3
Tolerância a Falhas e Replicação

A utilização de computação para aplicações críticas implica na demanda por sistemas com alta disponibilidade. A alta disponibilidade de um sistema está relacionada à capacidade de superar a ocorrência de eventos inesperados, evitando a perda de informação ou interrupção do serviço provido. Esses eventos inesperados são geralmente falhas que podem ocorrer por diferentes motivos: interrupção no fornecimento de energia, defeito de *hardware*, parada para manutenção, erros de programação ou erros de operação. Sistemas que conseguem manter a sua consistência e operação, mesmo na presença de falhas, são ditos Tolerantes a Falhas.

A computação distribuída permite o desenvolvimento de sistemas que executem em mais de uma máquina, se baseando na comunicação via rede. Esses sistemas podem ser compostos por aplicações clientes que acessam uma aplicação servidora. Se houver apenas uma aplicação servidora, o gerenciamento das interações entre os clientes e com os dados do sistema é mais simples. Entretanto, o sistema fica indisponível quando há qualquer interrupção na aplicação servidora. A técnica mais difundida para evitar esse tipo de vulnerabilidade é a replicação da aplicação servidora, visando tornar o sistema tolerante a falhas.

3.1
Replicação

A replicação consiste em criar cópias de um ou mais elementos de um sistema. Essas cópias são chamadas de réplicas e funcionam como um elemento único do sistema para aplicações externas a ele. Existem diferentes formas de replicação que diferem na maneira que as réplicas atendem as chamadas e em relação às características do elemento replicado às quais dão suporte. Em geral, a replicação consiste em instanciar réplicas de um mesmo elemento em diferentes máquinas, mantê-las consistentes e não afetar o comportamento padrão do sistema original. Dessa forma, esse elemento será tolerante a falhas de parada, pois, se uma máquina com uma de suas réplicas deixar de operar, as demais serão capazes e manter o sistema operando.

3.2 Comunicação em Grupo

A comunicação com as réplicas é o elemento chave da replicação e consiste no envio de mensagens para múltiplos destinatários (*multicast*) e gerenciamento de membros de grupos. O envio de mensagens *multicast* possibilita a comunicação de um cliente com diversas réplicas de um servidor de forma transparente. Além disso, uma ferramenta de comunicação em grupo deve permitir a ordenação das mensagens entregues para os membros do grupo.

O gerenciamento de membros do grupo simplifica a manutenção da lista de réplicas e oferece uma localização única para o grupo de réplicas. Como é possível incluir e remover membros do grupo, é necessário identificar quando a lista de membros mudou. Para isso, o gerenciamento de membros de grupo utiliza o conceito de visão, em que cada mudança na lista de membros gera uma nova visão do grupo. A manutenção dessas visões auxilia na determinação do momento de entrega de mensagens *multicast*. Nesses casos, uma ferramenta de comunicação em grupo que apresenta sincronia de visão garante que as mensagens são entregues para todos os membros do grupo, ou, se houver uma troca de visão durante a entrega das mensagens, para nenhum, sendo entregues apenas para os membros da próxima visão como representado no figura 3.1.

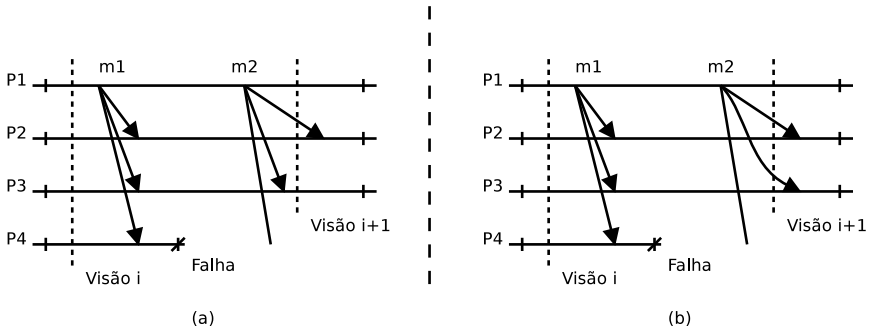

Figura 3.1: Exemplos de ferramenta de comunicação em grupo sem sincronia de visão *(a)* e com sincronia de visão *(b)*.

