia somente esses processos executando no momento da falha. Um processo local pode ter sido criado por outro processo remoto. Nesse caso, tal processo local podia estar executando quando o nó falhou. Esse processo não está registrado na TEA, pois não estava em estado PREPARADO e não estava ligado a nenhum processo local. Na verdade, quando o nó se recupera ele não tem como saber, de imediato, sobre a existência de tal processo. Durante a fase de sua recuperação, o nó recebe mensagens PREPARADO do nó que havia criado esse processo que estava em execução no momento da falha. Somente ao obter tal mensagem, o nó em recuperação descobre a existência desse processo e decide a abortar a ação. Os demais nós são informados sobre a decisão e também a abortam.

Esse protocolo não possui mecanismos para evitar acúmulos de informação sobre as ações, tais como, mensagens RECEBI utilizadas nos demais protocolos, as quais permitem a um nó

apagar informações sobre determinada ação finalizada. O protocolo assume a existência de um mecanismo de coleta de lixo para esta finalidade.

Correção do Protocolo

Inicialmente, será demonstrada a correção do protocolo durante execuções em que não ocorrem falhas.

Quando um nó envolvido na ação a aborta, os demais nós envolvidos também a abortam. Um nó, ao abortar uma ação, apaga de sua memória volátil todas as informações sobre a ação e envia mensagens ABORTE a todos os nós do sistema. Logo, todos os nós envolvidos abortam a ação.

Será demonstrado que se um nó valida a ação, então todos os nós que a finalizam também a validam. A demonstração é feita por contradição. Seja um nó N em cuja tabela TVA todos os processos estão em estado PREPARADO. Nessa circunstância N valida a ação de acordo com o protocolo. Considere que haja algum processo P_j participante da ação e que não esteja incluído na TVA de N. Essa suposição poderia dar origem a uma inconsistência, pois N validaria a ação de modo independente e os demais nós poderiam abortar ou validar a ação, de acordo com a decisão de P_j. Porém, se P_j participa da ação, ele está diretamente conectado a algum outro processo P_k (que pode ser pai ou filho de P_j). Se o identificador de P_k pertence à tabela TVA de N, é porque P_k está em estado PREPARADO. O estado de P_k torna-se PREPARADO em N quando P_j executa a primitiva prepare-validação (caso P_k seja local a N) ou quando N recebe uma mensagem PREPARADO para o processo P_k (caso P_k) seja remoto. Em ambos os casos, os identificadores dos processos diretamente conectados a P_k são passados como parâmetros; mas está sendo considerado que N não conhece P_j, logo P_k não pode estar em estado PREPARADO. Porém todos os processos da TVA de N estão em estado PREPARADO. Logo, conclui-se que o identificador de P_k também não pode estar na TVA de N. De modo análogo pode-se concluir que todos os processos ligados a P_k, e por conseguinte todos os processos envolvidos na ação também não podem pertencer a TVA de N, o que é obviamente uma contradição.

Será considerada, a seguir, a correção do protocolo em situações em que podem ocorrer falhas.

Quando um nó X aguarda o recebimento de mensagens dos demais nós ele pode detectar término do tempo de espera por essas mensagens. X pode proceder de dois modos distintos conforme os estados de seus processos locais:

- se todos os processos locais estiverem em estado PREPARADO, X periodicamente envia mensagens TEMPO-ESPERA a todos os nós que ele conhece. Nesse caso, X fica bloqueado enquanto não receber mensagens dos demais nós que lhe permitam prosseguir a execução do protocolo;
- se houver algum processo local em estado ATIVO, X aborta a ação localmente. Não há

risco dessa decisão ser inconsistente, pois os demais nós ou não possuem o identificador desse processo ativo em suas TVAs ou possuem mas ele está em estado ATIVO. Logo, nenhum outro nó outro irá validar a ação.

Portanto, quando um nó detecta a falha de outro nó, ou a falha na transmissão de uma mensagem, se ele finaliza a ação, ele o faz de modo consistente com os demais nós.

Quando um nó Y se recupera de uma falha, conforme mostrado na Seção 4.2, se ele não possui TEA para uma dada ação, ele irá abortar a ação. Deve-se demonstrar que nesse caso nenhum dos demais nós envolvidos na ação podem tê-la validado. Um nó valida a ação somente quando possui na sua TVA todos os processos envolvidos na ação e eles estão em estado PREPARADO. Porém, nenhum outro nó pode estar com sua TVA nessa situação, pois, nesse caso, todos os processos locais a Y estariam no estado PREPARADO e portanto haveria uma TEA para a ação no nó Y uma vez que um nó grava dados na TEA sempre que todos os seus processos locais ficam todos em estado PREPARADO. Portanto, os demais nós devem estar executando a ação, ou alguns de seus processos locais estão em estado PREPARADO. Cada um desses nós irá abortar a ação por receber de Y a mensagem ABORTE em resposta a uma mensagem TEMPO-ESPERA, que o nó possa ter enviado caso ele também esteja se recuperando, ou por detectar término do tempo de espera por uma resposta a uma mensagem PREPARADO que ele possa ter enviado a Y. Portanto, nesse caso, todos os nós que finalizam a ação a abortam.

Se o processo recuperador do nó Y encontra a TEA para uma dada ação com todos os processos preparados, ele periodicamente envia a mensagem TEMPO-ESPERA a todos os nós que ele conhecia no instante da falha. Se os demais nós tiverem finalizado a ação, eles respondem às mensagens TEMPO-ESPERA a decisão correspondente à finalização executada, ou seja, ABORTE ou VALIDE. Se os demais nós não haviam finalizado a ação quando Y se recupera, Y continua a execução do protocolo, enviando mensagens TEMPO-ESPERA e atualizando as suas tabelas locais (TVA e TEA) sempre que receber mensagens PREPARADO como respostas às mensagens TEMPO-ESPERA. Nesse caso, se Y finaliza a ação ele também o faz de acordo com a finalização dos demais nós.

Tempo de Execução

O protocolo distribuído de duas fases tem o menor tempo de execução em relação aos demais protocolos de duas fases. Todas as mensagens podem, pelo menos teoricamente, ser enviadas paralelamente em uma única unidade de tempo de execução do protocolo, considerando uma execução sem falhas em que a ação é validada.

Número de Mensagens

O número de mensagens enviadas pelo protocolo é $N(S-1)$ durante uma execução sem falhas em uma rede ponto-a-ponto para uma ação que é validada. N é o número de processos

envolvidos na ação e S é o número de nós em que esses processos são executados. Quando há somente um processo por nó, $S = N$ e o total de mensagens é $N(N-1)$. Obviamente, quanto maior o número de processos concorrentes por nó envolvidos na ação, menor é o número de mensagens trocadas entre os nós.

Se o protocolo é executado em uma rede de broadcast, então, o número de mensagens enviadas é S.

Número de Gravações no Log

Durante uma execução sem falhas do protocolo em que a ação é validada, cada nó força a gravação de um registro PREPARADO sempre que os seus processos locais ficam todos em estado PREPARADO, e força a gravação de um registro VALIDADO quando a ação é validada no nó. Portanto, o total de gravações em um nó é $tot = \sum_{i=1}^{s} w_i + s$, onde s é o número de nós envolvidos na ação e w_i é o número de vezes em que os processos locais do nó entram em estado PREPARADO.

4.3 Protocolo de Validação com Substitutos Para o Coordenador

Nesta seção é apresentado um protocolo de quatro fases derivado a partir dos protocolos de duas fases. Esse protocolo é utilizado como parte do sistema de banco de dados distribuídos SDD-1 [HS80]. O protocolo foi projetado para executar em uma rede particular denominada RelNet, a qual é um subsistema do SDD-1. A RelNet possui as seguintes facilidades além daquelas citadas na Seção 2.1:

1. ordenação das mensagens enviadas de um nó a outro;
2. um nó A pode, através de uma primitiva, ordenar a outro nó B para que se comporte como se estivesse falho. Quando isso ocorre, o nó B simula uma situação de falha, apagando de sua memória volátil todos dados e iniciando as rotinas de recuperação;
3. há um mecanismo que permite a um nó A ser avisado quando um nó B muda do estado funcionando para o estado falho, ou vice-versa.

Os processos criados por uma ação utilizam os bancos de dados distribuídos pelos nós da rede. Um processo pode ativar um ou mais processos locais ou remotos em favor da ação.

Descrição do Protocolo

O Protocolo de Validação com Substitutos para o Coordenador visa diminuir as situações de bloqueio do protocolo de duas fases original por meio de cópia do processo coordenador em

mais de um nó da rede. O protocolo estende o protocolo original de duas fases para quatro fases, as quais estão descritas a seguir:

Primeira Fase

O processo coordenador cria remotamente m processos que irão funcionar como seus substitutos caso ele falhe. Inicialmente, o estado desses substitutos é o estado de ABORTO. Sejam $S_1, S_2, \cdots, S_m$ estes processos, cada um dos quais ativados em nós distintos da rede. Ao criar o processo S_i, o coordenador o informa sobre os identificadores dos demais processos substitutos e dos processos subordinados que participam da ação.

Após requerer aos nós remotos a criação dos processos substitutos, o coordenador aguarda a chegada de mensagens desses nós indicando a criação dos processos. Se alguma mensagem não chegar durante o tempo de espera, o coordenador envia uma mensagem ao nó em atraso, forçando-o a se comportar como se realmente tivesse falhado.

Segunda Fase

A segunda fase corresponde à primeira fase do protocolo original de duas fases, ou seja, é quando o coordenador envia mensagens PREPARE aos subordinados e grava um registro ABORTADO ou VALIDADO.

Os subordinados também comportam-se como na primeira fase do protocolo original de duas fases.

Terceira Fase

O coordenador envia mensagens com a decisão a todos os substitutos e aguarda uma mensagem de cada um deles indicando o recebimento da decisão. Se o coordenador não receber a mensagem de algum substituto durante o tempo de espera, ele envia mensagem forçando esse substituto a se comportar como se tivesse falhado.

Cada substituto ao receber uma mensagem com a decisão VALIDE muda do estado de ABORTO para o estado de VALIDAÇÃO. A mudança de um estado para outro não requer nenhum tipo de gravação em memória estável; o substituto apenas grava em sua memória volátil uma indicação que está em um do dois estados. Quando um nó que continha processos substituto se recupera de uma falha, ele deixa de fazer parte do mecanismo de substituição.

Quarta Fase

A quarta fase corresponde à fase em que o coordenador envia mensagens contendo a decisão a todos os subordinados. O funcionamento do protocolo nessa fase é o mesmo do protocolo de duas fases tanto no nó coordenador como em cada um dos subordinados.

Os processos substitutos são ativados somente se ocorrer a falha do coordenador ou de outros substitutos. Cada substituto $i, 1 < i \leq m$ permanece observando o funcionamento do substituto $i-1$. Essa observação é feita através do mecanismo que permite a um nó ser avisado quando outro nó muda do estado funcionando para o estado falho. O substituto 1 observa o funcionamento do coordenador. Um processo substituto ao detectar a falha de seu predecessor, passa a observar o processo que o predecessor falho observava. Quando o coordenador falha, o substituto que o observa assume o controle do protocolo e finaliza a ação de acordo com o estado em que se encontra neste instante. O mecanismo de observação garante que somente um processo substitua o coordenador em um dado instante. A explicação para esta afirmação é a seguinte: seja A um substituto tal que A observa B que é o observador de um processo C (que pode ser o processo coordenador). Se A detecta a falha de B antes de passar a observar o processo C, A força a falha de B. Desse modo, fica garantido que cada processo seja observado por apenas um processo por vez.

O protocolo, como foi descrito acima funciona corretamente se o substituto do coordenador não falhar durante o envio da mensagem com a decisão. Se ele falha, há o risco do substituto que o observava estar em um estado diferente e, portanto, irá enviar uma decisão diferente. O risco dos substitutos estarem em estados diferentes ocorre porque o coordenador pode ter falhado durante a terceira fase, quando as mensagens com a decisão são enviadas aos substitutos. Assim, alguns substitutos podem ter recebido a decisão VALIDE e mudado para o estado de VALIDAÇÃO, enquanto outros não a receberão, permanecendo no estado inicial de ABORTO. Com o objetivo de evitar esse risco, quando um substituto assume o controle do protocolo, ele primeiro envia a decisão aos substitutos de identificadores mais altos, aguarda mensagens deles indicando o recebimento da decisão e só então inicia o envio da decisão aos subordinados. Desse modo o substituto do coordenador garante que se ele falhar os seus possíveis substitutos continuarão transmitindo a mesma decisão aos subordinados.

Tratamento de Falhas

Em [HS80], onde o protocolo é descrito não há referência sobre o procedimento de um nó ao se recuperar de uma falha. Entretanto, como o controle do protocolo pode estar com o coordenador ou com um dos substitutos, quando um subordinado em estado PREPARADO se recupera, ele deve enviar mensagens PERGUNTANDO ao coordenador e se este não responder, então, mensagens PERGUNTANDO devem ser enviadas sucessivamente aos substitutos até que uma decisão seja recebida.

Se o processo coordenador se recupera de uma falha e se ele encontra um registro VALIDADO no log, mas não encontra um registro de FIM para uma dada ação, então é necessário que ele consulte os substitutos, pois pode ocorrer que ele tenha falhado após gravar o registro VALIDADO, mas antes de enviar as mensagens com a decisão aos subordinados. O processo que o substituiu estava em estado de ABORTO e portanto esse processo deve ter enviado a decisão ABORTE aos subordinados. Neste caso, é necessário o coordenador desfazer os efeitos locais da ação. Portanto, o coordenador, nestas circunstâncias, não pode responder a

nenhuma mensagem PERGUNTANDO vinda de um subordinado enquanto não se certificar da decisão que os substitutos podem ter tomado.

Correção do Protocolo

A correção do protocolo é garantida, desde que não ocorra falha de partição da rede. Em caso de execução sem falhas do protocolo, o coordenador controla a execução do protocolo até o final. Portanto, nesse caso, a execução do protocolo é semelhante ao do protocolo de duas fases hierárquico e portanto os argumentos utilizados para sua correção são os mesmos. Quando o coordenador falha, os processos substitutos são unânimes quanto à decisão a ser tomada, ou seja todos que assumem o controle do protocolo tomam a mesma decisão, conforme foi mostrado acima. Quando o coordenador se recupera de uma falha, há apenas um caso em que ele pode ter tomado uma decisão divergente dos substitutos, que é a situação em que ele valida a ação, e falha antes de emitir a decisão aos substitutos. Conforme foi mostrado, nesse caso ele acata a decisão dos substitutos. Desse modo, os subordinados ao enviarem mensagens PERGUNTANDO ao coordenador e aos substitutos obterão a mesma resposta de todos eles.

Quando ocorre falha de partição da rede, entretanto, o protocolo pode causar finalizações inconsistentes. Seja a ocorrência de uma falha de partição da rede durante a terceira fase do protocolo e que a decisão do coordenador tenha sido a de validar a ação. O coordenador envia mensagens VALIDE a todos os substitutos, porém apenas aqueles que se encontram na mesma partição as recebem. Se o substituto de menor número estiver em uma outra partição, ele pode não receber a mensagem VALIDE[3]. Esse substituto irá observar falhas de todos os substitutos e inclusive a falha do coordenador, logo ele irá assumir o controle do protocolo naquela partição. Porém esse substituto ainda não tinha recebido a decisão VALIDE do coordenador antes da ocorrência de falha de partição, portanto ele está no estado de ABORTO. Como conseqüência ele irá enviar mensagens ABORTE a todos os subordinados da partição à qual ele pertence. Obviamente, os processos que estão em partições diferentes terminarão a ação de modo diferente, o que é uma inconsistência. Logo, apesar do protocolo tolerar mais situações de falhas do que o protocolo hierárquico, por exemplo, ele é inconsistente em caso de falha de partição da rede. Essa inconsistência não ocorre nos demais protocolos de duas fases estudados, pois na falta do coordenador, o protocolo simplesmente fica bloqueado.

Tempo de Execução

O valor h corresponde à altura da árvore de processos que participam da ação. O tempo total do protocolo em unidades de tempo é $4(h+1)$: 1 para o envio de mensagens ordenando

[3]Apesar da rede garantir a entrega da mensagem através da duplicação de "buffers", pode ocorrer que os nós onde estão esses "buffers" não estejam na mesma partição na qual se encontra o processo substituto em questão. Como conseqüência, a entrega da mensagem só é garantida quando não há falha de partição da rede.

a criação dos processos substitutos, 1 para o recebimento das mensagens avisando da criação dos processos, h para o envio de mensagens PREPARE aos subordinados, h para o envio dos votos dos subordinados ao coordenador, 1 para o envio da decisão do coordenador aos substitutos, 1 para as mensagens dos substitutos acusando o recebimento da decisão, h para o envio da decisão aos subordinados e h para o envio dos sinais RECEBI dos subordinados.

Número de Mensagens

Considerando M o número de processos substitutos e N o número de processos subordinados, o número de mensagens enviadas é $4M+4N-4$, que corresponde a M mensagens para criação dos processos substitutos, M mensagens de respostas indicando a criação desses processos, $N-1$ mensagens PREPARE aos processos envolvidos na ação, $N-1$ votos, M mensagens com a decisão enviadas aos substitutos, m mensagens dos substitutos indicando o recebimento da decisão, $N-1$ mensagens com a decisão aos subordinados envolvidos na ação e $N-1$ mensagens RECEBI enviadas pelos subordinados.

Número de Gravações no Log

O número de gravações em log do protocolo é o mesmo de um protocolo hierárquico em que a árvore de processos possui altura um. Portanto, o coordenador grava dois registros: o registro VALIDADO cuja gravação é forçada e o registro FIM que é gravado de modo assíncrono. Cada subordinado força a gravação dos registros PREPARADO e VALIDADO e grava de modo assíncrono o registro FIM.

4.4 Protocolo BPA - "Byzantine Presumed Abort"

Nesta seção é descrito o protocolo apresentado por Mohan, Strong e Finkelstein. [MSF83] que utiliza um protocolo de consenso tolerante a falhas bizantinas como parte da segunda fase. O BPA é resultado de alterações feitas no protocolo PA apresentado na Seção 3.7.

Descrição do Protocolo

O protocolo BPA a ser descrito nesta seção assume que o sistema subjacente seja constituído por uma rede de clusters. Um *cluster* é um conjunto de processadores ligados em rede, cujo objetivo é o de fornecer alta disponibilidade de dados. A conectividade e a taxa de transmissão entre os processadores de um mesmo cluster é considerada alta, quando comparadas com a conectividade e taxa de transmissão entre processos de clusters diferentes.

As ações são executadas segundo o modelo de árvore de processos, como nos protocolos anteriores, porém uma ação pode ter processos executando em processadores de diferentes

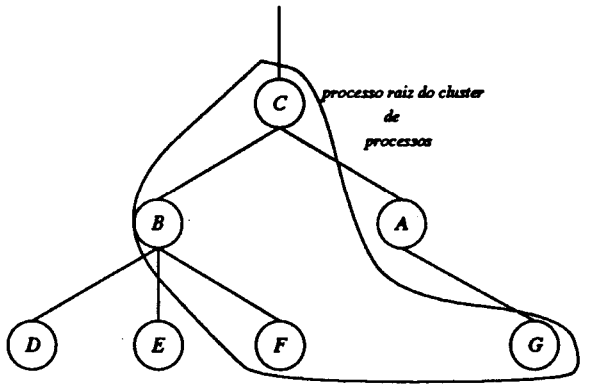

Figura 4.4: Exemplo de árvore de processos em que alguns dos processos pertencem a um cluster

clusters. O cluster que contém o processo raiz da árvore (ou coordenador) é denominado *cluster-raiz*. A figura 4.4 mostra um exemplo de árvore de processos em que o processo C é a raiz e os processos B, C, D, F e G pertencem ao cluster-raiz.

As redes internas aos clusters e a rede que os interliga garantem a entrega de mensagens, desde que não ocorra falha de partição. Considera-se a existência de um mecanismo capaz de detectar falha de partição de um cluster e de um subsistema que inclui um protocolo para obtenção de consenso entre os processadores de um cluster. Esse protocolo de consenso deve ser tolerante a falhas bizantinas. Esse subsistema deve manter e gerenciar um *log para consenso* que contém a história dos consensos obtidos. O protocolo para obtenção de consenso é utilizado apenas entre processadores de um mesmo cluster. Ele não é utilizado entre clusters distintos por dois motivos [MSF83]:

1. a conectividade entre processos de clusters distintos não é considerada alta suficiente para se obter tolerância satisfatória a falhas bizantinas;
2. o custo para obtenção de um consenso a nível do sistema como um todo seria muito alto;

Antes de iniciar a primeira fase do protocolo, o processo coordenador estima com base na altura da árvore de processos o tempo máximo de duração da primeira fase. A seguir, descreve-se o funcionamento do protocolo durante as duas fases.

Primeira Fase

Durante esta fase, o protocolo BPA é semelhante ao protocolo PA, exceto quanto ao tipo de informações passadas junto com a ordem PREPARE e com os votos. Junto com a ordem

PREPARE é enviado o valor máximo estimado para duração da primeira fase e junto com o voto SIM de um processo X é enviada uma lista de identificadores conhecidos por X de processos que executam no cluster-raiz. Cada processo subordinado obtém essa lista a partir da união das listas recebidas junto com os votos SIM de seus filhos. Se esse processo subordinado também pertence ao cluster-raiz, ele insere o seu próprio identificador na lista resultante. A nova lista é incluída na mensagem contendo o voto SIM a ser enviada ao processo pai do subordinado. Na figura 4.4, assumindo que todos os processos votam SIM, B envia junto com seu voto o seu identificador e o identificador de F. Desse modo, ao receber os votos de seus filhos, se eles forem todos SIM o coordenador tem conhecimento de todos os processos que pertencem ao seu cluster e que participam da ação.

Cada processo pertencente ao cluster-raiz que vota SIM ativa um relógio logo após emitir o seu voto. Esse relógio dispara sinal quando o tempo máximo estimado para obtenção do consenso expira. Se o consenso sobre a validação da ação é obtido antes do sinal do relógio, o processo valida a ação. Se o tempo expira, então, o processo aborta a ação.

