Estudo sobre Comunicação de Grupos para Tolerância a Falhas

Luiz Angelo Steffenel

Estudo sobre Comunicação de Grupos para Tolerância a Falhas

Luiz Angelo Barchet Estefanel
angelo@inf.ufrgs.br

1 Introdução

O desenvolvimento de sistemas distribuídos freqüentemente exige o emprego de diversas técnicas a fim de prover garantias sobre esses sistemas. Tais técnicas incluem sincronização de processos, consistência, disponibilidade e tolerância a falhas. Para adicionar tolerância a falhas, mantendo confiabilidade e desempenho, a complexidade da implementação aumenta consideravelmente. Uma forma de modelar tais características de modo mais simplificado é através do paradigma de *comunicação de grupos*.

Neste modelo, pode−se representar situações onde há a necessidade de realizar comunicação entre processos que trabalham de forma cooperativa. Um **grupo** é compreendido como o conjunto dos processos cooperantes, reunidos de alguma maneira definida pelo usuário ou pelo sistema, e pode ser endereçado como sendo uma unidade única.

Cada processo membro de um grupo pode ser considerado um elemento independente, que não tem acesso direto aos dados dos outros membros, e para isso necessita trocar informações através de mensagens. Ao utilizar o modelo de grupos para abstrair um conjunto de processos, considera−se que quando uma mensagem é enviada para o grupo, todos os seus membros a recebem.

Ainda sobre a abstração de grupos, são adicionados serviços especiais, tais como a associação dinâmica de membros, que têm por objetivo tornar transparente ao usuário o estado interno do grupo. Esses serviços também são utilizados para prover características como atomicidade, ordenação de mensagens e outras, essenciais à modelagem e ao funcionamento adequado de um sistema distribuído.

2 Características

Tanenbaum [TAN 92] descreveu as possíveis características do modelo de grupos, em relação à troca de mensagens entre os processos (integrantes do grupo e processos que acessam o grupos), classificando−os por suas estruturas internas:

Grupos Fechados: nesta definição, somente aos membros é permitido enviar mensagens para o grupo. Dessa forma, processos que não fazem parte do grupo não podem referenciá−lo como tal, embora possam contactar os membros individualmente.

Grupos Abertos: ao contrário dos grupos fechados, essa definição permite que elementos externos ao grupo possam contatá−lo e interagir com ele.

A escolha dos sistemas por um método ou outro depende de suas necessidades, e do nível de contato que o sistema terá com o meio externo (por exemplo, um servidor de arquivos distribuído provavelmente terá que aceitar requisições oriundas do “meio externo” ao sistema).

Já quanto ao relacionamento dos membros que compõem o grupo, a seguinte característica foi definida:

Grupos Não Hierárquicos: todos os membros integrantes do grupo têm representatividade semelhante, de modo que as decisões do grupo são obtidas através de uma votação ou consenso, sem a necessidade de uma hierarquia interna.

Grupo Hierárquico: em um grupo desse tipo, há um ou mais elementos que são destacados para realizar tarefas em níveis hierárquicos superiores, tais como a coordenação do processamento dos dados, contagem de votos, etc.

Sistemas que não estabelecem hierarquias podem ser estruturados de forma simétrica, diminuindo os riscos encontrados em sistemas com pontos únicos de falha (*single point of failure*). Entretanto, as técnicas para a obtenção de consenso são mais complexas, e o desempenho do sistema pode sair prejudicado. Sistemas hierárquicos podem ser estruturados logicamente sobre o problema a ser resolvido (por exemplo, a exploração de uma árvore de decisões), mas devem conter procedimentos para a eleição de um novo coordenador, caso seja desejado adicionar tolerância a falhas sobre os membros do sistema.