3.3 Detecção de falhas

A identificação de falhas nas réplicas do grupo procura identificar o momento em que uma delas deixa de operar. Existem diferentes algoritmos para determinar a falha de um elemento computacional. Um desses se baseia no envio periódico de mensagens por parte dos elementos que, uma vez recebidas pelo detector de falhas, irão indicar que o elemento está operando. Para funcionar dessa forma, os elementos e o detector tem que estipular um intervalo

máximo para o recebimento dessas mensagens. Esse intervalo deve ser definido de acordo com o ambiente em que o sistema opera, para evitar que existam identificações equivocadas da falha de um elemento. Existe a possibilidade também da função de detecção de falhas ser desempenhada por todos os elementos do sistema, evitando que a detecção seja suscetível a falhas. Nesse caso há uma sobrecarga maior na rede pois todos os elementos deverão trocar mensagens de detecção entre si.

A seguir iremos apresentar os tipos de replicação utilizados para replicação de objetos seguidos de uma breve comparação.

3.4 Tipos de Replicação

Existem dois tipos de replicação: replicação ativa e replicação passiva. Os conceitos de replicação podem ser usados tanto para replicar objetos quanto para replicar aplicações. Na descrição dos tipos de replicação dessa seção trataremos da replicação de objetos. Esses objetos possuem estado bem definido e métodos, que uma vez chamados podem alterar o estado do objeto. Além disso, os objetos podem ser parte de uma aplicação ou sistema maior e realizar operações sobre outros objetos durante a execução dos seus métodos. A seguir serão descritos em detalhes os tipos de replicação e suas características.

3.4.1 Replicação Ativa

Na replicação ativa são instanciadas diversas réplicas dos objetos, com o mesmo estado inicial, e cada uma realiza todas as operações chamadas por um cliente do objeto. Isso implica na repetição da chamada do cliente para todas as réplicas. Essa repetição pode ser feita de duas formas: o cliente envia as chamadas explicitamente para todas as réplicas ou o cliente envia a chamada para um objeto intermediário (por exemplo, uma ferramenta de comunicação em grupo) que repasse essa chamada para todas as réplicas. Em ambos os casos, se existirem resultados para as chamadas e o objeto replicado for realmente uma máquina de estados, os resultados das chamadas sobre cada uma das réplicas será igual e apenas o primeiro irá interessar. Entretanto, a recepção dos resultados deverá ser tratada de acordo com a forma que foi feita. Se o cliente realizou as chamadas diretamente ao objeto, o mesmo cliente deverá ser capaz de tratar o recebimento dos resultados, considerando apenas o primeiro resultado recebido e ignorando os demais. Se foi utilizado um objeto intermediário, esse deverá encaminhar apenas um resultado para o cliente, que, a princípio, desconhece a replicação. Dessa forma, se uma das réplicas falhar,

o funcionamento do objeto será mantido pelo ponto de vista do cliente que continuará tendo suas chamadas atendidas.

A replicação ativa considera que o objeto replicado pode ser visto como uma máquina de estados. Isso significa que a mesma sequência de operações sobre o objeto, a partir do mesmo estado inicial, irá levar ao mesmo estado final e resultados intermediários. No caso, o estado final se refere apenas ao estado do objeto, o estado global do sistema do qual esse objeto faz parte não pode ser alterado pelas suas operações. Sendo assim, operações sobre o objeto não podem resultar em efeitos colaterais sobre o sistema que faz parte, ou seja, as operações sobre esse objeto não podem realizar alterações no estado de outros objetos. A figura 3.2 exemplifica o caso em que o cliente envia a chamada para todas as réplicas.