Ao final da primeira fase, o coordenador decide quanto à finalização da ação com base nos votos recebidos. Se todos os votos forem SIM o coordenador decide validar a ação e inicia um protocolo para obtenção de consenso. Se, entretanto, a decisão é a de abortar a ação, o coordenador grava um registro de aborto mas não envia a decisão a nenhum processo que pertence ao seu cluster. Esses processos abortarão a ação quando os seus relógios internos dispararem. Portanto, somente os processos em estado PREPARADO que não pertencem ao cluster-raiz ficam bloqueados quando ocorrem falhas do seus processos pais. Se o coordenador decide validar a ação, ele comporta-se como um subordinado, forçando a gravação de um registro PREPARADO no log. O coordenador insere o seu identificador na lista de identificadores recebidos durante a primeira fase e inicia o protocolo para obtenção de consenso sobre a validação da ação. Esse protocolo envolve todos os nós do cluster-raiz, mesmo aqueles que não possuem processos participando da ação. Quando o protocolo de consenso termina e a decisão de validar a ação é obtida, o coordenador força a gravação de um registro VALIDADO no log. Cada nó do cluster-raiz informa aos seus processos que participaram da ação que a decisão de validá-la foi obtida. Ao ser informado sobre a decisão de validar a ação, cada processo do cluster-raiz força a gravação de um registro VALIDADO somente se tiver processos filhos que não pertençam ao esse cluster.

Segunda Fase

A segunda fase existe apenas se houver processos envolvidos na ação que executam em nós não pertencentes ao cluster-raiz. Nesse caso, a segunda fase é semelhante a do protocolo PA, exceto que ela inicia somente em cada processo do cluster-raiz que possui processos filhos não pertencentes a esse cluster. Assim, no exemplo da figura 4.4, se os processos A, D e E tivessem votado SIM durante a primeira fase, na segunda fase somente esses processos receberiam a mensagem com a decisão final. Os demais processos que participam do cluster-raiz são informados da decisão final pelo próprio protocolo de consenso.

Tratamento de Falhas

Em [MSF83] onde o protocolo BPA é descrito são dadas apenas algumas idéias sobre as operações a serem executadas durante a recuperação de um cluster: os dados alterados pelo cluster devem ser recuperados e os logs devem ser lidos para garantir que, se um dos nós tenha finalizado a ação, os demais também a finalizem de modo consistente.

Quando um nó se recupera de uma falha, se ele não pertence ao cluster-raiz correspondente a uma dada ação, ele executa as mesmas operações de recuperação executadas por um processo subordinado no protocolo PA. Se o nó pertence ao cluster-raiz, ao se recuperar de uma falha sua, ou de uma falha de partição do cluster em que ele pertencia à partição minoritária, ele copia o log de consenso de t nós e obtém o seu próprio log de consenso com base na união dos consensos desses t logs (t é o grau de confiabilidade do protocolo de consenso). Quando ocorre falha de partição do cluster, somente os nós que estão na partição majoritária continuam executando suas tarefas. Os nós da partição não majoritária, ao detectarem que estão nessa partição, suspendem suas operações com relação a transações distribuídas até o instante em que voltam a participar da partição majoritária. A partir desse momento, eles iniciam as medidas de recuperação apresentadas acima.

Correção do Protocolo

A correção do protocolo BPA deve ser analisada sob o aspecto dos tipos de falhas consideradas pelo protocolo. Dentro do cluster-raiz o protocolo deve tolerar qualquer tipo de falha.

Fora do cluster-raiz, considera-se que apenas falhas de parada possam ocorrer, visto que os nós externos ao cluster-raiz executam o protocolo PA.

A correção do protocolo pode ser demonstrada, portanto, com base nas seguintes afirmações:

a. dentro do cluster-raiz, o BPA tolera qualquer tipo de falha, incluindo as falhas bizantinas, dentro do grau de confiabilidade t do protocolo.

b. fora do cluster-raiz o BPA tolera qualquer número de falhas de omissão.

A demonstração de correção da primeira afirmação pode ser feita por contradição. Sejam dois processos p e q ambos situados no cluster-raiz. Suponha uma situação em que o processo p valida a ação e o processo q a aborta. No protocolo BPA, o processo q pode validar uma ação somente se:

1. q obtém o consenso para validação da ação dentro do cluster-raiz, ou;

2. o nó onde o processo q executa, ao se recuperar de uma falha, copia os logs de consensos dos demais processadores e os os conteúdos deles indicam a obtenção do consenso em validar a ação.

Uma propriedade de um protocolo de consenso tolerante a falhas bizantinas é que se o consenso aparece no log de um nó, ele aparece nos logs dos demais nós envolvidos no protocolo. No BPA, todos os nós do cluster-raiz são envolvidos no protocolo de consenso, portanto o log do nó onde executa o processo p também possui o consenso sobre a validação da ação. Além disso, o consenso é obtido antes que o relógio de p dispare. Portanto, p não pode ter abortado a ação.

Pode-se demonstrar que a afirmação (b) está correta pelo fato de que a comunicação entre os nós que estão fora do cluster-raiz e a comunicação deles com os nós dentro desse cluster se faz através do protocolo PA. Portanto, a correção da afirmação (b) se reduz à demonstração da correção do protocolo PA, a qual já foi apresentada na Seção 3.7.

Tempo de Execução

O tempo de execução varia de acordo com o número de subordinados que pertencem ao cluster-raiz. A seguir são apresentados os limites de tempo de execução para ações de leitura apenas e ações de escrita. T_c é o tempo de execução do protocolo de consenso, N é o número de processos envolvidos na ação e h é a altura da árvore de processos.

O tempo de execução de ações totalmente de leitura é $2h$ (h para mensagens PREPARE e h para as mensagens com o voto LEITURA). Nesse caso, não é necessário ativar o protocolo de consenso.

O tempo mínimo de execução de ações que alteram os dados é $4h$, o que corresponde à execução apenas do protocolo PA e ocorre quando não há processos subordinados no cluster-raiz. O tempo máximo de execução é de $2h + T_c$.

Número de Mensagens

O número de mensagens é calculado como a soma do número de mensagens do protocolo PA com o número de mensagens do protocolo de consenso utilizado. N_c é o número de processos envolvidos e que executam no cluster-raiz. N_c é o número de mensagens enviadas pelo protocolo de consenso e N é o número de processos envolvidos na execução da ação.

Em ações totalmente de leitura, o número de mensagens é $2N - 2$ ($N - 1$ mensagens PREPARE e $N - 1$ mensagens COM VOTOS LEITURA)

Nas ações em que todos os processos alteram dados, o número de mensagens é $4N - 2N - 2N_c + N_c$. Esse valor corresponde a $N - 1$ mensagens PREPARE, $N - 1$ mensagens de voto, $N - N_c$ mensagens com a decisão, $N - N_c$ sinais RECEBI e N_c mensagens do protocolo de consenso.

Número de Gravações no Log

O número de gravações no log é idêntico ao do protocolo PA, não contando, no entanto, as gravações feitas pelo protocolo de consenso.

Capítulo 5

Protocolos com Mais de Duas Fases

Neste capítulo, são apresentados outros protocolos de validação que não se incluem na categoria de protocolos de duas fases.

Será utilizada a padronização abaixo na descrição dos protocolos apresentados:

1. descrição do Protocolo;
2. tratamento de Falhas;
3. correção do Protocolo;
4. tempo de Execução;
5. número de Mensagens.

A primeira seção apresenta o protocolo de três fases [BVHNG87]. A seção 5.2 descreve o protocolo proposto por Duchamp [Duc88]. A seção 5.3 descreve o protocolo ODCP [YA91] e as duas últimas seções descrevem protocolos tolerantes a falhas derivados do protocolo ODCP.

5.1 Protocolo de Três Fases

Na Seção 2.3.4 foi mostrado que o protocolo de duas fases é bloqueante porque não satisfaz às condições do Teorema 2.3.1.

A seguir é apresentado o protocolo de Três Fases, que é um protocolo não-bloqueante se ocorrem apenas falhas de nós e essas falhas não impedem a comunicação entre dois nós operacionais.

Quando um subordinado detecta término do tempo de espera por uma mensagem do coordenador, ele ativa um protocolo de finalização caso esteja em um estado que não lhe permita decidir de modo independente. Através do protocolo de finalização, os subordinados interagem entre si na busca de uma decisão.

Descrição do Protocolo

Primeira Fase

O coordenador inicia o protocolo enviando mensagens PREPARE a todos os subordinados. Nessa mensagem são incluídos os identificadores de todos processo que participam da ação. A primeira fase do protocolo é semelhante à primeira fase do protocolo de duas fases, diferindo desse último apenas nos seguintes aspectos:

- as mensagens PREPARE contém os identificadores de todos os processos que participam da ação;
- se os votos obtidos pelo coordenador forem todos SIM, ele não decide validar de imediato a ação. Nesse caso o coordenador entra em um estado intermediário denominado PRÉ-VALIDADO através da gravação forçada em log de um registro PRÉ-VALIDADO.

Esse registro contém os identificadores da ação, do coordenador e dos subordinados.

Fase Intermediária

Para obter-se um protocolo não-bloqueante derivado do protocolo de duas fases é necessário que o protocolo derivado satisfaça as seguintes exigências:

a. quando um processo está em um estado final (ABORTADO ou VALIDADO), nenhum processo pode estar em estado final diferente;

b. quando um processo está no estado VALIDADO, os demais estão em estado VALIDÁVEL.

A exigência (b) pode ser satisfeita graças à inclusão do estado PRÉ-VALIDADO citado anteriormente, o qual corresponde a uma nova fase. Essa fase é intermediária entre a primeira e a segunda fase do protocolo de duas fases.

Após entrar em estado PRÉ-VALIDADO, o coordenador inicia a fase intermediária, enviando mensagens do tipo PRÉ-VALIDE a todos os subordinados. Ao obter uma mensagem PRÉ-VALIDE, cada subordinado entra em estado PRÉ-VALIDADO, forçando a gravação de um registro correspondente e envia como resposta a mensagem PRÉ-VALIDADO ao coordenador.

Ao obter todas as mensagens PRÉ-VALIDADO, o coordenador entra em estado VALIDADO, forçando a gravação do registro VALIDADO. Encerra-se, assim, a fase intermediária.

Pode-se observar que essa solução também satisfaz a exigência (a), pois se um processo está em estado VALIDADO, os demais estão em estados VALIDÁVEIS (i.e. estados VALIDADO ou PRÉ-VALIDADO). Logo, nenhum processo pode estar em estado ABORTADO.

Terceira Fase

A última fase do protocolo corresponde à segunda fase do protocolo de duas fases, em que o coordenador envia a decisão a todos os subordinados. Ao receber a decisão, cada subordinado envia o sinal RECEBI ao coordenador.

Tratamento de Falhas

Existem cinco situações em que um nó operacional pode detectar término do tempo de espera durante uma execução do protocolo:

a. quando o subordinado está aguardando mensagem PREPARE do coordenador. É o caso semelhante ao que ocorre nos protocolos de duas fases. Se detectar término do tempo de espera, o subordinado aborta a ação;

b. quando o coordenador está aguardando votos SIM (NÃO) dos subordinados. Nesse caso, também semelhante ao do protocolo de duas fases centralizado, o coordenador aborta a ação;

c. quando o subordinado está em estado PREPARADO aguardando a decisão ABORTE ou a mensagem PRÉ-VALIDE do coordenador. Nessa circunstância, o subordinado não pode tomar nenhuma decisão unilateral caso detecte a falha do coordenador. Portanto, um protocolo de finalização deve ser ativado;

d. o coordenador está aguardando mensagens PRÉ-VALIDADO dos subordinados. Nesse ponto, o coordenador já recebeu os votos SIM de todos subordinados, então, valida a ação e envia mensagens *valide* a todos os nós, mesmo após detectar termino de espera;

e. o subordinado está no estado PRÉ-VALIDADO, aguardando a mensagem valide. Nessa situação, o subordinado não pode validar a ação pois, pode haver um outro subordinado que está em estado PREPARADO. Essa é uma situação sutil, mas que pode ser explicada através do seguinte exemplo: suponha que o coordenador envie mensagem PRÉ-VALIDE a um subordinado X e falhe antes de enviá-la a outro subordinado Y. X está portanto, no estado PRÉ-VALIDADO (validável), enquanto Y permanece no estado PREPARADO (não-validável). Se X valida a ação localmente, o protocolo deixa de ser bloqueante, pois X muda para o estado de validação enquanto Y permanece em estado não-validável. Portanto, X só pode validar a ação quando tiver certeza de que os demais nós estão pelo menos em estado PRÉ-VALIDADO. Quando ocorre falha do coordenador nessa situação, o protocolo de finalização deve ser ativado pelo subordinado para que uma decisão possa ser obtida pelos subordinados operacionais.

A seguir, descreve-se o protocolo de finalização que permite que o protocolo de três fases ser não-bloqueante quando ocorrem falhas de nós apenas.

Protocolo de Finalização

O protocolo que é descrito a seguir baseia-se na eleição de um coordenador para continuar o processo de finalização. Uma vez eleito, o coordenador envia mensagens do tipo: QUAL-ESTADO aos demais processos que tornam-se seus subordinados. Cada subordinado responde uma mensagem correspondente ao tipo de estado em que se encontra (ABORTADO, VALIDADO, PRÉ-VALIDADO, OU PREPARADO). O coordenador ao obter as respostas dos processos subordinados, toma a decisão quanto ao término da ação com base nas seguintes regras:

Regra 5.1.1 *Se alguma resposta foi* ABORTADO, *o coordenador decide abortar a ação e envia mensagens* ABORTE *a todos os processos subordinados;*

Regra 5.1.2 *Se algum processo respondeu* VALIDADO, *o coordenador decide que a ação deve ser validada e envia mensagens* VALIDE *a todos subordinados;*

Regra 5.1.3 *Se todas mensagens recebidas foram* PREPARADO, *o coordenador aborta a ação enviando mensagem* ABORTE *a todos subordinados;*

Regra 5.1.4 *Se algum subordinado respondeu* PRÉ-VALIDADO, *mas nenhum respondeu* VALIDADO, *o coordenador não pode validar a ação de imediato pois, alguns outros subordinados podem estar em estado* PREPARADO. *O coordenador nesse caso, envia uma mensagem* PRÉ-VALIDE *a todos os processos que responderam* PREPARADO *e espera essas respostas,do tipo* PRÉ-VALIDADO. *Após receber essas respostas, o coordenador envia mensagens* VALIDE *a todos os subordinados. Caso um subordinado falhe antes de emitir a mensagem contendo o seu estado, o coordenador ignora o subordinado e baseia sua decisão nos estados dos subordinados operacionais. Se todos os subordinados estiverem falhos, o coordenador decide com base no seu próprio estado somente.*

A tolerância a falhas do protocolo é garantida do seguinte modo: se o coordenador falha quando os subordinados estiverem aguardando mensagens dele, qualquer subordinado que detecta a falha (por término de espera), ativa o algoritmo para eleição de um novo coordenador.

Um algoritmo de eleição pode ser obtido com base em uma seqüência dos endereços dos processos que executam a ação atômica. Pode-se, por exemplo, considerar que em caso de falha, o processo com endereço de valor menor seja o eleito. Uma vez que cada processo contem os identificadores dos demais[1], a eleição pode ser feita por um processo assim que ele

[1]Esses identificadores fazem parte da lista transmitida com a mensagem PREPARE.

detectar a falha do coordenador. Cada processo mantém um conjunto F de processos que ele considera que estejam operacionais. Ao detectar a falha, o processo retira o identificador do coordenador falho do conjunto F e verifica qual o menor valor de identificador entre os processos que permaneceram no conjunto. Se o identificador obtido for o seu, ele se elege o novo coordenador, senão, envia mensagem NOVO-COORDENADOR ao processo de menor valor de identificador, informando-lhe da sua eleição.

Devido à possibilidade de dois subordinados quaisquer detectarem a falha do coordenador em tempos diferentes, um deles pode eleger um novo coordenador mais rápido do que o outro, com isso, o outro subordinado pode receber mensagens do novo coordenador antes mesmo de detectar a falha do coordenador anterior. Se o identificador de coordenador embutido em uma mensagem recebida por um processo for maior que o identificador que o processo considera ser do coordenador atual, significa que esse último falhou.

O novo coordenador reinicia o protocolo, enviando mensagens *qual-estado* aos demais processos operacionais.

O coordenador baseia sua decisão somente nas mensagens com os estados dos subordinados. Portanto, os subordinados que falham antes de enviarem mensagens são desprezados pelo coordenador.

Protocolo de Recuperação

Quando um processo se recupera após ter falhado em estado ABORTADO, VALIDADO, ou antes de entrar em estado PREPARADO, ele não necessita se comunicar com os demais processos. Nos dois primeiros casos, o processo em recuperação havia finalizado a ação antes da falha. No último caso, ele pode abortar a ação, pois não chegou a emitir o seu voto ao coordenador na primeira fase do protocolo de validação.

Os casos de recuperação em que o processo se encontra em estado PREPARADO ou PRÉ-VALIDADO são mais complicados devido à possibilidade de ter ocorrido uma falha de todos os processos.

Seja P um processo que falhou quando estava em estado PREPARADO ou PRÉ-VALIDADO. P, ao se recuperar, não pode tomar uma decisão de modo independente e não pode ativar o protocolo de finalização de imediato. Essa proibição ocorre pois os processos que falharam por último podem ter finalizado a ação. Esses processos podem continuar falhos, após a recuperação de P e de outros processos que falharam anteriormente. Se o grupo de processos recuperados ativar o protocolo de finalização, ou decidir unilateralmente finalizar a ação, a decisão pode ser diferente da decisão tomada pelos processos que continuam falhos.

O processo P em recuperação deve enviar mensagens PERGUNTANDO a todos os processos. Se algum processo tiver decidido e estiver operacional, ele envia a decisão e P, ao recebê-la, finaliza a ação.

Há, entretanto, a possibilidade de que todos os processos tenham falhado sem que uma

decisão tenha sido tomada. Nesse caso, o protocolo de finalização deve ser ativado. Porém, de acordo com a explicação anterior, um processo em recuperação pode ativar o protocolo de finalização somente após a recuperação de todos os processos que falharam posteriormente a ele. Os processos que falharam posteriormente a um processo P estão contidos no conjunto F do processo P. Portanto, a partir do momento em que P recebe mensagens PERGUNTANDO de todos os processos que pertencem ao seu conjunto F existente no momento da falha, P pode ativar o protocolo de finalização.

É necessário que cada processo armazene o seu conjunto F no log. Esse conjunto pode ser armazenado sempre que um registro tiver sua gravação forçada no log.

Correção do Protocolo de Três Fases e do Protocolo de Terminação

Deve-se demonstrar que o protocolo sempre termina quando não há falha total de processos e que a ação atômica termina em todos os processos de modo consistente.

Se não ocorre nenhuma falha do coordenador após iniciado o protocolo de finalização, o coordenador toma uma única decisão com base nos estados enviados pelos subordinados. Essa decisão não conflita com nenhum estado local dos processos envolvidos. Essa afirmação é verdadeira, pois está sendo considerado que o protocolo de finalização é ativado após a interrupção do protocolo de validação, portanto, no estado global do protocolo de validação existente no momento em que ocorreu a falha, não coexistem estados validados e estados não-validáveis. Como conseqüência, o coordenador utiliza para sua decisão uma única regra dentre as Regras 5.1.1, 5.1.2, 5.1.3 e 5.1.4. Desse modo, numa execução sem falha do protocolo de finalização, o protocolo termina e todos os processos abortam ou todos os processos validam a ação atômica.

Quando ocorrem falha sucessivas, a menos que todos o processos falhem, o protocolo sempre termina pois a cada nova falha, um novo coordenador é ativado. Se somente um processo permanecer operacional, ele termina a ação localmente com base no seu estado local. Se for um estado diferente de PRÉ-VALIDADO ou VALIDADO a ação é abortada. Em caso contrário ela é validada.

É necessário mostrar que a ação termina de modo consistente em todos os processos mesmo após sucessivas ativações do protocolo de finalização.

Deve-se notar que em qualquer ativação do protocolo de finalização, os processos que terminam a ação localmente tomam a mesma decisão quanto ao término. Isso se deve à inexistência de estados conflitantes quando da primeira ativação do protocolo e à utilização de uma única regra de terminação pelo coordenador, conforme discutido anteriormente.

É suficiente, portanto, mostrar que se houverem processos que terminam a ação localmente em uma ativação do protocolo de finalização, eles o fazem de modo consistente com as terminações feitas por outros processos em ativações anteriores e de modo consistente com as terminações feitas no protocolo de validação.

Deve-se mostrar que essa consistência se mantém qualquer que seja a regra de terminação utilizada pelo coordenador.

Regra 5.1.1: A aplicação dessa regra em uma ativação i significa que os processos que terminarem, abortarão a ação. O coordenador utiliza essa regra quando pelo menos um dos processos operacionais subordinados S está no estado de aborto. S pode estar no estado de aborto por um dos seguintes motivos:

1. S estava no estado ABORTADO no início da primeira ativação do protocolo de finalização, ou,
2. S recebeu uma mensagem ABORTE do coordenador na ativação anterior: $i-1$.

No primeiro caso, nenhum processo pode estar no estado de validação, pois todos os estados locais são resultantes da execução de um protocolo não-bloqueante.

No segundo caso, o coordenador da ativação $i-1$ só pode ter enviado a mensagem ABORTE a n se tiver recebido uma mensagem ABORTADO. Mas o coordenador na ativação $i-1$ só recebe uma mensagem ABORTADO se:

a. tiver ocorrido uma ativação anterior $i-2$ em que algum processo operacional m estava no estado de aborto ou (e)
b. não existe ativação anterior a $i-1$ ($i-1$ é a primeira ativação), mas existia algum processo m no estado ABORTADO quando o protocolo de validação foi interrompido.

