Protocolos para Controlar Dados Replicados em Sistemas de Computação Distribuídos

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. Nabor das Chagas Mendonça e aprovada pela Comissão Julgadora.

Campinas, 17 de Dezembro de 1993.

Prof. Dr. _____

Ricardo de Oliveira Anido

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.

Nabor das Chagas Mendonça

Protocolos para Controlar Dados Replicados em Sistemas de Computação Distribuídos

Dissertação apresentada em 17 de Dezembro de 1993.

Banca Examinadora:

Prof. Dr. Ricardo de Oliveira Anido — DCC–UNICAMP

Orientador

Prof. Dr. Pedro Sérgio de Souza — DCC–UNICAMP

Prof. Dr. Osvaldo Sérgio Carvalho — DCC–UFMG

A todos aqueles que estão sempre procurando desafios.

Agradecimentos

A meus pais, pelo apoio irrestrito e fundamental.

A meus irmãos, que mesmo de tão longe nunca deixaram de "torcer" e acreditar em mim.

A meus ex-professores Mônica e Fares, pelo incentivo inicial, e aos demais professores, ex-professores e companheiros do Departamento de Ciência da Computação da Universidade do Amazonas.

Ao professor Ricardo Anido, meu orientador, pelas sugestões e correções sempre precisas e valiosas.

Aos amigos que conheci na Unicamp, vários atualmente buscando novos objetivos em outros lugares, que conviveram comigo esses dois anos e meio de mestrado e tiveram de "agüentar" todas as minhas "baixarias" nem sempre bem recebidas; em especial, à turma do vôlei, pelos jogos e cervejadas tão animados.

Aos companheiros de outras universidades que se deram o incômodo de procurar e me enviar os artigos que tanto precisava; em particular, Altigran e João Marcos (UFMG), Edson (COPPE/UFRJ), Naranker Dulay e Kostas Karagianids (Imperial College).

À CAPES e à IBM do Brasil, pelo aporte financeiro.

À Déia, por ter compartilhado comigo "baús" de momentos agradáveis e estimulantes nesses últimos meses; sei que minha alegria de ver este trabalho realizado certamente se estende a ela também.

Abstract

This dissertation studies several protocols for controlling replicated data in distributed systems. The protocols considered to be more significative are grouped into two distinct classes: protocols based on voting and protocols based on logical structures. Protocols from the same class are then described and compared.

As a result, a new control replica protocol is proposed which generalizes — and performs better than — most of the protocols based on logical structures. Moreover, the dissertation presents a list of important factors that must be observed by the system designers in order to make the search for a suitable control replica protocol easier.

Sumário

Esta dissertação estuda vários protocolos para a manutenção da consistência entre cópias de dados replicados em sistemas distribuídos. Os protocolos considerados mais significativos são agrupados em duas classes distintas: protocolos baseados em votação e protocolos baseados em estruturas lógicas. Protocolos de uma mesma classe são então descritos e comparados.

Como resultado dos estudos realizados, é proposto na dissertação um novo protocolo de controle de réplicas que generaliza — e supera — grande parte dos protocolos baseados em estruturas lógicas. É sugerido ainda um conjunto de fatores que deve ser considerado pelos projetistas de um sistema distribuído quando da escolha do protocolo de controle de réplicas mais adequado.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A atual proliferação de sistemas de computação distribuídos (ou simplesmente sistemas distribuídos), tanto no universo acadêmico quanto nos meios comerciais, pode ser atribuída principalmente aos constantes avanços tecnológicos na área de microeletrônica. Estes avanços permitiram que componentes de elevados desempenho e eficiência, como processadores extremamente velozes, memórias com enorme capacidade de armazenamento e redes de comunicação com alta confiabilidade e taxa de transmissão de dados, estejam disponíveis a preços significativamente reduzidos. No entanto, tais avanços só ganham influência na medida em que tornam possível o desenvolvimento de sistemas que consigam satisfazer as cada vez mais sofisticadas necessidades dos usuários. Por sofisticação, entenda-se aplicações consideradas não (ou pouco) convencionais; por exemplo, as que requerem elevada disponibilidade de recursos, troca de informações entre locais geograficamente dispersos, alta interação gráfica, ou armazenamento e manipulação de objetos complexos como som e imagem.

Tratar logicamente um conjunto de computadores interligados por uma rede de comunicação como um único e integrado sistema de computação (distribuído) produz uma série de benefícios [LPS83, CD88]. Esses benefícios, juntos, facilitam o suporte a essas novas e sofisticadas aplicações, tornando um sistema distribuído o ambiente ideal para aplicações desta natureza; entre os principais deles, citam-se:

- **Melhor desempenho**: uma aplicação pode ser processada em paralelo por mais de um processador, propiciando maior *throughput* (ou número de requisições de serviços atendidas por unidade de tempo).
- **Tolerância a falhas**: a falha de um ou alguns componentes do sistema não necessariamente implica na interrupção de todas as atividades em andamento; a atividade que depende de um componente que falhou, por exemplo, poderá ser realocada para outro componente ainda em operação.

- **Compartilhamento de recursos**: recursos disponíveis a um computador, como periféricos ou mesmo arquivos, podem ser compartilhados pelos outros computadores do sistema.

- **Expansibilidade**: o sistema pode ter novos recursos adicionados de acordo com o crescimento da demanda por serviços, sem a necessidade de onerosas alterações nos programas que implementam os serviços já existentes; o limite da expansão estaria na capacidade da rede de comunicação.

Embora um sistema distribuído apresente esta série de vantagens, aspectos como exigência de software mais complexo, dependência da capacidade e confiabilidade da rede, e menor segurança decorrente do fácil compartilhamento de dados podem ser apontados como desvantagens de um sistema distribuído em comparação a um sistema centralizado [Tan92].

Dos benefícios descritos, alguns podem ser obtidos diretamente pelo uso de uma técnica conhecida como *replicação de dados*. Esta técnica consiste em armazenar cópias (réplicas) de um mesmo dado lógico em diferentes computadores. Isto permite, através da possibilidade de que o acesso a dados replicados seja feito localmente (ou a partir de um computador próximo), reduzir o tráfego na rede e melhorar o tempo de resposta do sistema. Mais: consegue-se melhor *throughput* com o tratamento em paralelo de diferentes requisições de acesso a um mesmo dado. Outra grande vantagem da replicação é maior tolerância a falhas, uma vez que o dado replicado, estando fisicamente armazenado em mais de um computador com probabilidade independente de falha, terá suas chances de estar disponível (e de não ser perdido) aumentadas.

Quando comparada ao caso convencional em que o dado é armazenado em um único local, a replicação onera a realização das operações de acesso ao dado, visto que, para garantir a consistência[1] entre as réplicas, qualquer alteração sofrida pelo dado replicado deve ser refletida em todas as suas cópias. Portanto, enquanto a replicação propicia os vários benefícios citados anteriormente, ela, por outro lado, requer a existência de protocolos de controle – normalmente complexos e caros – para sincronizar todos os acessos ao dado replicado e, desse modo, conseguir manter as cópias em um estado consistente.

Nesta dissertação são estudados e comparados vários protocolos para o controle de dados replicados em sistemas distribuídos, agrupando-se em uma mesma classe os protocolos com características afins. Além disso, como resultado dos estudos realizados, é proposto um novo protocolo para o controle de dados replicados que generaliza um conjunto significativo dos protocolos estudados, e, mais ainda, consegue sobressair-se quando comparado a estes.

[1] A noção de consistência adotada na dissertação é descrita na seção 1.2.

Na seção seguinte define-se o modelo de sistema distribuído adotado na dissertação. Uma descrição mais detalhada de como acontece a replicação e da noção de consistência utilizada para as cópias é feita na seção 1.2. Os protocolos de controle tradicionais, os principais problemas que eles enfrentam e como esses problemas são resolvidos — ou não — estão na revisão da seção 1.3. Por fim, a seção 1.4 mostra a organização do restante da dissertação.

1.1 Modelo Adotado

1.1.1 Sistema Distribuído

Um sistema distribuído é definido como uma coleção de computadores autônomos que não compartilham uma memória global, e se comunicam apenas pela troca de mensagens enviadas através de uma rede de comunicação. Cada processador possui associada uma memória local e é denominado um *nó* do sistema.

A característica crucial para se distinguir um sistema distribuído de uma rede de computadores tradicional, também constituída de um conjunto de computadores interligados por algum mecanismo de comunicação, é a *transparência* [TGGL82]. Em um sistema distribuído, os usuários e programadores de aplicações enxergam o sistema como um todo centralizado, ficando escondida (transparente) a distribuição física dos recursos. Desse modo, os serviços fornecidos pelo sistema é que encarregar-se-ão de mapear um recurso logicamente centralizado requisitado pelos usuários em um ou mais recursos físicos (possivelmente) dispersos pelos nós. Em uma rede de computadores, em contraste, os próprios usuários têm a incubência de indicar a localização física de cada recurso requisitado; no caso de haver dados replicados, por exemplo, a manutenção da consistência entre as cópias deve ser tarefa de todas as aplicações que tenham acesso a eles.

1.1.2 Falhas

Considera-se que os nós do sistema e os canais (ou *links*) da rede de comunicação são *fail-stop* [SS83], ou seja, em caso de falha simplesmente param de funcionar até que o devido reparo seja realizado, não se comportando maliciosamente. Falhas de nós e/ou canais da rede podem levar ao *particionamento* do sistema [DGMS85]. Durante o particionamento, o sistema fica subdividido em grupos disjuntos de nós denominados *partições*. Os nós em uma mesma partição conseguem se comunicar entre si, mas nós em partições diferentes ficam isolados uns dos outros. O particionamento é uma situação particularmente difícil de ser contornada pelos serviços fornecidos em um

sistema distribuído porque, quando um nó qualquer não consegue ser contactado, é impossível determinar se ele falhou ou está isolado em outra partição.

Considera-se também que cada nó do sistema está independentemente operacional com probabilidade p ($0 < p < 1$), ou seja, em um dado instante, um nó qualquer tem probabilidade $1 - p$ de não conseguir ser contactado devido a falhas. Esta probabilidade p indica a *confiabilidade* dos nós e é usada no cálculo da disponibilidade do dado resultante da replicação.

1.2 Replicação de Dados

Um dado é replicado armazenando-se duas ou mais cópias em diferentes nós do sistema, uma cópia em cada nó. Inicialmente, a cada cópia é atribuído um *número de versão* que é incrementado toda vez que a cópia sofre uma alteração. Isto possibilita a identificação da cópia mais recentemente alterada, ou cópia corrente, como sendo aquela com o maior número de versão.

A replicação dos dados é tarefa exclusiva do administrador do sistema. A ele compete decidir quantas cópias um dado a ser replicado deve possuir e em quais nós elas devem ser armazenadas, informações cujo estabelecimento depende de fatores como a demanda pelo dado, o número total de nós do sistema e a capacidade de armazenamento de cada nó, entre outros. Essas informações, uma vez estabelecidas, devem estar disponíveis apenas ao serviço encarregado de mapear as operações (lógicas) de leitura e escrita requisitadas pelos usuários sobre o dado replicado em operações (físicas) sobre as cópias. A este serviço denomina-se *protocolo de controle de réplicas* ou *protocolo para controlar dados replicados.*

Através do protocolo de controle de réplicas, portanto, é possível alterar a quantidade e localização da cópias de um dado replicado sem que sejam necessárias alterações nos programas das aplicações que realizam (e já realizavam) operações sobre ele. Isto porque as aplicações, como também os próprios usuários finais, efetuam operações sobre um mesmo dado lógico, ficando transparente o mapeamento (das operações lógicas requisitadas para operações físicas sobre as cópias) realizado pelo protocolo de controle.

Ao protocolo de controle de réplicas concerne apenas a manutenção da consistência entre as cópias; como realizar (automaticamente ou não) a replicação dos dados, considerando-se as características do sistema, de maneira a otimizar a utilização dos recursos — maximizando a disponibilidade do dado e minimizando o tráfego na rede, por exemplo — é um outro campo de pesquisa que tem sido foco de poucos trabalhos até o presente momento e está fora do escopo desta dissertação.

1.2.1 Consistência

Uma vez que as cópias representam fisicamente um mesmo dado lógico, o protocolo de controle de réplicas é considerado correto se ele as mantém em um estado consistente. A noção de consistência considerada na dissertação engloba dois aspectos [LPS83]: *consistência mútua* e *consistência interna*.

As cópias de um dado replicado estão mutuamente consistentes se elas são idênticas; como é difícil satisfazer esta restrição a todo momento, normalmente apenas exige-se que as cópias convirjam para um mesmo estado consistente ao final de um período predeterminado. Até que ponto é tolerável ter cópias momentaneamente divergentes depende da natureza das aplicações visadas.

A consistência interna refere-se à informação contida em cada cópia e corresponde à mesma noção de consistência a que está submetido um dado não replicado; envolvem-se, nesse caso, os conceitos tradicionais de atomicidade, serialização (*serializability*) e integridade semântica inerentes ao mecanismo de controle de concorrência utilizado pelo sistema.

A grande maioria dos protocolos para controlar dados replicados estudados nesta dissertação, assim como a maioria dos trabalhos existentes na literatura, preocupa-se somente com a manutenção da consistência mútua entre as cópias, partindo do pressuposto de que a consistência interna será mantida por algum protocolo de controle de concorrência (ver [BG81] e [BHG87] para uma descrição detalhada de vários protocolos dessa natureza). O critério de correção normalmente utilizado para a manutenção da consistência mútua é o de *one-copy serializability* (ou "serialização de uma cópia") [BG83], pelo qual duas operações conflitantes (escrita/escrita, escrita/leitura ou leitura/escrita) devem ser realizadas sobre conjuntos de cópias com intersecção não nula, ou seja, devem acontecer com pelo menos uma cópia ciente da existência de ambas operações. Este critério garante que operações realizadas sobre cópias de um dado replicado terão acesso a valores equivalentes aos que seriam vistos por essas mesmas operações caso o dado não estivesse replicado. Os protocolos de controle de réplicas tradicionais apresentados a seguir, bem como todos os protocolos revistos nos capítulos 2 e 3, mais o novo protocolo proposto no capítulo 4, adotam esse critério de correção.

1.3 Protocolos Tradicionais

1.3.1 Protocolos Simples

Uma solução simples para se implementar um protocolo de controle de réplicas correto é mapear uma operação de escrita sobre um dado replicado em operações de escrita sobre todas as suas cópias; com isso, uma operação de leitura pode ser realizada em qualquer uma das cópias, preferencialmente na cópia mais próxima ao nó que requisitou a consulta [TGGL82]. Esta solução é conhecida como protocolo *read-one-write-all* (ROWA). Embora correto, o protocolo ROWA possui a indesejável característica de não tolerar a falha de um único nó, pois isto já impediria a realização de operações de escrita.

O protocolo de *cópias acessíveis* [BG84] modifica o protocolo ROWA de modo que apenas as cópias acessíveis (armazenadas em nós que não estejam inoperantes) sejam alteradas por uma operação de escrita. Uma lista contendo as cópias que participaram da última operação de escrita deve ser mantida e, quando necessário, renovada por todos os nós para evitar que uma operação de leitura seja realizada sobre uma cópia desatualizada. Mesmo sendo mais robusto que o protocolo ROWA, o protocolo de cópias acessíveis também não tolera falhas que levem ao particionamento do sistema — o protocolo necessita do conhecimento sobre quais nós falharam e quais nós permanecem em funcionamento, o que é impossível de ser obtido quando o sistema está particionado.

A seguir, discute-se o problema de controlar dados replicados em sistemas particionados, e mostra-se a abordagem utilizada por alguns protocolos tradicionais.

1.3.2 Protocolos Tolerantes ao Particionamento do Sistema

Durante situações de particionamento do sistema, uma vez que os nós de partições distintas ficam incomunicáveis entre si, existe a possibilidade de que operações de escrita independentes, ou seja, operações que não tomam conhecimento uma da outra, sejam realizadas sobre cópias de um mesmo dado replicado em diferentes partições, ocasionando inconsistências. Duas abordagens para lidar com o particionamento do sistema têm sido tradicionalmente seguidas pelos protocolos de controle [DGMS85]: a *otimista* e a *pessimista*.

Na abordagem otimista, é permitido que cópias em diferentes partições sejam alteradas; conforme as falhas são reparadas e as partições deixam de existir, as cópias (ou as transações que atuaram sobre elas) antes separadas são comparadas e iniciam-se então procedimentos para fazê-las convergir para um estado consistente.

Tais procedimentos em geral constituem-se em cancelamentos e/ou novas execuções de transações conflitantes.

Na abordagem pessimista, apenas uma partição, chamada *partição privilegiada*, pode alterar as cópias; as outras partições têm suas cópias bloqueadas até o reparo das falhas, quando os nós da partição privilegiada poderão enviar para os outros nós anteriormente isolados em outras partições as alterações ocorridas durante o particionamento.

Os termos "otimista" e "pessimista" usados para designar as duas abordagens descritas referem-se à freqüência com que as falhas ocorrem e ao tempo esperado de duração do particionamento do sistema. Na abordagem otimista, supõe-se que as falhas são raras e, caso aconteçam, serão rapidamente reparadas; supõe-se também que a momentânea divergência entre as cópias não causará maiores problemas nem incorrerá em procedimentos de correção muito onerosos. No outro extremo, a abordagem pessimista considera que as falhas podem ser freqüentes e que o particionamento poderá durar um tempo indeterminado, sendo a consistência entre as cópias vital a todo instante.

Na prática, os pressupostos da abordagem otimista são raramente encontrados, principalmente no que se refere à possibilidade de que transações sejam canceladas facilmente. Um exemplo claro dessa dificuldade são sistemas que executam transações de efeito externo, como a emissão de cédulas em um caixa eletrônico de um banco ou a realização de um corte de uma peça em uma fábrica automatizada. Por este motivo, a grande maioria dos protocolos de controle de réplicas existentes na literatura segue a abordagem pessimista, priorizando a consistência em detrimento da disponibilidade. Esta proporção também é refletida na dissertação, sendo pessimistas os próximos protocolos tradicionais a serem revistos e todos os protocolos tratados nos capítulos 2, 3 e 4, esse último apresentando um novo protocolo. Alguns protocolos otimistas são sucintamente descritos no capítulo 5.

Cópia Primária [Sto79]

Neste protocolo, uma das cópias é designada a cópia primária do dado replicado, e como tal torna-se responsável pelo controle de todos os acessos requisitados. Uma operação de escrita é sempre realizada sobre a cópia primária, a qual encarrega-se de propagar as alterações para as outras cópias; uma operação de leitura pode ser realizada sobre qualquer cópia, desde que a cópia primária tenha dado a devida permissão. No caso de um particionamento do sistema, a partição privilegiada será aquela contendo o nó com a cópia primária, e somente a partição privilegiada, portanto, poderá alterar suas cópias.

As desvantagens desta solução são a necessidade de um procedimento adicional para a eleição de uma nova cópia primária se o nó que armazenava a anterior falhar.

e o fato de o nó com a cópia primária tornar-se um foco de congestionamento na rede. Outro problema ocorre se não é possível distinguir a falha de um nó da falha de canais da rede de comunicação; neste caso, existe a chance de que a falha de um ou mais canais da rede seja tomada como a falha do nó com a cópia primária e, desse modo, uma outra cópia primária pode ser eleita. De maneira oposta, pode-se tomar a falha do nó com a cópia primária como falha de canais da rede, o que deixaria o dado inacessível por tempo indeterminado.

Token [MW82]

Este protocolo é muito semelhante ao protocolo de cópia primária, exceto que a cópia primária agora é identificada como a cópia armazenada no nó que possui um objeto especial chamado *token*. O *token* pode ser passado livremente para outros nós, possibilitando a mudança de localização da cópia primária mesmo quando nenhum nó tenha falhado. Havendo um particionamento, a partição privilegiada será aquela contendo o nó com o *token*.

Tal como na solução anterior, é necessário ainda um procedimento adicional para criar novamente o *token* caso ele seja perdido devido a falhas. O problema do congestionamento é amenizado e o tempo de resposta melhorado com a possibilidade de a cópia primária ser movida para locais onde as requisições de acesso são mais freqüentes.

Votação Simples [Tho79]

Esta foi a primeira solução para a manutenção da consistência entre as cópias com controle realmente distribuído, ao contrário do controle centralizado dos protocolos de cópia primária e *token*, e ao mesmo tempo tolerante ao particionamento do sistema. Neste protocolo, tanto operações de escrita quanto operações de leitura são realizadas sobre um conjunto formado pela maioria ($\lfloor N/2 \rfloor + 1$) das N cópias do dado replicado. A esse conjunto de cópias denomina-se *quorum*. Como $2(\lfloor N/2 \rfloor + 1) > N$, é garantido que um quorum de escrita sempre terá uma intersecção não nula com todo quorum de leitura ou qualquer outro quorum de escrita, e que, se o sistema estiver particionado, apenas uma partição privilegiada existirá. O dado estará disponível enquanto puder ser contactada pelo menos a maioria das cópias.

A desvantagem deste protocolo está na degradação das operações de leitura, antes realizadas com o acesso a uma única cópia. Outra restrição é o fato de que, caso não haja uma partição contendo a maioria das cópias, o dado replicado ficará inacessível até que as falhas sejam reparadas e uma partição privilegiada com pelo menos esse número de cópias passe a existir.

Votação Ponderada [Gif79]

Neste protocolo, a cada uma das N cópias é atribuído um certo número de votos (ou peso), e os quorums de escrita e leitura são formados por quaisquer conjuntos de cópias cuja soma dos votos seja pelo menos r e w, respectivamente. Sendo v a soma total de votos atribuídos às cópias, os valores r e w devem ser definidos satisfazendo as seguintes restrições:

$$r + w > v \tag{1.1}$$
$$2w > v \tag{1.2}$$

A restrição 1.1 garante que há uma intersecção não nula entre todo quorum de leitura e todo quorum de escrita, assegurando assim que qualquer quorum de leitura conterá sempre uma cópia atualizada. Com o sistema particionado, essa restrição garante que o dado replicado não será lido em uma partição e alterado em outra. A restrição 1.2 evita que duas operações de escrita aconteçam independentemente; desse modo, no caso de o sistema estar particionado, a existência de apenas uma partição privilegiada é assegurada.

A flexibilidade para a definição dos valores r e w torna este protocolo uma generalização de outros protocolos descritos anteriormente. Atribuindo-se um voto a cada cópia, com $r = 1$ e $w = N$, o protocolo de votação ponderada corresponde ao protocolo ROWA; com $r = w = \lfloor N/2 \rfloor + 1$, a correspondência é com o protocolo de votação simples. Pode-se ainda atribuir maior número de votos aos nós mais confiáveis, aumentando-se as chances de que quorums possam ser formados.

Tal como no protocolo de votação simples, esta solução possui a desvantagem de degradar as operações de leitura uma vez que também é exigida a formação de um quorum antes da realização dessas operações, além de deixar o dado inacessível para escrita se não houver uma partição contendo pelo menos w votos. Há ainda o fato de o controle não ser plenamente distribuído caso tenha sido feita uma associação não uniforme de votos, o que implica em sobrecarga para os nós de maior peso.

1.4 Organização da Dissertação

A seguir é descrito o conteúdo do restante da dissertação.

O capítulo 2 trata da revisão e análise de protocolos de controle de réplicas que propõem melhorias ao protocolo de votação (ponderada) tradicional.

O capítulo 3 revisa protocolos que utilizam estruturas lógicas para formar os quorums. Esses protocolos consideram sistemas distribuídos de larga escala e visam

melhorar a distribuição da carga (e o desempenho, conseqüentemente) do sistema através da redução do tamanho dos quorums.

O novo protocolo proposto, denominado protocolo de votação hierárquica estendida, também utiliza uma estrutura lógica na formação dos quorums; ele generaliza e supera grande parte dos protocolos revisados no capítulo 3. A descrição detalhada do novo protocolo é feita no capítulo 4.

O capítulo 5 revisa sucintamente alguns protocolos que partem de princípios um pouco diferentes daqueles nos quais se baseiam os protocolos revistos nos capítulos anteriores. Como exemplo, citam-se protocolos que seguem a abordagem otimista, que adotam novos critérios de correção (que não o *one-copy serializability*), ou que fazem uso de conhecimento semântico das transações para manter a consistência entre as cópias.

A dissertação é concluída no capítulo 6 com a enumeração dos fatores que devem ser considerados na escolha de um protocolo de controle de réplicas para um sistema distribuído, e com uma discussão sobre em quais circunstâncias, considerando-se os fatores enumerados, o protocolo de votação hierárquica estendida proposto na dissertação é a melhor opção para os projetistas do sistema.

Para facilitar e melhorar o entendimento dos protocolos revisados e analisados na dissertação, os capítulos 2, 3 e 4 são concluídos com uma seção especial de notas bibliográficas. Estas seções contêm referências a trabalhos não abordados por questões de espaço, trabalhos que deram origem aos protocolos revisados no capítulo, ou ainda extensões mais recentemente feitas a esses protocolos.

Capítulo 2

Protocolos Baseados em Votação

Dentre os protocolos de controle de réplicas que seguem a abordagem pessimista para lidar com o particionamento do sistema, a grande maioria certamente é composta por protocolos que melhoram, com respeito a diferentes aspectos, o protocolo de votação tradicional. Este capítulo revisa e compara vários protocolos desse tipo, aqui denominados protocolos baseados em votação. Para facilitar as comparações, protocolos com características afins são revisados conjuntamente (em uma mesma seção).

A próxima seção descreve protocolos que visam reduzir o espaço de armazenamento das cópias. Protocolos que implementam o acesso ao dado de maneira mais eficiente a partir da manutenção de visões convergentes dos nós sobre a topologia corrente da rede de comunicação são descritos na seção 2.2. A seção 2.3 trata de protocolos que buscam aumentar a disponibilidade do dado através da adaptação do tamanho dos quorums às mudanças ocorridas na configuração do sistema. Por fim, uma análise comparativa dos protocolos revisados é feita na seção 2.4.

2.1 Reduzindo o Espaço de Armazenamento

Um dos custos associados aos benefícios da replicação dos dados é a alta ocupação de espaço de armazenamento: um dado replicado com N cópias necessita N vezes mais espaço para ser armazenado do que o mesmo dado em um ambiente sem replicação. A idéia central dos dois protocolos descritos nesta seção é modificar o protocolo de votação tradicional para obter os benefícios de um dado replicado possuindo N cópias mas com a ocupação física de um espaço equivalente a apenas M delas ($M \leq N$).

2.1.1 Votação com Testemunhas

O protocolo de votação com testemunhas [Pâr86] propõe a redução do espaço de armazenamento do dado replicado através da substituição de algumas réplicas por meros registros contendo apenas informações sobre o estado corrente de uma cópia, como o número de versão, por exemplo, e não o dado propriamente dito. A cada um desses registros denomina-se uma *testemunha*. Nesse esquema, um dado lógico com N réplicas tem na verdade M cópias de fato e $N - M$ testemunhas.

A cada testemunha é atribuído um número específico de votos, tal como é feito a uma cópia convencional, sendo, portanto, permitido que testemunhas participem normalmente do processo de votação para a formação dos quorums. A restrição adicional é que todo quorum deve possuir pelo menos uma cópia corrente (evitando-se, assim, que um quorum possa ser formado apenas por testemunhas, ou por testemunhas e cópias desatualizadas). Esta restrição é necessária porque uma testemunha, não contendo dados, não pode ser usada para consultas nem para atualizar cópias que tenham perdido alterações recentes. A desvantagem desta restrição adicional, que é o preço a ser pago pelo uso de testemunhas, é o fato de haver situações em que um dado replicado com N cópias convencionais estaria acessível enquanto o mesmo dado com M cópias e $N - M$ testemunhas estaria bloqueado. Exemplos de tais situações são mostrados a seguir.

Considere um dado replicado com três cópias e duas testemunhas, com cada entidade (cópia ou testemunha) possuindo um voto. Suponha agora que duas entidades estão inacessíveis e os números de versão das outras três são 5, 5 e 8; nesta configuração, os números de versão das duas entidades inacessíveis não podem ser menores que 8, uma vez que a oitava alteração deve ter envolvido pelo menos três entidades, nem maiores que 8, já que qualquer alteração subseqüente teria envolvido mais que duas delas. Desse modo, as configurações possíveis para as três entidades que podem ser contactadas, com respeito a quais são cópias e quais são testemunhas (identificadas pelo número de versão sublinhado), são:

[5, 5, 8],
[5, 5, 8],
[5, 5, 8],
[5, 5, 8],
[5, 5, 8],
[5, 5, 8] e
[5, 5, 8].

Em três ([5, 5, 8], [5, 5, 8] e [5, 5, 8]) das sete configurações possíveis não existe nenhuma cópia corrente — embora haja uma testemunha atualizada. Nessas três

configurações, o dado já ficaria bloqueado devido à restrição adicional que a presença de testemunhas impõe ao processo de votação; não havendo testemunhas, o bloqueio do dado claramente não aconteceria. Segundo análises probabilísticas feitas em [Pár86], situações assim são raras e não impedem que a disponibilidade do dado no protocolo de votação com testemunhas seja comparável à disponibilidade no protocolo de votação tradicional.