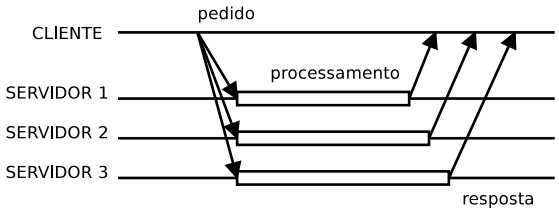

Figura 3.2: Troca de mensagens e processamento de chamada usando replicação ativa.

Esse tipo de replicação tem o potencial de apresentar o menor atraso sobre a realização de uma chamada, pois a réplica mais rápida determina o tempo para realização da operação e o atraso é determinado apenas pelo envio para todas as réplicas. Entretanto, esse tipo de replicação apresenta o maior consumo em termos de processamento, pois todas as réplicas irão realizar todas as chamadas.

Caso uma réplica se recupere ou uma nova réplica seja incluída no conjunto de réplicas, as demais réplicas devem ser capazes de informar o seu estado para a atualização na nova réplica. Nesses casos, enquanto atualizam a nova réplica, chamadas do cliente deverão acumuladas e realizadas sobre a nova réplica também. Como o objeto depende apenas da sequência das chamadas para determinar seu estado, as chamadas devem ser ordenadas de forma que todas as réplicas realizem as chamadas na mesma ordem. Essa ordenação depende do consenso das réplicas em relação a ordem das chamadas e pode ser feito por um elemento único ou de acordo com a comunicação entre as réplicas [10].

Os resultados enviados pelas réplicas que são ignorados podem ser usados para sobrepor falhas que gerem resultados incorretos. Isso porque se observados todos os resultados, é possível determinar o resultado que a maioria obteve.

Sendo assim, seria possível tolerar falhas bizantinas [11]. Essas falhas são caracterizadas por envio de valores incorretos em mensagens e podem ter diferentes causas. Uma réplica que, por algum motivo, tenha sofrido alguma interferência externa e possui estado diferente das demais, pode corromper o sistema se o seu resultado for considerado. Entretanto, se o sistema de replicação considerar os valores de retorno até que a maioria dos resultados indique o mesmo valor, esse valor pode ser enviado como resultado para o cliente, mascarando a falha, e mantendo a operação do sistema.

Por depender do comportamento de máquina de estado para os elementos replicados nesse tipo de replicação, a replicação ativa não pode ser usada quando as operações desses elementos podem ter comportamento não-determinístico. Operações desse tipo podem apresentar resultados diferentes a cada chamada, mesmo que os valores de entrada e a ordem seja a mesma. Nesse caso, as réplicas passariam a apresentar estado interno diferente e a replicação deixaria de ser consistente. Por outro lado, se o elemento realizar operações que apresentem algum tipo de efeito colateral em outros elementos do sistema, a replicação dessa operação pode gerar efeitos indesejados. Para replicar elementos que apresentem essas características deve ser usado um modelo de replicação diferente.

3.4.2 Replicação Passiva

A replicação passiva consiste em apenas uma réplica realizar as operações. Nesse tipo de replicação as réplicas desempenham funções diferentes, existe a figura de uma réplica principal (primária) e de réplicas reservas (secundárias). Sendo assim, um sistema que utilize esse tipo de replicação não precisa ter comportamento determinístico, pois as chamadas são realizadas apenas uma vez na réplica principal. As mudanças no estado da réplica principal causadas por essas chamadas são enviadas para as réplicas secundárias em mensagens de atualização. Essas mensagens possuem todas as informações necessárias para manter o estado das réplicas secundárias consistente em relação à réplica principal. Dessa forma, em casos de falha da réplica principal, um réplica secundária pode assumir o seu papel e passar a realizar as operações.

Apesar de permitir a recuperação da operação de uma réplica, a replicação passiva é suscetível à perda de uma ou mais operações, dependendo de como o envio das mensagens de atualização é feito. O momento do envio e a forma de utilização dessas mensagens determinam os sub-tipos da replicação passiva. Esses sub-tipos são: replicação passiva fria, replicação passiva morna e replicação passiva quente.