Assim, de modo recursivo, se há algum processo que aborta a ação em uma ativação i do protocolo de finalização é porque houveram outros processos que abortaram a ação em ativações anteriores ou (e) quando o protocolo de validação foi interrompido algum processo estava no estado ABORTADO. Em ambos os casos, todos os processos que terminaram a ação, abortaram.

Regra 5.1.2: O coordenador de uma ativação i utiliza essa regra quando recebe uma mensagem VALIDAÇÃO. Isto significa que na ativação i existe um processo n que está no estado de validação. De modo análogo ao caso da Regra 5.1.1, n está nesse estado por um dos seguintes motivos:

1. n estava no estado de validação quando o protocolo de validação foi interrompido;
2. n mudou para o estado de validação após receber uma mensagem VALIDE do coordenador da ativação $i-1, (i>1)$.

De modo recursivo, em todas as ativações anteriores ou antes da primeira ativação, todos os processos que terminam a ação a validam.

Regra 5.1.3: quando o coordenador utiliza essa regra em uma execução i do protocolo de finalização, todos os processos que terminam a ação em i abortam. Deve-se mostrar

que em nenhuma execução anterior nenhum processo tenha validado a ação localmente. Será mostrado, que se isso ocorre, não há como existir processos em estado PREPARADO na execução i.

Em uma execução anterior $j, j < i$, possivelmente parcial, do protocolo de finalização, alguns processos podem ter validado a ação somente se o coordenador enviou a mensagem VALIDE a eles durante j. O coordenador pode tomar essa decisão somente nos casos em que se aplicam as Regras 5.1.2 e 5.1.4. Se a Regra 5.1.2 for utilizada, como vimos, todos os processos operacionais estão em estado VALIDADO ou em estado PRÉ-VALIDADO. Uma vez que o processo entra em estado validável ele não volta a um estado não-validável em execuções posteriores do protocolo, portanto na execução i todos os processos estarão em estado PRÉ-VALIDADO ou em estado VALIDADO, o que é uma contradição.

Se a Regra 5.1.4 for utilizada na execução j, então os processos que estavam em estado não-validável irão mudar para o estado validável após receberem a mensagem PRÉ-VALIDAÇÃO. Posteriormente, dentro da mesma execução i (caso o coordenador não falhe nesse intervalo), os processos que mudaram para o estado validável receberão uma mensagem VALIDE. Portanto, nas execuções subseqüentes (inclusive na execução i) todos os processos que permanecerem operacionais irão validar a ação o que é uma contradição. Logo, se a Regra 5.1.3 for usada em uma execução i do protocolo de finalização, não podem existir processos que validaram a ação em execuções anteriores.

Regra 5.1.4: quando essa regra é aplicada em uma execução i do protocolo de finalização, todos os processos que terminam a ação a validam. Será mostrado novamente por contradição que nenhum processo pode ter abortado em uma execução anterior. Suponha que algum processo tenha abortado a ação em alguma execução $j, j < i$. Foi visto que processos terminam consistentemente a ação em uma execução e foi mostrado para o caso da Regra 5.1.1 que quando um processo aborta a ação em uma execução, todos os processos que terminarem a ação nas execuções seguintes também a abortam. Portanto, se um processo aborta a ação na execução j, todos os processos que terminam a ação na execução i também abortam, o que contradiz a suposição da Regra 5.1.4.

Conclui-se portanto, que o protocolo de finalização centralizado termina consistentemente em todos processos.

A execução conjunta dos dois protocolos apresentados nesta seção evita que os nós operacionais fiquem bloqueados. Entretanto, a demonstração da correção dos protocolos dada acima vale apenas quando ocorrem falhas de nós que não impossibilitem a comunicação entre dois nós operacionais. O seguinte exemplo ilustra a possibilidade de finalizações inconsistentes, caso ocorra falha de partição no sistema. Suponha que ocorra uma falha de comunicação e que o sistema fique divido em três partições: A = {coordenador}, B = {subordinados preparados} e C = {subordinados pré-validados}. Um ou mais subordinados pertencentes às partições B e C ativam o protocolo de finalização. Um novo coordenador é eleito em cada uma das partições B e C. De acordo com a descrição do protocolo de finalização, o coordenador de B decide abortar a ação e o coordenador de A decide validá-la.

Os protocolos descritos acima não se aplicam na prática, pois não é possível a um processo deduzir através da detecção de término do tempo de espera se houve uma falha de comunicação ou se a mensagem não chegou devido a falha do processo emissor. Portanto, há sempre o risco de finalizações inconsistentes.

Protocolos mais realistas podem ser obtidos a partir do protocolo de três fases, sacrificando-se a resistência ao bloqueio, mas garantindo que as finalizações sejam todas consistentes entre si. Na próxima seção é apresentado um protocolo com essas características.

Tempo de Execução

Uma execução sem falhas do protocolo para uma ação que é validada possui no máximo seis etapas de envio de mensagens: 1) envio de mensagens PREPARE; 2) envio dos votos; 3) envio de mensagens PRÉ-VALIDE do coordenador; 4) envio de mensagens PRÉ-VALIDADO dos subordinados; 5) envio das mensagens VALIDE, 6) envio dos sinais RECEBI dos subordinados. Logo, o tempo de execução corresponde a cinco unidades de tempo.

Número de Mensagens

Seja N o número de processos envolvidos na execução da ação. O número de mensagens enviadas pelo protocolo durante uma execução sem falhas do protocolo em que a ação é validada é $6(N-1)$ que corresponde a: $N-1$ mensagens PREPARE, $N-1$ mensagens de votos, $N-1$ mensagens PRÉ-VALIDE, $N-1$ mensagens PRÉ-VALIDADO, $N-1$ mensagens VALIDE e $N-1$ mensagens RECEBI.

5.2 Protocolo de Duchamp

O problema de decisões inconsistentes surge quando dois ou mais coordenadores coexistem em partições distintas. É necessário garantir que dois coordenadores que decidem tomem a mesma decisão. A solução para esse problema consiste em permitir que um coordenador tome uma decisão somente se ele puder se comunicar com um quorum mínimo de processos[2].

O protocolo de Duchamp [Duc88], apresentado nesta seção, utiliza dois tipos de quorum: um quorum para validação e um para o aborto. A execução do protocolo requer três fases. A primeira e a última fases são semelhantes às do protocolo de duas fases. A segunda fase é a fase de obtenção de quorums. Não é utilizado um protocolo de finalização separado. Os subordinados operacionais que não podem finalizar a ação de modo independente ativam novamente o protocolo na tentativa de obterem uma decisão.

[2]Uma variação dessa solução corresponde a utilizar o limite mínimo como sendo a maioria, em vez de quorum.

Descrição do Protocolo

Primeira Fase

A primeira fase do protocolo de Duchamp é semelhante à do protocolo de três fases descrito na seção anterior. Entretanto, o coordenador envia também na mensagem PREPARE o tamanho dos quorums de aborto e de validação. O registro PREPARADO, tanto dos subordinados como do coordenador, deve conter: os identificadores de todos os processos envolvidos na ação e os tamanhos dos quorums.

Ao obter os votos dos subordinados, o coordenador entra em estado PRÉ-VALIDADO se todos os votos forem SIM; caso contrário ele entra em estado PRÉ-ABORTADO. Em ambos os casos, o coordenador força a gravação de um registro correspondente ao seu novo estado. A seguir, o coordenador inicia a fase intermediária, ou fase de replicação.

Fase Intermediária ou Fase de Replicação

Nesta fase, o coordenador tenta formar um quorum correspondente ao estado em que se encontra; isto é, um quorum de aborto se estiver em estado PRÉ-ABORTADO ou um quorum de validação se estiver em estado PRÉ-VALIDADO. O coordenador inicia esta fase emitindo mensagens-convite do tipo PRÉ-VALIDE ou PRÉ-ABORTE, de acordo com o estado que ocupa.

Cada subordinado preparado ao receber uma mensagem PRÉ-VALIDE (PRÉ-ABORTE), entra no estado correspondente, forçando a gravação de um registro PRÉ-VALIDADO (PRÉ-ABORTADO). Dessa forma, o subordinado se declara participante do quorum correspondente a esse estado. Em seguida, o subordinado envia a mensagem PRÉ-VALIDADO (PRÉ-ABORTADO) ao coordenador.

Se um processo participa na formação de um quorum, ele não pode participar de quorum oposto durante a execução do protocolo para a mesma ação. Ao obter as mensagens contendo os estados dos subordinados, o coordenador tenta obter um quorum de aborto ou de validação, incluindo na contagem o seu próprio estado. Se N é o número de processos envolvidos na ação, A é o número de processos requerido para formar o quorum de aborto e V o número requerido para formar o quorum de validação, a seguinte relação deve prevalecer entre esses parâmetros:

$$V + A = N + 1,\ V, A > 1. \tag{5.1}$$

Se a relação 5.1 é satisfeita, não há a possibilidade dos dois quorums coexistirem para uma mesma ação.

Se o coordenador obtém um quorum, ele toma a decisão correspondente ao quorum obtido e finaliza a ação entrando no estado final correspondente.

Terceira Fase

A terceira fase é semelhante à segunda fase do protocolo de duas fases, porém, algumas alterações são necessárias. Os subordinados não podem apagar de sua memória volátil as informações referentes a ação logo após terem enviado o sinal RECEBI ao coordenador. É necessário que os subordinados mantenham dados sobre a ação em sua memória volátil enquanto houver a possibilidade de existir outro coordenador tentando obter quorum. Essa restrição é necessária para garantir que o protocolo seja consistente. Se assim não fosse, um subordinado que tenha apagado informações sobre uma ação após validá-la pode vir a receber uma mensagem do tipo PRÉ-ABORTE de um outro coordenador. Como esse subordinado não tem mais informações sobre a ação, ele passa a participar do quorum de aborto, gravando um registro PRÉ-ABORTADO, o que pode causar a formação de um quorum de aborto. Nesse caso, o outro coordenador toma uma decisão diferente da primeira.

Para evitar o problema acima, os subordinados mantêm informações sobre a ação na memória volátil até que recebam uma mensagem ESQUEÇA do coordenador.

Ao receber todas mensagens RECEBI, o coordenador apaga de sua memória volátil as informações sobre a ação e envia mensagem ESQUEÇA a todos os subordinados. Ao receber mensagens RECEBI de todos os subordinados, o coordenador sabe que todos os processos envolvidos na ação a finalizaram e, portanto, não há mais a possibilidade haver outro coordenador tentando obter quorum.

Tratamento de Falhas

Quando o coordenador detecta término do tempo de espera pelo voto de um subordinado na primeira fase, ele comporta-se como se tivesse recebido o voto NÃO daquele subordinado.

Quando um subordinado detecta término do tempo de espera por uma mensagem do coordenador, ele se transforma em um novo coordenador, mantendo-se no estado em que estava ao detectar a falha. A seguir, o novo coordenador tenta fazer com que os demais processos fiquem no mesmo estado em que ele está. Assim, se o novo coordenador estava preparado quando detectou a falha do coordenador anterior, ele envia mensagens PREPARE a todos os processos. Se estiver em estado PRÉ-VALIDADO (PRÉ-ABORTADO), o novo coordenador envia mensagem PRÉ-VALIDE) (PRÉ-ABORTE) a todos os subordinados. A comunicação do novo coordenador com os demais processos é possível graças a lista de identificadores que cada processo recebe com as mensagens PREPARE.

Um processo também se torna coordenador se ele se recupera de uma falha. Nesse caso, o processo em recuperação comporta-se como no caso de um subordinado operacional que se transforma em coordenador[3].

Se ao se recuperar um subordinado não encontra registro correspondente a uma ação ele

[3]No caso específico em que um processo falha antes de entrar em estado PREPARADO, ele muda para o estado PRÉ-ABORTADO ao se recuperar e torna-se um coordenador.

a aborta. Se for encontrado um registro VALIDADO ou ABORTADO, o processo em recuperação comporta-se como um coordenador e envia mensagens com a decisão correspondente ao seu estado aos demais processos.

Devido à ocorrência de falhas pode haver, em qualquer instante, dois ou mais coordenadores para uma mesma ação. A presença de vários coordenadores é referida por Duchamp como um *duelo* entre coordenadores. Para os subordinados, a presença de coordenadores duelistas tem como conseqüência apenas o recebimento de mensagens duplicadas ou de mensagens que não afetam o estado do processo receptor. Por exemplo, se um subordinado em estado PRÉ-VALIDADO recebe uma mensagem PREPARE ele apenas envia o voto SIM, mas continua no estado PRÉ-VALIDADO. Se um processo em estado PRÉ-VALIDADO (PRÉ-ABORTADO) recebe uma mensagem PRÉ-ABORTE (PRÉ-VALIDE), ele permanece no estado PRÉ-VALIDADO (PRÉ-ABORTADO) e envia a mensagem PRÉ-VALIDADO (PRÉ-ABORTADO) como resposta. Os estados possíveis em que um subordinado fica aguardando uma mensagem do coordenador são: PREPARADO, PRÉ-VALIDADO e PRÉ-ABORTADO.

O tratamento dado por um coordenador a mensagens recebidas de outro coordenador é um pouco mais complicado e depende dos estados ocupados pelos coordenadores duelistas. Se um coordenador que recebe uma mensagem está em um estado *mais adiantado*[4] com relação ao estado do coordenador emissor da mensagem, o coordenador receptor ignora a mensagem e continua tratando o emissor como se ele fosse um subordinado. Se o coordenador receptor está menos adiantado que o emissor, ele comporta-se como um subordinado do emissor, mas continua sendo um coordenador. Vejamos o exemplo de um coordenador em estado PREPARADO que recebe uma mensagem VALIDE de outro coordenador. O coordenador receptor valida a ação, envia o sinal RECEBI ao emissor e volta a atuar como coordenador, enviando mensagens VALIDE a todos os seus subordinados. Se ambos, receptor e emissor, estão igualmente adiantados, então o receptor responde como se fosse um subordinado. Portanto, se um coordenador em estado VALIDADO recebe uma mensagem VALIDE ele envia uma mensagem RECEBI como resposta.

Correção do Protocolo

Todos os processos que finalizam ação o fazem segundo a mesma decisão. As justificativas para essa afirmação se devem-se aos seguintes fatos:

1. uma decisão é obtida somente se um dos quorums (de aborto ou de validação) for obtido;

2. os tamanhos dos quorums obedecem à relação 5.1. Portanto, se em um dado instante um quorum de validação (aborto) é obtido por um coordenador, um quorum diferente não pode ser obtido por outro coordenador duelista;

[4] Um estado p é *mais adiantado* que um estado q, se p está mais próximo de um estado final do que q.

3. a participação de um processo em um quorum é definitiva, ou seja, uma vez que um processo entra no estado PRÉ-VALIDADO (PRÉ-ABORTADO), ele não muda para o estado oposto(PRÉ-ABORTADO (PRÉ-VALIDADO)). Esse fato é garantido pelo uso de mensagens ESQUEÇA, conforme descrito na seção anterior, e pelos registros PRÉ-ABORTADO e PRÉ-ABORTADO gravados no log.

Os três fatos garantem que se um quorum é formado, ele é único durante a execução do protocolo para uma dada ação e que a decisão tomada é de acordo com esse quorum.

O protocolo de Duchamp é, contudo, bloqueante quando ocorrem falhas de nós ou falhas de partição que impedem a obtenção de um quorum.

Tempo de Execução e de Número de Mensagens

Uma execução sem falhas do protocolo de Duchamp é semelhante a uma execução do protocolo de três fases. Portanto, o tempo de execução corresponde a cinco unidades de tempo e o número de mensagens é $6(N-1)$, onde N é o número de processos que participam da ação.

5.3 Protocolo de Validação Descentralizado e Ótimo (ODCP)

Os protocolos apresentados nesta e nas próximas seções diferem bastante dos protocolos descritos anteriormente. Tratam-se de protocolos distribuídos em que o modelo de comunicação entre os processos é um hipercubo.

Nesta seção é descrito o protocolo ODCP que utiliza o número mínimo de mensagens possível em um protocolo distribuído. O ODCP assume que o número de processos da rede seja $N = 2^k$, para algum k $\in N$. A cada processo X é associado um endereço que é um número binário com k dígitos da forma: $(x_1, x_2, ...x_k)$. Se o número N de processos da rede não é uma potência de dois, utiliza-se o menor número $M > N$, com $M = 2^k$, para algum natural k e trata-se o sistema como uma rede formada por M processos lógicos com alguns processos físicos correspondendo a dois processos lógicos.

Na descrição do protocolo será utilizada a seguinte definição:

Definição 5.3.1 *Para cada processo* X, $0 \leq X \leq N-1$, *cujo o endereço é* $(x_1, x_2, ...x_i, ...x_k)$, *denomina-se* complemento i-ésimo de X *o processo* $\overline{X}_i$ *cujo endereço é* $(x_1, x_2, ...\overline{x}_i, ...x_k)$.

O protocolo é caracterizado por várias etapas (*rounds*) de trocas de mensagens entre os processos. A idéia central do protocolo é a seguinte: na etapa i, $1 \leq i \leq k$, todo processo X, $0 \leq X \leq N$ troca mensagem com o processo $= \overline{X}_i$.

O diagrama de estados do autômato do ODCP em cada processo está mostrado na Figura 5.1.

Descrição do Protocolo

As ações executadas em cada estado no processo X são as seguintes:

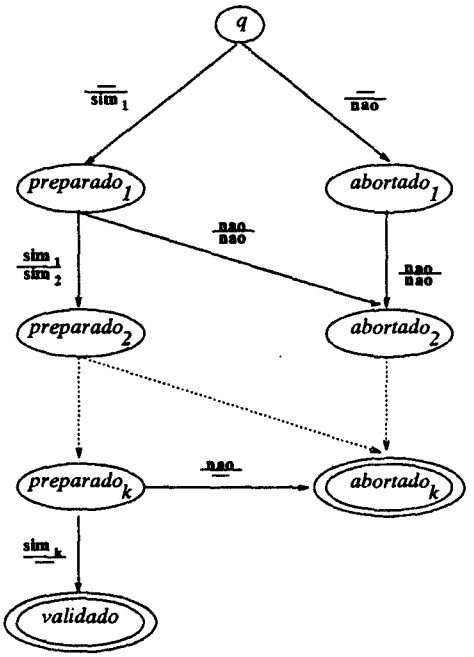

Figura 5.1: Autômato Finito Correspondente ao ODCP

No estado inicial (estado q), após terminar a execução local de parte da ação, o processo X decide se concorda com o aborto ou validação da ação. Se a decisão for de abortar, X envia uma mensagem NÃO$_1$ ao processo $\overline{X}_1$ e move para o estado ABORTADO$_1$. Se concordar em validar, X envia mensagem SIM$_1$ ao processo $\overline{X}_1$ e move para o estado PREPARADO$_1$.

No estado PREPARADO$_i$, $1 \leq i \leq (k-1)$, se X recebe uma mensagem SIM$_i$, X inicia a etapa de mensagens $i+1$, enviando mensagem SIM$_{i+1}$ ao processo $\overline{X}_{i+1}$ e muda para o estado PREPARADO$_{i+1}$. Se X recebe uma mensagem NÃO$_i$, inicia a etapa de mensagens $i+1$, enviando mensagem NÃO$_{i+1}$ ao processo $\overline{X}_{i+1}$ e muda para o estado ABORTADO$_{i+1}$.

Se X está no estado PREPARADO$_k$, então, se ele recebe uma mensagem SIM$_k$, ele muda para o estado final VALIDADO e efetua a validação da ação localmente. Se uma mensagem NÃO$_k$ for recebida, X aborta a ação localmente e muda para o estado ABORTADO$_k$.

No estado ABORTADO$_i$, $1 \leq i \leq (k-1)$, o processo X envia uma mensagem NÃO$_{i+1}$ ao processo $\overline{X}_{i+1}$ e muda para o estado ABORTADO$_{i+1}$, qualquer que seja o tipo de mensagem

recebida.

Os estados ABORTADO$_k$ e VALIDADO$_k$ são os estados finais de aborto e validação, respectivamente. Ao entrar em um novo estado, um processo X grava no log um registro correspondente a esse estado (ABORTADO, VALIDADO, PREPARADO).

Correção do Protocolo ODCP

O ODCP sempre termina quando não ocorrem falhas de nenhuma espécie. Pode-se mostrar a veracidade dessa afirmação com base na demonstração dos seguintes fatos:

a. se um processo está em um estado ABORTADO ele irá por fim, abortara ação;

b. se um processo não muda para nenhum estado ABORTADO, ele irá validar a ação.

No caso (a), quando um processo X está em um estado ABORTADO$_i$, $1 \leq i \leq (k-1)$, a única transição possível em cada etapa j, $i \leq j \leq k$ é enviar mensagens NÃO e mudar do estado ABORTADO$_j$ para o estado ABORTADO$_{j+1}$. Como está sendo considerado que não ocorrem falhas durante as k etapas, na k-ésima etapa X mudará do estado ABORTADO$_{k-1}$ para o estado ABORTADO$_k$, que é o estado final de aborto.

No caso (b), se um processo X nunca muda para um estado ABORTADO durante a execução do protocolo, é porque quando X está no estado q na primeira etapa ele envia mensagens SIM a $\overline{X}_1$ e muda para o estado PREPARADO$_1$. Além disso, como está desconsiderada a ocorrência de falhas e como pelo protocolo todos os processos enviam mensagens em cada etapa $i, 2 \leq i \leq k$, na etapa k, X está no estado PREPARADO$_k$ e ao receber uma mensagem SIM$_k$ de $\overline{X}_{k-1}$, irá para o estado VALIDADO e, portanto, validará ação.

Deve-se mostrar também que as terminações em cada processo são consistentes entre si, isso é:

1. se um processo aborta a ação, todos os processos irão também abortá-la;

2. se nenhum processo decide abortar a ação, todos os processos irão finalmente validá-la.