Além da vantagem de reduzir o espaço de armazenamento, a utilização de testemunhas agiliza a realização de operações de escrita uma vez que a atualização de uma testemunha significa simplesmente o incremento de seu número de versão. Tendo custos desprezíveis de armazenamento, testemunhas ainda podem ser criadas ou realocadas mais facilmente do que cópias convencionais.

2.1.2 Votação com Fragmentação

Diferentemente do protocolo de votação com testemunhas, o protocolo de votação com fragmentação [AA90b] reduz o espaço de armazenamento das réplicas fragmentando o dado e replicando parcialmente seus fragmentos. Mais especificamente, o dado é dividido em N fragmentos e a replicação desses fragmentos é feita de forma que:

(1) cada fragmento seja armazenado em M ($M \leq N$) nós, e

(2) cada nó armazene M fragmentos distintos.

Dessa maneira, o dado é distribuído por N nós mas, como cada fragmento é replicado em M destes nós, o espaço físico ocupado é equivalente a apenas M cópias. O conjunto dos M fragmentos distintos armazenados em um mesmo nó é denominado um *segmento*. O número mínimo de segmentos necessários, no pior caso, para a reconstrução total do dado (isto é, o menor número de segmentos que sempre contém todos os N fragmentos) é denotado por S; é fácil ver que $S = N - M + 1$. A figura 2.1 mostra um exemplo de um dado replicado com três cópias armazenadas em cinco nós ($M = 3$, $N = 5$ e, portanto, $S = 3$). Note que o dado pode ser reconstruído somente com os segmentos s_1 e s_4 ou s_2 e s_5, mas isto não é verdade para qualquer par de segmentos. Por outro lado, quaisquer três segmentos contêm todos os fragmentos e são suficientes para a reconstrução do dado.

A cada segmento é atribuído um número de versão, tal como é feito às cópias convencionais, e os tamanhos q_r e q_w dos quorums de leitura e escrita, respectivamente, referem-se agora ao número mínimo de segmentos a ser contactado para que a operação correspondente possa ser realizada. Os valores de q_r e q_w são definidos seguindo as restrições:

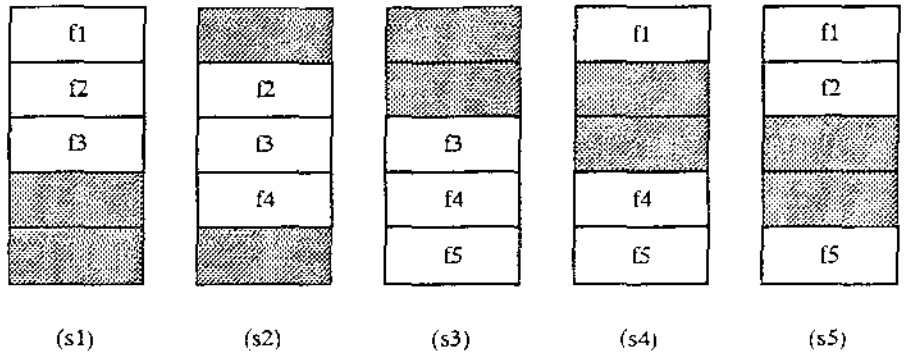

Figura 2.1: Um dado replicado com 3 cópias distribuídas por 5 segmentos.

$$S \leq q_r \leq N \tag{2.1}$$

$$max(S, \lfloor N/2 \rfloor + 1) \leq q_w \leq N \tag{2.2}$$

$$N + S \leq q_r + q_w \leq 2N \tag{2.3}$$

A restrição 2.1 assegura que uma operação de leitura terá acesso a todos os N fragmentos do dado. A restrição 2.2 garante que uma operação de escrita será realizada sobre todos os fragmentos e que quaisquer duas operações de escrita terão pelo menos um segmento em comum. Finalmente, a restrição 2.3 garante que qualquer par de quorums leitura/escrita terá pelo menos S segmentos em comum, ou seja, que pelo menos uma réplica de cada um dos N fragmentos do dado será comum aos dois quorums; como conseqüência, é garantido que toda operação de leitura será realizada sobre segmentos atualizados.

Comparado ao protocolo de votação tradicional, fazendo-se $N/2 < M \leq N$, o protocolo de votação com fragmentação necessita dos mesmos tamanhos de quorums para operações de escrita em detrimento dos tamanhos dos quorums para operações de leitura, ou vice-versa. Isto acontece devido à restrição 2.3 exigir pelo menos S segmentos comuns a qualquer par leitura/escrita de quorums, em contraste com a exigência de uma única cópia comum aos dois quorums no protocolo de votação tradicional.

Esta exigência a mais da restrição 2.3 pode ser contornada com a integração de um mecanismo de propagação [WB84] ao sistema. Para implementar o mecanismo de propagação, cada nó deve manter um *log* de eventos que correspondem às operações de leitura e escrita realizadas sobre o dado replicado. Toda comunicação no sistema passa então a ser feita exclusivamente através de: a) mapeamento das requisições de operações sobre o dado e de seus conseqüentes resultados em operações de inserção de eventos no *log* (mantido localmente), e b) intercâmbio de cópias dos *logs* entre

os nós. Ao receber a cópia de um outro *log*, um nó atualiza seu *log* com eventos sobre os quais ainda não tenha conhecimento e descarta aqueles eventos que já são conhecidos por todos (esses eventos já conhecidos são identificados examinando-se os *logs* recebidos dos outros nós). Dessa maneira, é possível o abrandamento da restrição 2.3 para

$$N + 1 \leq q_r + q_w \leq 2N \tag{2.4}$$

com a exigência de que a intersecção entre os quorums seja agora de apenas um segmento. A garantia de que o dado será totalmente reconstruído, uma vez que não há mais a intersecção mínima de S segmentos, está no fato de cada *log* conter eventos relativos ao dado como um todo, e não apenas ao segmento armazenado no nó. Em [AA90b] é demonstrado que os segmentos contactados obedecendo-se à restrição 2.4 mais as informações contidas no *log* são suficientes para a reconstrução total do dado.

No pior caso, o tamanho de um *log* pode crescer até o equivalente a uma cópia completa do dado (quando todo o conteúdo do dado foi gerado por operações de escrita cujos eventos correspondentes ainda permanecem no *log*). Desse modo, quando as operações de escrita são muito freqüentes em relação à freqüência com que as cópias dos *logs* são intercambiadas, o espaço total de armazenamento pode chegar a N vezes o tamanho do dado original, o mesmo espaço necessário utilizando-se o protocolo de votação tradicional. Com o dado sofrendo poucas alterações, os *logs* crescem muito menos e, nesse caso, consegue-se as vantagens de um dado replicado com N cópias ocupando-se um espaço consideravelmente menor (M vezes o tamanho do dado original, no melhor caso).

2.2 Capturando a Topologia Corrente da Rede

No capítulo anterior foi visto que o protocolo de cópias acessíveis implementa eficientemente uma operação de leitura com o acesso a uma única cópia, para isso realizando uma operação de escrita sobre todas as cópias que puderem ser contactadas. Infelizmente, essa idéia não pode ser aplicada quando é considerada a possibilidade de haver particionamento do sistema. Mais precisamente, esta impossibilidade acontece porque nós e canais de comunicação estão sujeitos a falhas, e a detecção imediata dessas falhas nem sempre é possível para todos os nós que permanecem operando corretamente; o mesmo também é verdade para a detecção, após as falhas serem reparadas, do retorno dos componentes ao funcionamento.

Como conseqüência da detecção não imediata das falhas e dos reparos dos componentes por todos os nós, a topologia lógica do sistema (que pode ser representada

pelos grupos de nós que compõem as partições existentes) é alterada dinamicamente diante da ocorrência de falhas. Estas alterações na topologia lógica do sistema podem provocar situações nas quais os nós têm visões inconsistentes sobre com quais outros nós estão aptos a se comunicar, ou seja, sobre quais outros nós compõem a partição que os inclui. Um nó, por exemplo, pode não ter detectado a recuperação da comunicação entre partições (ou o surgimento de um novo particionamento) e concluir erradamente que ainda não pode se comunicar com nós de sua própria partição (ou que pode se comunicar com nós de partições diferentes).

Os dois protocolos revisados nesta seção conseguem realizar as operações de acesso ao dado replicado com a mesma eficiência do protocolo de cópias acessíveis e ainda tolerar o particionamento do sistema. Para isso, ambos protocolos (o segundo na verdade é uma generalização do primeiro) possuem um mecanismo que permite aos nós compartilharem visões consistentes sobre a topologia da rede. A descrição de como isto é implementado pelos dois protocolos é feita em seguida.

2.2.1 Votação com Partições Virtuais

O protocolo de votação com partições virtuais [ASC85] constitui-se de duas camadas distintas: a camada de gerenciamento das visões ou *partições virtuais* (PVs) que os nós compartilham e que indicam quais nós conseguem se comunicar entre si, e a camada de gerenciamento das réplicas, encarregada de controlar os acessos ao dado replicado e mapear as operações lógicas requisitadas em operações físicas sobre as cópias.

Quando um nó percebe alguma mudança na configuração do sistema (causada pela falha de um componente, por exemplo), ele inicia a execução da primeira fase de um protocolo especial da camada de gerenciamento de PVs, chamado protocolo de criação de PV, enviando aos demais nós do sistema solicitações de engajamento à nova PV a ser criada. Caso a maioria dos nós responda positivamente ao protocolo de criação, a nova PV é efetivada e o nó iniciante passa para a segunda fase do protocolo, enviando os atributos da nova PV para os nós que concordaram com sua criação. São três os atributos de uma PV: um identificador, univocamente gerado pelo nó que deu início ao protocolo de criação, e que serve para assegurar que cada nó pertencerá a uma única PV de cada vez; um conjunto de membros potenciais, que são os nós que concordaram com a criação da nova PV; e um conjunto de membros de fato, formado pelos nós que realmente fazem parte da PV (inicialmente, os conjuntos de membros potenciais e de membros de fato são iguais). O identificador e o conjunto de membros potenciais são atributos estáticos, estabelecidos na primeira fase de execução do protocolo, e são os mesmos para todos os membros potenciais. O conjunto dos membros de fato é dinâmico e nem sempre poderá ser o mesmo

para todos os nós da PV. Isto acontece porque um membro potencial deixa de ser um membro de fato simplesmente alterando o valor de uma variável local, o que possibilita um nó, ao suspeitar de alguma mudança na topologia vigente, retirar-se independentemente de sua atual PV e iniciar a criação de uma nova. A detecção de mudanças na topologia da rede é feita com mais exatidão através do envio periódico de mensagens "sonda", tarefa de outro protocolo da camada de gerenciamento de PVs chamado protocolo sonda.

Os protocolos da camada de gerenciamento das réplicas encarregam-se de verificar a *accessibilidade* do dado e de mapear as operações lógicas sobre o dado em operações físicas sobre as cópias. Um dado replicado é considerado *accessível* a um nó i se a maioria das cópias desse dado está armazenada em nós pertencentes à mesma PV de i; com isso, garante-se que apenas uma PV por vez terá acesso ao dado. Note que a verificação de accessibilidade é feita sem a necessidade de qualquer comunicação externa. O mapeamento das operações lógicas em operações físicas segue o mesmo esquema do protocolo ROWA, e é feito através das duas regras seguintes:

Regra de leitura Um nó i realiza uma operação de leitura verificando se o dado está accessível a ele e, se estiver, enviando uma requisição de leitura para *alguma* cópia armazenada em qualquer nó da mesma PV de i.

Regra de escrita Um nó i realiza uma operação de escrita verificando se o dado está accessível a ele e, se estiver, enviando uma requisição de escrita para *todas* as cópias armazenadas em nós da mesma PV de i.

A garantia de que apenas uma partição (real) poderá proclamar-se privilegiada advém do fato de ser exigida a maioria dos nós tanto para a criação de uma nova PV quanto para que o dado seja considerado accessível dentro de uma PV vigente.

2.2.2 Votação com Partições Virtuais Generalizada

Este protocolo [AT89] modifica o protocolo anterior generalizando o processo de formação dos quorums e eliminando a camada de gerenciamento de PVs. Em seguida, além da descrição dessas modificações, são apresentadas estratégias, também presentes em [AT89], para se determinar o melhor momento para a criação de uma nova PV.

Seja v uma PV. No protocolo de votação com partições virtuais generalizada, um dado replicado é considerado accessível para leitura ou escrita em v se A_r ou A_w cópias, respectivamente, estão armazenadas em nós pertencentes a v. Estes

valores, chamados *limites de acessibilidade*, são estabelecidos seguindo a restrição $A_r + A_w > N$, onde N é o número total de cópias do dado; esta restrição garante que qualquer conjunto de cópias usado em um acesso para leitura sempre conterá uma cópia atualizada. Note que a ausência da restrição $2A_w > N$ permite que o dado seja alterado concorrentemente em mais de uma partição, o que não é possível no protocolo original. Para cada PV v são definidos um quorum de leitura $q_r(v)$ e um quorum de escrita $q_w(v)$. Estes quorums são usados no mapeamento das operações lógicas em físicas e devem satisfazer as restrições (considera-se $n(v)$ o número de nós pertencentes a v que contêm uma cópia do dado):

$$q_r(v) + q_w(v) > n(v) \tag{2.5}$$

$$2q_w(v) > n(v) \tag{2.6}$$

$$1 \leq q_r(v) \leq n(v) \tag{2.7}$$

$$A_w \leq q_w(v) \leq n(v) \tag{2.8}$$

As restrições 2.5 e 2.6 garantem a consistência entre as cópias de uma mesma PV; as restrições 2.7 e 2.8 compatibilizam o tamanho do quorums definidos para a PV com os valores dos limites de acessibilidade do dado. O protocolo de votação com partições virtuais generalizada corresponde ao protocolo de votação com partições virtuais original quando $A_r = A_w = \lfloor N/2 \rfloor + 1$, $q_r = 1$ e $q_w = n(v)$.

A eliminação da camada de gerenciamento de PVs presente no protocolo original é decorrente da constatação de que o protocolo de criação de uma nova PV executado em duas fases é redundante: o próprio protocolo de controle de concorrência (o protocolo de validação em duas fases [Gra78], por exemplo) normalmente já é executado assim. Em vista disso, as PVs passam a ser tratadas como objetos sujeitos ao controle de concorrência e criadas através da execução de transações específicas para este fim. Uma transação de criação de PV é encarregada de contactar os outros nós e de, caso a maioria concorde com a criação da nova PV, estabelecer os atributos da nova PV e enviá-los para os outros membros potenciais (que não o nó iniciante da transação de criação de PV). Quanto às estratégias para se determinar o melhor momento de criação de uma nova PV, três são sugeridas: a estratégia "agressiva", na qual uma nova PV é criada tão logo quanto uma mudança na topologia da rede seja detectada; a estratégia "por objetos", onde somente cria-se uma nova PV se objetos de alta prioridade correntemente inacessíveis tornar-se-ão acessíveis na PV a ser criada; e a estratégia "sob demanda", na qual, analogamente à estratégia "por objetos", somente cria-se uma PV se transações de alta prioridade correntemente incapacitadas de serem executadas poderão ser executadas na nova PV.

A grande vantagem deste protocolo generalizado em relação ao protocolo original é sua enorme flexibilidade para balancear, de acordo com as características do ambiente utilizado, os custos de comunicação inerentes à realização de operações de leitura

e escrita sobre o dado replicado. Isto deve-se ao fato de o protocolo de votação com partições virtuais generalizada permitir que diferentes limites de acessibilidade do dado para leitura e para escrita sejam definidos, e que cada PV tenha seus próprios quorums.

2.3 Aumentando a Disponibilidade do Dado

Embora tolerante a falhas de um certo número de nós, canais da rede de comunicação e ao particionamento do sistema, o protocolo de votação tradicional não permite que operações sejam realizadas sobre o dado replicado se o número mínimo exigido de cópias para formar os quorums não consegue ser contactado. A princípio, supondo-se um dado replicado com N cópias, uma operação que requer um quorum de Q cópias consegue ser realizada enquanto houver pelo menos Q cópias podendo ser contactadas o que significa uma tolerância a falhas de até $N-Q$ nós que armazenam uma cópia do dado. Esta asserção não é verdadeira, porém, se é considerada a possibilidade de o sistema particionar-se, uma vez que pode haver Q ou mais cópias acessíveis distribuídos de forma tal que nenhuma partição contenha mais que $Q-1$ cópias. A mesma objeção vale para o caso em que diferentes quantidades de votos são atribuídas às cópias, visto que as partições também podem não conter um conjunto de cópias cuja soma dos votos seja suficiente para formar o quorum exigido.

É fácil constatar, portanto, que quanto maior o tamanho do quorum, menores serão as chances de conseguir que ele seja formado ante a ocorrência de falhas. Desse modo, pode-se dizer que a disponibilidade do dado diminui quando os componentes falham devido ao fato de os quorums, nesse caso, ficarem proporcionalmente maiores. Isto acontece porque apenas as cópias acessíveis podem compor os quorums, e não mais todos as cópias do dado replicado. A idéia básica dos protocolos revistos nesta seção, chamados protocolos dinâmicos, é alterar dinâmica e automaticamente o tamanho dos quorums quando falhas são detectadas de forma a tentar evitar situações nas quais o dado não possa ser lido ou alterado. O primeiro protocolo descrito faz isso através da alteração dinâmica dos números de votos atribuídos às cópias; o terceiro protocolo é uma extensão do segundo e ambos alteram o tamanho dos quorums formando-os a partir apenas das cópias que participaram da mais recente operação de escrita realizada.

2.3.1 Votação com Reatribuição Dinâmica de Votos

Neste protocolo [BGMS89], toda vez que falhas são detectadas altera-se a quantidade de votos atribuída a cópias pertencentes à partição privilegiada. A idéia é fazer com

que a nova atribuição de votos diminua as chances de que não se consiga formar os quorums caso novas falhas venham a acontecer.

Suponha que o dado replicado possui quatro cópias (A, B, C e D). Suponha também que inicialmente foi dada às cópias a seguinte atribuição de votos: $v_A = v_B = v_C = 1$ e $v_D = 2$. Sob essa atribuição, qualquer conjunto de três cópias ou qualquer de duas cópias das quais uma é a cópia D tem a maioria dos votos e, portanto, pode ser usado como um quorum de escrita. Suponha agora que o sistema particionou-se e deixou a cópia D isolada das outras três. Nesse instante, o conjunto formado pelas cópias A, B e C tem a maioria dos votos e constitui a partição privilegiada. Caso um novo particionamento aconteça isolando a cópia C das cópias A e B, nenhuma das três partições ({A, B}, {C} ou {D}) terá a maioria dos votos e o dado ficará bloqueado. Entretanto, poder-se-ia evitar essa situação de bloqueio se a quantidade de votos das cópias da partição {A, B, C} fosse aumentada antes do segundo particionamento. Por exemplo, a nova atribuição de votos poderia ser: $v_A = v_B = v_C = 5$ (a cópia D permaneceria com dois votos pois não pertencia à partição privilegiada). Dessa maneira, após o segundo particionamento a partição {A, B} teria a maioria dos votos (10 de um total de 17) e assim o bloqueio do dado seria evitado. A nova partição privilegiada {A, B} poderia ainda fazer uma nova atribuição de votos para $v_A = 15$ e $v_B = 5$, o que propiciaria a tolerância de um terceiro particionamento que isolasse as quatro cópias — a partição privilegiada seria aquela contendo a cópia A (com 15 votos de um total de 27).

A partir desse exemplo, observa-se que a partição privilegiada pode dinamicamente reatribuir os votos de suas cópias e assim aumentar seu poder de voto, conseqüentemente aumentando também as chances de o dado continuar acessível diante da hipótese de falhas subseqüentes.

A reatribuição dinâmica de votos pode acontecer basicamente de duas maneiras: com "consenso da maioria", onde os nós da partição privilegiada entram em acordo sobre uma nova atribuição de votos para as cópias através da eleição de um nó coordenador; ou com "autonomia", onde cada nó toma sua própria decisão sobre quando e em quanto mudar o número de votos de sua cópia independentemente dos outros nós. Nos dois casos, deve existir uma política para a escolha da nova atribuição de votos de forma a melhor compensar o poder de voto perdido com a ocorrência de falhas.

Aqui são revisados somente protocolos para a reatribuição de votos com autonomia. A reatribuição com consenso da maioria pode ser implementada através da eleição de um nó coordenador (ver [GM82] para algoritmos de eleição) e, a partir da informação que os nós enviam sobre a visão que eles têm da topologia lógica do sistema, da escolha de uma atribuição ótima de votos para as cópias [BGM87, KS88a, CAA89], tópicos que estão fora do escopo desta dissertação.

Reatribuição Dinâmica de Votos com Autonomia

Dois protocolos são apresentados: no primeiro, os nós podem apenas aumentar os votos de suas cópias; no segundo, os nós tanto podem aumentar quanto diminuir os votos. O nó que decide alterar os votos de sua cópia é considerado o nó *iniciante* da reatribuição de votos; os outros nós informados pelo nó iniciante da a nova atribuição são os nós *participantes*.

Protocolo para aumentar os votos

Neste protocolo, cada nó mantém um vetor contendo o número de votos de todas as cópias (inclusive a sua própria) do dado; este vetor é enviado junto (ou como) a resposta positiva a uma solicitação de votos para a realização de algum evento (pedido de acesso ao dado, por exemplo). O protocolo para os nós iniciante e participantes é descrito a seguir:

Iniciante Solicita votos dos outros nós e atualiza seu vetor de votos com os vetores de votos recebidos; se obtém a maioria dos votos (sob a atribuição de votos vigente), envia o novo (aumentado) número de votos de sua cópia para os outros nós; se obtém confirmação de recebimento de um grupo de nós contendo a maioria dos votos (ainda sob a atribuição de votos vigente), efetiva a alteração no seu vetor de votos local.

Participantes Registram a alteração recebida no vetor de votos local e enviam uma confirmação de recebimento ao nó iniciante.

Protocolo para aumentar e diminuir os votos

Como o número de votos das cópias pode aumentar ou diminuir, torna-se necessário neste protocolo que cada nó mantenha, além do vetor de votos, um vetor de versões que indique a versão do número de votos de todas as cópias do dado. Os dois vetores devem ser enviados em resposta a qualquer solicitação de votos. A necessidade do vetor de versões é mostrada em seguida.

O nó iniciante verifica se houve alteração no número de votos das outras cópias comparando os vetores de votos recebidos com seu vetor de votos local. Quando o número de votos de uma cópia armazenada em um nó i de um dos vetores de votos recebidos é menor que o número de votos dessa mesma cópia no vetor de votos local do nó iniciante, dois fatos podem ser deduzidos: que o nó i diminuiu os votos de sua cópia, ou que o nó i aumentou os votos de sua cópia mas esse aumento ainda não foi efetivado. O vetor de versões permite identificar quais dos dois eventos ocorreu: se a versão do número de votos da cópia do nó i no vetor de versões recebido é maior

que a versão do número de votos da mesma cópia no vetor de versões local então o nó i de fato diminuiu os votos de sua cópia; caso contrário, há um aumento de votos da cópia do nó i em andamento.

O protocolo para aumentar os votos, ligeiramente modificado para incluir a possibilidade de os votos decrescerem, é descrito a seguir.

Iniciante Solicita votos dos outros nós e atualiza seu vetor de votos e seu vetor de versões com os vetores de votos e versões recebidos; envia seus vetores de votos e versões mais o novo (aumentado) número de votos de sua cópia para os outros nós; se obtém confirmação de recebimento de um grupo de nós contendo a maioria dos votos (sob a atribuição de votos vigente), efetiva a alteração e incrementa de um a versão do número de votos de sua cópia (nos vetores de votos e versões locais).

Participantes Atualizam, se necessário, seus vetores de votos e versões locais; incrementam de um a versão do número de votos da cópia do nó iniciante no vetor de versões local.

O protocolo para diminuir os votos é o que segue.

Iniciante Solicita votos dos outros nós e atualiza seu vetor de voto e seu vetor de versões com os vetores de votos e versões recebidos; diminui o número de votos de sua cópia e incrementa de um a versão correspondente; envia seus vetores de votos e versões para os outros nós.

Participantes Atualizam, se necessário, seus vetores de votos e versões locais.

O protocolo para apenas aumentar os votos é mais simples porém requer um mecanismo para reestabelecer os valores originais dos votos toda vez que eles atingem um certo limite superior. O protocolo para tanto aumentar quanto diminuir os votos é mais flexível e permite, por exemplo, o retorno aos números de votos originais após o reparo das falhas; a desvantagem é sua maior complexidade e a necessidade de que mais informações sejam mantidas nos nós, e maiores mensagens tenham de ser enviadas. Em [BGMS89] são apresentadas provas minuciosas de que ambos os protocolos podem ser executados concorrentemente sem violar as restrições impostas ao processo de formação dos quorums.

A seguir são indicadas algumas políticas para saber-se quando e em quanto alterar a quantidade de votos das cópias.

Políticas de Reatribuição de Votos

Uma nova atribuição de votos deve ser feita em duas situações: quando é detectada a saída de um nó da partição privilegiada (devido à falha de um nó ou a um particionamento do sistema), e quando um nó torna a fazer parte dela. No primeiro caso, um nó (ou mais) da partição privilegiada deve(m) aumentar o número de votos de sua(s) cópia(s) para tentar restaurar o poder de voto perdido. Seja i o nó que saiu e v_i o número de votos da cópia armazenada no nó i. O aumento dos votos da partição privilegiada deve ser de pelo menos $2v_i$ votos (v_i para suplantar os votos perdidos do nó i e mais v_i para cobrir o acréscimo no total de votos atribuídos). No segundo caso, o nó que está retornando à partição privilegiada deve também aumentar os votos de sua cópia, ou os nós que haviam aumentado os votos de suas cópias devem diminuí-los, restaurando assim o poder de voto anterior à falha.

Vale ressaltar que a formação correta dos quorums diante da possibilidade de o número de votos das cópias ser alterado é garantida pelos protocolos para reatribuição de votos; as políticas para a escolha da nova atribuição de votos não afetam essa garantia — o único risco é, no pior dos casos, elas levarem a uma nova atribuição de votos pouco proveitosa para tolerar falhas subseqüentes.

2.3.2 Votação Dinâmica

A idéia do protocolo de votação dinâmica [JM90] é formar os quorums apenas a partir das cópias correntes do dado, e não a partir de quaisquer cópias, como acontece no protocolo de votação tradicional; um quorum de escrita, por exemplo, deve ser formado por pelo menos a maioria dos votos de cópias que estiverem atualizadas, ou seja, cópias que participaram da última operação de escrita realizada. Como o número de cópias correntes pode variar com a ocorrência de falhas, o tamanho dos quorums é dinâmico, aumentando ou diminuindo de acordo com as mudanças na configuração do sistema.

Associam-se a cada cópia do dado, além do número de versão e do número de votos, dois atributos adicionais: a *cardinalidade de alteração* e o *nó privilegiado*. A cardinalidade de alteração indica quantos nós (ou, no caso de haver cópias com diferentes números de votos, a soma dos votos das cópias que) participaram da mais recente operação de escrita realizada sobre o dado replicado, e deve ser enviada em resposta a cada solicitação de votos; o nó privilegiado identifica qual nó da partição privilegiada vigente, escolhido por um mecanismo de eleição ou através de uma ordenação linear previamente imposta aos nós, será usado como critério de "desempate" caso a partição privilegiada vigente subdivida-se em duas partições menores com exatamente o mesmo número de votos (nessa situação, a nova partição

privilegiada será aquela contendo o nó privilegiado) este atributo é obviamente desnecessário quando a cardinalidade de alteração das cópias da partição privilegiada é um número ímpar. Em seguida é apresentado um exemplo do funcionamento do protocolo de votação dinâmica.

Considere um dado replicado contendo cinco cópias armazenadas nos nós A, B, C, D e E, com um voto atribuído a cada cópia. Os cinco nós estão linearmente ordenados (A>B>C>D>E) para que, quando necessário, o nó privilegiado possa ser determinado. Para simplificar será considerada somente a realização de operações de escrita, as quais exigem um quorum de pelo menos a maioria das cópias correntes. Suponha agora que o dado sofreu nove alterações e que nenhuma falha ocorreu no sistema (os nós, portanto, constituem uma única partição). Sejam NV (número de versão), CA (cardinalidade de alteração) e NP (nó privilegiado) os atributos das cópias, esta configuração hipotética poderia ser representada por

	A	B	C	D	E
NV	9	9	9	9	9
CA	5	5	5	5	5
NP	-	-	-	-	-

Neste ponto, suponha que o nó C inicia uma operação de escrita e descobre que consegue comunicar-se apenas com os nós D e E. Como C ainda pertence a uma partição privilegiada (que possui três das cinco cópias que participaram da última alteração feita ao dado), o quorum consegue ser formado. A nova configuração passa a ser

	A	B	C	D	E
NV	9	9	10	10	10
CA	5	5	3	3	3
NP	-	-	-	-	-

Suponha que o nó C inicia uma outra operação de escrita e descobre que consegue comunicar-se apenas com o nó D. Como C e D formam uma maioria dentre os três nó que participaram da última alteração, a operação consegue ser realizada. Note que isto não mais seria possível no protocolo de votação tradicional porque C e D não são maioria em relação a todos os cinco nós. A configuração muda então para

	A	B	C	D	E
NV	9	9	11	11	10
CA	5	5	2	2	3
NP	-	-	C	C	-

Suponha que C e D realizaram mais quatro alterações sobre o dado e em seguida descobriram-se isolados um do outro, levando ao surgimento da configuração

	A	B	C	D	E
NV	9	9	15	15	10
CA	5	5	2	2	3
NP	-	-	C	C	-

Nessa configuração, a partição constituída unicamente pelo nó C ainda é capaz de realizar operações de escrita sobre o dado uma vez que C, de acordo com a ordenação linear imposta aos nós, era o nó privilegiado da partição privilegiada anterior.