Na replicação passiva fria réplicas secundárias são instanciadas apenas após a identificação da falha da réplica principal. Por esse motivo, o estado global e as mensagens de atualização do estado da réplica principal são armazenadas em meio não-volátil e acessível pelas máquinas onde serão instanciadas as nova réplicas. A réplica principal pode ser configurada também para salvar o estado completo em um ponto específico da execução ou após um intervalo determinado de tempo. Essa modificação evita que sejam acumuladas mensagens complementares ou que se anulem, por exemplo mensagens que ajustam o valor do mesmo campo.

Casos em que o tempo de recuperação tem que ser menor, é utilizada a replicação passiva morna. Nesse tipo de replicação, as réplicas secundárias executam em conjunto com a réplica primária. Dessa forma, as mensagens de atualização são enviadas diretamente para as réplicas secundárias, mas só são aplicadas durante a recuperação. Assim como na replicação passiva fria, é possível enviar o estado completo para sincronizar as réplicas secundárias com a réplica principal. Isso é utilizado para reduzir o número de mensagens de atualização acumuladas e o tempo gasto na recuperação da operação.

Na replicação passiva quente as mensagens de atualização são aplicadas assim que são recebidas e o envio de mensagens e atualização de estado são considerados parte da operação. Para isso, a réplica principal não envia uma resposta para o cliente até que todas as réplicas secundárias confirmem o recebimento da mensagem de atualização e tenham atualizado o seu estado. Por esse motivo, esse é a forma de replicação passiva com o maior impacto no desempenho do sistema. O sistema fica tão lento quanto sua réplica mais lenta. A figura 3.3 contém um exemplo de chamada usando a replicação passiva quente.

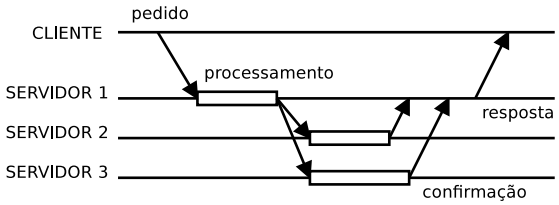

Figura 3.3: Troca de mensagens e processamento de chamada usando replicação passiva quente.

O impacto no desempenho do sistema que utilize esse tipo de replicação pode ser minimizado flexibilizando algumas garantias [12]. Essa flexibilização pode ser feita no momento em que a resposta é enviada para o cliente e no número de confirmações necessárias para responder uma chamada. A flexibilização do momento da resposta consiste em modificar o comportamento

das réplicas secundárias para que respondam com a mensagem de confirmação assim que receberem as mensagens de atualização, deixado para atualizar o estado após o envio da confirmação. Por outro lado, se o sistema abrir mão da capacidade de tolerar a falha simultâneas de uma réplica principal e de uma réplica reserva, é possível responder a chamada para o cliente assim que a primeira réplica reserva confirmar o recebimento da atualização. Na figura 3.4 é possível observar a combinação dessas flexibilizações. Em (a) a resposta é enviada para o cliente assim que a primeira réplica secundária confirmar a atualização do seu estado. Em (b) as réplica secundárias enviam a mensagem de confirmação assim que recebem a mensagem de atualização, processando a atualização em seguida. Em (c) a resposta é enviada assim que que a primeira réplica secundária confirma o recebimento da mensagem de atualização. Finalmente, em (d) a réplica principal tem o comportamento mais próximo ao observado sem a replicação e, consequentemente, o menos tolerante a falhas. Nesse caso, assim que o processamento é concluído o resultado é enviado para o cliente. Somente após o envio da resposta as mensagens de atualização são enviadas para as réplicas secundárias. Apesar de ser o menos tolerante a falhas, essa flexibilização é observada em sistemas em que a consistência do estado é mais importante que a perda de algumas operações.