Sobreposição de Grupos: quando é permitido a um ou mais processos pertencerem a mais de um grupo, pode ocorrer o fenômeno da sobreposição dos grupos. O suporte a grupos sobrepostos significa que deve ser dado tratamento especial à ordem das mensagens enviadas para os elementos comuns entre os grupos. Como os sistemas normalmente tratam a questão de ordenamento e atomicidade de comunicação apenas dentro de um grupo, operações concorrentes sobre um elemento comum podem causar inconsistências de dados com os demais membros do grupo. Uma ferramenta de comunicação de grupo deve prever se irá suportar grupos sobrepostos, ou impedirá isso através de restrições de projeto ou implementação.

Um outro caso crítico, imaginado pelo autor deste resumo, refere−se à uma característica específica do sistema de comunicação Isis. No Isis, quando um processo é considerado suspeito de ter falhado, é ordenado a ele que se suicide, para que não ocorram situações de falhas temporárias ou arbitrárias. Em um sistema que implemente essa mesma atitude com relação aos objetos suspeitos, e permita sobreposição de grupos, deverá haver um controle muito rígido sobre os membros de todos os grupos, pois o suicídio forçado de um processo irá acarretar um esforço adicional dos demais grupos sobrepostos, para detectar e eliminar o processo falho do seu quadro de membros.

Grupos Dinâmicos: são grupos que permitem a manutenção do quadro de integrantes do sistema, tornando possível a adição de novos membros em situações de sobrecarga, ou a remoção de elementos que comportem−se inadequadamente. Para isso, o sistema que gerencia o grupo deve conter procedimentos para atualizar constantemente a lista desses elementos, e no exemplo de um sistema não hierárquico, também garantir que todos elementos tenham conhecimento da nova lista.

Grupos Estáticos: nesta organização, o *membership* (conjunto de membros de um grupo) não muda durante todo o tempo de vida do sistema. Mesmo em um caso de falha de um dos membros, o *membership* não muda para refletir esta falha, ou seja, o componente que falhou permanece como membro do grupo depois da falha e antes de uma possível recuperação [GUE 97]. Isso induz à necessidade de adicionar ferramentas de detecção de elementos falhos, e as primitivas de comunicação providas pelo grupo devem ser capazes de contornar a situação e operar mesmo que de forma reduzida sob a presença destes elementos, para garantir a conclusão do processamento.

A atualização do *membership* em sistemas dinâmicos deve refletir de modo mais acurado o real estado dos processos membros. Assim, utiliza−se o conceito de **visão**, que é o estado do *membership* em um instante de tempo determinado. A união das diversas visões do grupo, no decorrer do tempo, é chamado de **histórico** do grupo.
Cabe ao sistema de comunicação de grupo determinar o momento em que as visões são definidas. A alteração da visão no meio de uma comunicação pode levar a uma inconsistência sobre os elementos do grupo. Por isso normalmente os sistemas implementam a atualização da visão somente antes da execução de uma comunicação, e a visão para essa operação é mantida até o fim da execução dessa instrução. Mais conceitos sobre visões serão apresentados na seção 4.
A classificação de Tanenbaum é relativamente simples e desprovida de considerações sobre o tipo de aplicações que referenciam o grupo. Por isso, Liang *et al*. (*apud* [NUN 98]) estenderam a classificação de Tanenbaum para incluir o conceito de objetos e grupos de objetos. Os grupos de processos são considerados gerentes que controlam os objetos de um determinado grupo, e um grupo de objetos é composto por um conjunto de objetos compartilhando uma ou mais características comuns (estado interno), interagindo e se auto-coordenando (todo grupo de objetos tem um processo gerente) para fornecer uma interface externa uniforme.
Comparando as definições de grupos de processos e grupos de objetos, ressaltam−se certas diferenças que devem ser cuidadosamente estudadas na modelagem de um sistema de comunicação de grupo. Membros do grupo de processos controlam a maneira pela qual é obtido acesso aos recursos de um dado grupo de objetos além da forma pela qual eles coordenam, entre si, a consistência no grupo de objetos. Os membros também fazem a interface entre os usuários e os recursos que eles utilizam. Mensagens para o grupo (*multicast*) correspondem a mensagens para o grupo de processos e não para o grupo de objetos. Essa diferença foi detectada na prática por Guerraoui *et al*. [GUE 98].
De acordo com a taxonomia de Liang [NUN 98], a estrutura interna dos grupos, do ponto de vista da aplicação, é feita segundo a presença ou a ausência de homogeneidade dos objetos e operações, onde ser **homogêneo** significa que todos os membros do grupo mantêm réplicas dos objetos ou operações, e ser **heterogêneo** significa possuir objetos ou operações diferentes.
Quanto ao comportamento externo, os grupos podem ser **determinísticos** ou **não-determinísticos**. Basicamente um grupo determinístico requer uma alta confiabilidade na comunicação do grupo para manter consistência entre os membros. Tais grupos costumam requerer informações completas sobre o controle dos membros dos grupos, além de interações atômicas e ordenadas. Ao contrário, grupos não determinísticos suportam comunicações não confiáveis, e as interações de grupo inconsistentes e não confiáveis são controladas pela aplicação, o que aumenta a flexibilidade da aplicação, mas complica seu desenvolvimento.