Este exemplo mostra que falhas de nós e canais da rede podem ocorrer em tal ordem que o protocolo de votação dinâmica consegue realizar operações que normalmente seriam rejeitadas pelo protocolo de votação tradicional. O contrário também é verdade. Continuando o exemplo anterior, suponha que o nó C falha e os outros quatro nós se reagrupam formando uma única partição. Nesse caso, o protocolo de votação dinâmica deixaria o dado bloqueado enquanto que o protocolo tradicional permitiria que a nova partição, composta pelos nós A, B, D e E, o alterasse. Através de uma análise estocástica, em [JM90] é demonstrado que o protocolo de votação dinâmica, em geral, propicia maior disponibilidade do que o protocolo de votação tradicional. Por fim, quanto à definição de tamanhos diferentes para os quorum de leitura e escrita (seguindo as restrições de intersecção tradicionais) duas soluções são sugeridas em [JM90]: manter uma tabela em cada nó indicando os melhores tamanhos de quorum para cada valor possível da cardinalidade de alteração, ou deixar a escolha do tamanho dos quorums como encargo do nó que iniciar a operação de escrita.

2.3.3 Votação Dinâmica Topológica

Similarmente ao protocolo de votação dinâmica, o protocolo de votação dinâmica topológica [PL88] também forma os quorums a partir somente das cópias correntes do dado replicado. A diferença significativa entre os dois protocolos está na capacidade

deste segundo, em alguns casos, tirar proveito do conhecimento específico sobre a topologia da rede para aumentar ainda mais a disponibilidade do dado.

A cada cópia estão associados, além do número de versão (NV), um número de operação (NO), incrementado de um após a realização de uma operação de escrita ou leitura (em contraste ao número de versão, que é incrementado apenas após operações de escrita); e um conjunto de partição (CP), indicando quais nós participaram da última alteração feita ao dado (em contraste ao atributo cardinalidade de operação do protocolo anterior, que indica apenas a quantidade de nós). Os números de versão e operação são usados pelo nó solicitando acesso ao dado para a identificação das cópias correntes, e o conjunto de partição é usado para verificar se as cópias armazenadas nos nós que responderam à solicitação de acesso constituem uma maioria em relação total de cópias correntes identificadas[1]. O dilema de a partição privilegiada vigente subdividir-se em duas novas partições com exatamente a mesma quantidade de cópias também é resolvido através de uma ordenação linear previamente imposta aos nós. Todos esses atributos são enviados pelos nós como resposta positiva a uma requisição de acesso ao dado.

O aumento na disponibilidade do dado obtido por este protocolo advém do fato de existirem redes de comunicação de alta confiabilidade imunes a situações de particionamento, caso das redes de *broadcast* como a rede *Ethernet*. São comuns sistemas distribuídos constituídos de várias redes desse tipo, aqui chamadas *segmentos* (o termo rede passa a ser usado para nomear o conjunto de todos os segmentos); segmentos são conectados uns aos outros por meio de nós denominados repetidores (ou *gateways*). Em sistemas que adotam essa topologia, nós localizados em um mesmo segmento nunca ficarão isolados uns dos outros, portanto, nunca pertencerão a partições diferentes; na prática, a única maneira de acontecer particionamento é através da falha de nós repetidores. É com o conhecimento de quais cópias estão armazenadas em nós de um mesmo segmento que o protocolo de votação dinâmica topológica consegue, em comparação ao protocolo de votação dinâmica anterior, maior disponibilidade para o dado replicado. O modo como isto acontece é mostrado no exemplo a seguir.

Considere um dado replicado com quatro cópias armazenadas nos nós A, B, C e D (A>B>C>D) de um sistema distribuído cuja topologia é composta por três segmentos ($s1$, $s2$ e $s3$) e dois nós repetidores (X e Y), conforme mostra a figura 2.2. Os nós A e B estão localizados no mesmo segmento $s1$, e os nós C e D estão cada um nos seus próprios segmentos $s2$ e $s3$, respectivamente. X é o nó repetidor conectando os segmentos $s1$ e $s2$, e Y é o nó repetidor que conecta o segmento $s1$ ao segmento $s3$. Os dois nós repetidores são os únicos pontos de particionamento

[1] O protocolo de votação dinamica topológica considera que a cada cópia é atribuído um único voto, e exige um quorum formado pela maioria das cópias correntes para operações tanto de leitura quanto de escrita.

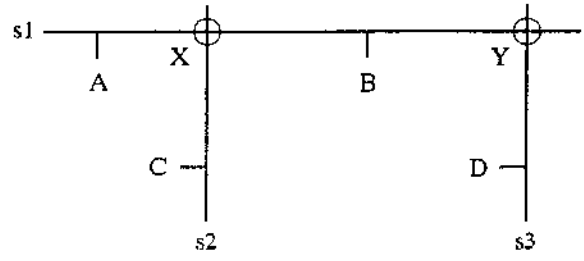

Figura 2.2: Uma rede com três segmentos conectados por dois nós repetidores.

do sistema; decorre disto o fato de as únicas partições possíveis no sistema serem os grupos de nós {{A, B, C}, {D}}, {{A, B, D}, {C}} ou {{A, B}, {C}, {D}}.

Considere agora que os nós repetidores X e Y falharam e que o sistema encontra-se particionado da seguinte maneira: {A, B}, {C} e {D}, com {A, B} sendo a partição privilegiada. Uma possível configuração para o conjunto de cópias do dado seria:

	A	B	C	D
NV	15	15	8	11
NO	15	15	8	11
CP	{A, B}	{A, B}	{A, B, C, D}	{A, B, D}

Suponha que nessa configuração o nó A falha. Pelo protocolo de votação dinâmica anterior o dado já ficaria bloqueado; isto porque o nó B sozinho não pode proclamar-se a partição privilegiada uma vez que ele tem menor prioridade que o nó A, de acordo com o critério de "desempate" utilizado (ordenação linear), e é incapaz de reconhecer se o nó A falhou ou se a comunicação entre os dois foi interrompida. A situação muda se o protocolo conhece a topologia da rede: como o nó B não consegue comunicar-se com o nó A, e é sabido que os dois localizam-se no mesmo segmento, o nó B pode prontamente concluir que o nó A falhou e assim, sem qualquer risco para a consistência das cópias, proclamar-se a nova partição privilegiada.

Em geral, se m cópias do dado replicado estão armazenadas em nós de um mesmo segmento, um nó qualquer que deseja o acesso ao dado deve solicitar permissão de acesso de apenas uma dentre essas m cópias para certificar-se de que as outras $m-1$ cópias não estão envolvidas em alguma outra operação conflitante. Na formação dos quorums, este fato implica na possibilidade de um nó tomar para si votos de outros nós que tenham asseguradamente falhado, considerando-se, é claro, que a topologia da rede (quais nós localizam-se em quais segmentos) é conhecida. As circunstâncias nas quais este "aproveitamento" de votos pode acontecer são formalizadas em se-

guida.

Seja P_k o conjunto de partição de um nó k e R_k o conjunto de nós com os quais o nó k verificou poder se comunicar. O nó k estará apto a tomar para si os votos de todos os nós $j \in P_k$ que estejam no mesmo segmento de um nó $i \in P_k \cap R_k$. (O nó i, portanto, participou da última alteração feita ao dado com a conivência do nó k e está operando normalmente.)

A consistência mútua entre as cópias é garantida pois o protocolo de votação dinâmica topológica não permite que um nó participante da última alteração sofrida pelo dado, e que esteja correntemente fora de operação, tenha seu voto aproveitado por duas ou mais tentativas conflitantes de formação de quorum. Como observação final, vale salientar que nós repetidores armazenando cópias do dado são um problema em potencial devido ao fato de pertencerem simultaneamente a dois segmentos distintos. Uma solução simples para este problema é pressupor que todo nó repetidor está sempre associado a apenas um segmento da rede.

2.4 Análise dos Protocolos

Nesta seção, todos os protocolos revisados neste capítulo são analisados com respeito às melhorias que propõem (e aos possíveis novos custos que impõem) ao protocolo de votação tradicional. Para fins de comparação, mantém-se o mesmo critério de classificação dos protocolos usado na revisão feita ao longo das três seções anteriores.

2.4.1 Reduzindo o Espaço de Armazenamento

A idéia do protocolo de votação com testemunhas de designar algumas cópias do dado replicado como sendo entidades com poder de voto mas sem conter o dado propriamente dito é muito interessante, e o objetivo de diminuir o custo de armazenamento do dado mantendo alta disponibilidade é plenamente alcançado, principalmente quando o número de cópias é pequeno. Isto fica bem claro no exemplo a seguir.

Suponha que pretende-se replicar um dado muito grande (o número de cópias deve ser mínimo, portanto) para ser obtida tolerância à falha de pelo menos um nó. Para isso, o protocolo de votação tradicional exigiria que o dado tivesse pelo menos três cópias (com ambos os quorums de leitura e escrita de tamanho igual a dois), pois com duas cópias o protocolo não toleraria a falha de nenhum nó durante uma operação de escrita. Se em vez das três cópias fossem usadas duas cópias e uma testemunha, o grau de tolerância a falha das operações permaneceria o mesmo e o

espaço de armazenamento equivaleria a apenas duas vezes o tamanho do dado original. É importante ressaltar, no entanto, que a utilização de testemunhas prejudica a disponibilidade se o dado possui muitas cópias e as testemunhas estão em número próximo ao tamanho dos quorums; nesse caso, aumentam-se as chances de estar acessível um grupo de nós contendo apenas testemunhas e cópias desatualizadas, o que impediria a formação de quorums.

O protocolo de votação com fragmentação é mais adequado para as situações em que, devido a restrições locais, o espaço de armazenamento dos nós deve ser mantido reduzido; o motivo desta adequação está no fato de uma cópia inteira do dado não ser armazenada em cada nó, mas somente fragmentos (ou um segmento) dele. A maior desvantagem do protocolo, porém, é que, devido aos *logs* do mecanismo de propagação, quando o dado é freqüentemente alterado o espaço de armazenamento pode crescer e tornar-se até mesmo igual ao necessário no protocolo de votação tradicional. Desse modo, embora inicialmente a fragmentação reduza os custos de armazenamento, os *logs* exigem que os nós tenham espaço suficiente para armazenar uma cópia inteira do dado caso exista a possibilidade da freqüência de ocorrência das alterações aumentar significativamente. Poder-se-ia melhorar o desempenho do protocolo de votação com fragmentação através da criação de alguns "segmentos testemunhas": isto diminuiria ainda mais o espaço total de armazenamento do dado e, devido ao mecanismo de propagação, facilitaria a atualização de segmentos que tivessem perdido alterações recentes.

2.4.2 Capturando a Topologia Corrente da Rede

A maior vantagem do protocolo de votação com partições virtuais é conseguir realizar um operação de leitura com o acesso de uma única cópia e ainda tolerar o particionamento do sistema (já que uma operação de escrita continua sendo realizada sobre a maioria das cópias). O protocolo de votação com partições virtuais generalizada dá mais flexibilidade ao protocolo original, permitindo que os quorums de leitura e escrita de cada partição virtual possam ser modificados de acordo com a freqüência com o dado é lido e alterado, e elimina a redundância do processo de validação em duas fases na criação de uma nova partição virtual. Uma observação interessante é que pode-se considerar esses dois protocolos também protocolos dinâmicos: como os quorums são definidos em relação às cópias de uma partição virtual, e o número de cópias pertencentes a uma partição virtual pode variar, o tamanho real dos quorums claramente se altera ante mudanças na configuração do sistema. Uma limitação em relação aos protocolos dinâmicos revisados na seção 2.3, porém, é que, como uma partição virtual, para manter a consistência mútua, deve conter pelo menos a maioria das cópias do dado, e um quorum de escrita de cada partição virtual deve ser formado por pelo menos a maioria das cópias dessa partição, o dado não conseguirá ser

alterado caso não haja pelo menos 1/4 (a maioria da maioria) dos nós funcionando corretamente.

Quanto aos custos inerentes à manutenção das partições virtuais, constata-se facilmente que a tarefa de atualizá-las, seja através de protocolos especiais ou de transações específicas, é consideravelmente onerosa. No primeiro caso, há a redundância do procedimento de validação; no segundo, as transações específicas podem prejudicar (e até cancelar, se elas chegarem a causar *deadlocks*, por exemplo) a execução das outras transações em andamento. Portanto, a eficiência desses dois protocolos depende fortemente da relação entre a freqüência com que as falhas acontecem e a freqüência com que o dado é alterado: se as falhas são raras em relação às alterações, os procedimentos de reconfiguração (protocolos especiais ou transações específicas) serão pouco acionados e o desempenho do sistema não será (ou será muito pouco) afetado; por outro lado, se as falhas são freqüentes quando comparadas às alterações, partições virtuais terão de ser criadas sistematicamente e a penalização resultante no desempenho do sistema poderá ser comprometedora.

2.4.3 Aumentando a Disponibilidade do Dado

O grande mérito dos protocolos dinâmicos é a capacidade de alterar o tamanho dos quorums diante da detecção de mudanças na configuração do sistema. No protocolo de votação com reatribuição dinâmica de votos, por exemplo, o tamanho dos quorums é diminuído aumentando-se o número de votos das cópias da partição privilegiada, e é aumentado reestabelecendo-se o poder de voto original quando as falhas são reparadas; nos protocolos de votação dinâmica, a alteração do tamanho dos quorums dá-se pela retirada do poder de votos das cópias que ficam de fora da partição privilegiada, e pela reposição desse poder de voto para as cópias que voltam a atualizar-se.

A idéia de alterar dinamicamente o número de votos atribuído inicialmente aos nós foi proposta originalmente visando sistemas que utilizam o mecanismo de votação para obter exclusão mútua; em conseqüência, os protocolos de reatribuição de votos, no caso específico do controle de dados replicados, representam custos adicionais de comunicação uma vez que ainda deve existir um protocolo de controle de concorrência para manter a consistência interna das cópias. Em contraste, os protocolos de votação dinâmica fazem a "reatribuição dos votos" implicitamente a cada realização de uma operação de escrita, para isso embutindo (*piggybacking*) informações adicionais dentro das próprias mensagens do protocolo de controle de concorrência; a desvantagem é que mudanças na configuração do sistema ocorridas entre duas operações de escritas consecutivas (o que é possível caso as falhas sejam muito freqüentes em relação à freqüência com que o dado é alterado) não são interceptadas, o que pode prejudicar a disponibilidade do dado. Para esse problema ser contornado, em [JM90] é sugerida

a realização periódica de alterações "nulas" a fim de melhor acompanhar o ritmo com que as mudanças acontecem; esta solução, entretanto, implica em mais custos de comunicação e interfere na execução de outras transações "normais", podendo inclusive aumentar as chances de surgirem *deadlocks*.

É válido ressaltar, por fim, que as idéias de reduzir os custos de armazenamento (votação com testemunhas e votação com fragmentação) podem ser naturalmente aplicadas aos protocolos dinâmicos. Na verdade, pode-se considerar a redução do espaço de armazenamento das cópias uma característica ortogonal à manutenção de visões convergentes sobre a topologia corrente do sistema e ao aumento da disponibilidade do dado conseguido através da alteração dinâmica do tamanho dos quorums.

Notas Bibliográficas

O protocolo de votação com testemunhas foi estendido em [Pâr90] com a definição de um novo tipo de testemunha que pode ser armazenada em memória volátil. Uma generalização do protocolo de votação com fragmentação, onde números diferentes de fragmentos do dado podem ser distribuídos pelos nós, é apresentada como uma das vantagens do mecanismo de votação proposto em [AAC91] (este mecanismo é revisado no capítulo 5). O protocolo proposto em [ES83] pode, de certa forma, ser considerado dinâmico pois, devido aos seus dois modos de funcionamento (como o protocolo ROWA quando não há falhas, e como o protocolo de votação tradicional quando falhas são detectadas), o tamanho dos quorums varia em resposta a mudanças na configuração do sistema. Versões anteriores do protocolo de votação dinâmica estão presentes em [JM87a] e [JM87b].

Capítulo 3

Protocolos Baseados em Estruturas Lógicas

O custo de realização de uma operação sobre um dado replicado é diretamente proporcional ao tamanho do quorum exigido pelo protocolo de controle. Quanto maior o número de cópias a ser contactado para que a permissão de acesso ao dado seja obtida, maior será o número de mensagens transmitidas pela rede e, conseqüentemente, maiores serão a sobrecarga e o tempo de resposta do sistema.

Nos protocolos baseados em votação, revistos no capítulo anterior, o tamanho dos quorums cresce linearmente com o número total de cópias do dado — um quorum de escrita no protocolo de votação simples, por exemplo, necessita de pelo menos $N/2+1$ cópias de um total de N cópias do dado replicado —, o que implica em (indesejáveis) elevados custos de comunicação quando o número total de cópias é consideravelmente grande[1]. Teoricamente, poder-se-ia reduzir o tamanho dos quorums nos protocolos baseados em votação através da atribuição de uma grande parte dos votos a apenas algumas das cópias; esta solução não é aconselhável, porém, pois ela centraliza o controle do dado replicado sobre um pequeno conjunto de cópias, propiciando menor tolerância a falhas e limitando a distribuição da carga pelo sistema — o que vai de encontro aos objetivos principais da replicação dos dados.

Este capítulo revisa vários protocolos que mantêm a consistência entre as cópias exigindo quorums menores que os necessários nos protocolos baseados em votação. A idéia básica desses protocolos, aqui denominados protocolos baseados em estruturas lógicas, é organizar logicamente as cópias do dado replicado em estruturas como grades, árvores ou hierarquias, e formar os quorums a partir da disposição das cópias

[1] Embora não seja razoável imaginar atualmente um dado replicado com mais do que, especula-se, 10 cópias, esse número certamente crescerá bastante no futuro em vista da crescente evolução e proliferação dos sistemas distribuídos.

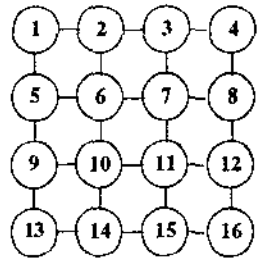

Figura 3.1: 16 cópias organizadas em uma grade de dimensões 1×4.

dentro dessas estruturas; os quorums obtidos dessa maneira normalmente não podem ser formados pelo protocolo de votação tradicional sob nenhuma atribuição de votos e são, em geral, de tamanho sublinear, resultando em operações de acesso ao dado de custos significativamente reduzidos, principalmente para um número grande de cópias.

Cada protocolo é descrito com respeito à formação e ao tamanho dos quorums, e à disponibilidade do dado; essa última sendo representada pelo cálculo da probabilidade de que um quorum da operação requisitada consiga ser formado a partir da estrutura lógica em questão. Para cada protocolo é também apresentada uma prova de correção (baseada na prova de correção do artigo original).

A descrição dos protocolos está assim distribuída no restante do capítulo: a próxima seção descreve um protocolo que organiza logicamente as cópias do dado em uma estrutura de grade; a seção 3.2 descreve um protocolo cuja estrutura lógica é uma hierarquia na qual as cópias localizam-se no nível mais baixo; uma estrutura de grade hierárquica é utilizada pelo protocolo descrito na seção 3.3; a seção 3.4 descreve um protocolo que utiliza uma estrutura de árvore; finalizando o capítulo, a seção 3.5 apresenta uma análise dos protocolos revisados.

3.1 Protocolo de Quorum em Grade

O protocolo de quorum em grade [CAA90] organiza logicamente as N cópias do dado replicado em uma grade (ou matriz) de m linhas e n colunas ($N = m\ n$). O conjunto $\{C_1, C_2, \ldots, C_N\}$ das N cópias do dado replicado é representado logicamente em uma estrutura de grade através do conjunto $\{G(i,j) \mid i = 1, \ldots, m \text{ e } j = 1, \ldots, n\}$. A correspondência entre uma posição da grade $G(i,j)(1 \leq i \leq m \text{ e } 1 \leq j \leq n)$ e a cópia C_k equivalente a essa posição dá-se pela expressão $k = (i-1) * n + j$. A figura 3.1 mostra um exemplo de 16 cópias organizadas em uma grade de dimensões 1×1.

3.1.1 Formação e Tamanho dos Quorums

Um quorum de leitura no protocolo de quorum em grade é formado por pelo menos uma cópia de cada uma das colunas, sendo portanto de tamanho n, e desse modo é representado por qualquer conjunto de cópias da grade da forma $\{G(i,j) \mid i \in \{1,\ldots,m\}$ e $j = 1,\ldots,n\}$ (ou seja, i, que representa a linha, pode assumir qualquer valor entre 1 e m, e j, que representa a coluna, deve assumir todos os valores de 1 a n). Exemplos de quorums de leitura na grade mostrada na figura 3.1 são {1, 6, 3, 12}, {9, 10, 15, 4} e {5, 2, 7, 16}.

Um quorum de escrita é formado por um quorum de leitura mais todas as cópias de qualquer uma das colunas, sendo representado por todos os conjuntos de cópias da forma $\{G(i,j) \mid i \in \{1,\ldots,m\}$ e $j = 1,\ldots,n\} \cup \{G(i,j) \mid i = 1,\ldots,m$ e $j \in \{1,\ldots,n\}\}$. Como toda linha da grade tem uma cópia em comum com todas as colunas, o tamanho do quorum de escrita é $m + n - 1$. Exemplos de quorums de escrita na grade mostrada na figura 3.1 são {1, 5, 9, 13, 6, 3, 12}, {9, 2, 6, 10, 14, 15, 4} e {5, 2, 7, 4, 8, 12, 16}.

Quando $m \simeq n$, os tamanhos dos quorums de leitura e escrita são $\sqrt{N}$ e $2\sqrt{N}-1$, respectivamente, ambos da ordem de $O(\sqrt{N})$.

3.1.2 Prova de Correção

De acordo com o critério de *one-copy serializability*, deve ser provado que os quorums de operações conflitantes gerados pelo protocolo de quorum em grade têm pelo menos uma cópia em comum. Isto é feito provando-se os dois lemas a seguir.

Lema 3.1 *O protocolo de quorum em grade garante que qualquer quorum de leitura tem pelo menos uma cópia em comum com qualquer quorum de escrita.*

Prova. Como um quorum de leitura é formado por pelo menos uma cópia de cada uma das colunas, e um quorum de escrita contém todas as cópias de alguma coluna, é garantido que qualquer quorum de leitura tem pelo menos uma cópia em comum com qualquer quorum de escrita. □

Lema 3.2 *O protocolo de quorum em grade garante que quaisquer dois quorums de escrita têm pelo menos uma cópia em comum.*

Prova. Como todo quorum de escrita contém um quorum de leitura, a prova deste lema decorre diretamente da prova do lema 3.1. □

3.1.3 Disponibilidade do Dado

Considerando-se que cada nó do sistema está independentemente acessível com probabilidade p, a disponibilidade do dado replicado para operações de leitura (A_r) em uma grade de dimensões $m \times n$ é dada por:

$$A_r(p) = [1 - (1-p)^m]^n,$$

onde $(1-p)^m$ é a probabilidade de todas as cópias de uma coluna estarem inacessíveis, e $1-(1-p)^m$ é a probabilidade de pelo menos uma cópia de uma coluna estar acessível.

A disponibilidade do dado para operações de escrita (A_w) no protocolo de quorum em grade é calculada pela diferença entre a probabilidade de pelo menos uma cópia estar acessível em todas as colunas (disponibilidade para leitura) e a probabilidade de pelo menos uma cópia, mas não todas, estar acessível também em todas as colunas (que é o caso em que consegue-se um quorum de leitura mas não consegue-se contactar todas as cópias de alguma coluna). Essa disponibilidade é dada por

$$A_w = [1 - (1-p)^m]^n - [1 - p^m - (1-p)^m]^n,$$

onde $1 - p^m$ é a probabilidade de todas as cópias de uma coluna não estarem simultaneamente acessíveis.

3.2 Protocolo de Votação Hierárquica

Este protocolo [Kum91] forma os quorums de acesso ao dado replicado utilizando uma hierarquia lógica de vértices de m níveis. Os vértices folhas do nível mais baixo da hierarquia correspondem às cópias do dado replicado; os vértices dos níveis mais altos representam grupos lógicos de vértices do nível imediatamente inferior. Em geral, um vértice de um nível i ($1 \leq i \leq m$) da hierarquia representa um grupo lógico de l_i vértices do nível $i-1$. Por exemplo, o vértice raiz do nível m representa um grupo lógico de l_m vértices do nível $m-1$; cada vértice do nível $m-1$ filho do vértice raiz, por sua vez, representa um grupo lógico de l_{m-1} vértices do nível $m-2$, e assim sucessivamente, até os vértices folhas do nível 0 (ver figura 3.2). O número total de cópias nessa estrutura é $\prod_{i=1}^{m} l_i$.

A cada cópia do dado replicado é associado um identificador da forma $o_{i_1 i_2 \cdots i_m}$ ($1 \leq i_j \leq l_j$ e $1 \leq j \leq m$); este identificador indica, dentro da hierarquia de vértices, a rota do vértice raiz da hierarquia à cópia. A figura 3.3 mostra um exemplo de uma hierarquia de 2 níveis ($l_1 = l_2 = 3$) com 9 cópias no nível 0; nessa hierarquia, as cópias são identificadas como o_{11}, o_{12}, o_{13}, o_{21}, o_{22}, o_{23}, o_{31}, o_{32} e o_{33}. De modo similar, em uma hierarquia de 3 níveis com 27 cópias no nível 0, as cópias seriam identificadas como $o_{ijk}(1 \leq i, j, k \leq 3)$.

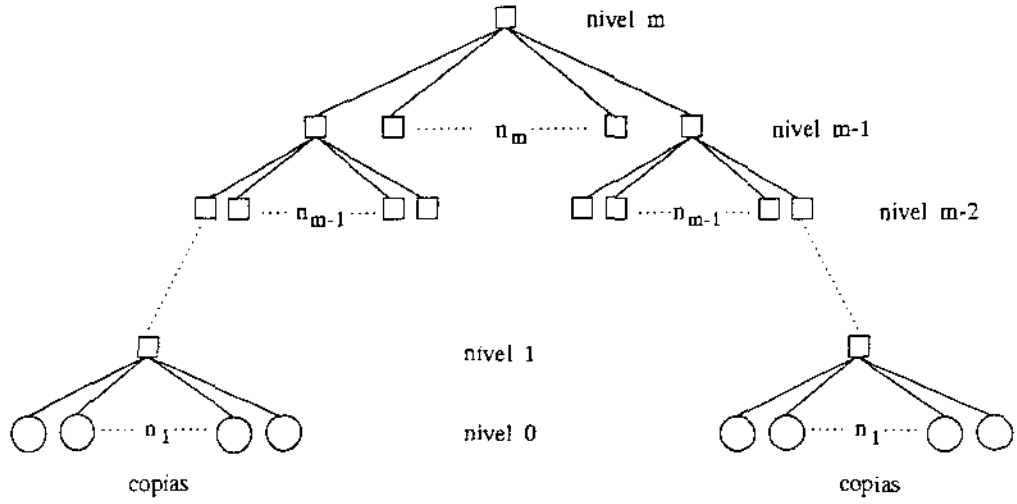

Figura 3.2: Uma hierarquia lógica de vértices de m níveis.

3.2.1 Formação e Tamanho dos Quorums

Um quorum de leitura é formado no protocolo de votação hierárquica se for obtida a devida permissão do vértice raiz da hierarquia no nível m. O vértice raiz dará permissão de acesso para leitura se ele receber a mesma permissão de pelo menos r_m vértices dentre seus vértices filhos do nível $m-1$; cada um desses vértices filhos do nível $m-1$, por sua vez, dará permissão de acesso para leitura se também for obtida a mesma permissão de pelo menos r_{m-1} vértices dentre seus vértices filhos do nível $m-2$, e assim sucessivamente, até que as permissões sejam pedidas diretamente as cópias do dado (vértices folhas do nível 0). Dessa maneira, as permissões de acesso para leitura dadas pelas cópias são propagadas de baixo para cima na hierarquia até atingirem o vértice raiz no nível mais alto, resultando em um quorum de leitura formado por $\prod_{i=1}^{m} r_i$ cópias.

Um quorum de escrita é formado similarmente, agora definindo-se para cada nível i $(1 \leq i \leq l)$ o número mínimo w_i de vértices filhos do nível $i-1$ dos quais um vértice do nível i deve receber permissão de acesso para escrita para que ele próprio possa dar essa mesma permissão. Um quorum de escrita possui, portanto, tamanho $\prod_{i=1}^{m} w_i$.