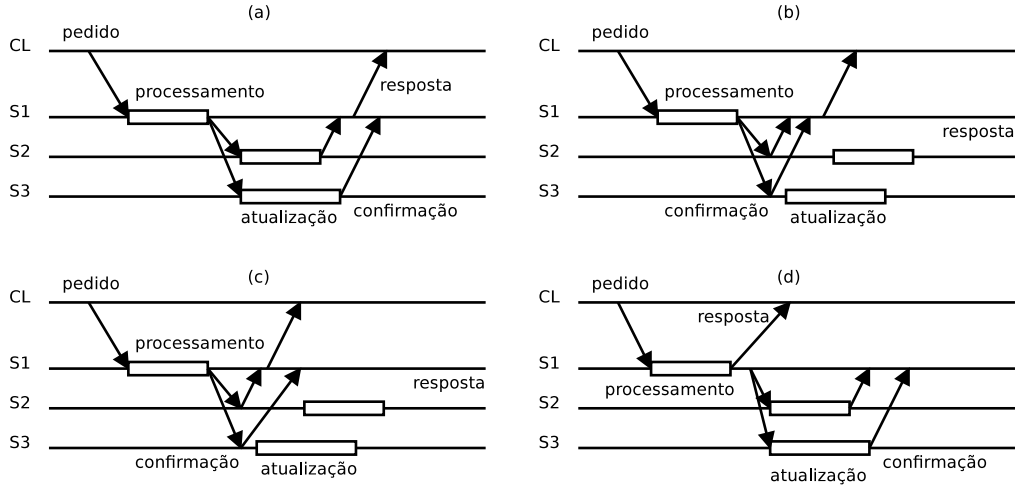

Figura 3.4: Flexibilização de garantias para replicação passiva quente.

Existe também a replicação semi-passiva [13], que utiliza o consenso circular [14] para determinar a atualização do estado. Nesse tipo de replicação o cliente da aplicação precisa conhecer todas as réplicas para realizar uma chamada, mas não precisa saber qual é a réplica principal ou identificar se uma réplica falhou. A cada chamada, a réplica coordenadora do consenso fará o papel de réplica primária, realizando a operação, e enviará o resultado da operação como o resultado do consenso, além de enviar a resposta para o

cliente. Em casos de falhas, as réplicas secundárias irão identificar a ausência de um consenso e irão passar para uma nova rodada de consenso, em que uma nova réplica irá fazer o papel de coordenadora. Essa forma de replicação sugere a utilização de intervalos pequenos para identificação de falhas, mas não considera a recuperação de uma réplica. Nesse tipo de replicação as réplicas com um índice alto de atrasos devem ser mantidas no fim da lista de possíveis coordenadores, implicando no gerenciamento da ordenação customizada e não circular das réplicas.

3.4.3 Comparação Entre os Tipos de Replicação

O tipo de replicação passiva a ser usado depende das características do objeto replicado. Para determinar o melhor tipo de replicação, devem ser observadas características como: armazenamento de estados intermediários, custo de processamento, uso de banda de rede e tempo de recuperação.

O armazenamento de estados intermediários consiste em armazenar o estado completo, após um intervalo de tempo ou em pontos do código pré-definidos, para a instanciação de uma nova réplica primária. Essa característica implica na possibilidade de perda de operações, pois as operações realizadas após o último ponto de armazenamento cujo resultado ainda não tenha sido armazenado em um estado intermediário terão que ser realizadas novamente em casos de falha. Enquanto a replicação passiva fria e replicação passiva morna apresentam essa característica, a replicação passiva quente não conta com esses estados pois as réplicas secundárias são atualizadas a cada chamada realizada. Na replicação ativa também não apresentam armazenamento de estados intermediários, cada réplica é atualizada realizando as operações que recebe. Sendo assim, apenas a replicação passiva fria e a replicação passiva morna apresentam sobrecarga em relação a essa característica.

Em relação ao processamento a replicação passiva apresenta o potencial de ter o menor impacto no desempenho. Enquanto na replicação ativa todas as réplicas realizam todas as operações e o processamento total é proporcional ao número de réplicas, na replicação passiva apenas uma réplica executa as operações e o impacto no processamento é proporcional ao tamanho do estado ou da atualização que tem que ser aplicada nas réplicas secundárias.