Antes da demonstração da afirmação (1) acima, será apresentada a definição a seguir que auxiliará na demonstração:

Definição 5.3.2 *Para todo processo $X, 0 \leq X \leq N-1$, define-se $G(X,i)$ como sendo o conjunto de processos cujos endereços são da forma: $(*, ..., *, x_i, ..., x_k)$, onde $1 \leq i \leq k$ e o endereço do processo X é: $(x_1, x_2, ..., x_{i-1}, x_i, ...x_k)$ e $1 \leq i \leq k$.*

Em outras palavras, o conjunto $G(X,i)$ para um dado processo $X(x_1, x_2, \cdots, x_i, \cdots, x_k)$ é o conjunto formado pelos processos cujos $i-1$ primeiros bits dos seus endereços pertencem

ao conjunto resultante de todas permutações dos bits $(x_1, x_2, \cdots, x_{i-1})$ e os bits restantes são do endereço: $(x_i, \cdots, x_k$.

Deve-se notar que para todo processo $X, 0 \leq X \leq N-1$, $G(X,i)$ é a união de $G(X, i-1)$ e $G(\overline{X}_{i-1}, i-1)$. Se for considerado que o processo X possui o endereço: $(x_1, x_2, ..., x_{i-1}, x_i, ...x_k)$; então o processo $\overline{X}_{i-1}$ possui o endereço: $(x_1, x_2, ..., , \overline{x}_{i-1}.x_i, ..., x_k)$. Uma vez que os processos em $G(X,i)$ possuem os endereços da forma:

$$(\overbrace{*, ..., *}^{i-1}, x_i, ..., x_k)$$

que inclui endereços da forma:

$$(\overbrace{*, ..., *}^{i-2}, x_{i-1}, x_i, ..., x_k) = G(X, i-1)$$

ou endereços da forma:

$$(\overbrace{*, ..., *}^{i-2}, \overline{x}_{i-1}, x_i, ..., x_k) = G(\overline{X}_{i-1}, i-1)$$

$G(X,i)$ é um subconjunto de $G(X,i-1) \cup G(\overline{X}_{i-1}, i-1)$. Como, $G(X,i-1)$ e $G(\overline{X}_{i-1}, i-1)$ são subconjuntos de $G(X,i)$, $G(X,i)$ è igual a união de $G(X, i-1)$ e $G(\overline{X}_{i-1}, i-1)$.

Quanto à demonstração das afirmações (1) e (2) acima, a primeira pode ser mostrada por indução. A hipótese indutiva é a seguinte:

Hipótese Indutiva: Se um processo X decide abortar a ação, todos os processos em $G(X, i+1)$ recebem uma mensagem NÃO após a i-ésima etapa de envio de mensagens onde $1 \leq i \leq k$.

Caso Base: $i = 1$, ou seja, X está no estado q e decide abortar. X envia mensagem NÃO_1 a $Z = \overline{X}_1$ e muda para o estado ABORTADO_1. $Z = \overline{X}_1$ envia mensagem a X cujo conteúdo depende se Z está em ABORTADO_1 ou PREPARADO_1. Qualquer que seja o conteúdo da mensagem recebida, o processo X muda para o estado ABORTADO_2. Após receber a mensagem NÃO_1 de X, o processo Z muda para o estado ABORTADO_2. Portanto, após a primeira etapa, $X_1 = (x_1, x2, \ldots, x_k)$ e $Z = \overline{X}_1 = (\overline{x}_1, x2, \ldots, x_k)$ estarão ambos (que constituem $G(X,2)$) em estado ABORTADO_2.

Passo da Indução:

Suponha que a hipótese seja verdadeira para $i = l-1$, onde $l \leq k$. Logo, após a $(l-1)$-ésima etapa de mensagens, todos os processos de $G(X,l)$ cujos endereços são da forma $(\underbrace{*, \ldots, x_l}_{l-1}, \ldots, x_k)$ terão recebido mensagens NÃO e estarão em um estado ABORTADO de

acordo com a hipótese indutiva. Portanto, durante a etapa l, cada processo de $G(X,l)$ enviará mensagens NÃO a um a um processo correspondente de $G(\overline{X}_l,l)$ cujos endereços são da forma: $(*,\ldots,*,\overline{x}_l,\ldots,x_k)$. Logo, ao final da etapa l, todos os processos em $G(\overline{X},l) \cup G(X,l)$, e portanto $G(X,l+1)$, estarão em algum estado ABORTADO$_i$. Uma vez em um estado ABORTADO$_i$, um processo entrará na etapa k no estado final ABORTADO$_k$. ■

A afirmação (2) pode ser demonstrada pelos mesmos argumentos utilizados na demonstração do caso (b) referente à demonstração do término do protocolo, vista anteriormente.

Tempo de Execução

O tempo de execução corresponde ao número de etapas do protocolo. O número de etapas do ODCP é igual ao número de dígitos binários que compõem os endereços dos processos. Para N processos, $N = 2^k$, para algum inteiro positivo k, o número de etapas será $k = \log_2 N$.

Número de Mensagens

Em uma etapa, cada processo envia uma mensagem, portanto, são N processos que enviam mensagens em $\log_2 N$ etapas, logo o número total de mensagens é $N \log_2 N$. Em [YA87] é demonstrado que o ODCP é um protocolo distribuído ótimo com relação ao número de mensagens enviadas.

Propriedades do ODCP

Apesar de não suportar nem mesmo uma única falha de nó, o ODCP possui duas propriedades que permitem estendê-lo de modo a obter-se protocolos que oferecem tolerância a falhas.

Propriedade 5.3.1 *Se dois processos X e Z trocam mensagens na etapa $i \geq 1$, então eles enviarão as mesmas mensagens e irão para estados idênticos nas etapas subseqüentes.*

Esta propriedade pode ser demonstrada por indução no número de etapas de envio de mensagens. Consideremos dois processos X e Z que trocam mensagens entre si durante a etapa i. Do protocolo ODCP, $X = \overline{Z}_m$ e $Z = \overline{X}_m$.

Hipótese Indutiva.

Na etapa $j > i$ os processos X e Z enviam mensagens idênticas e mudam para estado idênticos.

Base da indução $j = i + 1$.

Se X e Z enviam mensagens NÃO na etapa i, então pelo ODCP eles irão para o estado ABORTADO$_{i+1}$ e enviarão mensagens NÃO na etapa $i+1$. De modo análogo, se ambos enviam mensagens SIM na etapa i, passarão para o estado PREPARADO$_{i+1}$ e enviarão SIM na etapa $i+1$.

Passo da Indução. Suponha que a hipótese seja válida para toda etapa $m, i < m \leq l$. Na etapa m, o processo X troca mensagem com o processo $\overline{X}_m$ e o processo Z troca mensagem com o processo $\overline{Z}_m$. Uma vez que o processo $\overline{X}_m$ troca mensagem com o processo $\overline{Z}_m$ na etapa $i, i < m \leq l$, então, pela hipótese indutiva, a mensagem recebida pelo processo X será a mesma recebida pelo processo Z da etapa $i+1$ à etapa l. Os dois processos enviarão, portanto, a mesma mensagem e irão para o mesmo estado na etapa $l+1$. ■

Propriedade 5.3.2 *Para todo processo $X, 0 \leq X \leq N-1$, todos os processos em $G(X,i)$ enviam a mesma mensagem e movem para o mesmo estado na etapa $i, 1 \leq i \leq k$.*

Essa propriedade é conseqüência da propriedade 5.3.1 e pode ser demonstrada também por indução no número de etapas.

Hipótese Indutiva. Para todo processo X, todos os processos em $G(X,l)$ enviam a mesma mensagem e movem para o mesmo estado na etapa l, onde $l \geq 1$.

Base da indução $l = 1$.

Nesse caso $G(X,l) = X$, a hipótese é verdadeira.

Passo da Indução.

Considere que a hipótese seja verdadeira para todo $i \leq l$. Todos os processos em $G(X,l)$ enviam a mesma mensagem e mudam para o mesmo estado na etapa l. Do mesmo modo, todos os processos em $G(\overline{X},l)$ enviam a mesma mensagem e mudam para o mesmo estado na etapa l. Como cada processo em $G(X,l)$ troca mensagem com um processo correspondente em $G(\overline{X},l)$ na etapa l, de acordo com a propriedade 1, e como $G(X,l+1) = G(X,l) \cup G(\overline{X},l)$, todos os processos em $G(X,l+1)$ enviarão a mesma mensagem e mudarão para o mesmo estado na etapa $l+1$. Portanto, para cada processo $X, 0 \leq X \leq N-1$, todos processos em $G(X,i)$ enviam a mesma mensagem e mudam para o mesmo estado na etapa $i, 1 \leq i \leq k$. ■

A obtenção gradual de consenso é a base para a construção de protocolos tolerantes a falhas a partir do ODCP. Cada elemento de um conjunto $G(X,i)$ de processos que concordam entre si serve como substituto, caso algum outro elemento dentro do mesmo conjunto falhe durante uma etapa do protocolo. Desse modo, a execução do protocolo não é interrompida quando um nó falha, pois na próxima etapa o substituto envia uma mensagem para dois processos: o processo para o qual ele deveria enviar e o processo para o qual o processo falho deveria enviar a mesma mensagem.

O protocolo descrito na próxima seção utiliza a idéia de se usar um processo substituto para a obtenção de tolerância a falhas.

5.4 Protocolo de Validação Distribuído e Tolerante a Falha (PVDTF)

O protocolo PVDTF utiliza, em cada processo, a variável *término-espera* e o vetor *substituto*. A variável *término-espera* contém o número de vezes que o processo detecta término do tempo de espera durante uma execução do protocolo. O valor inicial de *término-espera* é zero. O vetor *substituto*, em um processo X, armazena os endereços dos processos que X está substituindo. Quando X detecta a falha de um processo Y, ele soma um ao valor *término-espera* e insere o endereço de Y na próxima posição livre do vetor *substituto*.

Os seguintes tipos de mensagens são utilizados em adição àqueles empregados no protocolo ODCP:

PERGUNTANDO(X,i): Quando um processo X detecta um término de espera na etapa i, ele envia uma mensagem PERGUNTANDO(X,i) para descobrir o estado em que o processo falho estava.

RECUPERANDO(X): Ao se recuperar de uma falha, um processo X pode enviar uma mensagem RECUPERANDO(X) para descobrir a decisão final quanto ao término da ação.

VALIDE: Um processo, ao receber uma mensagem RECUPERANDO(X), envia uma mensagem VALIDE caso a decisão final tenha sido a de validar a ação.

ABORTE: Um processo, ao receber uma mensagem RECUPERANDO(X), envia uma mensagem ABORTE caso a decisão final tenha sido a de abortar a ação.

A idéia básica do protocolo é a seguinte: Em uma etapa $i, 2 \leq i \leq k$, se um processo X detecta a falha do processo $\overline{X}_i$, então X insere no vetor *substituto* o endereço $\overline{X}_i$ e envia mensagens PERGUNTANDO(X, i) a todos os processos em $G(\overline{X}_i, i)$ com o objetivo de saber qual mensagem X iria receber de $\overline{X}_i$ na etapa i. Ao receber alguma resposta, X passa enviar as mensagens que $\overline{X}_i$ iria enviar nas etapas subseqüentes. Esse procedimento funciona somente se a seguinte restrição for satisfeita:

Restrição 5.4.1 *Se um processo X com endereço $(x_1, x_2, ..., x_k)$ falha na etapa $i, i > 1$, então, pelo menos um dos processos em $G(X, i)$ devem estar funcionando na etapa i.*

O protocolo PVDTF, descrito a seguir, garante tolerância a múltiplas falhas, desde que seja obedecida a restrição 5.4.1.

Descrição do Protocolo

Um processo X no estado inicial emite o seu voto após terminar a execução local da ação. Se X não concorda em validar a ação, ele envia mensagem NÃO para o processo $\overline{X}_1$ e muda para o estado ABORTADO$_1$. Se concordar em validar a ação, X envia a mensagem SIM$_1$ para o processo $\overline{X}_1$ e muda para o estado PREPARADO$_1$. Se falhar, ao recuperar-se muda para o estado de aborto ABORTADO$_k$.

No estado PREPARADO$_1$, ao detectar término de espera, X atribui *término-espera* = 1 e *substituto*[1]= $\overline{X}_1$, envia mensagem NÃO ao processo $\overline{X}_2$ e ao processo $\overline{substituto[1]}_2$, e muda para o estado ABORTADO$_2$. Ao receber uma mensagem SIM$_1$, envia uma mensagem SIM$_2$ ao processo $\overline{X}_2$ e muda para o estado PREPARADO$_2$. Ao receber uma mensagem NÃO$_1$, envia uma mensagem NÃO$_2$ ao processo $\overline{X}_2$ e muda para o estado ABORTADO$_2$. Se falhar, ao recuperar, envia uma mensagem RECUPERANDO(X) aos demais processos e muda para o estado especial RECUPERANDO.

No estado PREPARADO$_i$, $2 \leq i \leq (k-1)$, ao detectar término de espera, X atribui *término-espera* = *término-espera* + 1 e *substituto*[*término-espera*] = $\overline{X}_i$ e envia mensagem PERGUNTANDO(X, i) a todos processos em $G(\overline{X}_i, i)$. Ao receber uma mensagem PERGUNTANDO(Z, j), envia uma mensagem SIM$_j$ a Z. Ao receber uma mensagem SIM$_i$, envia uma mensagem SIM$_{i+1}$ ao processo $\overline{X}_{i+1}$ (se *término-espera* > 0, envia, também, mensagem SIM$_{i+1}$ para os processos $\overline{substituto[j]}_{i+1}, j = 1, 2, ...$*término-espera*) e muda para o estado PREPARADO$_{i+1}$. Ao receber uma mensagem NÃO, envia uma mensagem NÃO$_{i+1}$ ao processo $\overline{X}_{i+1}$ (se *término-espera* > 0, envia, também, mensagem NÃO$_{i+1}$ para os processos $\overline{substituto[j]}_{i+1}, j = 1, 2, ...$*término-espera* e muda para o estado ABORTADO$_{i+1}$. Ao se recuperar de uma falha, envia mensagem RECUPERANDO(X) aos demais processos e muda para o estado RECUPERANDO.

No estado PREPARADO$_k$, ao detectar um término de espera, X envia mensagem PERGUNTANDO$(\overline{X}_k, k)$ a todos os processos em $G(X, k)$. Ao receber uma mensagem PERGUNTANDO(Z, j), X envia uma mensagem SIM$_j$ a Z. Ao receber uma mensagem SIM$_k$, muda para o estado VALIDADO. Ao receber uma mensagem NÃO, muda para o estado ABORTADO$_k$. Quando falhar, ao recuperar, envia mensagem RECUPERANDO(X) aos demais processos e muda para o estado RECUPERANDO.

No estado ABORTADO$_1$, ao detectar término de espera, X atribui *término-espera* = 1 e *substituto*[*término-espera*] = $\overline{X}_1$, envia mensagem NÃO$_1$ ao processo $\overline{X}_2$ e ao processo $\overline{substituto[1]}_2$ e muda para o estado ABORTADO$_2$. Ao receber uma mensagem de qualquer tipo, envia uma mensagem NÃO ao processo $\overline{X}_2$ e muda para o estado ABORTADO$_2$. Ao se recuperar de uma falha, muda para o estado ABORTADO$_k$.

No estado ABORTADO$_i$, $2 \leq i \leq (k-1)$, ao detectar um término de espera, X atribui *término-espera* = *término-espera* + 1 e *substituto*[*término-espera*] = $\overline{X}_i$. Ao receber uma mensagem PERGUNTANDO(Z, j), envia uma mensagem NÃO a Z. Ao receber uma mensagem SIM$_i$ ou NÃO$_i$, envia uma mensagem NÃO$_{i+1}$ ao processo $\overline{X}_{i+1}$(se *término-espera* > 0, envia, também, mensagem SIM$_{i+1}$ para os processos $\overline{substituto[1]}_{i+1}, j = 1, 2, ...$*término-espera*) e muda para o estado ABORTADO$_{i+1}$. Quando falhar, ao recuperar muda para o estado ABORTADO$_k$.

No estado ABORTADO$_k$, ao receber uma mensagem PERGUNTANDO(Z, j), X envia uma mensagem NÃO a Z. Ao receber uma mensagem RECUPERANDO(Z), envia uma mensagem ABORTE ao processo Z.

No estado VALIDE Ao receber uma mensagem PERGUNTANDO(Z, j), X envia uma men-

sagem SIM$_j$ ao processo Z. Ao receber uma mensagem RECUPERANDO(Z), envia uma mensagem VALIDE ao processo Z.

No estado RECUPERANDO, ao receber uma mensagem VALIDE, X muda para o estado VALIDADO. Ao receber uma mensagem ABORTE, muda para o estado ABORTADO$_k$.

Correção do Protocolo PVDTF

Quando não ocorrem falhas, o PVDTF é idêntico ao ODCP, portanto é necessário analisar sua correção apenas quando ocorrem falhas.

O PVDTF é tolerante a múltiplas falhas, desde que a restrição 5.4.1 seja respeitada. Essa afirmação pode ser demonstrada do seguinte modo: quando o processo X detecta a falha do processo $\overline{X}_i$, na etapa i, ele enviará mensagens PERGUNTANDO(X,i) para todos os processos de $G(\overline{X}_i, i)$. Desde que a restrição seja mantida, pelo menos um processo irá responder ao processo X. Pelo protocolo, todos os receptores subseqüentes de $\overline{X}_i$ receberão de X a mesma mensagem que $\overline{X}_i$ enviaria se estivesse operacional. Se X também falhar, $\overline{X}_{i+1}$ detectará a falha de X na etapa $i+1$. Desde que a restrição ainda seja satisfeita, $\overline{X}_{i+1}$, após enviar PERGUNTANDO($\overline{X}_{i+1}, i+1$) a todos os processos pertencentes a $G(X, i+1) = G(X, i) \cup G(\overline{X}_i, i)$, irá receber pelo menos uma resposta e o protocolo poderá ser continuado, com os processos operacionais terminando a ação consistentemente.

Os processos falhos, ao se recuperarem, abortam a ação imediatamente, caso esteja em um dos estados q ou ABORTADO$_i$, $(1 \leq i \leq k)$. Se estiverem em algum estado PREPARADO$_i$, $1 \leq i \leq k$, enviam mensagens RECUPERANDO a todos os processos e mudam para o estado RECUPERANDO. Desde que a restrição 5.4.1 tenha sido observada, os processos operacionais estarão no mesmo estado final e portanto enviarão mensagens idênticas em respostas às mensagens RECUPERANDO. Logo, os processos em recuperação mudarão para o mesmo estado em que os processos operacionais estão, após a chegada da resposta. Fica mostrada assim a correção do protocolo.

Além de ser tolerante a múltiplas falhas quando é obedecida a restrição 5.4.1, o PVDTF ainda funciona corretamente, bloqueando, quando a restrição não pode ser satisfeita; isto é, quando X detecta uma falha de $\overline{X}_i$ na etapa i, e todos os processos em $G(\overline{X}_i, i)$ estão falhos, o protocolo não funciona de modo errado, mas permanece bloqueado até que algum processo pertencente a $G(\overline{X}_i, i)$ ou o próprio $\overline{X}_i$ se recupere e possa responder a X[5].

Em [YA91] é proposta uma alteração do protocolo PVDTF com o objetivo de tolerar algumas situações de falhas de partição. O protocolo é obtido retirando-se as operações

[5]É feita uma pequena ressalva com relação a essa propriedade mostrada por Yuan e Agrawala ([YA91]): para que o protocolo possa continuar após o bloqueio, é necessário que o processo X saiba, de algum modo, em que estado os processos $G(\overline{X}_i, i)$, estão para que X possa prosseguir o protocolo e terminar a ação. Essa informação pode ser passada a X pelos processos que estavam falhos, através da mensagem RECUPERANDO, incluindo nela os estado em que estavam quando falharam. Como essa mensagem é enviada a todos os processos, X também a receberá e poderá então se desbloquear.

de recuperação existentes no protocolo PVDTF. Após uma falha de partição, o protocolo continua a funcionar desde que em cada partição haja pelo menos um processo em $G(\overline{X}_i, i)$ quando a falha ocorre durante a etapa i de envio de mensagens.

Capítulo 6

Protocolo de Validação Semibloqueante

Conforme mostrado na Seção 2.3.4, nos protocolos de duas fases os subordinados em estado PREPARADO ficam bloqueados quando o coordenador falha antes de tomar uma decisão. Por outro lado, os protocolos ditos não-bloqueantes não são muito utilizados em sistemas gerenciadores de ações atômicas. O pouco uso desses protocolo deve-se principalmente a dois fatores:

- tais protocolos sempre requerem três ou mais fases em qualquer execução e, portanto, o tempo de execução e o número de mensagens enviadas é maior do que nos protocolos de duas fases, mesmo em execuções sem falhas;
- não existem protocolos não-bloqueantes se há a possibilidade de ocorrer falhas de partição múltipla na rede. Logo a tolerância a falhas desses protocolos não compensa a sua ineficiência.

Neste capítulo é proposto um protocolo intermediário entre os protocolos de duas fases e os protocolos não-bloqueantes. Durante uma execução sem falhas, o protocolo comporta-se como um protocolo de duas fases. Quando surgem falhas no sistema, o protocolo passa a funcionar como um protocolo não-bloqueante. Devido a esse comportamento intermediário, o protocolo é denominado protocolo semibloquenante. O protocolo semibloqueante apresentado neste trabalho é uma extensão do protocolo descrito em [RA93].

O protocolo proposto baseia-se no modelo hierárquico de comunicação entre processos. O protocolo reduz as situações de bloqueio e em determinadas circunstâncias tolera falhas de partição da rede. O modelo hierárquico é utilizado para copiar, nos descendentes e nos ascendentes de um processo, a intenção desse processo em validar ou abortar a ação. Tais cópias permitirão ao protocolo tolerar diversas situações de falhas, incluindo falhas do coordenador e falhas de partição da rede.

Durante uma execução sem falhas do protocolo, o número de mensagens emitidas é comparável ao do protocolo de duas fases hierárquico descrito na Seção 3.6.