A figura 3.4 mostra o algoritmo para construir, no protocolo de votação hierárquica, quorums que exijam para cada nível i $(1 \leq i \leq m)$ da hierarquia a obtenção de permissão de acesso de pelo menos q_i vértices do nível $i-1$. O algoritmo serve para formar tanto quorums de leitura quanto quorums de escrita, bastando-se substituir os valores q_i pelos valores r_i ou w_i, de acordo com a operação requisitada. Um quorum será formado com sucesso se o algoritmo retornar 1 após uma chamada

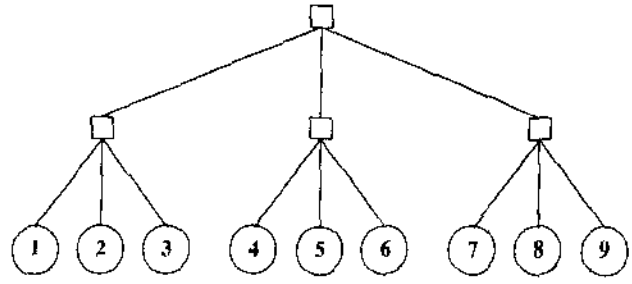

Figura 3.3: 9 cópias organizadas em uma hierarquia de 2 níveis.

onde o valor m (nível mais alto da hierarquia) é passado como argumento; caso o retorno seja 0, permissões de acesso não foram obtidas de um número mínimo de cópias e a operação é cancelada.

Para que os quorums de operações conflitantes se interceptem, os valores r_i e w_i para cada nível i ($1 \leq i \leq m$) da hierarquia devem ser definidos satisfazendo duas restrições idênticas às restrições presentes no protocolo de votação tradicional, que são:

$$r_i + w_i > l_i \tag{3.1}$$

$$2w_i > l_i \tag{3.2}$$

A restrição 3.1 garante a intersecção entre qualquer quorum de leitura e qualquer quorum de escrita, e a restrição 3.2 garante a intersecção entre quaisquer dois quorums de escrita (ver prova de correção na seção seguinte).

Fazendo-se $r_1 = 1$, $r_2 = 2$, $w_1 = 3$ e $w_2 = 2$ na hierarquia de 2 níveis mostrada na figura 3.3, exemplos de quorums de leitura são $\{1, 4\}$, $\{6, 7\}$ e $\{2, 8\}$, e exemplos de quorums de escrita são $\{1, 2, 3, 4, 5, 6\}$, $\{1, 2, 3, 7, 8, 9\}$ e $\{4, 5, 6, 7, 8, 9\}$. Da mesma forma, fazendo-se $r_1 = r_2 = w_1 = w_2 = 2$, exemplos tanto de quorums de leitura quanto de quorums de escrita são $\{1, 2, 4, 5\}$, $\{2, 3, 7, 8\}$ e $\{5, 6, 7, 9\}$.

Em [Kum91] é mostrado que quanto maior o número de níveis da hierarquia menor será o tamanho resultante dos quorums. Para que o número de níveis da hierarquia seja máximo, claramente o número de filhos por vértice deve ser mínimo. Como o menor número possível de vértices que pode satisfazer as restrições 3.1 e 3.2 ainda tolerando a falha (ou não recebimento da permissão de acesso) de algum deles é igual a 3, constata-se que a estrutura ideal é uma hierarquia onde $l_i = 3$ e $r_i = w_i = 2$ para todo nível i ($1 \leq i \leq m$) da hierarquia — a hierarquia, nesse caso, é uma árvore ternária. Nessa estrutura, $m = \log_3 N$ e os quorums de leitura e escrita

```
FormQuorum(i) {
   if (i == 0)
      if (copy o[i[1], i[2], ..., i[m]] grants permission)
         return(1);
      else
         return(0);
   else{
      num_examined = 0;
      num_permissions = 0;
      while (num_permissions< q_i) && (num_examined< l_i){
         num_permissions += FormQuorum(i - 1);
         num_examined++;
      }
      if (num_permissions==q_i)
         return(1);
      else
         return(0);
   }
}
```

Figura 3.4: Algoritmo para construir quorums no protocolo de votação hierárquica.

são ambos de tamanho 2^m; como $2^{\log_3 N} = N^{\log_3 2}$, os tamanhos de ambos os quorums são da ordem de $O(N^{0.63})$.

3.2.2 Prova de Correção

Deve ser provado aqui que quorums de operações conflitantes (formados a partir da permissão de acesso dada pelo vértice raiz da hierarquia) têm pelo menos uma cópia do dado (um vértice folha) em comum. Isto é feito através da prova dos dois lemas seguintes.

Lema 3.3 *Em uma hierarquia de m níveis, o protocolo de votação hierárquica garante que qualquer quorum de leitura tem pelo menos uma cópia em comum com qualquer quorum de escrita.*

Prova. Por indução nos níveis da hierarquia.

Base. Há pelo menos um vértice do nível $m-1$ em comum entre qualquer quorum de leitura e qualquer quorum de escrita formados no nível m. Esta asserção é facilmente verificada uma vez que os quorums de leitura e escrita são formados no nível m por r_m e w_m vértices do nível $m-1$, respectivamente, e, por 3.1, $r_m + w_m > l_m$.

Hipótese. Há pelo menos um vértice do nível $i-1$ em comum entre qualquer quorum de leitura e qualquer quorum de escrita formados a partir dos vértices filhos de um vértice do nível i ($1 \leq i \leq m-1$).

Passo. Deve ser mostrado que, se há um vértice do nível $i-1$ (dentre os vértices filhos de um vértice do nível i) em comum entre um quorum de leitura e um quorum de escrita, há também um vértice do nível $i-2$ comum aos dois quorums. Este passo garantirá que uma intersecção entre um quorum de leitura e um quorum de escrita no nível m implica, pela hipótese de indução, numa intersecção entre os dois quorums no nível 0, onde os vértices correspondem às cópias do dado.

Se há um vértice filho de um vértice do nível i em comum entre um quorum de leitura e um quorum de escrita, esse vértice é obviamente um vértice do nível $i-1$ que corresponde a um grupo lógico de l_{i-1} vértices do nível $i-2$. Nesse grupo também foram formados um quorum de leitura e um quorum de escrita; como, por 3.1, $r_{i-1} + w_{i-1} > l_{i-1}$, é garantido que há pelo menos um vértice do nível $i-2$ comum aos dois quorums.

Portanto, por indução, o lema é válido. □

Lema 3.4 *Em uma hierarquia de m níveis, o protocolo de votação hierárquica garante que quaisquer dois quorums de escrita têm pelo menos uma cópia em comum.*

Prova. Análoga à prova do lema 3.3. □

3.2.3 Disponibilidade do Dado

Devido ao fato de o protocolo de votação hierárquica, em relação a cada nível da hierarquia, ter um funcionamento semelhante ao protocolo de votação tradicional quando as cópias do dado possuem um voto cada uma, o cálculo da disponibilidade do dado no primeiro protocolo é realizado generalizando-se o cálculo da disponibilidade do dado no protocolo de votação tradicional para todos os níveis da hierarquia. Assim, primeiro é apresentado o cálculo da disponibilidade do dado no protocolo de votação tradicional; em seguida, e baseado nessa disponibilidade, apresenta-se o cálculo da disponibilidade do dado no protocolo de votação hierárquica.

No protocolo de votação tradicional, a probabilidade, denotada por $AV(N, q, \rho)$, de um quorum de q cópias, dentre um total de N cópias, ser formado quando cada cópia possui um único voto e está acessível com probabilidade ρ é dada pela seguinte expressão [AA89b]:

$$AV(N,q,p) = \sum_{j=q}^{N} \binom{N}{j} p^j (1-p)^{N-j}.$$

No protocolo de votação hierárquica, a probabilidade de um vértice do nível i ($1 \leq i \leq m$) da hierarquia dar permissão de acesso para uma operação de leitura é calculada de modo similar, substituindo-se, na expressão anterior, p pela probabilidade de um vértice filho do nível $i-1$ também dar essa mesma permissão, q por r_i, e N por l_i. Como os vértices folhas correspondem às cópias do dado replicado, considera-se que a probabilidade de eles darem permissão de acesso para leitura é p. Desse modo, a disponibilidade do dado para uma operação de leitura (A_r) no protocolo de votação hierárquica, ou seja, a probabilidade de o vértice raiz da hierarquia (no nível m) dar permissão de acesso para leitura, é calculada pela seguinte recorrência:

$$A_r(i) = \begin{cases} p & (i = 0) \\ AV(l_i, r_i, A_r(i-1)) & (i \geq 1) \end{cases}$$

Analogamente, a disponibilidade do dado para uma operação de escrita (A_w) é dada por:

$$A_w(i) = \begin{cases} p & (i = 0) \\ AV(l_i, w_i, A_w(i-1)) & (i \geq 1) \end{cases}$$

3.3 Protocolo de Quorum em Grade Hierárquica

Uma desvantagem do protocolo de quorum em grade é sua baixa disponibilidade para operações de escrita quando o número total de cópias do dado replicado (N) é grande: a probabilidade de se obter permissão de acesso de todas as cópias de alguma coluna da grade diminui drasticamente com o crescimento do número de cópias de cada coluna (decorrente do crescimento de N). Para solucionar este problema, o protocolo de quorum em grade hierárquica [KC91] propõe a organização das cópias em uma estrutura lógica que combina a idéia da estrutura de grade do protocolo de quorum em grade com a estrutura hierárquica do protocolo de votação hierárquica, e com isso melhora a disponibilidade para as operações de escrita ainda mantendo os tamanhos dos quorums da ordem de $O(\sqrt{N})$. A maneira como as cópias do dado

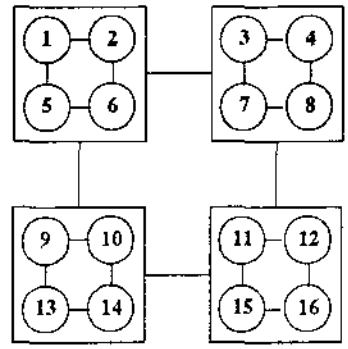

Figura 3.5: 16 cópias organizadas em uma grade hierárquica de 2 níveis.

são organizadas nesta estrutura, chamada grade hierárquica (ou h-grade), é descrita em seguida.

Similarmente ao protocolo de votação hierárquica, o protocolo de quorum em grade hierárquica organiza as cópias em uma hierarquia de l níveis na qual as cópias são os objetos do nível mais baixo (nível 0), e os objetos dos níveis mais altos são definidos logicamente a partir de objetos do nível imediatamente inferior. Um objeto lógico do nível 1 é definido como uma grade de dimensões $m_1 \times n_1$ constituída de $m_1.n_1$ cópias do dado replicado. Do mesmo modo, um objeto lógico do nível 2 é definido como uma grade de dimensões $m_2 \times n_2$ constituída de $m_2.n_2$ objetos do nível 1. Em geral, um objeto lógico do nível i ($1 \leq i \leq l$) representa uma grade de dimensões $m_i \times n_i$ constituída de $m_i.n_i$ objetos do nível $i-1$. O número total de cópias, portanto, é dado por $\prod_{i=1}^{l} m_i.n_i$. A figura 3.5 mostra um exemplo de um dado replicado com 16 cópias organizadas em uma estrutura de h-grade onde $m_1 = n_1 = m_2 = n_2 = 2$.

Uma h-grade na verdade pode ser vista como uma única grade global (ou g-grade) de dimensões $\prod_{i=1}^{l} m_i \times \prod_{i=1}^{l} n_i$ constituída apenas de cópias do dado. A correspondência entre uma posição (x, y) ($1 \leq x \leq \prod_{i=1}^{l} m_i$ e $1 \leq y \leq \prod_{i=1}^{l} n_i$) de uma cópia na g-grade e a posição k dessa mesma cópia no conjunto $\{C_1, C_2, \ldots, C_N\}$ das N cópias do dado replicado é facilmente feita através da expressão $k = (x - 1) \prod_{i=1}^{l} n_i + y$. Para simplificar a notação, na h-grade um objeto do nível i ($1 \leq i \leq l$) pertencente a uma grade de dimensões $m_i \times n_i$ é definido univocamente apenas em relação ao nível imediatamente superior $(i+1)$, sendo denotado por $b^{i-1}(x_i, y_i)$ ($1 \leq x_i \leq m_i$ e $1 \leq y_i \leq n_i$). Note que as cópias, no nível 0, são denotadas por $b^0(x_1, y_1)$. A correspondência entre a posição (x_i, y_i) ($1 \leq i \leq l$) de uma cópia na h-grade e a posição (x, y) dessa mesma cópia na g-grade é dada por $x = x_1 + \prod_{i=2}^{l}(x_i - 1)\, m_{i-1}$ e $y = y_1 + \prod_{i=2}^{l}(y_i - 1)\, n_{i-1}$.

Além das duas operações tradicionais de leitura (r) e escrita (w), o protocolo de quorum em grade hierárquica define uma terceira operação de acesso a dados replicados, chamada *escrita-cega* (b). Uma operação de escrita-cega é uma operação conflitante apenas com respeito a operações de leitura e de escrita, e não em relação a outras operações de escrita-cega[2]. Essa distinção entre uma operação de escrita-cega e uma operação de escrita tradicional é feita porque o quorum exigido para a primeira é menor que o quorum exigido para a segunda.

3.3.1 Formação e Tamanho dos Quorums

Um quorum de leitura é formado no protocolo de quorum em grade hierárquica obtendo-se no nível l da h-grade permissões de acesso para leitura de pelo menos um objeto do nível $l-1$ de cada uma das colunas da grade de dimensões $m_l \times n_l$ desse nível. Obter permissão de acesso para leitura de um objeto do nível $l-1$ significa conseguir formar um quorum de leitura na grade de dimensões $m_{l-1} \times n_{l-1}$ correspondente a esse objeto. Em geral, obter permissão de acesso para leitura de um objeto do nível i ($1 \leq i \leq l$) da h-grade consiste em obter essa mesma permissão de pelo menos um objeto do nível $i-1$ de cada uma das colunas da grade de dimensões $m_i \times n_i$ correspondente ao objeto do nível i. No nível 0, as permissões de acesso são pedidas diretamente às cópias. Um quorum de leitura formado na única grade do nível l ($b^l(1,1)$) é um quorum de leitura do protocolo de quorum em grade hierárquica, portanto é qualquer conjunto de cópias da h-grade definido através da recorrência a seguir, para a qual a grade $b^l(1,1)$ é passada como argumento.

$$q_r(b^i(x,y)) = \begin{cases} \{b^0(x_1,y_1) \mid x_1 \in \{1,\ldots,m_1\} \text{e} y_1 = \{1,\ldots,n_1\}\} & (i=1) \\ \{q_r(b^{i-1}(x_i,y_i)) \mid x_i \in \{1,\ldots,m_i\} \text{e} y_i = \{1,\ldots,n_i\}\} & (i>1) \end{cases}$$

Na h-grade mostrada na figura 3.5, são exemplos de quorums de leitura os conjuntos de cópias {1, 6, 7, 8}, {1, 2, 11, 12} e {9, 14, 15, 16}.

Um quorum de escrita-cega também é formado recursivamente, nível a nível, mas agora obtendo-se permissão de acesso para escrita-cega de todos os objetos do nível $l-1$ de alguma coluna da grade de dimensões $m_l \times n_l$ do nível l. A recorrência que define os quorums de escrita-cega é a que segue.

$$q_b(b^i(x,y)) = \begin{cases} \{b^0(x_1,y_1) \mid x_1 = \{1,\ldots,m_1\} \text{e} y_1 \in \{1,\ldots,n_1\}\} & (i=1) \\ \{q_b(b^{i-1}(x_i,y_i)) \mid x_i = \{1,\ldots,m_i\} \text{e} y_i \in \{1,\ldots,n_i\}\} & (i>1) \end{cases}$$

[2]Duas operações de escrita-cega, portanto, podem ser realizadas independentemente uma da outra.

Na h-grade mostrada na figura 3.5, são exemplos de quorums de escrita-cega os conjuntos de cópias {1, 5, 10, 14}, {3, 7, 11, 15} e {2, 6, 9, 13}.

Por fim, um quorum de escrita é formado unindo-se as cópias de um quorum de leitura com as cópias de um quorum de escrita-cega. O conjunto de cópias {1, 5, 6, 7, 8, 10, 14} é um exemplo de um quorum de escrita na h-grade mostrada na figura 3.5.

Mostra-se agora que os quorums exigidos pelo protocolo de quorum em grade hierárquica possuem tamanhos equivalentes aos tamanhos dos quorums exigidos pelo protocolo de quorum em grade original.

Lema 3.5 *Em uma h-grade de l níveis de dimensões $m_i \times n_i$ $(1 \leq i \leq l)$, os tamanhos dos quorums de leitura, escrita-cega e escrita são da ordem de $O(\sqrt{N})$.*

Prova. Suponha que $m_i = n_i = d_i$ para cada nível i $(1 \leq i \leq l)$ da h-grade. Nesse caso, como há N cópias do dado replicado no total, $\prod_{i=1}^{l}(d_i)^2 = N$. Como um quorum de leitura em um nível i é formado por d_i objetos do nível $i-1$, no geral, um quorum de leitura tem tamanho $\prod_{i=1}^{l} d_i$ (ou $\sqrt{N}$). Da mesma maneira, um quorum de escrita-cega também tem tamanho $\sqrt{N}$; sendo um quorum de escrita formado pela união de um quorum de leitura com um quorum de escrita-cega, seu tamanho é $2\sqrt{N} - 1$, uma vez que há uma intersecção não nula entre qualquer quorum de leitura e qualquer quorum de escrita-cega (ver prova de correção a seguir). Portanto, quando $m_i \simeq n_i$ para todo nível i $(1 \leq i \leq l)$ da h-grade, os quorums exigidos pelo protocolo de quorum em grade hierárquica possuem tamanhos da ordem de $O(\sqrt{N})$. □

3.3.2 Prova de Correção

Deve ser provado que a intersecção entre quorums de operações conflitantes formados na grade do nível l implica também em intersecção no nível 0, onde estão as cópias. Faz-se isto através da prova dos dois lemas seguintes.

Lema 3.6 *Em uma h-grade de l níveis, o protocolo de quorum em grade hierárquica garante que qualquer quorum de leitura tem pelo menos uma cópia em comum com qualquer quorum de escrita-cega.*

Prova. Por indução nos níveis da h-grade.

Base. Há pelo menos um objeto do nível $l-1$ em comum entre qualquer quorum de leitura e qualquer quorum de escrita-cega formados na grade do nível l. Esta asserção é facilmente verificada uma vez que, na grade do nível l, um quorum de

leitura é formado por pelo menos um objeto do nível $l-1$ de cada uma das colunas, e um quorum de escrita-cega contém todas os objetos do nível $l-1$ de pelo menos uma das colunas.

Hipótese. Há pelo menos um objeto do nível $i-1$ em comum entre qualquer quorum de leitura e qualquer quorum de escrita-cega formados em uma grade do nível i ($1 \leq i \leq l-1$).

Passo. Se há um objeto do nível $i-1$ comum aos dois quorums, esse objeto é obviamente uma grade do nível $i-1$ composta de $m_{i-1}.n_{i-1}$ objetos do nível $i-2$ onde também foram formados um quorum de leitura e um quorum de escrita-cega. Por hipótese de indução, esses dois quorums formados na grade do nível $i-1$ também se interceptam, implicando no fato de também haver pelo menos um objeto do nível $i-2$ comum aos quorums de leitura e escrita-cega formados na grade do nível i. Continuando-se a indução, constata-se que os dois quorums também se interceptam no nível 0 e, portanto, têm pelo menos uma cópia do dado em comum.

Logo, por indução, o lema é válido. □

Lema 3.7 *Em uma h-grade de l níveis, o protocolo de quorum em grade hierárquica garante que qualquer quorum de escrita tem pelo menos uma cópia em comum com qualquer quorum de leitura, escrita-cega ou mesmo outro quorum de escrita.*

Prova. Como um quorum de escrita contém um quorum de leitura e um quorum de escrita-cega, a prova deste lema é decorrente da prova do lema 3.6. □

3.3.3 Disponibilidade do Dado

A disponibilidade do dado no protocolo de quorum em grade hierárquica para as operações de leitura (A_r) e escrita-cega (A_b) são calculadas nível a nível, de modo semelhante ao cálculo da disponibilidade do dado no protocolo de quorum em grade original, e expressas pelas duas recorrências a seguir.

$$A_r(i) = \begin{cases} p & (i = 0) \\ [1 - [1 - A_r(i-1)]^{m_i}]^{n_i} & (i \geq 1) \end{cases}$$

$$A_b(i) = \begin{cases} p & (i = 0) \\ 1 - [1 - [A_b(i-1)]^{m_i}]^{n_i} & (i \geq 1) \end{cases}$$

Para o cálculo da disponibilidade para operações de escrita primeiro definem-se duas probabilidades de interesse: a probabilidade de se obter acesso para leitura de alguma coluna de uma grade de dimensões $m_i \times n_i$ do nível i, dada por

$$P_r^{m_i,1}(i) = [A_r(i-1)]^{m_i},$$

e a probabilidade de se obter acesso para escrita de alguma coluna de uma grade de dimensões $m_i \times n_i$ do nível i, dada por

$$P_w^{m_i,1}(i) = [A_h(i-1)]^{m_i} - [A_h(i-1) - A_w(i-1)]^{m_i},$$

que é a diferença entre a probabilidade de se formar um quorum de escrita-cega em todos os objetos da coluna e a probabilidade de se formar apenas quorums de escrita-cega (e não quorums de escrita) também em todos os objetos da coluna.

Por fim, a disponibilidade para operações de escrita (A_w) em uma grade de dimensões $m_i \times n_i$ do nível i de uma h-grade é calculada pela diferença entre a probabilidade de se obter acesso para leitura de todas as colunas da grade e probabilidade de se obter apenas acesso para leitura (e não para escrita) também de todas as colunas da grade, sendo expressa pela recorrência:

$$A_w(i) = \begin{cases} \rho & (i = 0) \\ [P_r^{m_i,1}(i)]^{n_i} - [P_r^{m_i,1}(i) - P_w^{m_i,1}(i)]^{n_i} & (i \geq 1) \end{cases}$$

A prova do lema a seguir mostra que a disponibilidade do dado no protocolo de quorum em grade hierárquica utilizando uma h-grade de l níveis é maior (tendendo assintoticamente para 1 quando l cresce) do que a disponibilidade do dado no protocolo de quorum em grade original (cuja disponibilidade para operações de escrita tende assintoticamente para 0 quando N cresce).

Lema 3.8 *No protocolo de quorum em grade hierárquica, considerando-se que cada cópia está independentemente acessível com probabilidade $\rho > \rho^*$, a disponibilidade do dado cresce assintoticamente conforme mais níveis são adicionados à h-grade.*

Prova. Considere uma grade do nível 1 de dimensões $m_1 \times n_1$ constituída de objetos do nível 0 (cópias do dado). Nessa grade, a disponibilidade para operações de escrita $A_w(1)$é dada por

$$[1 - (1-\rho)^{m_1}]^{n_1} - [1 - \rho^{m_1} - (1-\rho)^{m_1}]^{n_1}.$$

Claramente, a disponibilidade para escrita de um objeto do nível 1 será maior do que a disponibilidade do objeto do nível 0 se $A_w(1) > \rho$. Reescrevendo-se essa disponibilidade como a função

$$f(x) = [1 - (1-x)^{m_1}]^{n_1} - [1 - x^{m_1} - (1-x)^{m_1}]^{n_1},$$

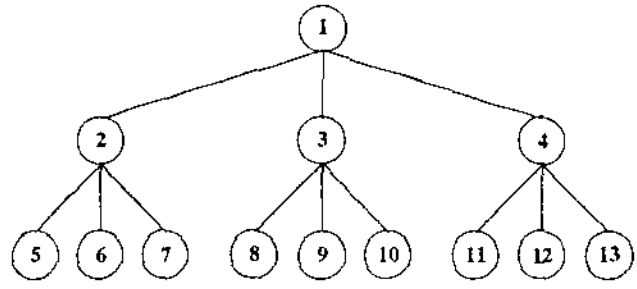

Figura 3.6: 13 cópias organizadas em uma estrutura de árvore.

verifica-se que existe um valor $\rho^* < 1$ tal que $f(x) > x$ para $\rho^* < x < 1$, uma vez que $f(1) = 1$, $f'(1) = 0$ e f é continuamente diferenciável. Note que ρ^* é estritamente menor que 1, embora seu valor exato dependa dos valores m_1 e n_1. Por exemplo, quando $m_1 = n_1 = 2$, tem-se que $f(x) = x^4 + 4x^3(1-x)$, e $f(x) > x$ para $x > 0.77$, isto é, $\rho^* = 0.77$.

O mesmo processo é repetido para o nível 2 substituindo-se ρ por $A_w(1)$, e m_1 e n_1 por m_2 e n_2; como $A_w(1)$ é maior que ρ, $A_w(2)$ é ainda maior que $A_w(1)$. Desse modo, conforme mais níveis vão sendo adicionados à h-grade, a disponibilidade para operações de escrita vai crescendo e tendendo assintoticamente para 1. Por fim, como os quorums de leitura e escrita-cega são subconjuntos dos quorums de escrita, suas respectivas disponibilidades também tendem assintoticamente para 1.

O lema, portanto, é válido. □

3.4 Protocolo de Quorum em Árvore

Neste protocolo [AA92b], as N cópias do dado replicado são organizadas logicamente como vértices de uma estrutura de árvore que possui altura h (número máximo de níveis) e grau d (número máximo de filhos de cada vértice). Para facilitar a notação, adota-se na descrição do protocolo a terminologia padrão — isto é, raiz, folhas, etc — para denotar os vértices da árvore; considera-se ainda que a árvore é completa, ou seja, cada vértice tem exatamente d filhos, o que resulta em $N = (d^h - 1)/(d-1)$. A figura 3.6 mostra um exemplo de uma árvore com $h = 3$ e $d = 3$ contendo um total de 13 cópias.

```
typedef struct TreeNode{
    COPY Node;
    struct TreeNode *Child[d];
} *TREE;

FormQuorum(TREE Tree, int Length, int Width)
{
    QUORUM ChildQuorum, TempQuorum;

    if (Length > Height(Tree))
        return(error);
    else if (Length == 0)
            return({});
        else if (Tree->Node grants permission) {
                /* Let TempQuorum be a set of children such that |TempQuorum|= Width */
                ChildQuorum = ⋃_{i∈TempQuorum} FormQuorum(Tree->Child[i], Length - 1, Width);
                return(Tree->Node ∪ ChildQuorum);
            }
            else{
                /* Let TempQuorum be a set of children such that |TempQuorum|= Width */
                ChildQuorum = ⋃_{i∈TempQuorum} FormQuorum(Tree->Child[i], Length, Width);
                return(ChildQuorum);
            }
}
```

Figura 3.7: Algoritmo para construir quorums no protocolo de quorum em árvore.

3.4.1 Formação e tamanho dos quorums

Os quorums de acesso ao dado replicado são formados na estrutura de árvore através do algoritmo mostrado na figura 3.7. O algoritmo é chamado passando-se a raiz da árvore e os valores l e w como argumentos; o conjunto de cópias retornado pelo algoritmo é denominado um quorum de comprimento l (número de níveis da árvore) e largura w (número de filhos de cada vértice), e é denotado pelo par $< l, w >$. A estratégia utilizada pelo algoritmo é tentar formar recursivamente o quorum incluindo a raiz e w de seus filhos, e para cada filho incluído, w de seus filhos, e assim sucessivamente, até uma profundidade l. Durante o processo de formação do quorum, se alguma cópia está inacessível em uma profundidade h' da raiz da árvore essa cópia é substituída por w quorums de comprimento $l - h'$ (e de mesma largura w), começando a partir dos filhos da cópia inacessível. A recursão termina com sucesso quando o comprimento do quorum a ser formado é zero. O algoritmo não consegue formar o quorum quando o comprimento do quorum excede a altura da sub-árvore remanescente; nesse caso, é retornado o conjunto vazio.

Um quorum de leitura (q_r) no protocolo de quorum em árvore é denotado pelo par $< l_r, w_r >$ e um quorum de escrita (q_w) é denotado por $< l_w, w_w >$. Analogamente ao protocolo de votação tradicional, onde há apenas uma dimensão para a definição de quorums, as dimensões (comprimento e largura) dos quorums de leitura e escrita no protocolo de quorum em árvore devem ser definidas satisfazendo certas restrições de forma a garantir a interseção entre os quorums de operações conflitantes. São elas:

$$l_r + l_w > h \tag{3.3}$$

$$w_r + w_w > d \tag{3.4}$$

$$2.l_w > h \tag{3.5}$$

$$2.w_w > d \tag{3.6}$$

Considere a árvore mostrada na figura 3.6 ($h = d = 3$) com a seguinte definição de quorums de leitura e escrita: $q_r = < 1, 2 >$ e $q_w = < 3, 2 >$. Nesse exemplo, no melhor caso um quorum de leitura é formado apenas pela raiz da árvore. Se raiz estiver inacessível, porém, um quorum de dimensões $< 1, 2 >$ pode ser formado a partir de quaisquer dois filhos da raiz, por exemplo {2, 3}, {2, 4} ou {3, 4}. Se dois ou mais filhos da raiz estão inacessíveis, esses filhos podem, por sua vez, ser substituídos por dois de seus filhos. Desse modo, se as cópias 1, 2 e 3 estão inacessíveis, então um quorum de leitura pode ser formado a partir da cópia 4 e de dois filhos tanto da cópia 2 quanto da cópia 3, por exemplo os conjuntos {4, 5, 6} e {4, 8, 10}. Finalmente, se as cópias 1, 2, 3 e 4 estão inacessíveis, um quorum de leitura ainda pode ser formado por dois filhos de pelo menos duas dentre as cópias 2, 3 e 4, por exemplo os conjuntos {5, 6, 8, 9}, {6, 7, 12, 13} e {8, 10, 11, 13}. Um quorum de escrita de dimensões $< 3, 2 >$ deve sempre incluir a raiz, dois de seus filhos e dois dos filhos de seus filhos. São exemplos de quorums de escrita os conjuntos {1, 2, 3, 5, 6, 8, 9}, {1, 2, 4, 6, 7, 12, 13} e {1, 3, 4, 9, 10, 11, 13}. Similarmente, com os dois quorums, de leitura e escrita, tendo dimensões $< 2, 2 >$, qualquer um dos conjuntos de cópias {1, 2, 3}, {1, 3, 4}, {1, 4, 5, 6} ou {2, 4, 6, 7, 12, 13}, por exemplo, seria suficiente para formar ambos.