O uso de banda de rede aumenta em qualquer tipo de replicação, mas a replicação passiva é a que apresenta o maior potencial de aumentar o impacto no desempenho em relação ao uso da bande de rede. Como mensagens de atualização são enviadas a todo momento ou o estado completo a cada intervalo de tempo, em casos que os parâmetros das operações são pequenos, o impacto

maior é na replicação passiva. Por outro lado, se os parâmetros das operações são grandes e o as alterações no estado são pequenas, a replicação passiva pode apresentar menor impacto na uso de banda de rede. Pode-se argumentar que na replicação ativa existem chamadas extras para ordenação de mensagens, mas, a não ser que o volume de chamadas e réplicas seja grande o suficiente para que as réplicas demorem muitos passos para atingir o consenso sobre a ordem das operações, o envio dessas mensagens é potencialmente menos custoso que o envio das operações e os seus parâmetros para todas as réplicas e o envio do resultado a partir de todas as réplicas.

O tempo de recuperação na replicação ativa é o menor possível. Nesse tipo de replicação a falha de uma das réplicas não interrompe a operação do sistema, pois os clientes continuam realizando suas chamadas para as demais réplicas em operação. O tempo de recuperação na replicação passiva depende do tempo para identificação da falha e eleição ou instanciação da nova réplica principal. O menor tempo de recuperação dentre os tipos de replicação passiva é observado na replicação passiva quente, em que não há a necessidade de instanciação de uma nova réplica ou de atualização de estado, sendo necessário apenas iniciar a operação da réplica eleita como réplica principal.

3.5 Abordagens para Replicação de Objetos Distribuídos

A integração de tecnologias de sistemas distribuídos com programação orientada a objetos levou a sistemas com objetos distribuídos. Esses sistemas funcionam como infra-estruturas de serviços genéricos a partir dos quais são desenvolvidos sistemas especializados. A replicação de sistemas desse tipo é feita no nível dos objetos e serviços, mas utiliza como base os conceitos de replicação ativa e passiva. Como se concentra na replicação de elementos no nível do estado e de chamadas de métodos do objeto, a replicação de objetos distribuídos não leva em consideração o processo que hospeda os objetos.

As infra-estruturas que iremos descrever utilizam o padrão de objetos distribuídos CORBA. O padrão CORBA foi criado para atender a demanda de portabilidade de código e interoperabilidade entre aplicações independente do sistema operacional e linguagem de programação. Objetos de sistemas que utilizam esse padrão podem interagir com objetos de outros sistemas baseado na localização e descrição da interface desses objetos. As interfaces são definidas em uma linguagem chamada de IDL que é mapeada de acordo com o padrão para diferentes linguagens. Essa linguagem comum e o protocolo de comunicação entre os objetos permite a criação de elementos conhecidos como ORBs. Esses ORBs são os intermediários da comunicação entre os objetos

CORBA e criam uma fronteira que abstrai a linguagem em que os objetos foram desenvolvidos e a sua implementação.

Infra-estruturas que oferecem replicação para sistemas distribuídos CORBA utilizam uma de três abordagens [15]: integração, serviço ou interceptação.

A abordagem de integração consiste em alterar o elemento responsável pela localização e envio de mensagens de forma que uma chamada seja encaminhada para várias réplicas sem o conhecimento do realizador da chamada. Além disso, o resultado da chamada é tratado para que apenas um seja retornado. Em sistemas que usem essa abordagem a existência de réplicas fica transparente para a aplicação cliente, mas os objetos criados não são compatíveis com sistemas que não usem essa abordagem. Isso ocorre pois os objetos passam a depender do comportamento específico dos elementos intermediários que encaminham as chamadas. Um exemplo de infra-estrutura que segue essa abordagem é o sistema *Electra* [16]. No Electra, o ORB é modificado para utilizar uma ferramenta de comunicação em grupo para encaminhar as chamadas realizadas pelos objetos distribuídos para todas as réplicas do grupo sem que o objeto cliente precise saber se o objeto servidor é replicado ou não.