6.1 Descrição do Protocolo Semibloqueante

6.1.1 Visão Geral do Protocolo

O coordenador (raiz da árvore de processos) inicia a execução do protocolo enviando ordens aos seus processos filhos para que votem pelo aborto ou validação da ação (primeira fase).

Ao receber a ordem de voto de seu processo pai, um processo intermediário propaga a ordem a seus processos filhos se ele concorda em validar a ação. O voto do processo intermediário, a ser enviado a seu processo pai, será o voto SIM, se ele concorda em validar a ação e todos os seus processos filhos votarem SIM. Se o processo intermediário não concorda em validar a ação ou recebe votos NÃO de seus filhos, ele emite o voto NÃO a seu processo pai e a decisão ABORTE a seus processos filhos. A decisão ABORTE enviada pelo processo intermediário propaga-se até os processos folhas.

Cada processo folha, ao receber uma ordem de voto, envia o seu voto a seu processo pai. Se receber uma decisão ABORTE, o processo folha aborta os efeitos locais da ação.

O coordenador toma uma decisão ao receber os votos dos processos que são seus filhos. A decisão é a de abortar a ação se pelo menos um dos votos recebidos é o voto NÃO. Se todos os votos são SIM, o coordenador decide validar a ação. Em seguida, o coordenador comunica aos seus processos filhos a decisão tomada (segunda fase). Mensagens com a decisão propagam do coordenador em direção às folhas da árvore de processos.

A descrição do protocolo apresentada acima corresponde a uma execução em que não ocorre nenhuma falha de processo ou falha de comunicação. Como pode ser constatado, uma execução sem falhas do protocolo semibloqueante é semelhante a do protocolo de duas fases hierárquico apresentado na Seção 3.6.

As maiores diferenças entre o protocolo semibloqueante e o de duas fases tradicional estão na capacidade do primeiro em tolerar falhas. A organização hierárquica dos processos e a informação implícita em cada voto permitem ao protocolo bloqueante suportar várias situações de falhas. Particularmente, permitem inferir, em determinadas situações, o voto dos processos subordinados falhos e a decisão do coordenador.

Se um processo intermediário I falha antes de emitir seu voto, este pode ser inferido com base nos votos dos filhos de I. Se há algum processo filho de I com voto SIM, é porque I concordaria em validar a ação. Se todos os filhos de I têm voto SIM, o voto de I com certeza seria SIM. Obviamente, se o voto de pelo menos um dos filhos de I for NÃO, ou se um deles tiver abortado a ação, o voto de I deveria ser o voto NÃO. Essa inferência pode ser ainda aplicada de modo recursivo para tentar descobrir qual seria o voto de um processo falho que é filho do processo I. O voto de I representa, portanto, uma combinação das intenções de

cada processo pertencente à subárvore com raiz no processo I. Por isso, o voto de I pode ser considerado como sendo a resposta dessa subárvore à ordem de voto. A definição completa de *resposta de uma subárvore* é dada na Seção 6.1.4.

Quando um processo (coordenador ou subordinado) detecta uma falha, o protocolo passa a funcionar de modo semelhante a um protocolo de três fases. O processo que detectou a falha tenta obter, com base nas respostas das subárvores cujas raízes são filhos do coordenador, uma intenção quanto à finalização da ação. Devido à ocorrência de falhas, é possível que dois processos obtenham intenções diferentes quanto a finalização da ação. Para transformar uma intenção em decisão, um processo deve obter um conjunto mínimo de processos que concordam com essa intenção. Esse número mínimo de processos deve ser obtido de tal modo que não seja possível obter um outro conjunto com número mínimo de processos que concordem com uma intenção contrária. Um conjunto com esse número mínimo de processos é denominado *quorum*. Existem, portanto, dois tipos de quorum: *quorum de aborto* e *quorum de validação*.

Uma tentativa de obter um quorum de um tipo pode ter como resultado, a obtenção de um quorum do tipo intencionado, ou de um quorum do tipo oposto, ou ainda, pode não obter quorum algum.

Um processo intermediário em estado PREPARADO que detecta término do tempo de espera pelo voto de um de seus filhos tenta obter um quorum de votos favorável ao aborto da ação. Caso consiga obter algum quorum, o processo intermediário decide finalizar a ação de acordo com o tipo de quorum obtido. Em seguida, o processo intermediário envia mensagens com a decisão a seus processos filhos. Se não conseguir obter um quorum o processo intermediário permanece aguardando que algum outro processo tome uma decisão.

Um processo subordinado S com voto SIM, ao detectar término do tempo de espera pela decisão que deveria ser enviada pelo seu processo pai, tenta, ele mesmo, obter uma decisão. S tenta inferir a decisão que o coordenador teria tomado, tentando obter as respostas das subárvores cujas raízes são filhos do coordenador. Se todas as respostas das subárvores forem SIM, S tenta obter um quorum para validação da ação. Se pelo menos uma das respostas for NÃO ou ABORTE, S aborta diretamente a ação.

O caso em que o processo subordinado com voto SIM não consegue obter a resposta de uma das subárvores é mais complexo e será tratado com mais detalhes na próxima seção (ver Regra PF 10).

O coordenador, ao detectar término do tempo de espera pelo voto de um de seus filhos, tenta obter a resposta da subárvore cuja raiz corresponde a esse processo filho. Ao obter uma resposta, o coordenador tenta obter um quorum correspondente ao aborto ou à validação da ação.

Ao obter um quorum desejado, o coordenador ou um subordinado efetiva a decisão correspondente ao quorum obtido.

O protocolo proposto tolera também alguns tipos de falha de partição da rede. Se em

uma partição há processos operacionais suficientes para inferir as respostas das subárvores cujas raízes são processos filhos do coordenador e se algum tipo de quorum pode ser obtido, então todos os processos operacionais dessa partição finalizam a ação.

6.1.2 Estados dos Processos

No protocolo semibloqueante, além dos estados: DESCONHECIDO, PREPARADO, ABORTADO E VALIDADO, utilizados também no protocolo de duas fases hierárquico, existem os estados PRÉ-VALIDADO e PRÉ-ABORTADO. Um processo entra em estado PRÉ-VALIDADO se concordar em participar de um quorum para validação da ação. De modo análogo, um processo que concorda em participar de um quorum para aborto da ação entra em estado PRÉ-ABORTADO. Um processo em estado PRÉ-VALIDADO não muda para o estado PRÉ-ABORTADO. O contrário também não ocorre. Ao entrar em estado PRÉ-VALIDADO (ou PRÉ-ABORTADO), um processo força a gravação no log de um registro correspondente a esse novo estado.

Os registros possivelmente encontrados no log são os seguintes: PREPARADO, ABORTADO, PRÉ-VALIDADO, PRÉ-ABORTADO e VALIDADO.

6.1.3 Tipos de Mensagens

Durante a execução do protocolo semibloqueante são utilizados os tipos de mensagens da Tabela 3.1. Além daqueles, quatro novos tipos de mensagens são utilizados:

- a ordem ESQUEÇA;
- o voto PREPARADO;
- o convite PRÉ-VALIDE;
- o convite PRÉ-ABORTE;
- o sinal RECUPERANDO.

A ordem ESQUEÇA indica a um processo que ele não necessita mais armazenar informações sobre a ação. A necessidade dessa mensagem será melhor explicada na Seção 6.1.7 que descreve a segunda fase do protocolo.

O voto PREPARADO é emitido por um processo intermediário que concorda em validar a ação, mas que ainda não tem recebido os votos de seus filhos. O voto PREPARADO é um voto particular de I e não pode ser considerado como resposta de uma subárvore com raiz em I. Um processo intermediário emite o voto PREPARADO somente após receber uma mensagem com a indagação PERGUNTANDO.

A mensagem PRÉ-VALIDE é utilizada pelo coordenador ou por um subordinado com voto SIM na tentativa de obter um quorum para validação da ação. Ao receber uma mensagem

PRÉ-VALIDE, um subordinado entra em estado PRÉ-VALIDADO se já não se encontrar nesse estado ou no estado PRÉ-ABORTADO. Ao entrar em estado PRÉ-VALIDADO, um subordinado envia uma mensagem PRÉ-VALIDADO ao remetente da mensagem PRÉ-VALIDE. Um processo em estado PRÉ-VALIDADO não muda de estado ao receber uma mensagem PRÉ-ABORTE.

A mensagem PRÉ-ABORTE é utilizada pelo coordenador ou por um subordinado com voto SIM na tentativa de obter um quorum para aborto da ação. Ao receber uma mensagem PRÉ-ABORTE, um subordinado entra em estado PRÉ-ABORTADO se já não se encontrar nesse estado ou no estado PRÉ-VALIDADO. Ao entrar em estado PRÉ-ABORTADO, um subordinado envia uma mensagem PRÉ-ABORTADO ao remetente da mensagem PRÉ-ABORTE. Um processo em estado PRÉ-ABORTADO não muda de estado ao receber uma mensagem PRÉ-VALIDE.

A mensagem PREPARE do protocolo semibloqueante é semelhante à mensagem PREPARE do protocolo hierárquico, mas contém uma informação adicional que é a árvore de processos que participam da ação. Essa informação é necessária para que os processos subordinados possam tentar obter uma decisão em caso de falha do coordenador.

A mensagem PERGUNTANDO é enviada a um processo com o objetivo de se obter o voto ou decisão desse processo. Essa mensagem é utilizada apenas em situações anormais.

A mensagem RECUPERANDO é enviada somente por um processo intermediário que se recupera de uma falha. Essa mensagem é enviada apenas se o processo intermediário estava em estado PREPARADO e se tiver recebido uma mensagem PERGUNTANDO após se recuperar.

A descrição das demais mensagens é idêntica à aquelas apresentadas na Seção 3.2.

6.1.4 Resposta de Uma Subárvore

A *resposta de uma subárvore com raiz em um processo* P indica a intenção dos processos pertencentes a essa subárvore quanto ao aborto ou validação da ação. A resposta pode corresponder ao voto de P, ou a uma decisão de P e é obtida com base nas mensagens enviadas pelos processos que constituem essa subárvore.

Existem dois modos de se obter a resposta de uma subárvore com raiz em um processo P. O primeiro deles corresponde a uma mensagem emitida por P durante uma execução sem falhas da primeira fase do protocolo. Essa mensagem corresponde ao voto de P. A mensagem emitida por P corresponde a uma resposta da subárvore, pois é compatível com os votos dos descendentes de P recebidos durante a primeira fase.

O segundo modo é utilizado quando não se pode obter a resposta da subárvore diretamente do processo P. Essa situação ocorre devido a três possibilidades:

a. o processo P ou os canais de comunicação sofrem falhas;

b. o processo P não recebe o voto de um de seus filhos e, portanto, o seu voto é PREPARADO;

c. a mensagem emitida pelo processo P é RECUPERANDO. Esta mensagem é emitida se P for um processo intermediário em estado PREPARADO e recebe uma mensagem PERGUNTANDO ao se recuperar de uma falha[1].

Nesses casos, a resposta da subárvore com raiz em P é obtida de modo recursivo, com base nas respostas das subárvores cujas raízes são processos filhos do processo P. Um processo X que não consegue obter uma resposta diretamente de P envia mensagens PERGUNTANDO aos processos filhos de P com o objetivo de obter as respostas das subárvores das quais eles são as raízes. A resposta da subárvore com raiz em P pode ser inferida pelo processo X com base nas respostas dessas subárvores e pode corresponder:

a. ao voto SIM, se as respostas de todas subárvores forem SIM;

b. ao voto NÃO, se pelo menos uma das respostas das subárvores for NÃO;

c. à decisão ABORTE, se pelo menos umas das respostas for ABORTE;

d. à decisão VALIDE, se pelo menos umas das respostas for VALIDE.

O segundo modo de se obter a resposta de uma subárvore, descrito acima, é utilizado pelo processo X de forma recursiva, enquanto, existir um processo em algum caminho entre P e um nó folha, do qual X não recebe uma mensagem, ou recebe uma mensagem PREPARADO, ou uma mensagem RECUPERANDO. Se esta recursão chega a um processo folha e este também não responde à mensagem PERGUNTANDO, o processo X conclui que a resposta da subárvore com raiz no processo P não pode ser obtida.

Exemplo

A Figura 6.1 ilustra a obtenção da resposta de uma subárvore com raiz no processo P. Na figura, o processo intermediário X com voto SIM detecta término do tempo de espera pela decisão do coordenador (Fig. 6.1a). X tenta obter, de modo recursivo, as respostas das subárvores cujas raízes são processos filhos do coordenador. Portanto, X envia mensagem PERGUNTANDO ao processo P (Fig. 6.1b). Ao detectar término do tempo de espera pela resposta de P (Fig. 6.1c), X envia mensagens PERGUNTANDO aos processos filhos de P (Fig. 6.1d). Porém, um desses processos, o processo S_3 também está falho (Fig. 6.1e). X tenta obter a resposta da subárvore com raiz em S_3, enviando mensagens aos descendentes desse processo (Fig. 6.1f). Com base nas respostas dos processos filhos de S_3, o processo X conclui que a resposta de S_3 é SIM. Como as respostas de S_1 e S_2 foram SIM, X consegue obter a resposta da subárvore P que é SIM.

[1] O voto PREPARADO e a mensagem RECUPERANDO não servem como resposta de uma subárvore com raiz em P, pois indicam apenas que P é um processo intermediário em estado PREPARADO que concorda em validar a ação, mas não traz nenhuma informação sobre as intenções dos descendentes do processo P.

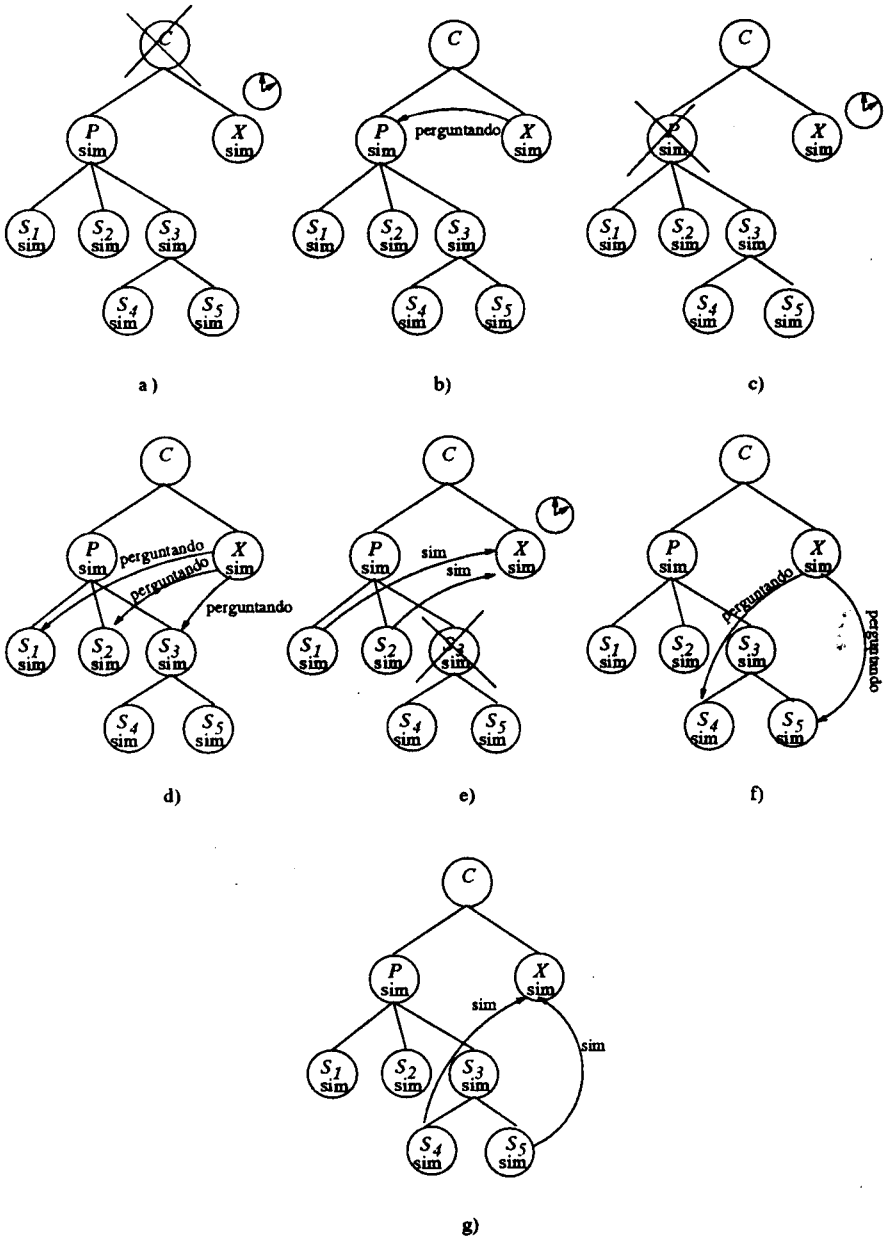

Figura 6.1: Exemplo de obtenção de resposta de uma subárvore após a ocorrência de falhas no sistema.

A Figura 6.2 ilustra uma situação em que a resposta de uma subárvore com raiz no processo P não pode ser obtida. No exemplo da figura tenta-se obter de modo recursivo a resposta da subárvore S_3, mas um dos processos folha dessa subárvore também está falho. Não é possível obter-se a resposta da subárvore S_3 e, como conseqüência, não é possível obter-se uma resposta para a subárvore com raiz em P.

Obs.: Um processo X pode receber mensagens PRÉ-VALIDADO ou PRÉ-ABORTADO em respostas às indagações PERGUNTANDO por ele enviadas. Essas mensagens funcionam como um convite para participação em algum quorum. Nesse caso, X interrompe a tentativa de obter a resposta da subárvore e passa a pertencer ao quorum correspondente à primeira dentre as mensagens PRÉ-VALIDADO ou PRÉ-ABORTADO recebidas.

6.1.5 Primeira Fase do Protocolo

Primeira Fase Quando Não Ocorrem Falhas

A primeira fase do protocolo é a fase em que os votos são coletados e uma decisão quanto à finalização da ação é obtida.

A descrição é feita por meio de um conjunto de regras ordenadas segundo a seqüência lógica de execução do protocolo.

Regra PF 1 *O coordenador ao iniciar a primeira fase entra em estado* PREPARADO *e envia mensagens com a ordem* PREPARE *a seus filhos.*

Regra PF 2 *Um processo intermediário I entra em estado* PREPARADO *se e somente se receber a ordem* PREPARE *de seu pai e concordar em validar a ação. Nesse caso, ele envia mensagens com a ordem* PREPARE *aos seus filhos.*

Regra PF 3 *Se um processo intermediário I ao receber a ordem* PREPARE *não concordar em validar a ação, ele entra em estado* ABORTADO, *envia o voto* NÃO *ao seu pai e a decisão* ABORTE *a seus filhos.*

O voto NÃO de I corresponde à resposta da subárvore com raiz em I.

Regra PF 4 *Um processo folha F entra em estado* PREPARADO *se e somente se receber a ordem* PREPARE *de seu pai e concordar em validar a ação. Nesse caso, F envia o voto* SIM *a seu processo pai. Se não concordar em validar a ação, F entra em estado* ABORTADO *e envia o voto* NÃO *a seu processo pai.*

O voto de um processo folha F corresponde à resposta da subárvore unitária com raiz em F.

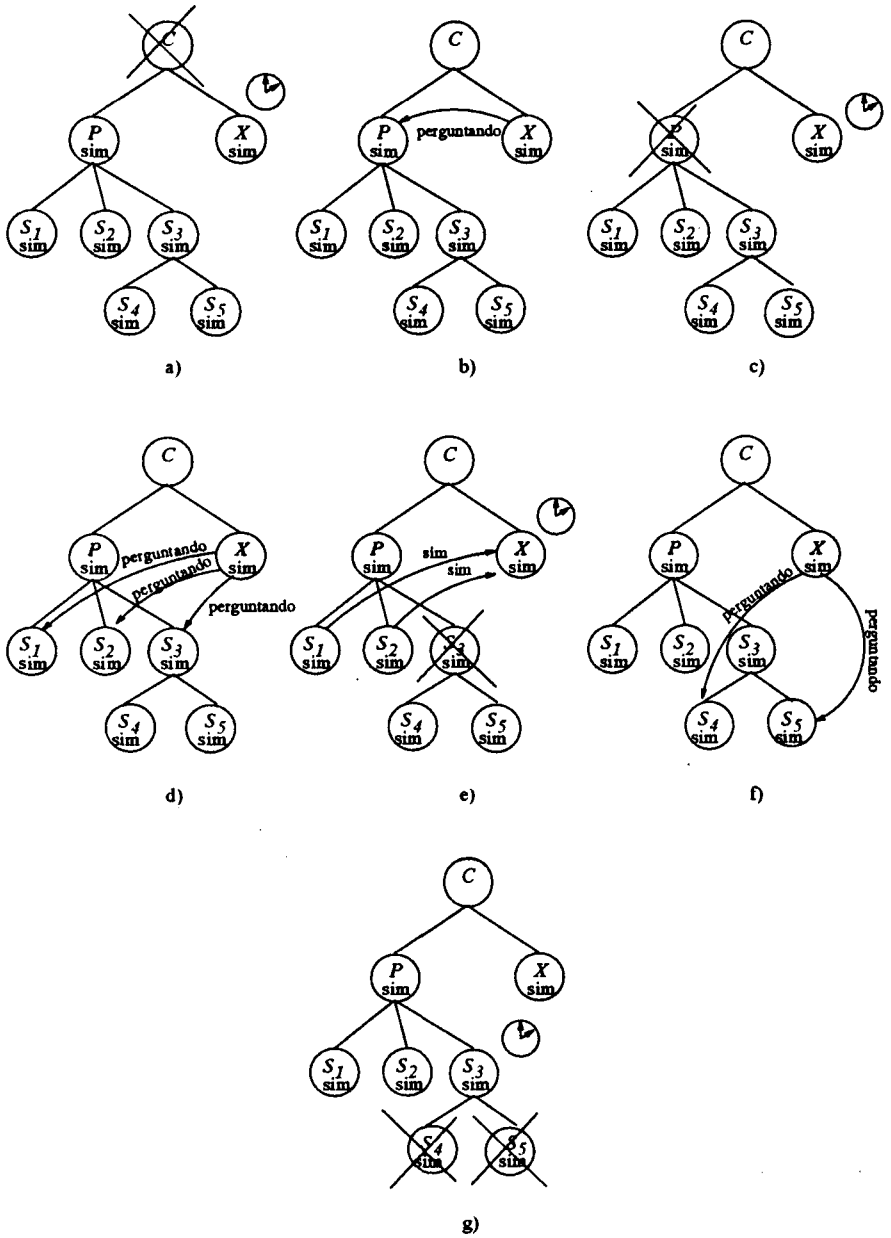

Figura 6.2: Exemplo em que não é possível obter a resposta de uma das subárvores.