O tamanho de um quorum de leitura varia de $[(w_r)^{l_r} - 1]/(w_r - 1)$, no melhor caso, quando todas as cópias do quorum são dos níveis mais altos da árvore, a $(w_r)^{l_r-1}.[(w_r)^{l_r} - 1]/(w_r - 1)$, no pior caso, quando as cópias são dos níveis mais baixos. De modo semelhante, o tamanho de um quorum de escrita varia de $[(w_w)^{l_w} - 1]/(w_w - 1)$ a $(w_w)^{l_w-1}.[(w_w)^{l_w} - 1]/(w_w - 1)$. Como o tamanho dos quorums é variável, para o cálculo do tamanho médio dos quorums introduz-se um parâmetro f que indica a fração de operações que utilizam a raiz da árvore nos seus quorums. Seja $MS_h(l, w)$ o tamanho médio de um quorum de dimensões $< l, w >$ em uma árvore de altura h.

O tamanho do quorum para uma árvore de altura $h+1$ é então:

$$MS_{h+1}(l,w) = [(1 + w.MS_h(l-1,w)] + (1-f).w.MS_h(l,w)$$

onde $MS_h(l,w)$ é $(w^l - 1)/(w-1)$ e $MS_h(0,w) = 0$. Note que o primeiro termo da relação de recorrência corresponde ao caso em que a raiz está incluída no quorum, enquanto que o segundo termo corresponde à situação em que a raiz não está incluída e quorums de comprimento l são formados em w sub-árvores. O parâmetro f, portanto, expressa todas as possibilidades de tamanho dos quorums, do melhor ($f = 1$) ao pior ($f = 0$) caso.

3.4.2 Prova de Correção

Os dois lemas seguintes mostram que quorums de operações conflitantes no protocolo de quorum em árvore têm pelo pelo menos um cópia em comum.

Lema 3.9 *Em uma árvore de altura h e grau d, o protocolo de quorum em árvore garante que qualquer quorum de leitura e qualquer quorum de escrita de dimensões $q_r = < l_r, w_r >$ e $q_w = < l_w, w_w >$, respectivamente, satisfazendo as restrições 3.3 e 3.4, têm pelo menos um cópia em comum.*

Prova. Por indução na altura da(s) (sub-)árvore(s).

Base. O lema vale para uma árvore de altura 1, uma vez que há apenas uma cópia na árvore.

Hipótese. Admita que o lema vale para árvores de altura h e quorums de dimensões tais que $l_r + l_w > h$. Note que o grau da árvore, d, é fixo.

Passo. Deve ser provado que o lema ainda vale para uma árvore de altura $h+1$ e quorums de dimensões tais que $l_r + l_w > h+1$ e $w_r + w_w > d$. Há dois casos a considerar.

Caso 1. O quorum de leitura q_r inclui a raiz da árvore, aqui denominada c_0. Neste caso, q_r deve incluir w_r quorums de comprimento $l_r - 1$ em quaisquer sub-árvores cujas raízes são filhos de c_0. O quorum de escrita q_w pode incluir c_0 ou não. Se ele a inclui, então os dois quorums têm uma cópia em comum. Se q_w não inclui c_0, então ele deve incluir w_w quorums de comprimento l_w em quaisquer sub-árvores cujas raízes são filhos de c_0. Como $w_r + w_w > d$, q_r e q_w devem ter pelo menos uma sub-árvore em comum. Nesta sub-árvore, a soma do comprimento dos quorums é $(l_r - 1) + l_w$. Mas $l_r + l_w > h+1$, e portanto $(l_r - 1) + l_w > h$. Qualquer sub-árvore com a raiz sendo um filho de c_0 tem altura h, e, portanto, pela hipótese de indução, os quorums de leitura e escrita formados na sub-árvore têm pelo menos uma cópia

em comum. Portanto, os quorums de leitura e escrita formados na árvore de altura $h+1$ também têm pelo menos uma cópia em comum.

Caso 2. O quorum de leitura q_r não inclui a raiz da árvore, c_0. Neste caso, q_r deve incluir w_r quorums de comprimento l_r em quaisquer w_r sub-árvores cujas raízes são filhos de c_0. O quorum de escrita pode incluir ou não c_0. Se ele não a inclui, então ele deve incluir w_w quorums de comprimento l_w em quaisquer sub-árvores cujas raízes são filhos de c_0. Pela restrição 3.4, deve haver uma sub-árvore, de altura h, comum aos dois quorums onde foram formados quorums de leitura e escrita de comprimentos l_r e l_w, respectivamente. Como $l_r + l_w > h$, pela hipótese de indução q_r e q_w se interceptam. Se q_w inclui c_0, então ele deve incluir w_w quorums de comprimento $l_w - 1$ em quaisquer sub-árvores cujas raízes são filhos de c_0. Novamente, deve haver uma sub-árvore, de altura h, comum aos dois quorums onde foram formados quorums de leitura e escrita de comprimentos l_r e $l_w - 1$, respectivamente. Como $l_r + l_w > h + 1$, $l_r + (l_w - 1) > h$. Pela hipótese de indução, os quorums de leitura e escrita se interceptam na sub-árvore e portanto também se interceptam na árvore de altura $h+1$.

Portanto, por indução, o lema é válido. □

Lema 3.10 *Em uma árvore de altura h e grau d, o protocolo de quorum em árvore garante que quaisquer dois quorums de escrita de dimensões satisfazendo as restrições 3.5 e 3.6 têm pelo menos um cópia em comum.*

Prova. Análoga à prova do lema 3.9. □

3.4.3 Disponibilidade do Dado

No protocolo de quorum em árvore, a disponibilidade do dado replicado em uma árvore tendo a cópia c_0 como raiz é expressa recursivamente em função da disponibilidade do dado nas sub-árvores cujas raízes são filhos de c_0, e calculada analogamente à disponibilidade do dado no protocolo de votação tradicional (ver seção 3.2.3). Assim, a disponibilidade para uma operação que requeira um quorum de dimensões $<l, w>$ em uma árvore de altura h e grau d é dada pela recorrência

$$A_h(l,w) = \begin{cases} \rho & (h=1) \\ \rho[AV(d,w,A_{h-1}(l-1,w))] + (1-\rho)[AV(d,w,A_{h-1}(l,w))] & (h>1), \end{cases}$$

onde o primeiro termo corresponde ao caso em que a raiz está acessível e deu a permissão de acesso, e o segundo termo corresponde à situação em que a raiz não deu permissão de acesso ou não pôde ser contactada.

3.4.4 Casos Particulares

Conforme visto na seção 3.4.1, o tamanho de um quorum no protocolo de quorum em árvore, na presença e ausência de falhas, varia tanto em função de seu comprimento quanto de sua largura. Em [AA92b], dois casos particulares de dimensões de quorums do protocolo de quorum em árvore são destacados:

QA1: As dimensões dos quorums de leitura são $< 1, (d+1)/2 >$ e as dos quorums de escrita são $< h, d/2+1 >$.

QA2: As dimensões dos quorums de leitura são $< 1, d >$ e as dos quorums de escrita são $< h, 1 >$.

No primeiro caso, denominado protocolo QA1[3], um quorum de leitura é formado por apenas uma cópia quando não há falhas. Se a raiz está inacessível, porém, o tamanho do quorum de leitura aumenta para $(d+1)/2$. Se mais cópias estiverem inacessíveis, o tamanho do quorum pode crescer até o máximo de $[(d+1)/2]^h$. Note que, em geral, $[(d+1)/2]^h < N/2$. Admitindo-se que 50% das operações de leitura utilizam a raiz — isto é, $f = 0.5$ —, o tamanho médio do quorum de leitura passa a ser $(h+1)/2$: como $h = \log_d[N(d-1)+1]$, isto implica em um quorum de leitura da ordem de $O(\log N)$. Os quorums de escrita, por outro lado, são todos de tamanho $[[(d+1)/2]^h - 1]/[(d-1)/2]$.

No segundo caso, denominado protocolo QA2, um quorum de leitura continua podendo ser formado por apenas uma cópia quando não há falhas, mas seu tamanho pode crescer até d^{h-1} na situação extrema em que somente as folhas da árvore estão acessíveis. Os quorums de escrita, entretanto, contrastando com o protocolo QA1, são formados por qualquer caminho de vértices na árvore ligando a raiz a alguma folha (ambas incluídas) e possuem assim tamanho h, da ordem de $O(\log N)$.

Como última observação, salienta-se que em [AA92b] é ainda descrita uma extensão do protocolo de quorum em árvore onde a estrutura lógica utilizada é uma árvore incompleta, ou seja, uma árvore cujos vértices não têm o mesmo número de filhos. Tal estrutura dá mais liberdade para a escolha do número total de cópias do dado replicado — utilizando-se árvores completas de grau 3, por exemplo, as quantidades de cópias possíveis limitam-se a números da forma $(3^h - 1)/2$, isto é, 1, 4, 13, 40, etc.

[3]Uma versão do protocolo de quorum em árvore tratando apenas dos casos particulares referentes ao protocolo QA1 é descrita em [AA90c].

3.5 Análise dos Protocolos

Esta seção analisa os protocolos revisados comparando-os com os protocolos baseados em votação e tecendo considerações sobre o grau de tolerância a falhas alcançado por cada um.

3.5.1 Estruturas Lógicas × Votação

Com respeito à abordagem adotada para manter a consistência mútua entre as cópias, os protocolos baseados em estruturas lógicas são idênticos aos protocolos baseados em votação: antes da realização de uma operação sobre um dado replicado devem ser obtidas permissões de um quorum, e quorums de operações conflitantes devem ter uma intersecção não nula. A principal diferença entre essas duas classes de protocolos está na maneira de como definir quais conjuntos de cópias são válidos para formar quorums. Para obter quorums menores, os protocolos baseados em estruturas lógicas impõem mais restrições à escolha de quais conjuntos de cópias podem ser usados como quorums do que os protocolos baseados em votação.

Como exemplo, suponha um dado replicado com N cópias sobre o qual deseja-se definir um operação $op1$ cujo quorum exigido tenha tamanho q. Nesse caso, através do protocolo de votação tradicional, considerando-se que a cada cópia é atribuído um único voto, qualquer conjunto de cópias de cardinalidade q seria suficiente para formar um quorum de $op1$. Nos protocolos baseados em estruturas lógicas, porém, apenas os conjuntos de cardinalidade q que contêm cópias localizadas em posições específicas da estrutura lógica utilizada poderiam ser usadas na formação desse mesmo quorum.

Destas restrições adicionais na formação dos quorums decorre o fato de, para quorums do mesmo tamanho, a disponibilidade do dado nos protocolos baseados em estruturas lógicas ser significativamente menor que a disponibilidade nos protocolos baseados em votação. Outra desvantagem dos protocolos baseados em estruturas lógicas é que as estruturas limitam as possibilidades de escolha para o número total de cópias do dado replicado, que passa a depender do formato e tamanho da estrutura[4]. Por outro lado, as estruturas lógicas permitem que quorums reduzidos sejam obtidos mesmo para operações conflitantes, o que não é possível no protocolo de votação tradicional. Nesse último, se o tamanho q de um quorum exigido para uma operação $op1$ é pequeno em relação a total de cópias N, o quorum de um operação $op2$ que conflita com $op1$ deve ser de no mínimo $N - q + 1$ (que é muito próximo de N).

[4]Estruturas incompletas, como as árvores incompletas descritas em [AA92b], funcionam como estruturas completas de maior tamanho nas quais algumas cópias estão continuamente inacessíveis, o que diminui ainda mais a disponibilidade do dado.

Como nos protocolos baseados em estruturas lógicas nem todos os conjuntos de cópias de cardinalidade q podem ser usados como quorums de $op1$, é possível obter quorums para $op2$ de tamanho consideravelmente menor que $N - q + 1$ através de novas restrições impostas aos conjuntos de cópias candidatos a quorums de $op1$ e $op2$, ainda assim garantindo a intersecção necessária entre os quorums das duas operações.

Uma característica importante de um protocolo de controle de réplicas é a sua capacidade para permitir uma distribuição uniforme da carga de trabalho pelo nós, uma vez que os pedidos de acesso ao dado replicado podem ser atendidos por diferentes conjuntos de cópias concorrentemente. Nos protocolos baseados em votação, considerando-se um voto por cópia, a carga é distribuída uniformemente pois cada cópia possui a mesma responsabilidade na manutenção da consistência mútua. Nesse aspecto, os protocolos de quorum em grade, de votação hierárquica e de quorum em grade hierárquica também distribuem a carga uniformemente, já que utilizam estruturas *simétricas*, nas quais as cópias do dado replicado participam de um número igual de quorums e, portanto, possuem a mesma importância para o algoritmo de construção de quorum; a distribuição de carga alcançada pelos protocolos baseados em estruturas lógicas, porém, é muito melhor devido ao fato de proporcionarem uma menor relação entre o tamanho dos quorums e o total de cópias do dado. O protocolo de quorum em árvore é desvantajoso com respeito a essa característica porque utiliza uma estrutura *assimétrica* na estrutura de árvore, as cópias localizadas nos níveis mais altos são mais solicitadas (e por isso participam de mais quorums) do que as cópias dos níveis mais baixos. No protocolo QA1, por exemplo, um quorum de leitura tem tamanho um quando não há falhas, mas isto não significa que a leitura poderá ser feita a partir de qualquer cópia; como apenas a raiz pode ser usada para formar um quorum de leitura de tamanho um, essa cópia certamente se tornará um foco de congestionamento do sistema se o dado replicado passar a ser muito requisitado.

A seguir analisa-se a implicação das restrições adicionais impostas à formação dos quorums no grau de tolerância a falhas dos protocolos baseados em estruturas lógicas.

3.5.2 Aspectos de Tolerância a Falhas

O grau de tolerância a falhas de um protocolo de controle de réplicas indica o número máximo de cópias do dado replicado que podem estar inacessíveis para que um quorum exigido por alguma operação de acesso ainda consiga ser formado. Nos protocolos baseados em estruturas lógicas, a possibilidade de formação de um quorum depende não só do número de cópias inacessíveis mas também da posição destas cópias na estrutura lógica utilizada. Por exemplo, se as cópias inacessíveis compõem virtualmente algum quorum de alguma operação $op1$ conflitante com uma operação

Protocolo	Operação	Grau de Tolerância a Falhas	
		Melhor Caso	Pior Caso
Grade	Leitura	$N - \sqrt{N}$	$\sqrt{N} - 1$
	Escrita	$N - 2\sqrt{N} + 1$	$\sqrt{N} - 1$
Votação Hierárquica	Leitura	$N - N^{0.63}$	$N^{0.63} - 1$
	Escrita	$N - N^{0.63}$	$N^{0.63} - 1$
Grade Hierárquica	Leitura	$N - \sqrt{N}$	$\sqrt{N} - 1$
	Escrita-cega	$N - \sqrt{N}$	$\sqrt{N} - 1$
	Escrita	$N - 2\sqrt{N} + 1$	$\sqrt{N} - 1$
Árvore	Leitura	$N - 1$	$\log N - 1$
	Escrita	$N - \log N$	0

Tabela 3.1: Grau de tolerância a falhas dos protocolos baseados em estruturas lógicas.

*op*2, um quorum de *op*2 não mais poderá ser formado uma vez que todo quorum de *op*2 tem pelo menos uma cópia em comum com todo quorum de *op*1, e, desse modo, a operação *op*2 sempre necessitará de pelo menos uma das cópias inacessíveis. Portanto, considerando-se que um quorum de *op*1 tem tamanho q_{op1} e que um quorum de *op*2 tem tamanho q_{op2}, para a realização da operação *op*1 um protocolo baseado em alguma estrutura lógica tolerará a falha de até $N - q_{op1}$ cópias no melhor caso, mas de apenas $q_{op2} - 1$ cópias no pior caso.

A tabela 3.1 mostra o número de cópias inacessíveis, no melhor e no pior caso, considerando-se que o número total de cópias N é condizente com as dimensões da estrutura, que os protocolos baseados em estruturas lógicas toleram na formação dos quorums de acesso ao dado. Ressalta-se novamente que esse número não indica qualquer conjunto de cópias com essa cardinalidade, mas apenas cópias inacessíveis localizadas em posições específicas na estrutura. Note que no protocolo de quorum em grade uma operação de leitura, no pior caso, não tolera a falha de $\sqrt{N}$ cópias se elas correspondem a alguma coluna da grade (mesmo que, como de fato acontece neste protocolo, todas as cópias de alguma coluna não sejam suficientes para formar quorum algum); isto explica-se porque, conforme depois definido pelo protocolo de quorum em grade hierárquica, uma coluna da grade pode ser usada como quorum de escrita-cega, operação inexistente no protocolo de quorum em grade.

Dependendo da freqüência com que o dado é alterado em relação à freqüência com que ele é lido, pode-se ajustar as estruturas lógicas de forma a aumentar o grau de tolerância a falhas (juntamente com a disponibilidade do dado) de uma operação em detrimento de outra(s). Nos protocolos de quorum em grade e em grade hierárquica, as dimensões da(s) grade(s) são os parâmetros usados para tal ajuste. No protocolo de quorum em grade, por exemplo, diminuindo-se o número

de colunas da grade através do aumento do número de linhas obtêm-se operações de leitura com quorums menores e maiores chances de serem formados em detrimento das operações de escrita, que passam a exigir quorums maiores e de formação mais difícil; este ajuste mostra-se útil quando o dado replicado é muito mais lido do que alterado. No protocolo de votação hierárquica e no protocolo de quorum em árvore, o ajuste é feito modificando-se, além do formato da própria estrutura, os quorums de cada nível e o comprimento e a largura dos quorums, respectivamente.

Notas Bibliográficas

A idéia de organizar os nós de um sistema distribuído em estruturas lógicas foi originalmente proposta para resolver eficientemente o problema de exclusão mútua: organizam-se os nós em planos projetivos finitos ou em uma grade em [Mae85]; em uma árvore qualquer em [Ray89], onde ainda é necessário o uso de uma mensagem especial como *token*; e em uma árvore binária em [AA89a] e, posteriormente, em [AA91]. Os protocolos de exclusão mútua de [Mae85] e [AA89a] foram primeiramente generalizados para protocolos de controle de réplicas em [AA90a] através de votação entre um conjunto de estruturas lógicas (grades e/ou árvores binárias). Mais tarde, cada um desses dois protocolos foi individualmente generalizado para controlar dados replicados com a definição de modos distintos (um para operações de leitura, que não necessitam de exclusão mútua para serem realizadas, e outro para operações de escrita, que necessitam de exclusão mútua) de formação de quorums na estrutura utilizada. A generalização do protocolo de [Mae85] resultou no protocolo de quorum em grade, e a generalização do protocolo de [AA89a] resultou no protocolo de quorum em árvore.

Dentre os trabalhos recentes na área, dois propõem estratégias para tornar os protocolos baseados em estruturas lógicas protocolos dinâmicos: [AA92a] utiliza a idéia das partições virtuais para, a partir da estrutura lógica, formar os quorums apenas com as cópias que puderem ser contactadas; e [RL92] sugere que a cada modificação detectada na configuração do sistema uma nova estrutura lógica seja construída apenas com as cópias correntes. Em [KM92] são apresentadas algoritmos para, dados o total de cópias e a proporção com que o dado é alterado em relação ao número de vezes que ele é lido, encontrar, no protocolo de votação hierárquica, a hierarquia e os quorums dentro de cada nível que minimizam os custos de comunicação em cada acesso ao dado replicado.

Capítulo 4

Protocolo de Votação Hierárquica Estendida

Este capítulo propõe um novo protocolo para controlar dados replicados em sistemas distribuídos. O novo protocolo, denominado protocolo de votação hierárquica estendida (ou protocolo VH+), segue a abordagem pessimista para manter a consistência mútua entre as cópias e é membro da classe de protocolos baseados em estruturas lógicas, revisada no capítulo anterior.

O protocolo VH+ organiza as cópias do dado replicado em uma estrutura lógica similar à estrutura utilizada pelo protocolo de votação hierárquica (ou protocolo VH), descrito na seção 3.2 – daí a escolha do nome VH+. Ambas estruturas, a do protocolo VH e a do protocolo VH+, são hierarquias de vértices nas quais as cópias do dado replicado correspondem aos vértices folhas, e que têm quorums definidos independentemente para cada nível. Os dois protocolos, no entanto, diferem nos seguintes pontos:

- na hierarquia do protocolo VH+, os vértices não necessariamente devem ter o mesmo número de filhos, como é exigido pelo protocolo VH; ou seja, nessa hierarquia, os vértices-folhas podem estar em níveis diferentes;
- o protocolo VH+ define operações de escrita-cega (como é feito pelo protocolo de quorum em grade hierárquica, descrito na seção 3.3), além das tradicionais operações de leitura e de escrita;
- para cada nível da hierarquia do protocolo VH+, apenas definem-se quorums para as operações de leitura e de escrita-cega; os quorums para operações de escrita são formados a partir da combinação, feita individualmente para cada nível, dos quorums definidos para as operações de leitura e escrita-cega.

Com essas extensões, o protocolo VH+ mostra-se uma generalização do protocolo de quorum em grade, do protocolo de quorum em grade hierárquica, de vários casos particulares do protocolo de quorum em árvore, bem como, é claro, do próprio protocolo VH original. Além de generalizar grande parte dos protocolos baseados em estruturas lógicas, que passam a poder ser implementados através de um único esquema, o protocolo VH+ ainda consegue, em muitas situações, obter melhores resultados do que aqueles obtidos por cada um desses protocolos individualmente. Uma descrição detalhada do protocolo VH+ e de como essa generalização se procede constituem o escopo do restante do capítulo.

A próxima seção descreve a estrutura hierárquica utilizada pelo protocolo VH+. A seção 4.2 mostra como os quorums são formados nessa estrutura e quais os tamanhos (número de cópias) de cada um. A seção 4.3 apresenta a prova de correção do protocolo VH+ e a seção 4.4 explica como calcular a disponibilidade do dado para as operações de acesso que o protocolo VH+ provê. Por fim, a seção 4.5 compara o protocolo VH+ com todos os protocolos baseados em estruturas lógicas vistos no capítulo anterior.

4.1 A Estrutura Hierárquica

No protocolo VH+, as N cópias do dado replicado são organizadas logicamente como os vértices folhas de uma hierarquia de vértices de m níveis. Os outros vértices da hierarquia (ou seja, os vértices não-folhas) representam grupos lógicos de vértices do nível imediatamente inferior. O vértice raiz é o vértice do nível mais alto da hierarquia, nível m, e os vértices folhas do nível mais baixo estão localizados no nível 0. Para cada nível i ($1 \leq i \leq m$) da hierarquia é definido um número l_i que indica o número máximo de vértices do nível $i-1$ que compõem o grupo lógico de vértices representado por um vértice do nível i; ou seja, um vértice do nível i pode ter *até* l_i filhos. Para cada vértice v da hierarquia, por sua vez, é definido um número n_v que indica o número real de filhos de v. Note que $n_v = 0$, se v é um vértice folha; $n_v = l_m$, se v é o vértice raiz; ou, para os outros vértices, $1 \leq n_v \leq l_i$ (onde i é o nível de v). Os vértices filhos do vértice v são denotados por $c_v[k]$ ($k = 1, 2, \ldots, n_v$).

Nesta hierarquia, o número total de vértices folhas (cópias do dado replicado) descendentes de um vértice v é dado pela seguinte recorrência:

$$Tot(v) = \begin{cases} 1 & (n_v = 0) \\ \sum_{k=1}^{n_v} Tot(c_v[k]) & (n_v > 0) \end{cases}$$

Portanto, $Tot(v) = N$ quando v é o vértice raiz. Se todos os vértices tiverem o número real de filhos igual ao número máximo de filhos permitido para o nível a que pertencem, então $\prod_{i=1}^{m} l_i = N$. Nesse caso, diz-se que a estrutura é uma hierarquia

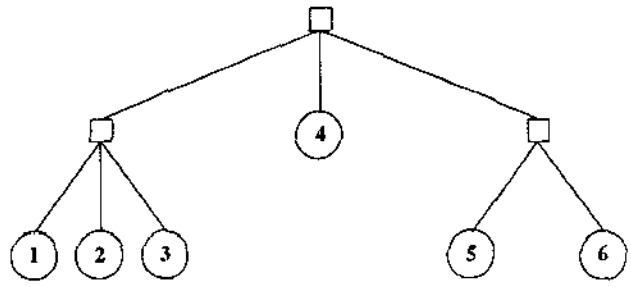

Figura 4.1: Seis cópias organizadas em uma hierarquia incompleta de 2 níveis.

completa, idêntica à hierarquia utilizada pelo protocolo VH. Se pelo menos um dos vértices não-folha possui menos filhos que o máximo permitido para o nível dele, diz-se que a hierarquia é incompleta.

Uma vantagem de se trabalhar com hierarquias incompletas é que elas permitem uma grande flexibilidade na escolha do número total de cópias do dado replicado, que no protocolo VH fica confinado a números da forma $\prod_{i=1}^{m} l_i$ ($l_i \geq 2$). Outra vantagem das hierarquias incompletas é que através delas é possível ao protocolo VH+ generalizar casos particulares do protocolo de quorum em árvore (ver seção 4.5.2 mais adiante), o que não pode ser feito somente com hierarquias completas. Uma desvantagem, porém, é o fato de, nas hierarquias incompletas, cópias correspondentes a vértices dos níveis mais altos terem mais responsabilidade no controle do dado replicado do que cópias correspondentes a vértices dos níveis mais baixos, prejudicando a distribuição uniforme da carga de trabalho pelos nós; as hierarquias incompletas, assim, são consideradas estruturas assimétricas, como também o são, aliás, as estruturas de árvores utilizadas pelo protocolo de quorum em árvore.

Para facilitar a notação, os valores $l_1, l_2, \ldots, l_m$ da hierarquia de vértices são denotados pelo vetor $\mathbf{l}=(l_1, l_2, \ldots, l_m)$. A figura 4.1 mostra um exemplo de uma hierarquia incompleta de 2 níveis ($\mathbf{l}=(3,3)$) contendo 6 cópias de um dado replicado.

4.2 Quorums de Acesso

Para cada nível i ($i = 1, 2, \ldots, m$) da hierarquia são definidos um quorum de leitura r_i e um quorum de escrita-cega b_i. Para garantir a interseção entre os quorums das operações de leitura e os quorums das operações de escrita-cega (ver prova de correção do protocolo na seção 4.3), os valores de r_i e b_i devem ser definidos obedecendo a restrição

$$r_i + b_i > l_i \tag{4.1}$$

Para que r_i e b_i sejam mínimos, esta restrição é reescrita como

$$r_i + b_i = l_i + 1 \tag{4.2}$$

Quando r_i e b_i não são mínimos ($r_i + b_i > l_i + 1$), pelo menos um dos dois quorums (leitura ou escrita-cega) conterá mais cópias do que o mínimo necessário para garantir uma interseção não nula entre eles, o que implica em custos de comunicação adicionais na formação dos quorums que podem ser perfeitamente desprezados. Com a adoção da restrição 4.2 em vez da 4.1, não faz-se necessário que os dois quorums sejam explicitamente definidos para cada nível: é suficiente a definição de um deles para que o outro seja calculado. No protocolo VII+, considera-se que para cada nível i ($i = 1, 2, \ldots, m$) da hierarquia apenas o quorum de leitura r_i é definido; o quorum de escrita-cega b_i, por 4.2, é calculado pela expressão $b_i = l_i - r_i + 1$. Os valores $r_1, r_2, \ldots, r_m$, novamente para facilitar a notação, são denotados pelo vetor $\mathbf{r}=(r_1, r_2, \ldots, r_m)$. Note que, como duas operações de escrita-cega não conflitam entre si, não é necessária a presença da restrição $2.bi > l_i$. Note também que não foram definidos quorums de escrita para os níveis da hierarquia; eles são deduzidos a partir da combinação dos quorums de leitura e de escrita-cega, conforme mostrado na seção seguinte.