A abordagem de serviço consiste em incluir serviços adicionais na infra-estrutura que fazem tarefas comuns à replicação, como comunicação em grupo e detecção de falhas. Sistemas que usam essa abordagem geram objetos distribuídos compatíveis entre infra-estruturas diferentes, mas obrigam o programador da aplicação a lidar com questões específicas da replicação. Uma infra-estrutura que usa essa abordagem é o OGS (*Object Group Service*) [17]. O OGS possui como principal característica possibilitar a configuração da replicação por método do objeto e por resolver a replicação ativa e passiva de forma comum, utilizando algoritmos de consenso [10].

Por último, a abordagem de interceptação se baseia em usar uma ferramenta externa ao sistema replicado para, no nível do sistema operacional, interceptar e identificar as mensagens trocadas entre os objetos distribuídos CORBA. Nessa abordagem, mensagens que utilizam esse protocolo específico são repassadas para réplicas pré-definidas e os resultados obtidos são tratados para que o objeto cliente não saiba da existência de réplicas. Como esses sistemas dependem de interação com o sistema operacional, soluções com essa abordagem não são portáveis entre sistemas operacionais. Um exemplo de sistema que use essa abordagem é o sistema *Eternal* [18]. O Eternal funciona como um componente de *software* associado aos binários da aplicação. Essa característica permite que ele seja usado com implementações de CORBA tanto em Java quanto C++.

A replicação de sistemas que usam objetos distribuídos pode ter que lidar com o estado do próprio ORB além do estado do objeto em si. No caso específico de CORBA, informações como a lista de clientes conectados ao objeto ou informações do último pedido enviado, podem interferir no comportamento do sistema e, portanto, devem ser replicadas. Por esse motivo, na maioria dos casos o envio do estado não pode ser feito diretamente entre réplicas criadas em infra-estruturas diferentes. Outro complicador é a necessidade de comunicação da aplicação replicada com elementos externos não replicados (por exemplo, sistemas legados ou bases de dados). Nesses casos, a infra-estrutura precisa disponibilizar serviços para intermediação dessas chamadas. Esses serviços farão o filtro de chamadas de forma que apenas uma requisição seja feita para o elemento externo, mas que o resultado fique disponível para todas as réplicas que venham a realizar a mesma chamada. Serviços como esse permitem lidar com o indeterminismo de chamadas externas, pois armazenam o resultado obtido da chamada. Entretanto, outra forma de indeterminismo é observada em sistemas distribuídos, por exemplo, chamadas baseadas no relógio da máquina ou interações com o usuário. Sendo assim, a infra-estrutura deve lidar com esse tipo de chamadas para dar suporte a replicação de objetos com operações não-determinísticas.

3.5.1 Comparação entre Abordagens de Replicação de Objetos Distribuídos

Cada uma das abordagens tem suas vantagens e desvantagens, cabe ao desenvolvedor identificar qual se adapta melhor ao sistema que está projetando. Tendo em mente que o objetivo de usar infra-estruturas prontas para objetos distribuídos é reduzir o tempo de desenvolvimento, facilitando o reuso de objetos e interação entre serviços, os principais pontos de comparação para abordagens de sistema distribuídos são [17]: transparência, facilidade de uso, portabilidade, interoperabilidade, desempenho e simplicidade.

A transparência está relacionada à capacidade de ocultar a existência de réplicas para o desenvolvedor, de forma que chamadas ou resultados pareçam originar de elementos únicos. As abordagens de integração e interceptação permitem a criação de sistemas com essa característica sem muita dificuldade, já a abordagem de serviço requer que a aplicação trate explicitamente de questões do gerenciamento de grupos e replicação de chamadas.

A facilidade de uso está relacionada aos passos adicionais necessários para o uso da replicação para o desenvolvimento dos sistemas. Essa característica permite que mais tempo seja dedicado a questões de segurança e confiabilidade do sistema em oposição ao aprendizado do mecanismo de replicação.