Regra PF 5 *Um processo intermediário I envia o voto* SIM *a seu pai se e somente se I estiver em estado* PREPARADO *e receber os votos de todos os seus filhos e eles forem* SIM.

Se um processo intermediário I em estado PREPARADO *recebe o voto* NÃO *de algum de seus filhos, ele entra em estado* ABORTADO, *envia o voto* NÃO *a seu pai e envia a decisão* ABORTE *a seus filhos cujos votos foram diferentes de* NÃO.

O voto de I representa a resposta da subárvore cuja raiz é o processo I. Os votos propagam-se no sentido folhas-raiz na árvore de processos.

Regra PF 6 *O coordenador toma uma decisão ao obter as respostas das subárvores cujas raízes são seus filhos. Tais respostas correspondem aos votos desses processos. A decisão é a de validar a ação se todas as respostas forem* SIM *e será a de abortar a ação se pelo menos uma das respostas for* NÃO.

Por enquanto, está sendo considerada uma execução sem falhas do protocolo, portanto, o coordenador não recebe nenhuma resposta do tipo VALIDE, ABORTE, PRÉ-VALIDADO ou PRÉ-ABORTADO.

Exemplo

A Figura 6.3 ilustra um exemplo de funcionamento sem falhas da primeira fase do protocolo para uma ação que envolve oito processos. Na Figura 6.3a, de acordo com a Regra PF 1, o coordenador C envia mensagens PREPARE a seus filhos I_1, F_1 e I_2. Na Figura 6.3b, conforme a Regra PF 2, o processo intermediário I_1 entra em estado PREPARADO. Ao entrar em estado PREPARADO, I_1 envia ordens PREPARE aos seus filhos F_2 e F_3 e permanece aguardando seus votos para que possa emitir o seu próprio voto a C. O processo I_2 não concorda em validar a ação, logo, pela regra PF 3, ele entra em estado ABORTADO, envia o voto NÃO a seu pai e a decisão aborte a seus filhos F_4 e F_5. Ainda na Figura 6.3b, o processo folha F_1, filho do coordenador não concorda com a validação da ação. De acordo com a Regra PF 4, ele entra em estado ABORTADO e envia o voto NÃO a seu pai. Na Figura 6.3c, os filhos de I_1 recebem a ordem PREPARE e ambos concordam com a validação da ação. Como F_2 e F_3 são processos folhas, eles emitem diretamente o voto SIM a I_1. Pela Regra PF 5, I_1 envia o voto SIM a C.■

Primeira Fase Quando Ocorrem Falhas

Durante a primeira fase, podem ocorrer falhas de processos e dos canais de comunicação. As falhas de um processo bem como as falhas de comunicação são detectadas por término do tempo de espera. Um processo operacional O aguarda a chegada de uma mensagem de um processo N durante o tempo de espera. Se a mensagem não chegar nesse intervalo de tempo, O assume que ocorreu uma falha em N ou no canal de comunicação que liga O a N.

Durante a primeira fase existem as seguintes situações em que um processo aguarda a mensagem de outro e que portanto pode ocorrer término do tempo de espera:

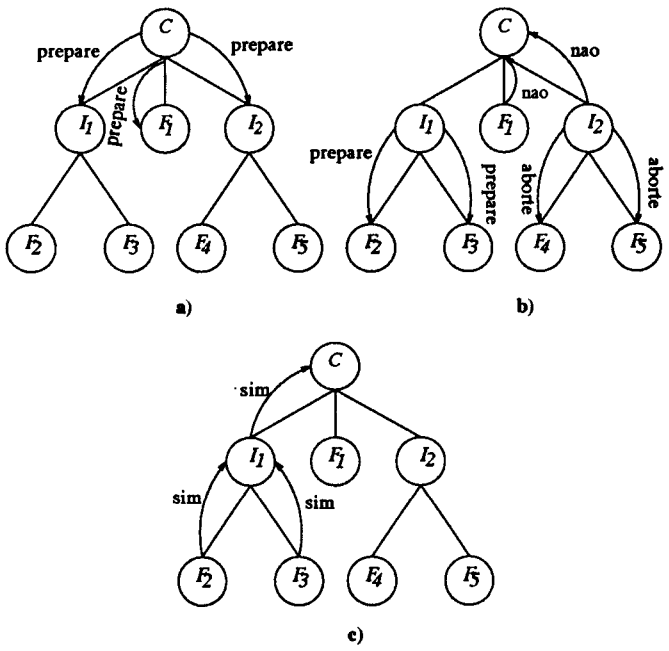

Figura 6.3: Ilustração do funcionamento sem falha da primeira fase do protocolo.

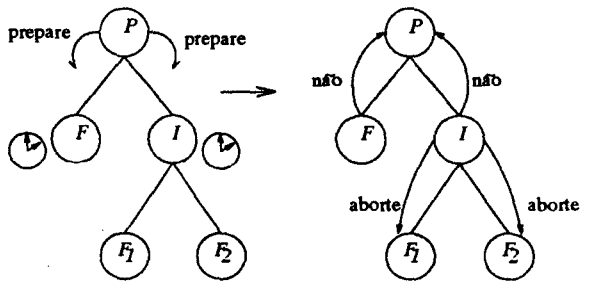

Figura 6.4: Ilustração do funcionamento do protocolo quando a mensagem PREPARE não chega durante o tempo de espera.

a. cada subordinado em estado DESCONHECIDO aguarda a mensagem PREPARE de seu pai;

b. cada processo intermediário em estado PREPARADO aguarda os votos de seus filhos;

c. um subordinado em estado PREPARADO e com voto SIM aguarda a decisão que deve vir de seu pai;

d. o coordenador aguarda os votos de seus filhos.

Quando ocorre o término do tempo de espera na situação descrita no item (a), o protocolo utiliza a seguinte regra:

Regra PF 7 *Se um processo subordinado em estado* DESCONHECIDO *não recebe a ordem* PREPARE *durante o tempo de espera, ele entra em estado* ABORTADO *e envia o voto* NÃO *a seu pai. Se ele for também um processo intermediário, ele envia a decisão* ABORTE *a seus filhos.*

Exemplo

A Figura 6.4 ilustra o comportamento dos subordinados quando ocorre término do tempo de espera estando eles em estado DESCONHECIDO e aguardando a chegada de uma mensagem do processo pai. O processo folha F, ao detectar término do tempo de espera, de acordo com a Regra PF 7, entra em estado ABORTADO e envia o voto NÃO a seu pai. O processo intermediário I, ao detectar término do tempo de espera, entra em estado ABORTADO, envia o voto NÃO a seu pai P e a decisão aborte a seus filhos F_1 e F_2. ■

Quando ocorre término do tempo de espera na situação do item (b) o processo intermediário utiliza a seguinte regra:

Regra PF 8 *Se os votos dos filhos de um processo intermediário I em estado* PREPARADO *não chegam durante o tempo de espera, I entra em estado* PRÉ-ABORTADO *e inicia uma tentativa para obter um quorum de aborto da ação. Caso consiga obter um quorum, I finaliza a ação de acordo com o tipo de quorum obtido e inicia a execução da segunda fase do protocolo. Se um quorum não for obtido I permanece aguardando a chegada de alguma decisão. Se após ter iniciado a tentativa de obtenção de um quorum de aborto I receber os votos (atrasados) de seus processos filhos, I os ignora e continua a tentativa se eles forem todos votos* SIM. *Se algum deles for o voto* NÃO, *I interrompe a tentativa de obtenção de quorum e decide abortar a ação. No último caso, o processo I inicia a execução da segunda fase do protocolo.*

Regra PF 9 *Um processo intermediário I que ainda não tenha recebido os votos de seus filhos, ao receber uma mensagem* PERGUNTANDO, *responde:*

a. o voto PREPARADO *se ainda não tiver detectado término do tempo de espera;*

b. o voto PRÉ-ABORTADO, *se já tiver iniciado uma tentativa de obter quorum de aborto.*

Em ambos os casos, se I recebe os votos de seus filhos após ter respondido a uma mensagem PERGUNTANDO, *I os ignora, se eles forem todos votos* SIM. *Isto significa que I não terá voto* SIM *nesse caso. Se um dos votos for* NÃO, *I decide abortar a ação e inicia a execução da segunda fase.*

Na situação especificada no item (c) acima, um processo subordinado em estado PREPARADO e com voto SIM aguarda a decisão que deve vir de seu pai. A regra abaixo indica como esse subordinado procede quando ele não recebe a decisão durante o tempo de espera:

Regra PF 10 *Se um subordinado S em estado* PREPARADO *com voto* SIM *não recebe uma decisão até o término do tempo de espera, ele mesmo tenta obter uma decisão. S tenta obter a resposta da árvore de processos cuja raiz é o coordenador através do envio de mensagens* PERGUNTANDO. *Caso o coordenador esteja inacessível, a resposta da árvore de processos corresponde a uma combinação das respostas das subárvores cujas raízes são processos filhos do coordenador conforme explicado na seção 6.1.4. Ao obter as respostas dessas subárvores, S decide abortar a ação se pelo menos uma das respostas for* NÃO *ou* ABORTE. *Se alguma resposta for* VALIDE, *S decide validar a ação. Se todas as respostas forem* SIM, *S tenta obter um quorum para validar a ação.*

Se S recebe uma mensagem do tipo PRÉ-ABORTADO *ou* PRÉ-VALIDADO, *ele entra no estado correspondente a primeira mensagem recebida. Caso não receba posteriormente nenhuma resposta* VALIDE, ABORTE *ou* NÃO, *S tenta obter um quorum correspondente ao seu estado (*PRÉ-ABORTADO *ou* PRÉ-VALIDADO*).*

Ao obter uma decisão, S inicia a execução da segunda fase do protocolo. Caso S não consiga obter a resposta de uma das subárvores, S inicia uma tentativa de obtenção de

quorum de aborto somente se tiver recebido pelo menos um voto PREPARADO *durante a tentativa de obter as respostas. Se não tiver recebido nenhum voto* PREPARADO *(i.e., os votos recebidos foram apenas votos* SIM*), o processo S não tenta formar quorum algum; apenas permanece aguardando o recebimento de alguma decisão.*

Na situação correspondente ao item (d), o coordenador aguarda os votos de seus processos filhos. Caso um dos votos não seja recebido durante o tempo de espera, o coordenador procede conforme descrito na regra abaixo:

Regra PF 11 *Se o coordenador em estado* PREPARADO *não recebe o voto de um de seus filhos durante o tempo de espera, ele tenta obter a resposta da subárvore cuja a raiz é esse processo. O coordenador envia mensagens* PERGUNTANDO *aos descendentes desses processos conforme explicado na Seção 6.1.4. Se a resposta dessa subárvore não puder ser obtida, o coordenador inicia uma tentativa para obtenção de um quorum de aborto da ação. Se o coordenador consegue obter as respostas de todas as subárvores e elas forem* SIM*, o coordenador inicia uma tentativa de obter um quorum para validação da ação. Se pelo menos uma das respostas for* NÃO *ou* ABORTE*, o coordenador decide abortar a ação.*

Se o coordenador recebe uma mensagem do tipo PRÉ-ABORTADO *ou* PRÉ-VALIDADO*, ele entra no estado correspondente ao tipo da primeira mensagem recebida. Caso não receba posteriormente nenhuma resposta* VALIDE*,* ABORTE *ou* NÃO*, o coordenador tenta obter um quorum correspondente ao seu estado (*PRÉ-ABORTADO *ou* PRÉ-VALIDADO*). Após obter uma decisão, o coordenador passa a executar a segunda fase do protocolo, através da aplicação da Regra SF 1 ou Regra SF 2.*

Caso não consiga obter a resposta de uma das árvores, o coordenador tenta obter um quorum de aborto para a ação até que consiga uma decisão.

6.1.6 Fase de Obtenção do Quorum

A tentativa de obtenção de um quorum corresponde a uma fase intermediária à primeira e à segunda fases do protocolo.

A fase de obtenção do quorum é necessária pois dois ou mais processos podem ter intenções diferentes quanto à finalização da ação, ao final da primeira fase, devido a ocorrência de falhas.

Um exemplo é dado na Figura 6.5, onde o processo intermediário I_1 possui voto SIM e detecta término do tempo de espera pela decisão do coordenador. De acordo com a Regra PF 10, o processo I_1 tenta obter a resposta da subárvore com raiz em I_2. O processo I_1 não consegue obter a resposta dessa subárvore, pois I_2 está em estado PREPARADO mas não possui voto e a comunicação entre I_1 e o processo folha F_3 não é possível. Portanto, a intenção de I_1 é a de abortar a ação. Contudo, I_1 não pode abortar diretamente a ação, pois outro processo pode obter a decisão de validá-la. Por exemplo, conforme ilustra a Figura

6.5b, o coordenador ao detectar término do tempo de espera pelo voto de I_2, tenta obter a resposta da subárvore com raiz em I_2, enviando mensagens aos processos folhas F_3 e F_4. A comunicação entre o coordenador e esses processos não está interrompida, portanto, o coordenador obtém o voto SIM como resposta da subárvore I_2 e a sua intenção será a de validar a ação, uma vez que ele recebe o voto SIM de I_1.

Para garantir que apenas um tipo de decisão seja tomada, um processo deve garantir a existência de um quorum de processos que concordem com sua intenção de finalizar a ação antes que uma decisão possa ser tomada.

Algoritmo para Obtenção de Quorum

O protocolo semibloqueante utiliza para obtenção de quorum o algoritmo proposto por Agrawal e Abbadi [AA91]. O algoritmo baseia-se nas definição de *coterie* que é dada a seguir.

Uma *coterie* C é um conjunto de conjuntos onde cada conjunto g é denominado *quorum*. Os quorums de uma coterie C obedecem às seguintes condições:

- *Propriedade de Interseção:* se g e h são quorums em C, então $g \cap h \neq \{\}$.
- *Propriedade de Minimalidade*: não há dois quorums g e h em C tais que g é um superconjunto de h.

Coteries podem ser utilizadas para obtenção de exclusão mútua em sistemas distribuídos. Por exemplo, para se obter acesso mutuamente exclusivo a um recurso da rede, um processo P deve requerer permissão de um quorum g de processos para realizar tal acesso. Se todos os processos de g concordarem, o processo P pode utilizar o recurso. Uma vez que qualquer par de quorums tem pelo menos um processo em comum, a exclusão mútua fica garantida.

O algoritmo de Agrawal e Abbadi é utilizado para obtenção de quorum em árvore de processos. Para simplificar a apresentação do algoritmo, será considerado que a árvore de processos é uma árvore binária completa. Posteriormente, o algoritmo será estendido para o caso de árvores genéricas.

A Figura 6.7 descreve o algoritmo de obtenção de quorum em árvore binária. Para tentar obter um quorum um processo chama a função recursiva ObterQuorum passando o identificador do processo raiz da árvore de processos como parâmetro. A função Consulta assume valor verdadeiro se um processo consultado concorda em participar do quorum e assume valor falso caso o processo não concorde em ser um membro do quorum ou não responde à consulta.

O algoritmo tenta construir um quorum formado por processos que pertençam a algum caminho que inicia com o processo raiz e termina com algum processo folha. Na árvore da Figura 6.6, os quorums possíveis segundo essa estratégia seriam $\{1, 2, 4\}$, $\{1, 2, 5\}$, $\{1, 3, 6\}$ e $\{1, 3, 7\}$. Observe que os quorums satisfazem as propriedades de interseção e minimalidade. Caso um processo P_i de um caminho esteja falho ou impossibilitado de se comunicar com o

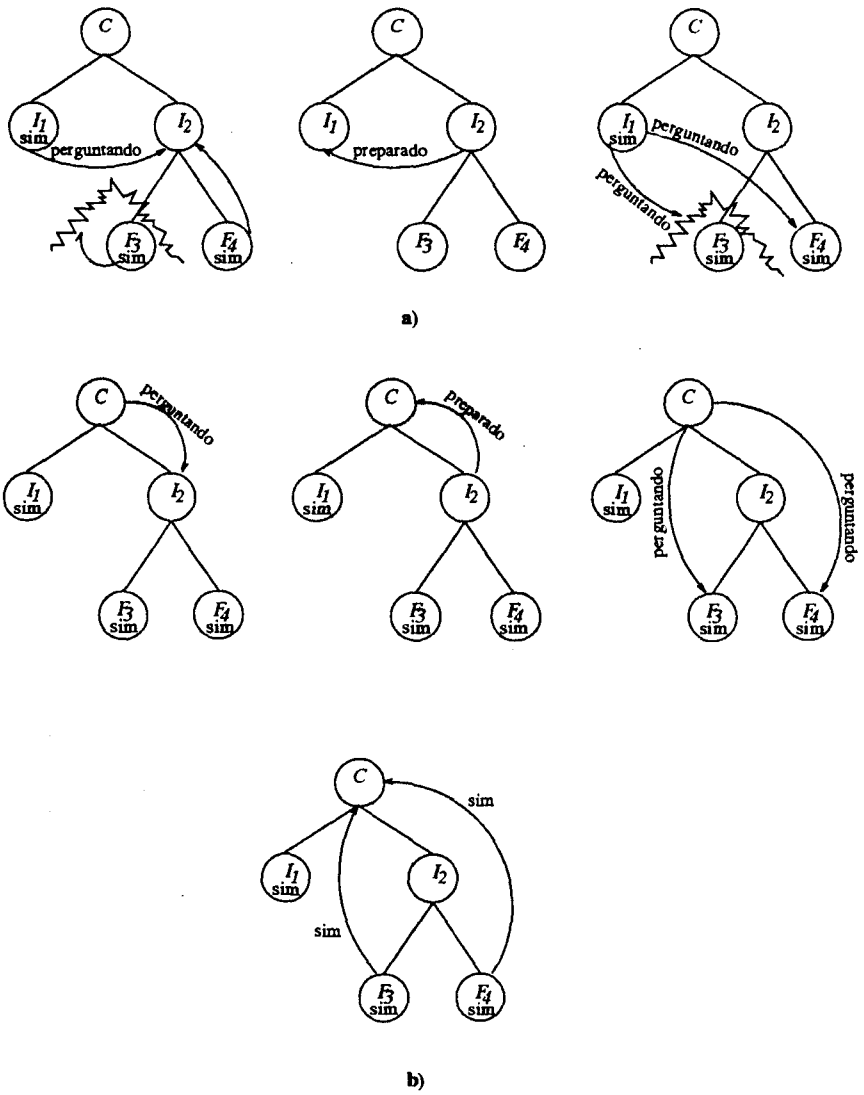

Figura 6.5: Exemplo em que um subordinado e o coordenador obtêm intenções opostas quanto à finalização da ação.

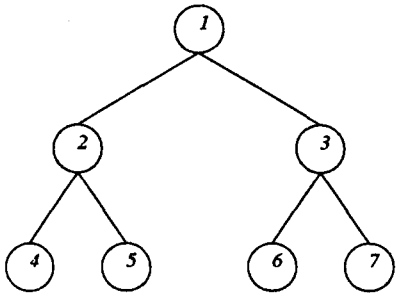

Figura 6.6: Exemplo de árvore de processos.

processo que tenta obter o quorum, o algoritmo tenta substituir P_i por dois caminhos com inícios nos dois nós filhos de P_i e términos em nós folhas. Por exemplo, na Figura 6.6, se não fosse possível a comunicação com o processo 1 (processo raiz), então os seguintes conjuntos seriam quorums válidos: {2, 4, 3, 6}, {2, 5, 3, 6}, {2, 4, 3, 7} e {2, 5, 3, 7}. Se os processos 2 e 3 estivessem inacessíveis, os seguintes quorums seriam válidos: {1, 4, 5} ou {1, 6, 7}. Se ambos os processos 1 e 2 estivessem falhos ou isolados, os quorums possíveis seriam: {4, 5, 3, 6} e {4, 5, 3, 7}. De modo semelhante, se ambos os processos 1 e 3 estivessem inacessíveis, os quorums possíveis seriam: {2, 4, 6, 7} e {2, 5, 6, 7}. Finalmente, se os processos 1, 2 e 3 estivessem incomunicáveis, o único quorum possível seria {4, 5, 6, 7}.

O conceito de coterie pode ser utilizado também com o objetivo de se obter exclusão mútua quanto à decisão de finalizar a ação: se um quorum de processos concorda com um tipo de decisão, a propriedade de interseção garante a impossibilidade de existir outro quorum que concorde com um tipo de finalização diferente.

O algoritmo da Figura 6.7 pode ser adaptado para atender a esse objetivo. Tal adaptação é feita utilizando-se as funções ObterQuorumAborto e ObterQuorumValidação em substituição à função ObterQuorum.

O algoritmo correspondente à função ObterQuorumAborto é mostrado na Figura 6.9. A função ObterQuorumValidação é semelhante à função ObterQuorumAborto, exceto que a função de consulta é a função ConsultaValidação.

A função ConsultaValidação convida um processo a participar da formação de um quorum de validação, através do envio de mensagem PRÉ-VALIDE a esse processo. De modo análogo, a função ConsultaAborto tenta obter um participante para um quorum de aborto, enviando uma mensagem PRÉ-ABORTE.

O tipo quorums utilizado pelas funções ObterQuorumAborto e ObterQuorumValidação é mostrado na Figura 6.8.