4.2.1 Formação dos Quorums

Um vértice não-folha v da hierarquia dará permissão de acesso ao dado replicado para uma operação requisitada se ele receber essa mesma permissão de um conjunto de seus filhos suficiente para formar o quorum exigido para essa operação no nível em que v está localizado; caso v seja um vértice folha, ele deve pedir essa permissão da própria cópia do dado replicado à qual ele corresponde. Desse modo, um vértice não-folha v do nível i ($1 \leq i \leq m$) dará permissão de acesso para leitura (escrita-cega) se ele receber permissão de acesso para leitura (escrita-cega) de pelo menos r_i (b_i) de seus filhos, vértices do nível $i - 1$; caso alguma operação (leitura ou escrita-cega) tenha o quorum do nível i maior que o número real de filhos de v ($r_i < n_v$ ou $b_i < n_v$), v não dará a permissão de acesso para essa operação. A figura 4.2 mostra o algoritmo para construir quorums de leitura ou escrita-cega no protocolo VII+ (nesse algoritmo, a operação *op* significa leitura ou escrita-cega, dependendo da operação requisitada); ele deve ser acionado tendo como argumento o vértice raiz da hierarquia.

```
FormQuorum_op(v, i)
{
    if (n_v == 0)
        if (copy corresponding to vertex v grants permission)
            return({v});
        else
            return({});
    if (n_v ≥ op_i){
        /* Let TempQuorum be a set of v's children such that |TempQuorum|= op_i */
        Quorum_op = ⋃_{v∈TempQuorum} FormQuorum_op(v, i − 1);
        return(Quorum_op);
    }
    else
        return({});
}
```

Figura 4.2: Algoritmo para construir quorums de leitura ou escrita-cega no protocolo VII+.

Agora descreve-se a combinação dos quorums de leitura e escrita-cega de cada nível para formar o quorum de escrita. Para que um vértice não-folha de um nível i da hierarquia dê permissão de acesso para escrita ele deve (a) receber permissão de acesso para escrita de uma quantidade de seus filhos equivalente ao menor quorum definido para o nível i, isto é, $min(r_i, b_i)$, e (b) receber permissão de acesso para a operação de maior quorum do nível i (que é leitura, se $r_i > b_i$, ou escrita-cega, caso contrário) de uma quantidade de seus filhos equivalente à diferença entre o maior e o menor quorum, isto é, $|\, r_i - b_i \,|$. Por exemplo, suponha que $l_i = 6$, $r_i = 1$ e, portanto, $b_i = 6$ para algum nível i da hierarquia. Nesse caso, um vértice não-folha v do nível i (suponha $n_v = l_i$) dará permissão de acesso para escrita se, dentre os vértices do nível $i - 1$ filhos de v, um dá permissão de acesso para escrita e cinco dão permissão de acesso para escrita-cega; da mesma maneira, se $r_i = 5$ e $b_i = 2$, seria suficiente que v recebesse permissão de acesso para escrita de dois de seus filhos juntamente com permissão de acesso para leitura de mais três deles. A permissão de acesso para escrita não é dada por um vértice v se algum dos quorums (leitura ou escrita-cega) definidos para o seu nível é maior que seu número real de filhos. O algoritmo para construir quorums de escrita no protocolo VII+ é mostrado na figura 4.3; ele é acionado de modo similar ao algoritmo anterior.

Similarmente ao protocolo VII, no protocolo VII+ a permissão para que uma operação de acesso ao dado replicado seja realizada é conseguida pedindo-se (e obtendo-se) essa mesma permissão ao (do) vértice raiz da hierarquia — daí os argumentos iniciais dos algoritmos de construção de quorum serem relativos a esse vértice. O vértice raiz propaga o pedido para seus filhos, que por sua vez o fazem

```
FormQuorum_w(v, i)
{
    if (n_v == 0)
        if (copy corresponding to vertex v grants permission)
            return({v});
        else
            return({});
    if (n_v ≥ max(r_i, bw_i)){
        /* Let TempQuorum be a set of v's children such that |TempQuorum|= min(r_i, bw_i) */
        Quorum_w ← ⋃_{c∈TempQuorum} FormQuorum_w(c, i − 1);
        /* Let TempQuorum be a set of v's children such that |TempQuorum|=|r_i − bw_i| */
        if (r_i > bw_i)
            Quorum_op ← ⋃_{c∈TempQuorum} FormQuorum_r(c, i − 1);
        else
            Quorum_op ← ⋃_{c∈TempQuorum} FormQuorum_bw(c, i − 1);
        return(Quorum_w ∪ Quorum_op);
    }
    else
        return({});
}
```

Figura 4.3: Algoritmo para construir quorums de escrita no protocolo VII+.

novamente para seus filhos, até que os pedidos recursivamente cheguem às cópias; as respostas aos pedidos (das cópias e dos outros vértices) são propagadas de volta, de baixo para cima na hierarquia, e uma resposta positiva do vértice raiz é dada apenas se o quorum necessário para operação requisitada foi formado com sucesso. A seção 4.3 apresenta provas de que quorums de operações conflitantes formados assim sempre têm uma intersecção não-nula.

Na hierarquia incompleta mostrada na figura 4.1, fazendo-se **r**=(1, 3) os conjuntos de cópias {1, 4, 5}, {2, 4, 6} e {3, 4, 5} são exemplos de quorums de leitura; {4} e {1, 2, 3} são exemplos de quorums de escrita-cega; e {1, 4, 6}, {2, 4, 5} e {1, 2, 3, 4, 5} são exemplos de quorums de escrita. Da mesma maneira, fazendo-se **r**=(1, 2) {1, 4}, {2, 5} e {4, 6} são exemplos de quorums de leitura, e {1, 2, 3, 4} é um (o único, na verdade) exemplo de quorum de escrita-cega ou escrita.

4.2.2 Tamanho dos Quorums

Nas hierarquias completas, os tamanhos dos quorums de leitura e escrita-cega do protocolo VII+ são calculados de maneira semelhante ao cálculo do tamanho dos quorums no protocolo VII. O tamanho de um quorum de leitura em relação a um nível i ($1 \leq i \leq m$) da hierarquia é dado por $S_r(i) = \prod_{j=1}^{i} r_j$; similarmente, o

tamanho de um quorum de escrita-cega é $S_b(i) = \prod_{j=1}^{i} b_j$. O tamanho de um quorum de escrita, por depender dos quorums de leitura e escrita-cega de cada nível, é dado pela seguinte recorrência:

$$S_w(i) = \begin{cases} 1 & (i = 0) \\ b_i\, S_w(i-1) \;+\; (r_i - b_i)\, S_r(i-1) & (i \geq 1 \;\; e \;\; r_i \geq b_i) \\ r_i\, S_w(i-1) \;+\; (b_i - r_i)\, S_b(i-1) & (i \geq 1 \;\; e \;\; r_i < b_i) \end{cases}$$

Nas hierarquias incompletas,devido às cópias poderem estar localizadas em níveis diferentes, os quorums podem ter tamanhos também diferentes — as cópias dos níveis mais altos são mais requisitadas para formar os quorums do que as cópias dos níveis mais baixos. Desse modo, expressam-se os tamanhos dos quorums de cada operação de acesso ao dado replicado na forma média (o tamanho médio do quorum, considerando-se todas as possibilidades de formação), mínima (o quorum de menor tamanho) e máxima (o de maior tamanho). Para isso, primeiramente são dadas algumas definições.

Alguns vértices não-folhas podem ter menos filhos do que o mínimo necessário para formar o quorum de alguma operação de acesso ao dado replicado definido para seus níveis; tais vértices, denominados *vértices deficientes*, evidentemente nunca poderão dar permissão de acesso para essa operação. Um vértice é considerado indiretamente deficiente se, embora seu número real de filhos possa ser suficiente para formar o quorum exigido, a diferença entre seu número real de filhos e o número de filhos (direta ou indiretamente) deficientes não o é. Para facilitar o reconhecimento dos vértices direta ou indiretamente deficientes, que podem ser previamente desconsiderados no cálculo do tamanho dos quorums, é definida a função de *possibilidade de quorum* $PQ_{op}(v, i)$, que retorna 1 se há possibilidade de formação de um quorum da operação *op* a partir de um vértice v de um nível i da hierarquia, ou 0 se não existe tal possibilidade, ou seja, se v é um vértice deficiente. Esta função é expressa pela recorrência

$$PQ_{op}(v, i) = \begin{cases} 1 & (n_v = 0 \;\; \text{ou} \;\; (\sum_{k=1}^{n_v} PQ_{op}(c_v[k], i-1)) \geq op_i) \\ 0 & (n_v > 0 \;\; e \;\; (\sum_{k=1}^{n_v} PQ_{op}(c_v[k], i-1)) < op_i) \end{cases}$$

Nessa recorrência, quando a operação *op* é uma leitura ou escrita-cega, substitui-se op_i por r_i ou b_i, convenientemente; quando *op* é uma escrita, substitui-se op_i por $max(r_i, b_i)$).

Seja v um vértice não-folha de um nível i da hierarquia, e *op* uma operação de acesso ao dado replicado. Define-se $V_{op}(v)$ como o conjunto de todos os filhos $c_v[k]$ $(k = 1, 2, \ldots, n_v)$ de v tais que $PQ_{op}(c_v[k], i-1) = 1$. Define-se também $n_v^{op} = |V_{op}(v)|$. Informalmente, enquanto n_v indica o número real de filhos de v, n_v^{op}

representa o número de filhos de v que não são deficientes (em relação a op) e que, portanto, podem ser considerados na formação de um quorum (definido para o nível i) da operação op. Claramente, $0 \leq n_v^{op} \leq n_v$.

Finalmente, mostram-se agora as recorrências para calcular o tamanho dos quorums no protocolo VII+. O tamanho (número de cópias) médio dos quorums de uma operação op (leitura ou escrita-cega) a partir de um vértice v localizado em um nível i da hierarquia é dado pela recorrência

$$S_{op}(v,i) = \begin{cases} 1 & (n_v = 0) \\ 0 & (n_v^{op} < op_i) \\ \dfrac{\sum_{|C(v)|=op_i} \sum_{c \in C(v)} S_{op}(c,i-1)}{\binom{n_v^{op}}{op_i}} & (n_v^{op} \geq op_i) \end{cases}$$

onde $C(v)$ é um subconjunto de $V_{op}(v)$ formado por $|C(v)|$ vértices não-deficientes filhos de v; e $|C(v)| = op_i$ representa todas as combinações possíveis de op_i filhos não-deficientes de v[1]. Desse modo, o tamanho médio dos quorums das operações de leitura e escrita-cega para toda a hierarquia é dado por $S_r(v,m)$ e $S_b(v,m)$, respectivamente, quando v é o vértice raiz.

O tamanho médio dos quorums das operações de escrita é calculado similarmente, mas considerando-se os quorums de leitura e escrita-cega de cada nível. A recorrência a seguir mostra esse cálculo.

$$S_w(v,i) = \begin{cases} 1 & (n_v = 0) \\ 0 & (n_v^w < op_i) \\ \dfrac{\sum_{|C(v)|=min(r_i,b_i)} \sum_{c \in C(v)} S_w(c,i-1)}{\binom{n_v^w}{min(r_i,b_i)}} + \dfrac{\sum_{|C(v)|=|r_i-b_i|} \sum_{c \in C(v)} S_{op}(c,i-1)}{\binom{n_v^{op}}{|r_i-b_i|}} & (n_v^w \geq op_i) \end{cases}$$

Nessa recorrência, op representa uma operação de leitura, se $r_i > b_i$, ou de escrita-cega, caso contrário.

O cálculo dos tamanhos mínimo e máximo dos quorums de uma operação op (leitura ou escrita-cega) em relação a um vértice v do nível i é mais simples, comparado

[1] O número total de combinações é dado por $\binom{n_v^{op}}{op_i}$.

com o cálculo do tamanho médio, uma vez que apenas *uma* combinação de filhos não-deficientes de v deve ser considerada. A recorrência que expressa esse cálculo é a seguinte:

$$S'_{op}(v,i) = \begin{cases} 1 & (n_v = 0) \\ 0 & (n_v^{op} < op_i) \\ \sum_{c \in C'(v)} S'_{op}(c, i-1) & (n_v^{op} \geq op_i) \end{cases}$$

Para que essa recorrência retorne o tamanho mínimo (máximo) dos quorums da operação op, $C'(v)$ deve ser formado pelos op_i filhos de v que possuam as menores (maiores) quantidades de vértices folhas como descendentes, que podem ser identificados pela função Tot.

Analogamente, o cálculo do tamanho mínimo e máximo dos quorums de escrita é dado pela recorrência

$$S_w(v,i) = \begin{cases} 1 & (n_v = 0) \\ 0 & (n_v^w < op_i) \\ \sum_{c \in C1'(v)} S'_w(c, i-1) \; + \; \sum_{c \in C2'(v)} S'_{op}(c, i-1) & (n_v^w \geq op_i) \end{cases}$$

onde $|C1'(v)| = min(r_i, b_i)$ e $|C2'(v)| = |r_i - b_i|$.

4.3 Prova de Correção

Para ser provada a correção do protocolo VH+, segundo o critério de *one-copy serializability*, deve-se mostrar que os quorums de acesso formados para operações conflitantes (leitura/escrita-cega, leitura/escrita e escrita/escrita) possuem pelo menos uma cópia do dado replicado em comum. Isto é feito através da prova dos dois lemas seguintes.

Lema 4.1 *Em uma hierarquia qualquer (completa ou incompleta) de vértices de m níveis, o protocolo VH+ garante que qualquer quorum de leitura e qualquer quorum de escrita-cega, cujos valores para cada nível i ($i = 1, 2, \ldots, m$) da hierarquia são r_i e b_i, respectivamente, definidos satisfazendo a restrição 4.2, têm pelo menos um vértice folha (ou seja, uma cópia do dado replicado) em comum.*

Prova. Por indução no número de níveis da hierarquia.

Base. O lema é válido para uma hierarquia de um nível pois nela todos os vértices folhas são filhos do vértice raiz, e, por 1.2, quaisquer quorums de leitura e de escrita-cega formados a partir dos filhos do vértice raiz têm pelo um desses filhos em comum.

Hipótese. O lema vale para hierarquias de até $m-1$ níveis.

Passo. Deve ser mostrado que o lema continua válido para uma hierarquia de m níveis.

Uma hierarquia de m níveis é composta por um vértice raiz r cujos filhos podem ser tanto vértices folhas quanto vértices não-folhas. Os vértices não-folhas podem ser vistos como os vértices raízes de outras (sub-)hierarquias de até $m-1$ níveis. Por 1.2, quaisquer quorums de leitura e escrita-cega formados a partir dos filhos de r têm pelo menos um desses filhos em comum. Se o filho de r comum aos dois quorums é um vértice folha, então o lema é válido. Se esse filho comum é um vértice não folha, os quorums de leitura e escrita-cega formados na hierarquia representada por ele devem ter pelo menos um vértice folha (dessa sub-hierarquia) em comum, o que é garantido pela hipótese de indução; portanto, nesse segundo caso o lema também é válido. □

Lema 4.2 *Em uma hierarquia qualquer de vértices de m níveis, o protocolo VII+ garante que quaisquer dois quorums de escrita têm pelo menos um vértice folha em comum.*

Prova. Como, para cada nível da hierarquia, um quorum de escrita contém tanto um quorum de leitura quanto um quorum de escrita-cega (conforme descrito na seção 4.2.1), e qualquer quorum de leitura tem uma interseção não-nula com qualquer quorum de escrita-cega, a prova deste lema é análoga à prova do lema anterior. □

4.4 Disponibilidade do Dado

Nas hierarquias completas, a disponibilidade do dado replicado para operações de leitura e escrita-cega no protocolo VII+ é calculada recursivamente, nível a nível, similarmente ao cálculo da disponibilidade do dado no protocolo VII. A seguinte recorrência é usada para calcular a disponibilidade do dado para uma operação op (leitura ou escrita-cega), em relação a um vértice v pertencente ao nível i da hierarquia, quando utilizam-se hierarquias completas (considera-se que cada cópia está acessível com probabilidade ρ):

$$A_{op}(v,i) = \begin{cases} \rho & (i=0) \\ AV(l_i, op_i, A_{op}(c_v[k], i-1)) & (i>0) \end{cases}$$

onde AV é a disponibilidade do dado no protocolo de votação tradicional (descrita juntamente com o cálculo da disponibilidade do dado no protocolo VII, na seção

3.2.3); e $c_v[k]$ $(1 \leq k \leq n_v)$ é qualquer filho de v, uma vez que, nas hierarquias completas, a disponibilidade é a mesma para todos os vértices de um mesmo nível.

A disponibilidade para operações de escrita é calculada de maneira semelhante, mas considerando-se agora tanto o quorum de leitura quanto o quorum de escrita-cega definidos para cada nível da hierarquia. Essa disponibilidade, em relação a um vértice v de um nível i, é dada pela diferença entre a probabilidade de se formar o maior quorum definido para o nível i $(max(r_i, b_i))$ e a probabilidade de se formar apenas esse (maior) quorum, isto é, a probabilidade de não serem obtidas permissões de acesso para escrita do número mínimo exigido $(min(r_i, b_i)$ de filhos de v. A recorrência a seguir expressa a disponibilidade do dado para operações de escrita no protocolo VII+, quando são consideradas apenas hierarquias completas.

$$A_w(v,i) = \begin{cases} \rho & (n_v = 0) \\ A_r(v,i) - AV(l_i, r_i, A_r(c_v[k], i-1) - A_w(c_v[k], i-1)) & \\ & (n_v > 0 \ \ e \ \ r_i \geq b_i) \\ A_b(v,i) - AV(l_i, b_i, A_b(c_v[k], i-1) - A_w(c_v[k], i-1)) & \\ & (n_v > 0 \ \ e \ \ r_i < b_i) \end{cases}$$

Nas hierarquias incompletas, o cálculo da disponibilidade do dado torna-se mais complexo (não sendo possível o uso do cálculo da disponibilidade do dado no protocolo de votação tradicional para simplificar as recorrências) pois não necessariamente a disponibilidade do dado é a mesma para vértices pertencentes a um mesmo nível. Para contornar essa dificuldade, inerente às hierarquias incompletas, adota-se uma estratégia semelhante àquela utilizada no cálculo do tamanho dos quorums. Assim, a disponibilidade do dado para uma operação *op* (leitura ou escrita-cega) no protocolo VII+, quando a estrutura utilizada é uma hierarquia qualquer, é dada pela recorrência:

$$A_{op}(v,i) = \begin{cases} \rho & (n_v = 0) \\ 0 & (n_v < op_i) \\ \sum_{|C'(v)| \geq op_i} [\prod_{c \in C(v)} A_{op}(c, i-1) * \prod_{d \notin C(v)} (1 - A_{op}(d, i-1))] & \\ & (n_v \geq op_i) \end{cases}$$

onde $C'(v)$ é um conjunto formado por $\mid C(v) \mid$ vértices filhos de v; e $\mid C(v) \mid \geq op_i$ representa todas as combinações possíveis de op_i ou mais filhos de v. Note que não é relevante o fato de os vértices filhos de v serem ou não deficientes; isto acontece porque a disponibilidade relativa a um vértice deficiente (segundo caso da recorrência) é sempre 0.

Similarmente, a disponibilidade do dado para operações de escrita, utilizando-se uma hierarquia qualquer, é dada por:

$$A_w(v,i) = \begin{cases} \rho & (n_v = 0) \\ 0 & (n_v < max(r_i, b_i)) \\ \\ A_r(v,i) - \sum_{|Y'(v)| \geq r_i} [\prod_{c \in C(v)} (A_r(c,i-1) - A_w(c,i-1)) \\ \qquad * \prod_{d \notin C(v)} (1 - A_r(d,i-1) + A_w(d,i-1))] \\ \qquad (n_v \geq max(r_i, b_i) \ \text{ e } \ r_i \geq b_i) \\ \\ A_b(v,i) - \sum_{|Y'(v)| \geq b_i} [\prod_{c \in C(v)} (A_b(c,i-1) - A_w(c,i-1)) \\ \qquad * \prod_{d \notin C(v)} (1 - A_b(d,i-1) + A_w(d,i-1))] \\ \qquad (n_v \geq max(r_i, b_i) \ \text{ e } \ r_i < b_i) \end{cases}$$

Como no protocolo VII, a disponibilidade do dado para as operações de leitura, escrita-cega e escrita no protocolo VII+, em relação a toda a hierarquia, é calculada passando-se como argumento da devida recorrência o vértice raiz.

4.5 Análise Comparativa

Esta seção compara o protocolo VII+ com os protocolos baseados em estruturas lógicas revisados no capítulo 3. Primeiramente, são considerados os protocolos VII, de quorum em grade e de quorum em grade hierárquica, os quais utilizam estruturas lógicas simétricas e podem ser generalizados pelo protocolo VII+ através de hierarquias completas. Na seqüência, é considerado o protocolo de quorum em árvore, que utiliza estruturas lógicas assimétricas e tem vários casos particulares generalizados pelo protocolo VII+ através de hierarquias incompletas.

4.5.1 Estruturas Lógicas Simétricas

Apesar de não existir a restrição $2.b_i > l_i$ para cada nível i da hierarquia de vértices no protocolo VII+, o que resultaria em uma intersecção não-nula entre quaisquer dois quorums de escrita-cega, nada impede que ela seja satisfeita -- desde que seja mantida, é claro, a restrição 4.2. Para isso, é necessário apenas definir os valores de r_i satisfazendo a restrição

$$r_i < l_i/2 + 1 \tag{4.3}$$

Nesse caso, todo quorum de escrita-cega pode ser usado como quorum de escrita; se forem consideradas apenas hierarquias completas, que são estruturas simétricas, verifica-se claramente que o protocolo VII+, com essa restrição adicional, corresponde exatamente ao protocolo VII original.

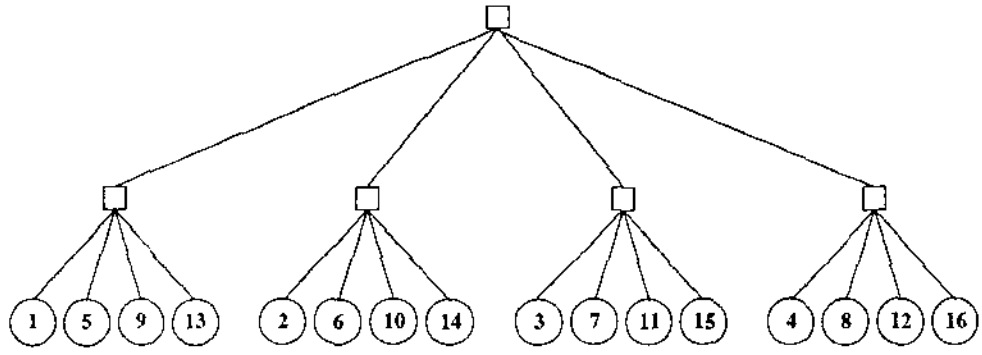

Figura 4.4: 16 cópias organizadas em uma hierarquia de 2 níveis.

O protocolo de quorum em grade é generalizado tratando-se as colunas da grade como vértices lógicos filhos do vértice raiz da hierarquia; cada um desses vértices filhos do vértice raiz, por sua vez, representa um grupo de vértices folhas que correspondem às cópias dentro de cada coluna da grade. Seja G uma grade de dimensões $x \times y$. Construindo-se uma hierarquia de vértices completa onde $m = 2$, $\mathbf{l}=(x, y)$ e $\mathbf{r}=(1, y)$, o protocolo VII+ gera os mesmos quorums gerados pelo protocolo de quorum em grade utilizando a grade G. A figura 4.4 mostra uma hierarquia de dois níveis do protocolo VII+ com $\mathbf{l}=(4,4)$ a qual, fazendo-se $\mathbf{r}=(1,4)$, corresponde à grade do protocolo de quorum em grade mostrada na figura 3.1. Esta generalização explica-se pois o processo de formação de quorums do protocolo de quorum em grade é, na verdade, uma votação em dois níveis: no nível de cima, vota-se pelas colunas da grade (um quorum de leitura necessita de todas as colunas e um quorum de escrita-cega necessita de apenas uma); no nível de baixo, a votação acontece entre as cópias de cada coluna (um quorum de leitura necessita de uma cópia e um quorum de escrita-cega necessita de todas); um quorum de escrita é formado então pela união de um quorum de leitura com um quorum de escrita-cega. É interessante ressaltar que, como pode ser percebido a partir desta generalização, na estrutura lógica de grade é irrelevante a informação sobre a quais linhas pertencem as cópias, sendo útil apenas a posição de cada uma em relação às colunas.

A generalização do protocolo de quorum em grade é facilmente estendida para o protocolo de quorum em grade hierárquica. Para cada nível da estrutura de grade hierárquica são construídos, conforme descrito no parágrafo anterior, dois níveis na hierarquia do protocolo VII+. Seja HG uma grade hierárquica de k níveis de dimensões $x_i \times y_i$ $(1 \leq i \leq k)$. Construindo-se uma hierarquia de vértices completa onde $m = 2.k$, $\mathbf{l}=(x_1, y_1, x_2, y_2, \ldots, x_k, y_k)$ e $\mathbf{r}=(1, y_1, 1, y_2, \ldots, 1, y_k)$, o protocolo VII+ gera os mesmos quorums gerados pelo protocolo de quorum em grade hierárquica utilizando a grade hierárquica HG. A figura 4.5 mostra uma hierarquia de quatro níveis com $\mathbf{l}=(2,2,2,2)$ a qual, fazendo-se $\mathbf{r}=(1,2,1,2)$, corresponde a

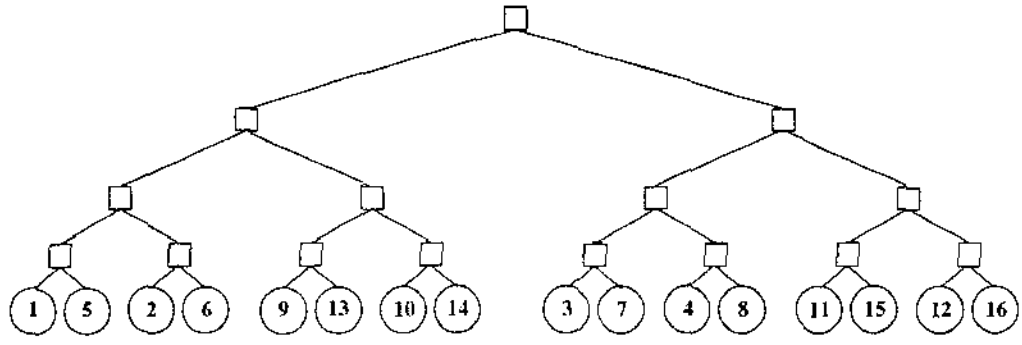

Figura 4.5: 16 cópias organizadas em uma hierarquia de 4 níveis.

grade hierárquica do protocolo de quorum em grade hierárquica mostrada na figura 3.5.

Para mostrar a superioridade do protocolo VII+ em relação aos três protocolos generalizados até aqui, faz-se agora uma comparação entre os protocolos com respeito ao número de cópias e ao tamanho dos quorums necessários para que uma disponibilidade do dado mínima seja alcançada. Essa comparação é possível uma vez que os três protocolos generalizados, bem como as configurações do protocolo VII+ que os generalizam, utilizam estruturas lógicas simétricas para formar os quorums. É tomada como referência uma comparação similar feita em [CAA90] entre o protocolo de votação tradicional e o protocolo de quorum em grade, a qual é descrita em seguida.

Os protocolos de votação tradicional[2] e de quorum em grade são comparados em [CAA90] com respeito ao número total de cópias do dado replicado e ao tamanho dos quorums de leitura e escrita necessários para que seja alcançada uma disponibilidade do dado mínima de 0.999999 para operações de leitura, e de 0.9955 para operações de escrita, considerando-se que **(a)** cada cópia está acessível com probabilidade 0.95 e **(b)** o dado replicado é alterado 20% das vezes em que é lido. O resultado da comparação mostra que, para alcançar tão alta disponibilidade, o protocolo de votação tradicional requer que o dado replicado tenha pelo menos 10 cópias ($q_r = 4$ e $q_w = 7$), enquanto o protocolo de quorum em grade requer no mínimo 30 cópias organizadas em uma grade de dimensões 6×5 ($q_r = 5$ e $q_w = 10$). Por outro lado, é constatado em [CAA90] que o protocolo de quorum em grade, por permitir uma menor relação entre os tamanhos dos quorums e o número total de cópias, realiza a distribuição da carga de trabalho com muito mais eficiência do que o protocolo de votação tradicional, provendo, portanto, operações de acesso ao dado replicado com tempos de

[2]Para que as cópias tenham a mesma responsabilidade no controle do dado replicado, é adotada no protocolo de votação tradicional a atribuição de um voto por cópia.

resposta significativamente menores. Constata-se ainda em [CAA90] que essa vantagem do protocolo de quorum em grade permanece mesmo aumentando-se o número de cópias no protocolo de votação tradicional.

Incluindo-se nessa mesma comparação o protocolo VH, o protocolo de quorum em grade hierárquica, e o protocolo VH+ utilizando apenas hierarquias completas, verifica-se que os protocolos VH e VH+ alcançam a disponibilidade mínima exigida já a partir de 14 cópias, enquanto que o protocolo de quorum em grade hierárquica somente a atinge com 30 cópias (organizadas em uma grade hierárquica de um nível de dimensões 6×5), exatamente quando ele corresponde ao protocolo de quorum em grade original de apenas um nível. A tabela 4.1 mostra todos os casos, para os protocolos VH, VH+, de quorum em grade e de quorum em grade hierárquica, com até 30 cópias, nos quais a disponibilidade do dado mínima exigida na comparação anterior é alcançada com a melhor relação entre os tamanhos dos quorums e o número total de cópias.