Abordagens que oferecem maior transparência são mais fáceis de usar no desenvolvimento pois a maior parte do trabalho da replicação fica na implantação do sistema replicado. A abordagem de serviço requer mais trabalho no desenvolvimento pois a configuração e instanciação de réplicas é feita no próprio código, mas os padrões e estruturas usadas são comuns para desenvolvedores.

A portabilidade pode ser avaliada em relação ao gerenciamento de grupos ou em relação ao código da aplicação criada. Essa característica é importante que a aplicação não fique dependente da infra-estrutura utilizada, tendo custos altos caso o desenvolvedor deixe de oferecer suporte a ela. A abordagem de integração requer que a aplicação faça uso de construções específicas para lidar com o gerenciamento de grupos, sendo assim o código da aplicação não é portável para outras implementações. No caso da abordagem de interceptação, a aplicação fica restrita aos sistemas operacionais suportados pela infra-estrutura, apesar do código da aplicação não depender da infra-estrutura, a sua replicação depende. Por especificar apenas serviços baseados em CORBA, a abordagem de serviço permite a criação de aplicações portáveis tanto em relação ao gerenciamento de grupos quanto em relação ao seu código.

A característica de interoperabilidade trata da possibilidade dos objetos de um sistema interagirem com objetos implementados usando outras infra-estruturas e objetos não replicados. Em sistemas com replicação de objetos distribuídos, a comunicação em grupo que vai determinar o desempenho geral de acordo com os algoritmos para envio de mensagens *multicast*. Sendo assim, sistemas que oferecem abstrações de alto nível (como transparência e interoperabilidade) têm uma eficiência menor que sistemas sem essas abstrações. Abordagens que tratem de forma especial o envio de chamadas para múltiplas réplicas apresentam o melhor desempenho, mas deixam de ser interoperáveis com demais infra-estruturas.

A simplicidade está associada à quantidade de elementos extras que a infra-estrutura oferece além do necessário para o funcionamento da aplicação. Nesse sentido, as abordagens de interceptação e de integração utilizam ferramentas para comunicação em grupo. Como essas ferramentas são desenvolvidas para a comunicação entre processos, fica a cargo das infra-estruturas disponibilizar camadas adicionais para adaptá-los à comunicação entre objetos. A abordagem de serviço se adapta melhor nesse sentido pois são disponibilizadas apenas as primitivas necessárias para comunicação em grupo para objetos e o serviço é incluído como um componente independente e opcional.

Além dessas considerações, um sistema tolerante a falhas deve ser planejado examinando [15]:

1. o estado relevante do sistema, isto é, os dados que deverão ser protegidos

na presença de falhas;

2. a granularidade apropriada para os objetos da aplicação, pois a replicação de objetos muito pequenos interfere com o desempenho e demanda por recursos;

3. os elementos críticos do processamento, isto é, operações que deverão ser realizadas sem interrupção, mesmo com falhas no sistema; e

4. o fluxo de dados entre elementos do sistema, ou seja, objetos que se comunicam com muita frequência deverão ser atribuídos ao mesmo processo.

3.6 Considerações Finais

Cada técnica de replicação possui características específicas que, dependendo do sistema que será replicado, podem ser interpretadas como vantagens ou desvantagens. Para um sistema ser tolerante a falhas, sua replicação tem que ser consistente, garantindo que as réplicas do sistema irão se comportar, pelo ponto de vista dos usuários ou aplicações clientes, como um elemento único. Por possuírem um motor interno para controle de processos, sistemas de gerenciamento de *workflow* podem ter seu estado global alterado sem uma chamada externa. Por esse motivo, esses sistemas não se comportam como um objeto replicado ou exatamente como um serviço de um sistema distribuído. Com isso em mente, no próximo capítulo iremos listar os requisitos, especificações e como foi implementado o mecanismo de replicação que desenvolvemos para sistemas de gerenciamento de *workflow*.