Ambas as funções de consulta podem devolver três tipos de resultados: TÉRMINO-TEMPO-ESPERA, PRÉ-VALIDADO e PRÉ-ABORTADO. Essas funções possibilitam maior versatilidade

```
Função ObterQuorum(árvore: ÁRVORE): conjunto_processos;
var esquerda, direita: conjunto_processos;
início
   se vazia(árvore) então retorne({}); senão
   se Consulta(árvore↑ .processo) então
   retorne({árvore↑ .processo} ∪ ObterQuorum(árvore↑.sub_árvore_esq);
   ou
   retorne({árvore↑ .processo} ∪ ObterQuorum(árvore↑.sub_árvore_dir); senão
   início
      esquerda ← ObterQuorum(árvore↑.sub_árvore_esq);
      direita ← ObterQuorum(árvore↑.sub_árvore_dir);
      se (esquerda = {} ou direita = {}) então
      (* Não foi possível obter um quorum *)
      termine; senão
      retorne(esquerda ∪ direita);
   fim
fim
```

Figura 6.7: Função para obter quorum em árvore em um sistema distribuído.

```
tipo
quorums = registro conjunto-aborto, conjunto-validação : conjunto-processos fim
```

Figura 6.8: Definição do tipo quorums.

```
Função ObterQuorumAborto(árvore: ÁRVORE, ref qaborto, ref qvalidação : lógico);
var resultado : enumeração {pré-validado, pré-abortado, término-tempo-espera}
x, y : quorums; qvalidação_esq, qaborto_esq: lógico;
início
   se vazia(árvore) então
   início
      x.conjunto_validação ← x.conjunto_aborto ← {}; qvalidação ← qaborto ← verdadeiro;
   fim
   senão início
         resultado ← ConsultaAborto(árvore ↑ .processo);
         se resultado ≠ término-tempo-espera então
         início
            x ← ObterQuorumAborto(árvore↑ .esquerda, qaborto, validação);
            se ((resultado = pré-validado) e (não qvalidação)) ou (resultado = pré-abortado) e
            (não qaborto)) então
            início
               y ← ObterQuorumAborto(árvore↑direita, qaborto, qvalidação);
               x.conjunto_aborto ← x.conjunto_aborto ∪ y.conjunto_aborto;
               x.conjunto-validação ← x.conjunto_validação ∪ y.conjunto_validação;
            fim
            se (resultado = pré-validado) e qvalidação então
            início
               qaborto ← falso;
               x.conjunto_validação ← x.conjunto.validação ∪ {árvore↑.processo};
            fim
            senão se (resultado = pré-abortado) e qaborto então
            início
               qvalidação ← falso;
               x.conjunto_aborto ← x.conjunto_aborto ∪ {árvore↑.processo};
            fim
            retorne (x);
         fim
         senão início
                  x ← ObterQuorumAborto(árvore↑ .esquerda, qaborto_esq, qvalidação_esq);
                  y ← ObterQuorumAborto(árvore↑.direita, qaborto, qvalidação);
                  se ((qvalidação_esq e qaborto) ou (qaborto_esq e qvalidação)) então
                  qvalidação ← qaborto ← falso;
                  senão início
                           x.conjunto_validação ← x.conjunto_validação ∪ y.conjunto_validação;
                           x.conjunto_aborto ← x.conjunto_aborto ∪ y.conjunto_aborto;
                           retorne (x);
                  fim
         fim
   fim
fim
```

Figura 6.9: Função ObterQuorumAborto.

ao protocolo semibloqueante, pois permitem a um processo obter um tipo de quorum oposto ao que ele pretende obter. Por exemplo, um processo P em estado PREPARADO e com voto SIM pode tentar obter um quorum de aborto para a ação que está sendo executada. O processo P pode chamar a função ObterQuorumAborto através do seguinte comando Q = ObterQuorumAborto(árvore, qvalidação, qaborto). Q.conjuntoAborto constitui um quorum de aborto se o parâmetro qaborto, passado por referência, contiver o valor verdadeiro. Um quorum de validação corresponde a Q.conjuntoValidação se qvalidação for verdadeiro. A impossibilidade de obtenção de qualquer quorum é identificada quando qvalidação = qaborto.

Uma vez obtido um quorum (de aborto ou validação), o processo P decide finalizar a ação de acordo com o tipo de quorum obtido. Em seguida, o processo P inicia a segunda fase do protocolo semibloqueante para propagar a decisão obtida.

Correção do Algoritmo de Obtenção de Quorum em Árvore

Demonstrar a correção das funções de obtenção de quorum consiste em demonstrar que o quorum em árvore resultante da execução possui as propriedades de intersecção e minimalidade. Essa demonstração é dada através da prova de correção do Teorema abaixo, apresentada por Agrawal e Abaddi [AA91]:

Teorema 6.1.1 *O quorum em árvore satisfaz as propriedades de interseção e minimalidade de coterie.*

Prova. A prova é por indução na altura da árvore binária que corresponde à árvore de processos:

Base. Seja uma árvore binária de altura 1, isto é, a árvore contem apenas um processo $s1$. Um quorum de aborto igual a $\{s1\}$ possui as propriedades de minimalidade e interseção. Igualmente, um quorum de validação possui as duas propriedades acima referidas.

Hipótese Indutiva: considera-se que o enunciado do Teorema seja verdadeiro para árvores de altura h.

Passo da Indução. Seja uma árvore binária de altura $h + 1$. Seja $s1$ a raiz dessa árvore. Pode-se verificar pelo algoritmo da Figura 6.9 que qualquer quorum de aborto da árvore pertence a uma das três classes:

1. $\{s1\} \cup \{$membros do quorum de aborto da subárvore esquerda$\}$;
2. $\{s1\} \cup \{$membros do quorum de aborto da subárvore direita$\}$ ou
3. {membros do quorum de aborto da subárvore esquerda} $\cup$ {membros do quorum de aborto da subárvore direita}.

Pela hipótese indutiva os quorums das subárvores direita e esquerda possuem as duas propriedades de coterie. Além disso, pode ser verificado que os quorums da classe 1 possuem

uma interseção não vazia com os quorums das classes 2 e 3. Porém, não são todos os membros da classe 1 que pertencem também às classes 2 e 3. Condições semelhantes ocorrem para as classes 2 e 3. Logo, o conjunto de quorums de aborto construído para uma árvore de altura $h + 1$ é uma coterie. Argumentos semelhantes podem ser utilizados para demonstrar que os quorums de validação obtido pela função ObterQuorumValidação também possui as propriedades de interseção e minimalidade. Portanto, as propriedades de coterie são satisfeitas por um quorum em árvore obtido de uma árvore binária de altura qualquer.

Comentários

O número de processos necessários para se obter um quorum é um fator importante, pois influencia no custo de execução do algoritmo, já que o número de mensagens é proporcional ao tamanho do quorum.

O algoritmo de obtenção de quorum em árvore requer no melhor caso $\lceil \log_2 n \rceil$ processos para formarem um quorum. Esse valor pode ocorrer em duas situações:

1. quando o quorum corresponde a um caminho que inicia no processo raiz e termina em um processo folha, ou

2. quando um processo no nível superior ao do processo folha está falho.

O algoritmo pode tolerar falhas de $n - \lceil \log_2 n \rceil$ processos específicos. Por exemplo, na Figura 6.6, o quorum $\{1, 2, 3, 4\}$ pode ser formado, mesmo que os processos $2, 3, 6$ e 7 estejam falhos ou tenham intenção contrária quanto ao tipo de finalização intencionada pelos processos desse quorum.

Conforme demonstrado em [AA91], no pior caso são necessários $\lceil (n + 1)/2 \rceil$ processos para obtenção de um quorum. Esse caso ocorre, por exemplo, quando apenas os processos folhas estão ativos e concordam em participar do quorum.

O algoritmo proposto por Agrawal e Abbadi apresenta, portanto, a propriedade de *degradação comportada* ("gracefull degradation"), o que significa que seu custo é considerado baixo quando não ocorrem falhas (apenas $\lceil \log_2 n \rceil$ mensagens) e aumenta com o acréscimo de ocorrências de falhas.

Em alguns casos, o algoritmo é incapaz de obter um quorum após a falha de apenas $\lceil \log_2 n \rceil$ processos. Por exemplo, se os processos $1, 2, 4$ Na Figura 6.6 estão falhos não é possível obter-se nenhum quorum em árvore. Entretanto, se outro método fosse utilizado, por exemplo o de obtenção de quorum por maioria, um quorum formado pelos processos $3, 5, 6, 7$ restantes poderia ser obtido.

Extensões

O algoritmo de obtenção de quorum em árvore pode ser estendido para árvores não binárias. Qualquer conjunto de processos que contém um caminho da raiz a um processo folha forma um quorum. Se um processo P desse caminho está falho ou não concorda em participar do quorum, então um quorum deve conter os processos pertencentes aos caminhos que iniciam nos filhos de P e terminam em processos folhas.

O algoritmo também pode ser estendido para ser utilizado com árvores não completas. Um processo pode ter um número arbitrário de processos filhos. Um quorum em árvore pode ser obtido, desde que um processo falho ou que não participa do quorum seja substituído por todos os caminhos possíveis que tenham início nos processos filhos desse processo e que terminem em processos folhas.

A última extensão proposta acima faz com que a propriedade de minimalidade seja violada. Por exemplo, considere uma árvore formada por dois processos P_1 e P_2. O processo P_2 é filho do processo P_1. De acordo com o algoritmo, a coterie correspondente a esta árvore seria $\{\{P_1, P_2\}, \{P_2\}\}$. Obviamente, os quorums não satisfazem à propriedade de minimalidade. Entretanto, essa propriedade é necessária para reduzir custos de envio de mensagens, mas não é necessária para garantir a exclusão mútua.

Se a minimalidade deve ser garantida (por questão de eficiência), pode-se alterar as funções de obtenção de quorum para que retornem uma indicação de erro quando ocorre a falha de um processo que possui apenas um descendente.

6.1.7 Segunda Fase do Protocolo Semibloqueante

A segunda fase do protocolo tem início com o envio de mensagens contendo a decisão quanto à finalização da ação. Essas mensagens são enviadas pelo coordenador e/ou por algum subordinado que tenha tomado uma decisão.

As primeiras duas regras descrevem o início da segunda fase quando ele ocorre no coordenador.

Regra SF 1 *Se o coordenador decide abortar a ação, ele entra em estado* ABORTADO *e envia a decisão* ABORTE *a todos os seus filhos cujos votos foram diferentes de* NÃO *ou que ainda não abortaram a ação. Caso o coordenador tenha enviado mensagens* PERGUNTANDO *ou tenha iniciado a fase de obtenção de quorum, ele envia mensagens a todos os processos cujas respostas foram diferentes de* ABORTE *ou* NÃO.

Regra SF 2 *Se o coordenador decide validar a ação, ele envia a decisão* VALIDE *a todos os seus filhos que ainda não validaram a ação e a todos os processos que responderam às mensagens* PERGUNTANDO *que o coordenador possa ter feito. Ao obter as mensagens* RECEBI *de todos os seus filhos, o coordenador entra em estado* DESCONHECIDO *e envia a mensagem* ESQUEÇA *a eles.*

Se um subordinado inicia a segunda fase, ele o faz segundo as as duas regras abaixo:

Regra SF 3 *Um subordinado S que tenha decidido abortar a ação procede do seguinte modo:*

1. *se S for um processo intermediário, ele envia mensagens aborte a todos os seus filhos cujos votos foram diferentes de* NÃO *e que ainda não decidiram abortar a ação;*

2. *S envia mensagens com a decisão* ABORTE *aos processos que enviaram respostas às indagações* PERGUNTANDO *e aos processos que enviaram mensagens durante a fase de obtenção de quorum. As mensagens com a decisão* ABORTE *são enviadas apenas àqueles processos que enviaram mensagens diferentes de* NÃO *ou* ABORTE.

Regra SF 4 *Um subordinado S que tenha decidido validar a ação procede do seguinte modo:*

- *se S for um processo intermediário, ele envia a decisão* VALIDE *a todos os seus filhos. Em seguida, S aguarda os sinais* RECEBI *de seus filhos e a ordem* ESQUEÇA *de seu pai;*

- *S envia a decisão* VALIDE *aos processos que enviaram respostas durante a fase de obtenção do quorum.*

Os processos intermediários ao receberem a decisão comportam-se do seguinte modo:

Regra SF 5 *Cada processo intermediário I ao receber a decisão:*

- *entra em estado* ABORTADO *se a decisão for* ABORTE. *Nesse caso, envia mensagens* ABORTE *a seus filhos; ou*

- *entra em estado* VALIDADO *se a decisão for* VALIDE. *Nesse caso, I executa os seguintes passos:*

  - *envia a decisão* VALIDE *a seus filhos,*
  - *envia o sinal* RECEBI *a seu pai e,*
  - *aguarda a ordem* ESQUEÇA *de seu pai e os sinais* RECEBI *de seus filhos.*

Regra SF 6 *Cada processo folha ao receber uma decisão* ABORTE *entra em estado* ABORTADO. *Se receber uma decisão* VALIDE, *entra em estado* VALIDADO, *envia o sinal* RECEBI *a seu pai e aguarda deste a ordem* ESQUEÇA.

Regra SF 7 *Um processo intermediário ao receber os sinais* RECEBI *de seus filhos e a ordem* ESQUEÇA *de seu pai, entra em estado* DESCONHECIDO *e envia mensagens* ESQUEÇA *a seus filhos.*

Regra SF 8 *Um processo folha, em estado* VALIDADO, *entra em estado* DESCONHECIDO *ao receber a ordem* ESQUEÇA

Após finalizar a ação, um processo pode ainda receber indagações PERGUNTANDO. Essas indagações podem vir de um processo em recuperação, ou de um processo que esteja tentando decidir sobre a finalização. Em qualquer caso, a seguinte regra é utilizada:

Regra SF 9 *Um processo em estado* ABORTADO, *ao receber uma indagação* PERGUNTANDO, *envia a decisão* ABORTE *como resposta. Se estiver em estado* VALIDADO, *responde:* VALIDE.

Há uma situação em que o coordenador inicia o protocolo diretamente pela segunda fase. Nesse caso ele não concorda com a validação da ação e a sua decisão não depende dos demais processos. Nesse caso, o coordenador age segundo a regra abaixo:

Regra SF 10 *Se o coordenador está em estado* DESCONHECIDO *e não concorda em validar a ação, ele entra em estado* ABORTADO. *A seguir, ele inicia diretamente a segunda fase enviando a decisão* ABORTE *a seus filhos, conforme a Regra SF 1.*

No protocolo semibloqueante, o estado DESCONHECIDO não possui um registro correspondente no log. Para evitar que haja confusão entre esse estado e o estado ABORTADO presumido, as mensagens RECEBI e ESQUEÇA são utilizadas.

As mensagens RECEBI são importantes para que o coordenador e os processos intermediários saibam até quando eles devem manter em suas memórias voláteis as informações sobre uma ação validada. Enquanto não receberem de seus filhos mensagens com o sinal RECEBI, tais processos não podem mudar para o estado DESCONHECIDO. Essa exigência é necessária pois um processo pode ter falhado antes de receber a mensagem com a decisão VALIDE. Esse processo, ao se recuperar, envia mensagens PERGUNTANDO ao processo pai conforme será mostrado na próxima seção. Se o processo pai estivesse no estado DESCONHECIDO, ele não possuiria nenhuma informação sobre a ação. Logo, ele enviaria a decisão ABORTE ao seu filho em recuperação. Obviamente, tal situação geraria uma inconsistência na finalização da ação. Portanto, um processo não folha não pode entrar em estado DESCONHECIDO enquanto não receber os sinais RECEBI de todos os seus filhos.

A mensagem ESQUEÇA é necessária, pois mais de um processo subordinado pode tentar decidir sobre a finalização em tempos diferentes. Suponha que um subordinado $S1$ entra em estado DESCONHECIDO antes que todos os processos tenham validado a ação. Posteriormente, um subordinado $S2$ pode enviar mensagens PERGUNTANDO a $S1$. Como $S1$ não mais possui informações sobre a ação, ele irá responder com uma mensagem contendo a decisão ABORTE. Isto, obviamente, gera finalizações inconsistentes em $S1$ e $S2$. Portanto, cada subordinado deve manter, em sua memória volátil, informações sobre a ação até receber a ordem ESQUEÇA do seu processo pai. Essa ordem indica que o processo pai sabe que todos os demais processos validaram a ação e que, portanto, não é mais necessário armazenar dados sobre a ação.

6.1.8 Recuperação de Processos Falhos

Ao se recuperar de uma falha, um processo lê os registros armazenados no log e acumula em sua memória volátil as informações sobre o estado de cada ação existente antes da ocorrência da falha.

As regras abaixo descrevem o comportamento do coordenador após ler o log:

Regra R 1 *Se ao se recuperar, o coordenador encontrar apenas um registro* PREPARADO, *ele reinicia a execução da primeira fase do protocolo a partir da Regra PF 11.*

Se encontrar um registro PRÉ-ABORTADO, *ou* PRÉ-VALIDADO, *o coordenador inicia uma tentativa de obter um quorum correspondente ao registro encontrado.*

Caso encontre um registro VALIDADO, *o coordenador executa as regras do coordenador na segunda fase do protocolo.*

Regra R 2 *O coordenador em recuperação, ao receber uma mensagem* PERGUNTANDO, *pode reagir dos seguintes modos:*

1. *se o coordenador encontrar apenas um registro* PREPARADO *no log para a ação, ele adia o envio de uma resposta até que uma decisão seja obtida;*
2. *se encontrar um registro* PRÉ-VALIDADO *ou um registro* PRÉ-ABORTADO, *o coordenador envia como resposta uma mensagem correspondente ao registro encontrado, isto é,* PRÉ-VALIDADO, *ou* PRÉ-ABORTADO;
3. *se não encontrar um registro algum no log correspondente a ação, o coordenador envia a mensagem* ABORTE *como resposta.*
4. *se o coordenador já havia validado a ação antes de ocorrer a falha, ele envia a mensagem valide como resposta.*

Regra R 3 *Se um processo subordinado ao se recuperar encontra apenas um registro* PREPARADO, *ele envia mensagens* PERGUNTANDO *a seu processo pai enquanto não receber uma decisão.*

Se encontrar um registro PRÉ-ABORTADO *ou* PRÉ-VALIDADO, *o subordinado inicia uma tentativa de obtenção de quorum correspondente ao registro encontrado.*

Um processo subordinado, ao encontrar um registro VALIDADO, *envia a mensagem* RECEBI *a seu pai. Se o subordinado for também um processo intermediário, ele envia mensagens* VALIDE *a todos os seus filhos. Em seguida o processo executa a Regra SF 7 se for um processo intermediário, ou a Regra SF 8 se for um processo folha.*

Regra R 4 *Se um processo em estado* VALIDADO *recebe uma mensagem* PRÉ-VALIDE, PRÉ-ABORTE *ou* PERGUNTANDO, *ele envia a mensagem* VALIDE *em resposta.*

Regra R 5 *Se um processo intermediário em estado* PREPARADO *recebe uma mensagem* PERGUNTANDO *ao se recuperar, ele envia a mensagem* RECUPERANDO *como resposta.*

Regra R 6 *Se um processo folha em estado* PREPARADO *recebe uma mensagem* PERGUNTANDO *ao se recuperar, ele envia a mensagem* SIM *como resposta.*

Qualquer processo em estado PRÉ-VALIDADO ou PRÉ-ABORTADO, que recebe uma mensagem PERGUNTANDO, envia mensagem correspondente ao estado em que se encontra.

6.2 Correção do Protocolo

Uma execução sem falhas do protocolo semibloqueante é semelhante à do protocolo de duas fases hierárquico. Os mesmos argumentos utilizados para aquele protocolo se aplicam ao protocolo semibloqueante em uma execução isenta de falhas.

A demonstração da correção do protocolo em execuções em que ocorrem falhas é feita com base na análise das *situações de conflito.* Denomina-se *situação de conflito* a possibilidade de dois processos diferentes tomarem decisões após a ocorrência de falha em uma execução do protocolo. Os processos envolvidos em uma situação de conflito são ditos *processos conflitantes.* As situações de conflito podem ser identificadas verificando-se os casos em que os processos podem vir a tomar decisões após uma ocorrência de falha do protocolo. Esses casos são os seguintes:

caso 1 o coordenador decide sem ter detectado a falha. Ocorre quando o coordenador toma uma decisão antes do surgimento da falha (ver Regra PF 6);

caso 2 o coordenador recebe mensagens NÃO ou ABORTE em resposta a uma tentativa de obtenção de respostas das subárvores cujas raízes são seus processos filhos (ver Regra PF 11);

caso 3 o coordenador tenta obter um quorum - ocorre quando o coordenador tenta obter as respostas das subárvores cujas raízes são seus processos filhos e recebe apenas mensagens SIM, PRÉ-VALIDADO ou PRÉ-ABORTADO ou PREPARADO (ver Regra PF 11);

caso 4 um processo subordinado com voto SIM tenta obter um quorum - ocorre quando um processo subordinado com voto SIM tenta obter as respostas das subárvores cujas raízes são processos filhos do coordenador e recebe apenas mensagens SIM, PRÉ-VALIDADO ou PREPARADO (ver Regra PF 10);

caso 5 um processo intermediário em estado apenas PREPARADO tenta obter um quorum de aborto após detectar término do tempo de espera pelo voto de um de seus processos filhos (ver Regra PF 8);

caso 6 um processo subordinado com voto SIM recebe alguma mensagem NÃO ou ABORTE durante a tentativa de obter respostas das subárvorescujas raízes são processos filhos do coordenador (ver Regra PF 10).

Os casos acima são os únicos em que um processo pode tomar decisão após a ocorrência de uma falha.