Note que a relação entre os tamanhos dos quorums e o número total de cópias obtida por todos os protocolos da tabela 4.1 é melhor do que a melhor relação que poderia ser obtida, para o mesmo número de cópias, no protocolo de votação tradicional. Note também que, ainda na tabela 4.1, as definições de quorums para cada nível das hierarquias do protocolo VH+ não poderiam ser usadas nas hierarquias do protocolo VH uma vez que em pelo menos um dos níveis a restrição 4.3 não é satisfeita. É válido salientar, por fim, que o protocolo VH+, por possuir maior flexibilidade na definição dos quorums de cada nível, obtém quorums de acesso de tamanhos menores que os tamanhos dos quorums obtidos pelo protocolo VH original.

4.5.2 Estruturas Lógicas Assimétricas

Para que o protocolo de quorum em árvore seja generalizado pelo protocolo VH+, a hierarquia de vértices deve ser construída de modo que os vértices folhas (correspondentes às cópias) tenham prioridades diferentes na formação dos quorums, como acontece na estrutura (assimétrica) de árvore. Isto é feito através de hierarquias incompletas, distribuindo-se os vértices folhas em níveis diferentes conforme descrito a seguir.

Seja t um árvore de altura h e grau d do protocolo de quorum em árvore; esta árvore t é composta por uma raiz que possui d filhos os quais, por sua vez, são as raízes de outras d sub-árvores de altura $h-1$ e grau também d. A hierarquia de vértices do protocolo VH+ equivalente a árvore t é construída criando-se dois filhos para o vértice raiz da hierarquia: um filho como um vértice folha correspondente à raiz de t, e outro filho como um vértice não-folha com d filhos: os filhos desse vértice não-folha constituem os vértice raízes de outras d sub-hierarquias, as quais

No. de Cópias	Protocolo	l_i			r_i			qr	qw
		l_1	l_2	l_3	r_1	r_2	r_3		
14	VH	7	2		4	1		4	8
	VH+	7	2		2	2		4	8
16	VH	4	4		2	2		4	9
	VH+	4	4		3	1		3	9
18	VH	3	3	2	2	2	1	4	8
	VH+	2	3	3	1	3	1	3	8
20	VH	5	4		2	2		4	12
	VH+	4	5		3	1		3	11
22	VH	11	2		4	1		4	16
	VH+	2	11		2	2		4	12
24	VH	4	3	2	2	2	1	4	12
	VH+	3	8		3	1		3	10
25	VH	5	5		3	2		6	12
	VH+	5	5		4	1		4	12
26	VH	13	2		5	1		5	18
	VH+	13	2		3	2		6	14
27	VH	3	3	3	2	2	1	4	12
	VH+	3	9		3	1		3	11
28	VH	7	4		4	1		4	16
	VH+	2	7	2	2	1	2	4	10
30	VH	5	3	2	3	2	1	6	12
	Grade	6	5		1	5		5	10
	Grade Hierárquica	6	5		1	5		5	10
	VH+	3	10		3	1		3	12

Tabela 1.1: Números de cópias e configuração da estrutura lógica que propiciam uma disponibilidade mínima exigida para os protocolos baseados em estruturas lógicas simétricas.

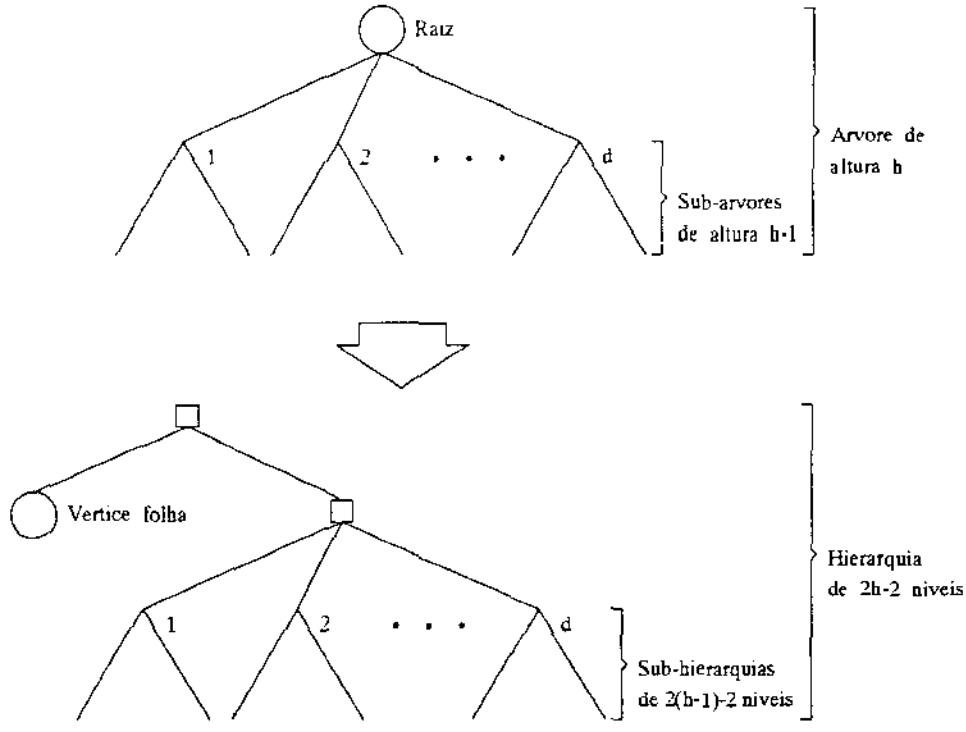

Figura 4.6: Esquema de transformação de uma estrutura de árvore em uma hierarquia de vértices.

são recursivamente construídas a partir das d sub-árvores de altura $h-1$, cujas raízes são os filhos da raiz de t (ver figura 4.6). Ao final desse processo recursivo de construção, a hierarquia de vértices resultante, equivalente à árvore t de altura h, terá $2h-2$ níveis com $\mathbf{l}=(d,2,d,2\ldots,d,2)$. A figura 4.7 mostra uma hierarquia de vértices do protocolo VH+ equivalente à árvore do protocolo de quorum em árvore mostrada na figura 3.6.

O vértice raiz de uma hierarquia de vértices construída desta maneira possui sempre dois filhos: um vértice folha u e um vértice não-folha v com l_{m-1} filhos. Nessa hierarquia, uma operação op que tenha o quorum do nível m igual a um ($op_m = 1$) já pode ser realizada com a permissão apenas da cópia correspondente ao vértice u; caso essa permissão não seja obtida, u pode ser substituído por v, o qual, para dar a permissão de acesso, deve antes recebê-la de pelo menos op_{m-1} dos seus l_{m-1} filhos. Similarmente, uma outra operação op com $op_m = 2$, para ser realizada, deve receber a permissão do vértice u e também do vértice v, essa última novamente significando a permissão de pelo menos op_{m-1} filhos de v. Desse modo, um quorum do protocolo de quorum em árvore de dimensões $< 1, w >$[3] ($1 \leq w \leq d$, com d sendo o grau da

[3]Um quorum no protocolo de quorum em árvore é definido com respeito a duas dimensões

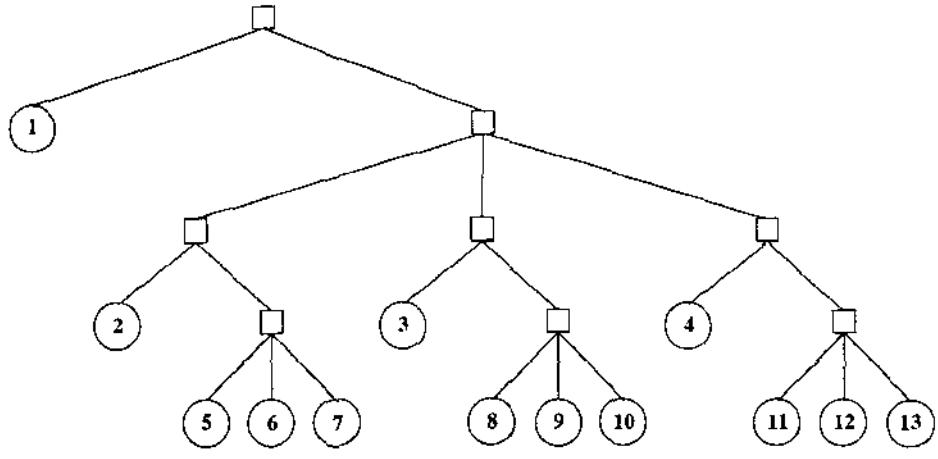

Figura 4.7: Uma hierarquia de vértices construída a partir de uma estrutura de árvore.

estrutura de árvore) definido para uma operação $op1$ pode ser obtido no protocolo VII+ fazendo-se $op1 = (w, 1, w, 1, \ldots, w, 1)$, e priorizando-se, durante o pedido das permissões de acesso, os vértices filhos que representam as menores sub-hierarquias (esse vértices podem ser identificados através da função Tot). Uma operação $op2$ que conflite com a operação $op1$ deve ter, novamente no protocolo de quorum em árvore, um quorum de dimensões $< h, d - w + 1 >$ (com h sendo a altura da estrutura de árvore); da mesma maneira, esse quorum para $op2$ é obtido no protocolo VII+ fazendo-se $op2 = (d - w + 1, 2, d - w + 1, 2, \ldots, d - w + 1, 2)$. Observe que, por não poderem prescindir do recebimento da permissão de acesso do vértice folha u (a raiz da estrutura de árvore do protocolo de quorum em árvore), quaisquer dois quorums formados para a operação $op2$ sempre têm pelo menos um vértice folha em comum.

Claramente, o quorum de dimensões $< 1, w >$ pode ser usado no protocolo de quorum em árvore para operações de leitura, o que equivaleria a $\mathbf{r}=(w, 1, w, 1, \ldots, w, 1)$ no protocolo VII+. Esses valores do vetor r resultariam, ainda no protocolo VII+, no vetor $\mathbf{b}=(d - w + 1, 2, d - w + 1, 2, \ldots, d - w + 1, 2)$ para as operações de escrita-cega, equivalendo ao quorum de escrita de dimensões $< h, d - w + 1 >$ no protocolo de quorum em árvore; como quaisquer dois quorums de escrita-cega do protocolo VII+, sob esse esquema, têm pelo menos uma cópia em comum, todo quorum de escrita-cega pode naturalmente ser usado como quorum de escrita. Portanto, estabelece-se assim uma equivalência total entre o protocolo VII+ e o todos os casos particulares do protocolo de quorum em árvore que utilizam quorums de dimensões $< 1, w >$ ou $< h, w >$ $(1 \leq w \leq d)$.

$< l, w >$: l indica o número de níveis da estrutura de árvore e w indica o número de filhos de cada vértice (ver seção 3.4.1).

Quorums do protocolo de quorum em árvore que exijam um número k ($1 < k < h$) de níveis da árvore não possuem equivalência no protocolo VH+ porque, para isso, seria necessário que os quorums das operações de leitura e escrita-cega, r_i e b_i, dos níveis i ($i = m, m-2, m-4, \ldots, 4, 2$), que contém apenas um vértice folha e outro não-folha, tivessem seus valores alterados dinamicamente (de 1 para 2 ou vice-versa, dependendo de quais cópias tenham dado ou não a permissão de acesso para a operação requisitada) durante o processo de construção do quorum. Tal alteração dinâmica das componentes dos vetores r e b não é possível no protocolo VH+. Esta limitação, entretanto, não tira o mérito da generalização "parcial" conseguida pelo protocolo VH+ sobre o protocolo de quorum em árvore, uma vez que todas as possibilidades de quorum do protocolo de quorum em árvore que conseguem realizar uma operação de leitura contactando apenas uma cópia, incluídos os protocolos QA1 e QA2 (destacados em [AA92b] e descritos na seção 3.4.4), podem ser facilmente generalizados pelo protocolo VH+.

Além da generalização, uma outra grande vantagem do protocolo VH+ em relação a esses casos particulares do protocolo de quorum em árvore é a maior flexibilidade para definir quorums. No protocolo de quorum em árvore, um quorum pode ser definido com respeito ao número de níveis da estrutura e ao número de filhos de cada vértice. No protocolo VH+, embora o número de níveis esteja restrito a *um* ou *todos*, o número de filhos de cada vértice que devem ser incluídos no quorum é definido individualmente para cada nível da hierarquia — contrastando com o protocolo de quorum em árvore, onde esse número é fixo para todos os níveis da estrutura. Um exemplo prático desta vantagem do protocolo VH+ é dado em seguida.

Na árvore mostrada na figura 3.6, apenas dois casos de quorums podem ser usados pelo protocolo de quorum em árvore para ser conseguido, no melhor caso, um quorum de leitura formado por apenas uma cópia: $q_r = <1,2>$ e $q_w = <3,2>$ (protocolo QA1), e $q_r = <1,3>$ e $q_w = <3,1>$ (protocolo QA2). No protocolo QA1, o tamanho do quorum de leitura varia de 1 a 4 cópias, e o tamanho do quorum de escrita é fixo em 7 cópias. No protocolo QA2, o tamanho do quorum de leitura passa a variar de 1 a 9 cópias, enquanto o tamanho do quorum de escrita é reduzido para apenas 3 cópias. Os resultados obtidos pelo protocolo QA1 são melhores do que os do protocolo QA1 no que se refere a operações de leitura (tanto na variação dos tamanhos dos quorums quanto na disponibilidade do dado, considerando-se, nesse segundo quesito, uma confiabilidade dos nós de 0.95); ocorre o inverso no que se refere a operações de escrita. Em ambientes onde o dado replicado é mais lido do que alterado, o protocolo QA1 é certamente mais vantajoso do que o protocolo QA2.

O protocolo VH+, utilizando a hierarquia de vértices da figura 4.7, gera exatamente os mesmos quorums gerados pelos protocolos QA1 e QA2, quando é utilizada por eles a estrutura de árvore de 3.6, fazendo-se $\mathbf{r}=(2,1,2,1)$ e $\mathbf{r}=(3,1,3,1)$, respectivamente. Note que os valores de r_i são idênticos para todos os níveis nos quais

o número máximo de filhos por vértice é igual ao grau da estrutura de árvore que originou a hierarquia ($i = 1$ e $i = 3$); isto acontece porque, como já salientado antes, o número de filhos de cada vértice a ser incluído em um quorum é fixo para todos os níveis da estrutura do protocolo de quorum em árvore. Uma vez que essa restrição não existe no protocolo VII+, os valores dos níveis i ($i = m-1, m-2, \ldots, 3, 1$) podem ser definidos diferenciadamente, suscitando novas possibilidades de formação de quorums. Por exemplo, fazendo-se, ainda na hierarquia da figura 4.7, r=(1, 1, 3, 1) consegue-se um quorum de leitura de tamanho variando entre 1 e 3 cópias, e um quorum de escrita-cega (que também pode ser usado para escrita) de tamanho fixo em 5 cópias; a disponibilidade resultante é aproximadamente igual à obtida pelo protocolo QA1. Portanto, em comparação ao protocolo QA1, essa alteração dos valores de r_1 e r_3 possibilita ao protocolo VII+ formar quorums de tamanhos menores (ainda mantendo, no melhor caso, um quorum de leitura formado por apenas uma cópia) com praticamente a mesma disponibilidade do dado. Por fim, é interessante ressaltar que esses tamanhos de quorums obtidos pelo protocolo VII+ na hierarquia da figura 4.7 não podem ser obtidos pelo protocolo de quorum em árvore na estrutura de árvore da figura 3.6 com nenhuma combinação de dimensões para os quorums.

Notas Bibliográficas

Durante o desenvolvimento da dissertação, foi estudada a organização lógica das cópias do dado replicado em outras estruturas alternativas; por exemplo, uma estrutura composta por anéis [MA93a], e uma estrutura de grade onde as linhas eram tratadas como anéis [MA93c]. Verificou-se então que não só essas estruturas alternativas mas também as estruturas (grades, hierarquias e árvores) utilizadas pelos protocolos baseados em estruturas lógicas poderiam ser generalizadas por hierarquias que eliminassem ou estendessem algumas características das hierarquias tradicionais. De início, foram generalizados apenas os protocolos baseados em estruturas simétricas através de hierarquias completas [MA93b, MA94]). Posteriormente, utilizando-se hierarquias incompletas, generalizaram-se casos particulares do protocolo de quorum em árvore.

Um outro trabalho, [KM92], sobre o qual tomou-se conhecimento recentemente, propõe um novo tipo de hierarquia que também generaliza os protocolos de quorum em grade e de quorum em grade hierárquica, para isso formando um quorum de escrita através da união explícita das cópias de uma quorum de leitura com as cópias de um quorum de escrita-cega. Embora correto, esse modo de formar o quorum de escrita não é estritamente necessário para garantir a interseção entre quorums de operações conflitantes: quando os quorum de escrita-cega contêm a maioria dos filhos dos vértices em todos os níveis da hierarquia, eles podem naturalmente ser

usados como quorums de escrita sem que seja preciso uní-los a nenhum quorum de leitura. Combinando os quorums de leitura e escrita-cega um nível de cada vez, o protocolo VH+ assegura que o quorum de escrita sempre terá o tamanho mínimo necessário para garantir que quorums de operações conflitantes tenham uma interseção não-nula. Ademais, a definição de hierarquias incompletas e a conseqüente generalização de casos particulares do protocolo de quorum em árvore, até onde se sabe, são novidades e estão sendo originalmente propostas neste capítulo.

Capítulo 5

Outras Soluções

Além dos protocolos baseados em votação e em estruturas lógicas, muitas outras soluções foram (e têm sido) propostas para o problema de se manter as cópias de um dado replicado em um estado consistente. O objetivo deste capítulo é dar uma idéia generalizada sobre como algumas dessas outras soluções mantêm a consistência entre as cópias, e assim melhor situar os protocolos de controle de réplicas revisados nos capítulos anteriores, incluído o protocolo VH+, dentro do amplo espectro de soluções existentes para este problema.

A descrição de cada uma das soluções consideradas é feita de forma sucinta, sem apresentar detalhes. A próxima seção descreve soluções que utilizam conceitos genéricos para definir os quorums. Dois protocolos otimistas são descritos na seção 5.2. Finalmente, a seção 5.3 enumera alguns protocolos que adotam estratégias não tão "convencionais" para garantir a consistência entre as cópias e ao mesmo tempo proporcionar um melhor tempo de resposta para as operações de acesso ao dado replicado.

5.1 Soluções Genéricas

Todos os protocolos que seguem a abordagem pessimista para controlar dados replicados revisados nos capítulos anteriores possuem um mecanismo para definir os conjuntos de cópias que serão usados como os quorums de acesso. A primeira solução descrita nesta seção generaliza os protocolos pessimistas eliminando a necessidade de tal mecanismo e presumindo que os quorums são previamente definidos. A segunda solução apresenta uma maneira de implementar a idéia genérica da primeira solução através de votação entre as cópias.

5.1.1 *Coteries*

Esta solução [GMB85] propõe que os conjuntos (ou grupos, para evitar confusão mais adiante) de cópias candidatos a quorums de acesso ao dado replicado sejam explicitamente definidos *a priori* pelo projetista do sistema, tornando-se desnecessário, portanto, qualquer mecanismo (como votação ou estruturas lógicas) para designar sobre quais cópias a operação requisitada será fisicamente realizada[1]. Ao conjunto formado pelos grupos de cópias candidatos a quorum denomina-se uma *coterie*[2]. Uma definição formal de uma *coterie* é dada em seguida.

Seja C o conjunto de cópias do dado replicado. Um conjunto de grupos de cópias S é uma *coterie* sobre C se e somente se

(i) $G \in S$ implica que $G \neq \emptyset$, e $G \subseteq C$.

(ii) Se $G, H \in S$ e G é candidato a quorum de leitura e H é candidato a quorum de escrita, ou G e H são ambos candidatos a quorums de escrita, então G e H devem ter pelo menos uma cópia em comum.

(iii) Não há dois grupos $G, H \in S$ tais que $G \subset H$.

A propriedade (i) restringe os candidatos a quorums ao conjunto C. A propriedade (ii) garante a interseção entre os quorums exigida pelo critério de *one-copy serializability*. Por fim, a propriedade (iii) reduz o tamanho dos grupos candidatos a quorum eliminando grupos que são super-conjuntos de outros grupos já incluídos na *coterie*. Como exemplo, imagine um dado replicado com cinco cópias representadas pelo conjunto C={1, 2, 3, 1, 5, 6}. Uma possível *coterie* sobre C, entre muitas outras, seria os conjuntos R={{1, 3, 5}, {2, 4, 6}} e W={{1, 2, 3, 4}, {3, 1, 5, 6}, {2, 3, 5, 6}}, com R e W representando os grupos de cópias candidatos a quorums de leitura e escrita, respectivamente.

Uma vez definida uma *coterie* para o conjunto de cópias do dado replicado, a cada operação de acesso requisitada o protocolo de controle das réplicas simplesmente solicita a devida permissão de um grupo de cópias pertencente à *coterie* e que seja candidato a quorum da operação requisitada. Se todas as cópias desse grupo dão a permissão pedida, a operação é realizada e a consistência entre as cópias fica garantida. Caso uma ou mais cópias do grupo não dê essa permissão ou não consiga ser

[1] Esta idéia foi inicialmente proposta visando obter exclusão mútua entre os nós de um sistema distribuído em [Lam78], mas, como pode ser constatado, ela pode ser facilmente estendida para o controle de dados replicados.

[2] Em francês, *coterie* significa "um círculo fechado de amigos que compartilham um interesse comum".

contactada, o protocolo tenta um outro grupo de cópias candidato ao quorum desejado, possivelmente aproveitando as permissões de acesso já recebidas anteriormente. Este processo é repetido até que um quorum seja formado ou não haja mais grupos candidatos disponíveis na *coterie*, esse último caso resultando no cancelamento da operação.

É fácil ver que todos quorum gerados pelo mecanismo de votação, que satisfazem o critério de *one-copy serializability*, constituem uma *coterie*. O contrário, porém, não é verdadeiro. Ou seja: há *coteries* para as quais não existe uma atribuição de votos equivalente no protocolo de votação tradicional (em [GMB85] é apresentada uma elegante prova desta característica das *coteries*; provas semelhantes podem ser usadas para mostrar que as *coteries* geradas pelos protocolos baseados em estruturas lógicas também não possuem uma atribuição de votos equivalente).

O número total de possíveis grupos de cópias candidatos a quorums em uma *coterie* é exponencial no total de cópias do dado replicado. Apesar disso, em [GMB85] os autores alegam que, para um número pequeno de cópias (até 5), é perfeitamente factível enumerar todas as possibilidades de *coteries* para o sistema e escolher aquela que propicia a maior disponibilidade para o dado replicado — com tal *coterie* podendo inclusive ser uma das que não poderiam ser geradas pelo protocolo de votação tradicional. Com mais do que 5 cópias, o número de *coteries* possível explode e enumerá-las torna-se inviável. Em [TN89], é proposto um esquema similar à idéia das *coteries* mas no qual a procura pelo melhor conjunto de grupos de cópias candidatos a quorums é formulada através de um problema de programação linear esparsa, o que viabiliza a procura pela melhor *coterie* para até 10 cópias.

5.1.2 Votação Multidimensional

O mecanismo de votação é um método simples e fácil de gerar quorums de acesso a um dado replicado. Entretanto, conforme visto no esquema de *coteries*, nem todos os quorums possíveis podem ser gerados através de votação. O esquema de votação multidimensional (ou votação MD) [AAC91] é apresentado como um método tão genérico quanto o esquema de *coteries* para a geração de quorums, e ao mesmo tempo simples e de fácil implementação como o mecanismo de votação tradicional (ou, a partir de agora, votação 1D). Neste esquema, os votos atribuídos às cópias e os quorums exigidos pelas operações de acesso ao dado são vetores k-dimensionais. Cada dimensão da atribuição de votos e dos quorums é similar à votação 1D, mas a flexibilidade de as exigências de formação de quorums nas diferentes dimensões poderem ser combinadas de variadas maneiras torna a votação MD mais poderosa do que a votação 1D tradicional.

Considere uma dado replicado com N cópias. Na votação MD, a atribuição de

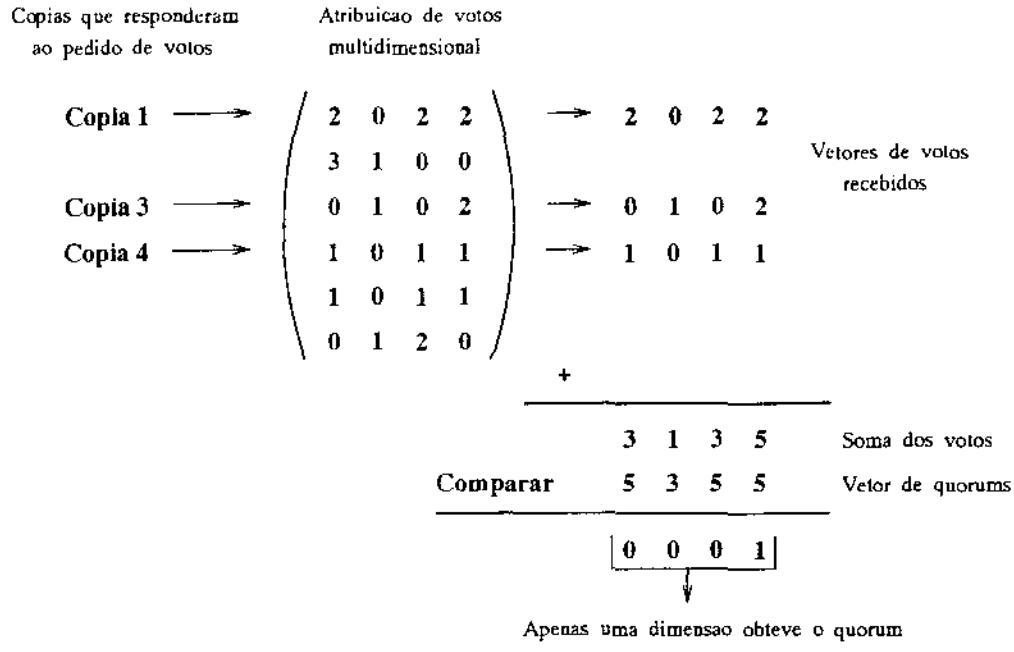

Figura 5.1: Exemplo do funcionamento do esquema de votação multidimensional.

votos $V_{N,k}$ feita às cópias do dado é uma matriz de dimensões $N \times k$ onde $v_{i,j}$ representa os votos do nó i na dimensão j ($v_{i,j} \geq 0$; $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, k$). Os quorums exigidos para cada dimensão por uma operação de acesso ao dado compõem um vetor $q_k = (q_1, q_2, \ldots, q_k)$ ($q_j > 0$; $j = 1, 2, \ldots, k$). É definido ainda um número l ($1 \leq l \leq k$) que indica o número de dimensões nas quais os quorums exigidos devem ser obtidos para que a respectiva operação de acesso possa ser realizada. Portando, há dois níveis de exigências na votação MD: votos e dimensões. No nível dos votos, o número de votos recebido para uma dimensão deve ser maior que ou igual ao quorum exigido nessa dimensão. No nível das dimensões, o número de dimensões que obtêm os quorums necessários deve ser maior ou igual a l. A exigência de que quorums sejam obtidos em l dentre um total de k dimensões é denotada pelo par (l,k). Note que (1,1) equivale à exigência presente na votação 1D.

A figura 5.1 mostra um exemplo do funcionamento da votação MD para um dado replicado com seis cópias. Nele, supõe-se que, em resposta a um pedido de acesso ao dado, apenas os votos das cópias 1, 3 e 4 são recebidos. Esses três vetores são somados e o vetor soma é então comparado com o vetor de quorums. A comparação é feita individualmente por dimensão e os resultados 1 ou 0 representam suficiência ou deficiência de votos, respectivamente. Nesse exemplo, apenas uma dimensão obteve o quorum necessário. Uma operação com exigência (1,1), portanto, poderia ser normalmente realizada. Operações com exigências (l,1) não poderiam ser realizadas

quando l=2, 3, e 4.

Agora mostra-se que qualquer *coterie* pode ser gerada por votação MD. Seja N o número de cópias do dado e k o número de grupos candidatos a quorums da *coterie*. A votação MD equivalente a essa *coterie* é montada da maneira que segue. Estabelece-se uma dimensão de votos para cada grupo da *coterie* — os vetores de votos, portanto, são k-dimensionais. Atribui-se para cada cópia i na dimensão j ($i = 1, 2, \ldots, N$ e $j = 1, 2, \ldots, k$) um voto se a cópia i pertence ao grupo correspondente à dimensão j, ou zero votos se ele não pertence a esse grupo. Define-se o vetor de quorums de modo que o quorum de cada dimensão seja igual ao número de cópias pertencentes ao grupo correspondente a essa dimensão. Através da exigência $(1,k)$ a votação MD produz exatamente os mesmos quorums presentes na *coterie* dada. Isto pode ser facilmente constatado uma vez que, com a exigência $(1,k)$, será suficiente a obtenção dos quorums necessários em qualquer uma das dimensões. Em todas as dimensões, cada uma equivalendo a um grupo da *coterie*, apenas os votos recebidos das cópias pertencentes ao grupo correspondente (os únicos diferentes de zero) podem perfazer o total exigido no vetor de quorums, que por sua vez é igual ao número de cópias desse grupo. É interessante destacar que, como muitas *coteries* podem ser geradas por votação ID, não necessariamente a atribuição de votos e o vetor de quorums na votação MD equivalentes a uma *coterie* dada devem ter tantas dimensões quanto for o número total de grupos desta *coterie*.