Uma situação de conflito corresponde à ocorrência simultânea de dois dos casos apresentados acima. Portanto, as situações de conflito possíveis são as seguintes:

Situações em que o coordenador decide sem detectar falha (caso 1)	**situação 1:** caso 1 × caso 4 **situação 2:** caso 1 × caso 5 **situação 3:** caso 1 × caso 6
Situações em que o coordenador recebe uma mensagem NÃO ou ABORTE (caso 2)	**situação 4:** caso 2 × caso 4 **situação 5:** caso 2 × caso 5 **situação 6:** caso 2 × caso 6
Situações em que o coordenador tenta obter um quorum (caso 3)	**situação 7:** caso 3 × caso 4 **situação 8:** caso 3 × caso 5 **situação 9:** caso 3 × caso 6
Situações que envolvem apenas os subordinados	**situação 10:** caso 4 × caso 4 **situação 11:** caso 4 × caso 5 **situação 12:** caso 4 × caso 6 **situação 13:** caso 5 × caso 5 **situação 14:** caso 5 × caso 6 **situação 15:** caso 6 × caso 6

Para demonstrar que todos os processos que decidem após a ocorrência de uma falha tomam a mesma decisão, é necessário e suficiente demonstrar que em nenhuma das quinze situações de conflito os processos tomam decisões diferentes entre si.

As situações cujas demonstrações de correção mais complicadas são aquelas que envolvem o caso 1. Nas situações 1 e 2, o coordenador decide sem obter quorum, enquanto um subordinado tenta obter um quorum para tomar decisão. A princípio, pode-se supor que a não necessidade de obter um quorum por parte do coordenador poderia fazer com que um subordinado e o coordenador tomassem decisões diferentes. Entretanto, pode ser demonstrado que isto não ocorre.

O coordenador decide abortar a ação somente se recebe um voto NÃO de um de seus filhos, pois está sendo considerado que a falha ocorre após o coordenador ter decidido. Portanto, todo subordinado que tenta obter as respostas das subárvores cujas raízes são processos filhos do coordenador obtém uma resposta NÃO, ABORTE, ou não obtém a resposta de uma das subárvores. No último caso, de acordo com a Regra PF 8, o subordinado tenta obter

um quorum de aborto da ação. Garante-se assim, que apenas quorums de aborto podem ser formados. Logo, todos os subordinados que estejam nos casos 4, 5 e 6 que decidirem, o fazem abortando a ação.

Considere a ocorrência do caso 1 em que o coordenador decide validar a ação antes da ocorrência da falha. Nesta condição, tem-se a garantia de que a situação 3 não ocorre, pois não há emissão de mensagens NÃO ou ABORTE por nenhum processo envolvido no protocolo. Além disso, não existe nenhum processo subordinado que esteja apenas em estado PREPARADO (isto é, sem voto SIM). Necessariamente, todos os subordinados possuem voto SIM, logo a situação 2 também não ocorre. A situação 1 pode ocorrer, entretanto um subordinado com voto SIM tentaria obter um quorum de aborto apenas no caso em que não consegue obter as respostas de uma das subárvores cujas raízes são processos filhos do coordenador. Obviamente, a possibilidade desse processo não obter a resposta de uma das subárvores existe. Porém, o protocolo garante que uma tentativa de obtenção de um quorum de aborto não ocorre. Especificamente, a Regra PF 10 assegura que um subordinado com voto SIM tenta obter um quorum de aborto somente se receber uma mensagem PREPARADO durante uma tentativa de obter as respostas dessas subárvores. Como todos os processos necessariamente possuem voto SIM, a tentativa de obter um quorum de aborto não ocorre.

Pode-se concluir que nas situações que envolvem o caso 1, isto é situações 1, 2 e 3, os processos que decidem tomam a mesma decisão.

As situações de conflito que não envolvem o caso 1 (ou que não envolvem o coordenador) podem ser classificadas nos seguintes grupos:

Grupo 1: formado por situações em que os dois processos conflitantes recebem mensagens NÃO ou ABORTE (situações 6 e 15);

Grupo 2: formado por situações em que um dos processos tenta obter um quorum e o outro recebe uma mensagem NÃO ou ABORTE (situações 4, 5, 6, 9, 12 e 14);

Grupo 3: formado por situações em que ambos os processos conflitantes tentam obter um quorum (situações 7, 8, 10, 11 e 13).

Nas situações do grupo 1 ambos os processos conflitantes finalizam a ação do mesmo modo, isto é, abortando-a.

Nas situações do grupo 2, o processo que recebe a mensagem NÃO, ou ABORTE aborta a ação. Não há a possibilidade de que o processo conflitante tente obter um quorum de validação, pois pelo menos uma das respostas das subárvores cujas raízes são processos filhos do coordenador é NÃO ou ABORTE. Nas situações que envolvem o caso 5, o processo conflitante tenta sempre obter um quorum de aborto, de acordo com a Regra PF 8. Se o processo conflitante não consegue obter a resposta de uma das subárvores cuja raiz é filho do coordenador, ele tenta obter um quorum de aborto de acordo com as Regras PF 10 e Regra PF 11. Logo, em todas as situações do grupo 2, todos os processos que decidem abortam a ação.

O grupo 3 é constituído por situações em que cada processo de um par de processos conflitantes tenta obter um quorum. Nestas situações, uma decisão é tomada apenas se um dos processos conflitantes obtiver um quorum. Conforme demonstrado na Seção 6.1.6, o algoritmo de obtenção de quorum em árvore utilizado pelo protocolo semibloqueante garante a impossibilidade de se obter dois quorums de tipos diferentes. Portanto, se dois processos em qualquer uma das situações do grupo 3 tomam decisões, essas decisões são as mesmas.

Os argumentos acima demonstram que em qualquer execução do protocolo semibloqueante, com ou sem falha, todos os processos que finalizam a ação o fazem do mesmo modo, isto é, todos a validam ou todos a abortam.

6.3 Otimizações

Nesta seção são propostas algumas otimizações com vistas a tornar o protocolo semibloqueante mais eficiente.

6.3.1 Economizando Mensagens em Processos de Leitura Apenas

Pode-se reduzir o número de mensagens enviadas durante a segunda fase do protocolo em casos de ações de leitura apenas ou ações parcialmente de leitura. O coordenador e os processos intermediários não enviam a decisão a processos que apenas lêem os dados. Também não necessitam ser enviadas as mensagens ESQUEÇA e RECEBI. Votos do tipo LEITURA-APENAS devem ser introduzidos no protocolo para identificar processos participantes da ação que não alteram dados.

As seguintes adaptações devem ocorrer no protocolo para que o mesmo possa tirar proveito das ações de leitura apenas.

1. se um processo intermediário *I* recebe somente votos LEITURA-APENAS e *I* não altera nenhum dado localmente, ele envia o voto LEITURA-APENAS ao seu processo pai. Se o processo *I* ou algum de seus processos filhos altera os dados, o voto emitido pelo processo *I* é obtido com base nas intenções desses processos que não são de leitura apenas, ou seja, o voto LEITURA-APENAS não influencia o voto do processo *I* neste caso.

2. se o coordenador recebe somente votos de LEITURA-APENAS de seus processos filhos e o coordenador não altera nenhum dado local, ao receber os votos de seus processos filhos, ele simplesmente apaga as informações sobre a ação de sua memória local. Não existe segunda fase do protocolo neste caso. Se o coordenador altera os dados locais ou alguns de seus filhos não são processos de leitura apenas, ele toma uma decisão com base nas intenções desses processos, ou seja os votos LEITURA-APENAS não influenciam na decisão. Neste caso, mensagens com a decisão são enviadas apenas aos processos cujos votos foram diferentes de LEITURA-APENAS.

3. processos subordinados de leitura apenas, ao entrarem em estado PREPARADO forçam a gravação de um tipo especial de registro no log. Esse registro contém apenas a identificação da ação e uma indicação de que ela não altera dados naquele processo.

As ações de leitura apenas, apesar de propiciarem uma redução de envios de mensagens, não evitam a gravação forçada no log do estado PREPARADO. Esta gravação se faz necessária para garantir a correção do protocolo. Se nenhuma gravação fosse executada, como ocorre no protocolo PA, um processo ao se recuperar poderia não saber identificar se ele era um processo de leitura apenas ou se ele havia abortado a ação antes da falha ocorrer. Essa dúvida ocorre tanto no protocolo semibloqueante quanto no protocolo PA, porém, no último, ela não afeta a correção do protocolo, pois não há a possibilidade do processo em recuperação receber uma mensagem PERGUNTANDO de um subordinado. No protocolo semibloqueante, entretanto, um processo de leitura apenas pode, ao se recuperar, receber uma mensagem PERGUNTANDO de um subordinado com voto SIM. Nesse caso, é necessário que o processo tenha alguma informação indicando que ele era de leitura apenas. Se assim não for, o processo em recuperação assume que havia abortado, caso não encontre nenhuma informação sobre a ação no log. Esta situação pode fazer com que o coordenador e um subordinado com voto sim tomem decisões diferentes. O coordenador pode validar a ação ao receber apenas votos SIM e o voto LEITURA-APENAS de um processo antes de sua falha; o subordinado com voto SIM aborta a ação ao receber uma mensagem aborte desse mesmo processo após ele se recuperar da falha. Portanto, devido à característica de presumir o aborto quando não há informação sobre a ação, o protocolo semibloqueante não pode evitar a gravação do estado PREPARADO em processos de leitura apenas.

6.3.2 Envio de Mais de Uma Mensagem em Uma Mesma Transmissão ("pigbacking")

Pode-se reduzir uma etapa de envio de mensagens no protocolo semibloqueante, evitando-se transmissões exclusivas para o transporte de mensagens ESQUEÇA. O envio dessas mensagens pelo coordenador ou por um processo intermediário não necessita ser imediatamente após esse processo receber as mensagens RECEBI de seus processos filhos. O coordenador ou um processo intermediário podem adiar o envio da mensagem ESQUEÇA e inseri-la no conteúdo da próxima mensagem a ser enviada ao processo filho ao qual ela deve ser enviada. Dessa forma pode-se obter uma redução no número de transmissões, através do aumento de tamanho da próxima mensagem a ser enviada.

6.3.3 Utilização de Balanceamento da Árvore de Processos

A árvore de processos que constituem a ação pode não ser uma árvore completa. Neste caso, pode-se obter uma nova estrutura a partir desta árvore, visando diminuir o tempo de execução do protocolo e o número de mensagens enviadas. Antes de iniciar a execução do protocolo, o

processo raiz da árvore de processos obtém uma a árvore balanceada correspondente àquela árvore. A árvore balanceada tem como raiz o mesmo processo que era raiz da árvore original. A execução do protocolo semibloqueante passa a tomar como base a árvore balanceada obtida. A árvore balanceada é propagada aos diversos processos envolvidos na ação através de sua inclusão nas mensagens PREPARE.

6.4 Eficiência do Protocolo Durante Execuções Isentas de Falhas

Serão utilizados na avaliação da performance do protocolo semibloqueante os mesmos critérios estabelecidos na Seção 2.3.6. A seguir, são feita avaliações do protocolo quanto ao tempo de execução e quanto ao número de mensagens enviadas.

6.4.1 Tempo de Execução

No modelo de cálculo de tempo de execução descrito na Seção 2.3.6, o tempo gasto para emitir mensagens em paralelo é o mesmo para emitir uma única mensagem. Portanto, o tempo de execução do protocolo para uma ação executada por uma árvore de processos de altura h é $5h$ que corresponde a h unidades de tempo para o envio de mensagens PREPARE até os nós folhas, h unidades de tempo para o envio de votos até o coordenador, h unidades de tempo para o envio de mensagens com a decisão, h unidades de tempo para o envio de mensagens RECEBI e h unidades de tempo para o envio de mensagens ESQUEÇA.

6.4.2 Número de Mensagens

O número de mensagens enviadas durante uma execução do protocolo é medido como sendo o número de mensagens enviadas pelos N processos em uma ação que é validada. Considerando a utilização da técnica de "pigbacking" para o envio das mensagens ESQUEÇA, o total de mensagens é $4(N-1)$ que corresponde a $N-1$ mensagens PREPARE, $N-1$ mensagens de votos, $N-1$ mensagens com a decisão e $N-1$ mensagens RECEBI.

6.4.3 Número de Gravações no Log

Durante uma execução sem falhas do protocolo semibloqueante, cada processo executa a gravação forçada do registro PREPARADO, e do registro VALIDADO, caso a ação seja validada. O registro de aborto é gravado caso a ação seja abortada, mas a sua gravação não é forçada no log. Em uma execução com falhas, cada processo força a gravação de no máximo três registros no log: o registro PREPARADO, o registro VALIDADO e um dos registros: PRÉ-VALIDADO ou PRÉ-ABORTADO.

Capítulo 7

Conclusão

O objetivo inicial do nosso trabalho era o de fazer um estudo detalhado do problema de validação de ações atômicas, bem como dos principais protocolos publicados que procuram resolvê-lo[1]. Em nossa opinião, esse objetivo foi realizado e tem como resultado os capítulos 2, 3, 4 e 5 desta dissertação.

Acreditamos que proporcionamos ao leitor uma visão detalhada do problema de validação de ações atômicas e dos protocolos que visam solucioná-los. Além disso, adotamos uma padronização na descrição dos protocolos estudados que acreditamos ser útil para estudos futuros, por exemplo, para uma análise comparativa dos protocolos estudados.

O estudo dos protocolos de validação também serviu de subsídio para a proposição de um novo protocolo, o protocolo semibloqueante, que acreditamos ser a segunda contribuição deste trabalho.

Este capítulo está divido em duas seções. A Seção 7.1 faz uma breve análise sobre as abordagens das soluções do problema de validação de ações atômicas discutidas neste texto. A Seção 7.2 encerra o texto com algumas sugestões de trabalhos futuros.

7.1 Comentários

O Capítulo 2 apresenta o problema de validação de ações atômicas distribuídas, descreve um modelo formal de funcionamento dos protocolos de validação e apresenta limitações quanto à solução do problema após a ocorrência de falhas de partição do sistema. A conclusão principal que se pode obter do Capítulo 2 refere-se à impossibilidade de haver protocolos de validação não-bloqueantes quando há falhas de partição ou de protocolos que permitem a recuperação independente de processos após uma falha deste tipo.

[1] O nosso estudo se restringe a ações atômicas de propósito geral. Não são abordados os casos de ações atômicas para sistema de tempo real e ações atômicas de longa duração.

As limitações dos protocolos de validação quanto à tolerância a falhas têm exercido influência sobre os projetistas de sistemas gerenciadores de ações atômicas. Espera-se que a ocorrência de falhas em um sistema distribuído seja uma exceção; portanto, não é compensador utilizar um protocolo do tipo não-bloqueante como protocolo principal para validação de ações atômicas no sistema.

Soluções alternativas têm sido pesquisadas e utilizadas. A mais comum baseia-se na priorização da eficiência em detrimento da tolerância a falhas. O Capítulo 3 descreve um conjunto de protocolos que adotam esta estratégia. São protocolos de duas fases que visam diminuir o número de mensagens a serem enviadas e o número de gravações em log. Deste modo, consegue-se obter maior vazão ("throughput") de ações finalizadas. Entretanto, quando ocorre uma falha no sistema, o comportamento destes protocolos é bem simplista: eles apenas abortam a ação. A grande desvantagem desta abordagem se deve à possibilidade de haver abortos desnecessários. O custo destes abortos geralmente é alto, especialmente no caso de ações que alteram vários itens de dados.

Alguns autores visam uma solução mais amena, utilizando um protocolo de duas fases com alguns recursos extras, de modo a permitir um aumento de tolerância a falhas sem descaracterizar o aspecto de duas fases do protocolo. Os protocolos descritos no Capítulo 4 utilizam-se desta solução. Sob o aspecto de tolerância a falhas esta abordagem é geralmente pouco eficaz, pois os protocolos de duas fases são inerentemente bloqueantes. Contudo, alguns protocolos conseguem bastante robustez ao custo de um aumento excessivo do número de mensagens. É o que ocorre, por exemplo, com o Protocolo de Validação com Substitutos para o Coordenador (Seção 4.3) e com o Protocolo BPA (Seção 4.4).

Outra solução, proposta por Duchamp [Duc88], sugere a utilização de dois protocolos: um protocolo de duas fases otimizado e um protocolo não-bloqueante. O programador tem a opção de escolher previamente qual protocolo a ser utilizado conforme as necessidades da aplicação. O protocolo não-bloqueante proposto por Duchamp é descrito no Capítulo 5, bem como outros protocolos que não são de duas fases.

A solução proposta neste texto consiste em utilizar um protocolo eficiente bloqueante, enquanto não ocorrem falhas no sistema. A partir do momento em que uma falha é detectada, o protocolo transforma-se em um protocolo não-bloqueante e uma terceira fase, intermediária entre a primeira e a segunda fases, é adicionada. Devido à capacidade de sofrer a transformação citada acima, o protocolo proposto é denominado protocolo semibloqueante.

Esta nova estratégia para tratar o problema de validação (pelo menos não se conhece outro protocolo que utiliza a mesma abordagem) possibilita ao protocolos semibloqueante eficiência e robustez em situações apropriadas. Durante execuções sem falha, o protocolo possui eficiência comparável à de protocolos de duas fases. Contudo, em execuções com falhas, o protocolo semibloqueante possui robustez de protocolos não-bloqueantes.

A versatilidade do protocolo semibloqueante é obtida pela organização lógica, em forma de árvore, dos processos que participam da ação. Esta estruturação permite aos descendentes e ancestrais de um processo terem informações sobre a intenção desse processo quanto à

finalização da ação.

O protocolo possui a propriedade de "degradação bem comportada" ("gracefull degradation"). Isto significa que o protocolo possui custos baixos de comunicação em execuções sem falhas. Com o aumento do número de falhas, aumentam os custos de comunicação.

O protocolo semibloqueante, contudo, possui tempo de execução maior do que o do protocolo de duas fases hierárquico, pois as mensagens mensagens RECEBI devem ser enviadas pelos processos filhos do coordenador somente após eles receberem mensagens RECEBI de seus filhos. Esta exigência não ocorre no protocolo de duas fases hierárquico e nos protocolos dele derivados (protocolos PC, PA, NPC, protocolo flexível e protocolo O2PC). Nestes protocolos, um processo filho do coordenador pode enviar mensagem RECEBI ao coordenador logo após entrar no estado correspondente à decisão recebida.

7.2 Sugestões para Trabalhos Futuros

Os capítulos 3, 4 e 5 apresentam descrições de protocolos, tomando como base o modelo de sistemas proposto no Capítulo 2. São feitas demonstrações das correções desses protocolos e avaliações com relação ao número de mensagens enviadas e o tempo de execução. Contudo, não é feito um estudo da probabilidade de bloqueio dos protocolos estudados. Também não foi feito este estudo para o protocolo semibloqueante proposto. Como um trabalho futuro, sugerimos um análise das probabilidades desses protocolos virem a se bloquear após a ocorrência de falhas.

Seria interessante que, em um trabalho futuro, o protocolo semibloqueante fosse implementado como parte de um sistema gerenciador de ações atômicas. Esta implementação seria importante para se obter uma avaliação realística da robustez e eficiência do protocolo proposto e também para possibilitar comparações mais efetivas com outros protocolos já implementados.

Bibliografia

[AA91] Divykant Agrawal e Amr El Abbadi. An Efficient and Fault-Tolerant Solution for Distributed Mutual Exclusion. *ACM Transactions on Computer Systems*, 9(1):1–20, February de 1991.

[Bue89] U. Buerger. A Flexible Two-phase Commit Protocol. *Computer Network and ISDN Systems*, 17:175–185, 1989.

[BVHNG87] P.A. Bernstein, Vassos Hadzilacos, e Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addisson-Wesley Publishing Compny, 1987.

[Duc88] Daniel Duchamp. *Transaction Management.* PhD thesis, Carnegie Mellon University, Pittsburg, PA 15213, December de 1988.

[Gra78] Jim Gray. Notes on Database Operating Systems. In *Operating Systems - An Advanced Course*, pp. 393–481. Springer Verlag, 1978.

[HS80] M. Hammer e D. Shipman. Reliability Mechanism for SDD1: A System for Distributed Databases. *ACM Trans. Database Syst.*, 5:431–466, December de 1980.

[LL93] Butler Lampson e David Lomet. A New Presumed Commit Optimization for Two Phase Commit. Technical Report CRL 93/1, Digital Equipment Corporation, Cambridge Research Laboratory, 1993.

[ML83] C. Mohan e B. Lindsay. Efficient Commit Protocols for The Tree of Process Model of Distributed Transactions. In *Proc. Second Ann. Symp. on Principles of Distributed Computing*, pp. 76–88. ACM, August de 1983.

[MSF83] C. Mohan, R. Strong, e S. Finkelstein. Method for Distributed Transaction Commit and Recovery Using Byzantine Agreement Within Clusters of Processors. In *Proceedings of Second SIGACT/SIGOPS Symposium on Principles of Distributed Computing.* ACM, Montreal Canada, August de 1983.

[RA93] Thierson Couto Rosa e Ricardo O. Anido. Protocolo Semibloqueante Para Validação de Ações Atômicas. *Anais do XXIII SEMISH*, 1993.

[Ram89] K.V.S. Ramarao. Complexity of Distributed Commit Protocols. *Acta Informatica Springer Verlag*, 26:577-595, 1989.

[RP90] Kurt Rothermél e Stefan Pappe. Open Commit Protocols for The Tree of Process Model. In *Proc. The Tenth International Conference on Distributed Computing Systems*, pp. 236-244. IEEE, Paris, France, June de 1990.

[Ske81] Dale Skeen. Nonblocking Commit Protocols. In *Proc ACM SIGMOD International Conference on Management of Data*, pp. 133-142. ACM, Michigan, 1981.

[SS83] D. Skeen e M. Stonebraker. A Formal Model of Crash Recovery in a Distributed System. *IEEE Trans. Software Engineering*, Se-9(3):219-227, May de 1983.

[YA87] Shyan-Ming Yuan e Ashok K. Agrawala. A Class of Optimal Decentralized Commit Protocols. Technical Report UMIACS-TR-87-31, University of Maryland, College Park, Maryland, 20742, 1987.

[YA91] Shyan-Ming Yuan e Ashok K. Agrawala. Fault-Tolerant Decentralized Commit Protocols. *Journal of Parallel and Distributed Computing*, 13:299-311, 1991.