Em [AAC91] é mostrado como as exigências da votação MD podem ser combinadas para gerar os mesmos quorums produzidos pelos protocolos baseados em estruturas lógicas[3], o que pode ser estendido também para o protocolo VH+, uma vez que todos os quorums formados por essa classe de protocolos também constituem um *coterie*. A desvantagem do esquema de votação MD, porém, está no fato de ele, por ser genérico de mais, não poder ser considerado um protocolo autônomo, mas sim um eficiente modo de implementar, através do mecanismo de votação, soluções já existentes. Para funcionar, a votação MD precisa inicialmente de uma *coterie* ou de uma estrutura lógica; nesse segundo caso, apesar de os quorums produzidos serem os mesmos que seriam formados pelo protocolo baseado em estrutura lógica correspondente, a votação MD não sabe qual a posição das cópias dentro da estrutura tomada como referência e, desse modo, não pode usar esse conhecimento para reduzir ainda mais os custos de comunicação inerentes a cada acesso feito ao dado replicado.

[3]Em [AAC91] ainda é mostrado como a votação MD pode ser usada para implementar a votação com fragmentação (ver seção 2.1.2) sem exigir que o número de fragmentos por segmento seja o mesmo para todos os nós.

5.2 Soluções Otimistas

Conforme explicado na seção 1.3.2, os protocolos de controle de réplicas considerados otimistas são aqueles que permitem, quando o sistema está particionado, o acesso às cópias de um mesmo dado replicado em mais de uma partição. Esta seção descreve dois protocolos otimistas: o primeiro apresenta estratégias para, após as falhas serem reparadas e as partições deixarem de existir, tanto detectar quanto eliminar inconsistência entre as cópias; o segundo propõe estratégias apenas para a detecção de inconsistência, não sugerindo nenhuma maneira de resolvê-la.

5.2.1 Protocolo Otimista de Davidson

Para que inconsistência entre cópias de um mesmo dado possa ser detectada e resolvida quando duas partições anteriormente existentes voltam a se reagrupar, este protocolo [Dav84] propõe que, tão logo as partições sejam detectadas, cada partição eleja um nó coordenador. Esse nó coordenador deverá construir (e manter, enquanto durar o particionamento) um grafo, denominado *grafo de precedência*, no qual os vértices correspondem às transações executadas na partição, e as arestas correspondem às relações de dependência entre essas transações. São dois os tipos de relações de dependência: "leu-de", quando uma transação leu um dado produzido por outra transação, e "leu-antes", quando uma transação leu um dado que foi posteriormente alterado por outra transação – considera-se que todo dado alterado por uma transação foi anteriormente lido pela mesma, ou seja, o conjunto dos dados que foram alterados pela transação é um subconjunto do conjunto dos dados que foram lidos por ela. A idéia é, uma vez reagrupadas as partições, usar os grafos de precedência construídos para detectar e resolver, caso haja, inconsistência entre as cópias. Isto é feito da maneira descrita a seguir.

Após a comunicação entre as partições ser restabelecida, um dos dois nós coordenadores constrói um grafo de precedência global juntando os dois grafos de precedência de cada uma das duas partições anteriores. A esse grafo global são adicionadas arestas, denominadas *arestas de interferência*, correspondentes à relação de dependência entre transações de diferentes partições. Uma aresta de interferência representa o fato que se uma transação leu um dado replicado em uma partição, ela deve preceder qualquer outra transação que tenha alterado o valor desse mesmo dado em outra partição. Inconsistência entre as cópias é então detectada pela existência de ciclos no grafo de precedência global. A eliminação da inconsistência detectada dá-se pela quebra de todos os ciclos do grafo através do cancelamento de uma ou mais transações correspondentes aos vértices que compõem os ciclos – a validação (*commit*) das transações executadas durante o particionamento do sistema, que garante a persistência das alterações dos dados feitas por elas, é desse modo postergada

até o reagrupamento das partições. Transações canceladas podem ser automaticamente reexecutadas ou simplesmente notificam-se os nós que deram origem a essas transações sobre os seus cancelamentos. Em seguida discute-se a questão da resolução da inconsistência entre as cópias.

Obviamente, a melhor estratégia para quebrar os ciclos do grafo de precedência global, e assim deixar as cópias em um estado consistente, é aquela que minimiza o número de transações a serem canceladas. Infelizmente, é mostrado em [Dav84] que quebrar os ciclos de um grafo minimizando o número de vértices retirados, estando associados ou não aos vértices pesos representando o custo de cancelamento das transações, é um problema NP-completo. Por outro lado, estratégias eficientes, embora nem sempre ótimas, podem ser elaborada, como a de quebrar inteligentemente primeiro os ciclos contendo apenas dois vértices e só depois quebrar os ciclos maiores.

5.2.2 Protocolo Otimista de Ramarao

A solução proposta por este protocolo de Ramarao [Ram89] para detectar inconsistência entre as cópias de um dado replicado após o reagrupamento de partições adota uma estratégia diferente da estratégia utilizada pelo protocolo de Davidson descrito na seção anterior. Neste protocolo é introduzida uma relação de precedência entre os dados, que depende de como os dados são manipulados pelas transações executadas durante o tempo em que o sistema permanece particionado, enquanto que no protocolo anterior estabelece-se uma relação de precedência entre as transações propriamente ditas. Portanto, certas informações devem ser associadas a todos os dados (ou seja, à todas as cópias do(s) dado(s) replicado(s)). A seguir descreve-se quais são essas informações e como elas são usadas para a detecção de inconsistência entre as cópias.

A cada dado d de uma partição P é associado um *conjunto-de-precedência*(d) constituído pelos nomes (ou qualquer outro tipo de identificador) de todos os dados armazenados em P que *precedem d*. Informalmente, para dois dados d_i, d_j armazenados em P é dito que d_i precede d_j se d_j é lido por uma transação T executada em P de modo que ou T altera d_i, ou T aparece estritamente depois, em alguma execução seqüencial equivalente para as transações executadas em P, de uma outra transação T' que altera d_i. Dessa forma, é mostrado em [Ram89] que quando duas partições P, P' voltam a se agrupar, as cópias dos dados replicados estarão mutuamente inconsistentes se existirem dados d_i, d_j tais que d_i precede d_j em P e d_j precede d_i em P'. A atualização dos conjuntos de precedência associados a cada dado e a posterior detecção de inconsistência entre as cópias podem ser feitas, tal como no protocolo anterior, através da escolha, para cada partição, de nós coordenadores designados

para realizar essa tarefa.

A diferença básica entre este protocolo e o protocolo de Davidson é que aqui o volume de informações mantidas e processadas para detectar inconsistência entre as cópias é proporcional à quantidade de dados armazenados em cada partição, enquanto que no protocolo de Davidson esse volume é proporcional ao número de transações executadas durante o tempo em que o sistema permanece particionado. Outra diferença está no fato de no protocolo anterior ser possível a identificação de quais transações causaram a inconsistência, enquanto que neste protocolo, embora saiba-se quais dados estão inconsistentes, não consegue-se identificar as transações que os deixaram nesse estado.

5.3 Soluções Não "Convencionais"

Todos os protocolos de controle de réplicas revisados até agora, incluídos os protocolos já descritos neste capítulo, compartilham o fato de considerarem o critério de *one-copy serializability* (1SR) como premissa básica para a manutenção da consistência mútua entre as cópias de um dado replicado. O que os diferencia, portanto, dentro de um contexto mais amplo, é apenas o modo pelo qual a consistência entre as cópias é assegurada. Esta seção enumera alguns protocolos de controle de réplicas que adotam outros critérios de correção (que não o 1SR) para manter as cópias em um estado consistente. O intuito é mostrar que soluções alternativas podem ser propostas para casos específicos e como isso torna-se possível obter resultados melhores que os normalmente obtidos pelos protocolos ditos "convencionais". Como cada novo critério de correção requer estudos minuciosos, o que está fora do escopo desta dissertação, os protocolos citados a seguir são descritos resumidamente, sem exemplos de funcionamento nem análises detalhadas.

5.3.1 Protocolo de Joseph e Birman

Nos protocolos de controle de réplicas pessimistas tradicionais, uma operação de acesso a um dado replicado deve ser realizada sobre um conjunto de cópias especificado pelo protocolo. Isto resulta em acessos com tempo de resposta muito longo, principalmente quando comparados aos acessos feitos em ambientes sem replicação, uma vez que uma operação lógica requisitada somente será considerada concluída após todas as operações físicas correspondentes terem sido realizadas sobre as cópias. Este protocolo de Joseph e Birman [JB86] implementa um esquema semelhante ao do protocolo ROWA, mas sem exigir que uma operação lógica de escrita, para ser considerada concluída, tenha que aguardar a efetivação em todas as cópias das alterações

feitas ao dado. A idéia é relaxar o nível de sincronização normalmente exigido por outros protocolos e permitir que as cópias sejam alteradas de modo assíncrono, concorrentemente com outras operações. Para isso, deve haver disponível no sistema um mecanismo de difusão atômica (*atomic broadcast*) confiável.

Uma operação lógica de escrita é realizada enviando-se para os outros nós o identificador (embutido nas mensagens do mecanismo de controle de concorrência) da operação lógica de escrita sendo realizada, e difundindo-se, através do mecanismo de difusão atômica, as alterações a serem feitas. Desse modo, não é necessário aguardar das outras cópias a confirmação de efetivação das alterações — o que fica garantido através do mecanismo de difusão atômica —, resultando em operações de escrita de baixíssimo tempo de resposta. Uma operação lógica de leitura é realizada com o acesso de qualquer cópia, preferencialmente aquela armazenada no próprio (ou mais próxima do) nó que requisitou a consulta. Em [JB86] é mostrado que, de acordo com esse esquema, as alterações feitas ao dado replicado por uma operação lógica OP no tempo lógico t_i [Lam78] em um nó S sempre chegarão a outro nó S' a tempo de serem usadas por outra operação OP' realizada no tempo lógico t_j ($t_i < t_j$). Portanto, as operação de acesso ao dado replicado sempre são realizadas sobre cópias atualizadas e com um tempo de resposta comparável ao que teriam em um ambiente sem replicação; mas ainda com as vantagens de o dado estar de fato replicado.

5.3.2 Protocolo de Kumar e Stonebraker

Este protocolo [KS88b] propõe explorar os aspectos semânticos das transações e com isso reduzir as restrições comumente impostas aos acessos a um dado replicado. O aspecto semântico adotado aqui é a propriedade de *comutatividade* pela qual certas operações aritméticas podem ser realizadas em qualquer ordem. Baseado nessa propriedade, é sugerido que as transações sejam pré-analisadas e classificadas em um desses quatro grupos: transações C, que *comutam* entre si e com quaisquer outras; transações PC, que, satisfeitas certas restrições de integridade, são executadas como transações C; transações NC1, que comutam entre si e com transações C; e transações NC2, que comutam apenas com transações C.

Durante a requisição de acesso ao dado replicado, o protocolo de controle exige quorums diferenciados para as transações de acordo com a classe a qual elas pertencem. Transações C não precisam formar quorum algum para obterem permissão de acesso ao dado, portanto executam muito mais rapidamente. Transações PC, uma vez satisfeitas as restrições de integridade previamente definidas para elas, também não precisam formar quorums. As classes NC1 e NC2, uma vez que não comutam entre si, devem formar quorums antes de cada acesso ao dado. Desse modo, considerando-se N o número de cópias do dado replicado, os quorums de acesso nc_1

e nc_2 para as transações das classes NC1 e NC2, respectivamente, devem ser definidos satisfazendo a restrição $nc_1 + nc_2 > N$. Note que a restrição $2.nc_1 > N$ não é necessária pois as transações NC1 comutam entre si.

É interessante salientar que, devido a propriedade de comutatividade, não é garantida por este protocolo uma execução seqüencial equivalente para todas as transações, o que normalmente é exigido por outros protocolos que adotam o critério 1SR. Contudo, também devido a essa mesma propriedade, é garantida por ele a consistência mútua entre as cópias do dado replicado.

5.3.3 Protocolo de Pu e Leff

Neste protocolo [PL91] é definido um novo critério de correção denominado *Epsilon-serializability* (ESR). Sob o critério ESR, duas classes de transações são definidas: transações de alteração (ou ETUs), compostas de pelo menos uma operação de escrita, e transações de consulta (ou ETQs), compostas somente de operações de leitura. ETUs são executadas seguindo o critério 1SR, enquanto que as ETQs não ficam sujeitas a quaisquer restrições quando da requisição de um acesso ao dado replicado. A idéia deste protocolo é efetivar as alterações causadas pela ETUs assincronamente, o que melhora o tempo de execução das ETUs mas, por outro lado, suscita a possibilidade de que as ETQs tenham acesso a dados desatualizados. Contudo, a eventual inconsistência entre as cópias será temporária uma vez que no final da execução das ETUs, que seguem o critério 1SR, os dados estarão em um estado consistente. Além disso, o protocolo provê meios para que o grau de inconsistência nas ETQs seja controlado de modo que a divergência entre as cópias possa ser confinada uma margem específica. Esse controle da inconsistência dá maior flexibilidade para a execução das transações sob o critério ESR: se o grau de inconsistência permitido é *nenhum*, a execução de todas as transações, ETUs e ETQs, claramente satisfaz o critério 1SR e nenhuma operação é realizada sobre cópias desatualizadas.

Em [PL91] é ainda analisado o desempenho de diferentes protocolos de controle de réplicas desenvolvidos para atuarem sob o critério ESR. Entre eles incluem-se protocolos que exploram aspectos semânticos das transações, como a propriedade de comutatividade utilizada no protocolo de Kumar e Stonebraker, e um protocolo que não restringe os acessos ao dado replicado mas mantém a consistência entre as cópias após a validação das transações, através do cancelando de transações conflitantes e da execução de transações de compensação correspondentes.

Capítulo 6

Conclusão

Após mais de quinze anos como foco de intensas pesquisas (os primeiros trabalhos na área surgiram no final da década de setenta), o problema de como controlar dados replicados em sistemas distribuídos continua sendo bastante estudado atualmente, o que leva à conclusão de que nenhuma das inúmeras soluções propostas até agora mostrou-se plenamente satisfatória. Dessas soluções, muitas estão fortemente relacionadas, algumas inclusive sendo melhorias ou extensões de outras previamente sugeridas. A maioria, entretanto, dentro de um contexto mais geral, constitui-se de protocolos distintos com respeito ao que é pressuposto e ao que é priorizado por cada um — ou seja, não existe o *melhor* protocolo de todos, mas sim o(s) protocolo(s) mais adequado(s) para determinadas circunstâncias. Esta constatação é corroborada pelos estudo e análise de um conjunto significativo de protocolos de controle de réplicas realizados nesta dissertação.

Desse modo, quando da escolha de um protocolo para controlar os dados que possam vir a ser replicados, os projetistas de um sistema distribuído devem ponderar sobre vários fatores (desde características dos dados e do sistema implementado até a natureza das transações que irão ter acesso aos dados replicados) antes de decidirem por um ou outro protocolo. A partir dos pressupostos e prioridades enfatizados por cada um dos protocolos estudados ao longo desta dissertação, conclui-se que os fatores mais importantes a serem considerados na escolha do protocolo mais adequado, não necessariamente nesta ordem, são:

- tamanho dos dados;
- número de nós do sistema;
- número necessário de cópias para cada dado replicado;
- espaço de armazenamento por nó;

- disponibilidade mínima esperada para as operações de acesso ao dado;
- grau de tolerância a falhas esperado;
- capacidade de transmissão da rede e de processamento do nós (que exigem um balanceamento da carga menos ou mais uniforme);
- relação *freqüência de ocorrência das falhas/freqüência com que o dado é alterado*;
- natureza das transações:
  - possibilidade de fácil cancelamento (não oneroso);
  - possibilidade de acesso a cópias desatualizadas;
  - relação *número de operações de leitura/número de operações de escrita*;
  - duração (transações longas ou curtas);
- número de transações processadas por unidade de tempo (*throughput*);

Seguem exemplos simples de como identificar o(s) protocolo(s) mais conveniente(s) baseando-se nos fatores enumerados acima. Se o dado replicado é muito grande, o que implica em um número de cópias pequeno, são mais indicados os protocolos de votação com testemunhas ou de votação com fragmentação; se a disponibilidade é fundamental e as falhas não são freqüentes, podem ser utilizados os protocolos dinâmicos ou os com procedimentos especiais de reconfiguração; já se as falhas são raras e as transações podem ser facilmente canceladas, é possível (e recomendada) a opção pelos protocolos otimistas.

Em sistemas distribuídos de grande porte, onde, espera-se, dados replicados terão cópias armazenadas em um número grande de nós, o nível de balanceamento da carga e a quantidade de mensagens enviadas/recebidas por cada operação realizada têm influência crucial no tempo de resposta do sistema. Nesses ambientes, portanto, será extremamente vantajoso a exigência de quorums de tamanhos reduzidos para as operações de acesso ao dado, o que pode ser obtido através dos protocolos baseados em estruturas lógicas sem o comprometimento do balanceamento da carga. Em vista disso, é destacada a vantagem do protocolo de votação hierárquica estendida (VH+) proposto nesta dissertação: ele generaliza (e supera, em muitas situações) praticamente todos os protocolos baseados em estruturas lógicas, produzindo quorums de tamanhos da ordem de $O(N)$, $O(N^{0.63})$, $O(\sqrt{N})$, $O(\log N)$, entre outros tamanhos intermediários possíveis, com N sendo o número total de cópias do dado replicado.

Como conclusão final, é interessante salientar que, apesar de utilizarem estruturas (grades, hierarquias completas e árvores) à primeira vista bastante diferentes, os

protocolos baseados em estruturas lógicas seguem um princípio básico comum: a estruturação das cópias em grupos hierarquicamente organizados. É este princípio comum, cuja descoberta é a maior contribuição desta dissertação, que permite o protocolo VH+ tratar em um único esquema genérico o protocolo de quorum em grade, de votação hierárquica, de quorum em grade hierárquica e vários casos do protocolo de quorum em árvore.

Bibliografia

[AA89a] D. Agrawal and El Abbadi A. An Efficient Solution to the Distributed Mutual Exclusion Problem. In *Proceedings of the* 8^{th} *Symposium on Principles of Distributed Computing*, pages 193–200. ACM, 1989.

[AA89b] M. Ahamad and M. H. Ammar. Performance Characterization of Quorum-Consensus Algorithms for Replicated Data. *IEEE Transactions on Software Engineering*, 15(1):492–495, April 1989.

[AA90a] D. Agrawal and A. El Abbadi. Exploiting Logical Structures of Replicated Databases. *Information Processing Letters*, 33(5):255–260, January 1990.

[AA90b] D. Agrawal and A. El Abbadi. Storage Efficient Replicated Databases. Technical Report TRCS90-5, Department of Computer Science, University of California, Santa Barbara, 1990. Published as "Reducing Storage for Quorum Consensus Algorithms", in *Proc. of the* 14^{th} *VLDB Conference*, pages 419-430, 1988.

[AA90c] D. Agrawal and A. El Abbadi. The Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data. In *Proceedings of the* 16^{th} *VLDB Conference*, pages 243–254, 1990.

[AA91] D. Agrawal and A. El Abbadi. An Efficient and Fault-tolerant Solution for Distributed Mutual Exclusion. *ACM Transactions on Computer Systems*, 9(1):1–20, February 1991.

[AA92a] D. Agrawal and A. El Abbadi. Resilient Logical Structures for Efficient Management of Replicated Data. In *Proceedings of the* 18^{th} *VLDB Conference*, pages 151–162, 1992.

[AA92b] D. Agrawal and A. El Abbadi. The Generalized Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data. *ACM Transactions on Database Systems*, 17(1):689–717, December 1992.

[AAC91] Mustaque Ahamad, Mostafa H. Ammar, and Shun Yan Cheung. Multi-dimensional Voting. *ACM Transactions on Computer Systems*, 9(4):399-431, November 1991.

[ASC85] Amr El Abbadi, Dale Skeen, and Flaviu Cristian. An Efficient, Fault-Tolerant Protocol for Replicated Data Management. In *Proceedings of the 4th Symposium on Principles of Database Systems*, pages 215-228. ACM, 1985.

[AT89] A. El Abbadi and S. Toueg. Maintaining Availability in Partioned Replicated Databases. *ACM Transactions on Database Systems*, 14(2):264-290, June 1989.

[BG81] Philip A. Berstein and Nathan Goodman. Concurrency Control in Distributed Database Systems. *ACM Computing Surveys*, 13(2):185-221, June 1981.

[BG83] Philip A. Bernstein and Nathan Goodman. The Failure and Recovery Problem for Replicated Databases. In *Proceedings of the 2nd Symposium on Principles of Distributed Computing*, pages 114-122. ACM, August 1983.

[BG84] Philip A. Bernstein and Nathan Goodman. An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases. *ACM Transactions on Database Systems*, 9(4):596-615, December 1984.

[BGM87] Daniel Barbara and Hector Garcia-Molina. The Reliability of Voting Mechanisms. *IEEE Transactions on Computers*, C-36(10):1197-1208, October 1987.

[BGMS89] Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster. Increasing Availability Under Mutual Exclusion Constraints with Dynamic Vote Reassignment. *ACM Transactions on Computer Systems*, 7(1):394-426, November 1989.

[BHG87] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Publishing Company, 1987.

[CAA89] Shun Yan Cheung, Mustaque Ahamad, and Mostafa H. Ammar. Optimizing Vote and Quorum Assignments for Reading and Writing Replicated Data. *IEEE Transactions on Knowledge and Data Egineering*, 1(3):387-397, September 1989.

[CAA90] S. Y. Cheung, M. Ammar, and M. Ahamad. The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data. In *Proceedings of the 6th International Conference on Data Engineering*, pages 438–445. IEEE, 1990.

[CD88] George F. Coulouris and Jean Dollimore. *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, 1988.

[Dav84] Susan B. Davidson. Optimism and Consistency in Partitioned Distributed Database Systems. *ACM Transactions on Database Systems*, 9(3):456–481, September 1984.

[DGMS85] Susan B. Davidson, Hector Garcia-Molina, and Dale Skeen. Consistency in Partioned Networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.

[ES83] Derek L. Eager and Kenneth C. Sevcik. Achieving Robustness in Distributed Database Systems. *ACM Transactions on Database Systems*, 8(3):354–381, September 1983.

[Gif79] D. K. Gifford. Weighted Voting for Replicated Data. In *Proceedings of the 7th Symposium on Operating Systems Principles*, pages 150–162. ACM, December 1979.

[GM82] Hector Garcia-Molina. Elections in a Distributed Computing System. *IEEE Transactions on Computers*, C-31(1):48–59, January 1982.

[GMB85] Hector Garcia-Molina and Daniel Barbara. How to Assign Votes in a Distributed System. *Journal of the ACM*, 32(4):841–860, October 1985.

[Gra78] J. N. Gray. Notes on Database Operating Systems. *Operating Systems: An Advanced Course, Lectures Notes in Computer Science*, 60:393–481, 1978.

[JB86] Thomas A. Joseph and Kenneth P. Birman. Low Cost Management of Replicated Data in Fault-Tolerant Distributed Systems. *ACM Transactions on Computer Systems*, 4(1):54–70, February 1986.

[JM87a] Sushil Jajodia and David Mutchler. Dynamic Voting. In *SIGMOD*, pages 227–237. ACM, May 1987.

[JM87b] Sushil Jajodia and David Mutchler. Enhancements to the Voting Algorithm. In *Proceedings of the 13th VLDB Conference*, pages 399–406. IEEE, 1987.

[JM90] Sushil Jajodia and David Mutchler. Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database. *ACM Transactions on Database Systems*, 15(2):230–280, June 1990.

[KC91] Akhil Kumar and Shun Yan Cheung. A High Availability $\sqrt{N}$ Hierarchical Grid Algorithm for Replicated Data. *Information Processing Letters*, 10(6):311–316, December 1991.

[KM92] Akhil Kumar and Kavindra Malik. Optimizing the Cost of Quorum Consensus through Hierarchies. Technical report, Graduate School of Management, Cornell University, 1992. Under review for ACTA Informatica.

[KS88a] Akhil Kumar and Arie Segev. Optimizing Voting-Type Algorithms for Replicated Data. *Lecture Notes in Computer Science*, 303:128–442, March 1988.

[KS88b] Akhil Kumar and Michael Stonebraker. Semantics Based Transaction Management Techniques for Replicated Data. In *SIGMOD*, pages 117–125. ACM, March 1988.

[Kum91] Akhil Kumar. Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data. *IEEE Transactions on Computers*, 40(9):996–1004, September 1991.

[Lam78] Leslie Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, 1978.

[LPS83] B. W. Lampson, M. Paul, and H. J. Siegert, editors. *Distributed Systems: Architecture and Implementation*. Spring-Verlag Berlin Heidelberg New York, 1983.

[MA93a] Nabor C. Mendonça and Ricardo O. Anido. Protocolo Hierárquico em Anel: um Esquema Eficiente para Leitura de Dados Replicados. In *Anais do 11o. Simpósio Brasileiro de Redes de Computadores*, pages 119–138, Campinas-SP, May 1993.

[MA93b] Nabor C. Mendonça and Ricardo O. Anido. Usando Votação Hierárquica Estendida para Controlar Dados Replicados: de Votação Simples a Estruturas Lógicas. In *Anais do V Simpósio de Computadores Tolerantes a Falhas*, pages 18–36, São José dos Campos-SP, October 1993.

[MA93c] Nabor C. Mendonça and Ricardo O. Anido. Utilizando uma Estrutura de Grade com Anéis para Reduzir o Custo de Acesso a Dados Replicados. In *Anais do XX Seminário Integrado de Software e Hardware*

(SEMISH), XII Congresso da SBC, pages 490-502, Florianópolis-SC, September 1993.

[MA91] Nabor C. Mendonça and Ricardo O. Anido. Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures. In *Proceedings of the 27th Hawaii International Conference on Systems Sciences (HICSS-27), Minitrack on Parallel and Distributed Databases*, Maui, Hawaii, 1991. To be published.

[Mae85] M. Maekawa. A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems. *ACM Transactions on Computer Systems*, 3(2):145-159, May 1985.

[MW82] Toshimi Minoura and Gio Wiederhold. Resilient Extended True-Copy Token Scheme for a Distributed Database System. *IEEE Transactions on Software Engineering*, SE-8(3):173-188, May 1982.

[Pâr86] Jehan-François Pâris. Voting with Witnesses: A Consistency Scheme for Replicated Files. In *Proceedings of the 6th Conference on Distributed Computing Systems*, pages 606-612. IEEE, June 1986.

[Pâr90] Jehan-François Pâris. Efficient Voting Protocols with Witnesses. *Lectures Notes in Computer Science*, 170:305-317, December 1990.

[PL88] Jehan-François Pâris and Darrel D. E. Long. Efficient Dynamic Voting Algorithms. In *Proceedings of the 4th International Conference on Data Engineering*, pages 268-275. IEEE, February 1988.

[PL91] Calton Pu and Avraham Leff. Replica Control in Distributed Systems: An Asynchronous Approach. In *SIGMOD*, pages 377-386. ACM, February 1991.

[Ram89] K. V. S. Ramarao. Detection of Mutual Inconsistency in Distributed Databases. *Journal of Parallel and Distributed Computing*, 6:498-511, 1989.

[Ray89] Kerry Raymond. A Tree-Based Algorithm for Distributed Mutual Exclusion. *ACM Transactions on Computer Systems*, 7(1):61-77, February 1989.

[RL92] Michael Rabinovich and Edward D. Lazowska. Improving Fault Tolerance and Supporting Partial Writes in Structured Coteries Protocols for Replicated Objects. In *SIGMOD*, pages 226-235. ACM, June 1992.

[SS83] R. D. Schlichting and F. B. Schneider. Fail-Stop Processors: An Approach to Designing Fault-Tolerant Computing Systems. *ACM Transactions on Computer Systems*, 1(3):222-238, August 1983.

[Sto79] Michael Stonebreaker. Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES. *IEEE Transactions on Software Engineering*, SE-5(3):188–191, May 1979.

[Tan92] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice-Hall, Inc., 1992.

[TGGL82] Irving L. Traiger, Jim Gray, Cesare A. Galtieri, and Bruce G. Lindsay. Transactions and Consistency in Distributed Database Systems. *ACM Transactions on Database Systems*, 7(3):323–342, September 1982.

[Tho79] Robert H. Thomas. Majority Concensus Approach to Concurrency Control for Multiple Copy Databases. *ACM Transactions on Database Systems*, 1(2):180–209, June 1979.

[TN89] Jin Tang and N. Natarajan. A Static Pessimistic Scheme for Handling Replicated Databases. In *SIGMOD*, pages 389–398. ACM, May 1989.

[WB84] Gene T. J. Wuu and Arthur J. Bernstein. Efficient Solutions to the Replicated Log and Dictionary Problems. In *Proceedings of the 3rd Symposium on Principles of Distributed Computing*, pages 57–66. ACM, 1984.