4 Modelos Semânticos de Execução

4.1 Sincronismo virtual

No modelo de comunicação de grupos, onde há constante troca de informações entre os componentes do grupo, é necessário manter dados e estados de cada membro consistentes. No caso em que membros falham, entram e saem do grupo dinamicamente, cada membro pode ter uma visão diferente da composição do grupo. Por esse motivo, o sincronismo

entre membros do grupo e a ordenação na entrega de mensagens por cada membro é necessário para se garantir um estado consistente.
O sincronismo virtual é um serviço semântico, introduzido pelo sistema ISIS [BIR94], que ordena as mudanças de *membership* do grupo em relação às mensagens difundidas no grupo. Este modelo garante que todos os processos pertencendo a um grupo percebem as mudanças de configuração que ocorrem em um mesmo tempo lógico. Além disso, todos os processos que pertencem a uma configuração entregam o mesmo conjunto de mensagens para aquela configuração. É garantido que uma mensagem é entregue (em todos os processos) na mesma configuração na qual ela foi difundida.
O sincronismo virtual garante uma ordenação entre mudanças de visão e difusão de mensagens, mas não considera a ordenação de mensagens entregues em uma mesma visão. Além disso, o sincronismo virtual considera o modelo de falhas *fail−stop*, de forma que quando um defeito é informado a qualquer membro, todos processos recebem esse evento. No caso de particionamento da rede, o sincronismo virtual garante que processos em apenas uma partição da rede, a partição primária, podem continuar operando. Processos em outras partições são bloqueados.
O conceito de sincronismo virtual pode ser descrito da seguinte forma:
cada grupo de processos tem associado a ele uma visão do grupo, na qual os membros são listados pela ordem em que eles se juntaram ao grupo;
mudanças na visão do grupo são relatadas na mesma ordem para todos os membros;
qualquer *multicast* enviado para o grupo é realizado entre duas visões e enviado para todos os membros do grupo. O gerenciamento dos membros do grupo é feito com a última visão do grupo recebida pelo processo que enviou a mensagem. A este procedimento é dado o nome de envio multicast com visão síncrona;
quando um processo junta-se ao grupo e a visão do grupo é relatada aos outros membros, este novo processo pode obter o estado corrente do grupo a partir de algum membro ou conjunto de membros pré-existente;
os *multicasts* são atômicos e ordenados, podendo esta ordenação ser total ou causal;
os defeitos são tratados segundo o modelo *fail-stop*, ou seja, se um defeito é relatado a qualquer processo, todos os processos vêem o mesmo evento;
um processo torna-se falho devido a um *crash* ou devido a um particionamento. Neste último caso, quando a partição é reparada, o processo que se tornou falho, somente pode se reunir ao grupo com um identificador de processo diferente, e após executar um protocolo especial de desconexão, disparado quando o processo perceber que pertence a uma partição minoritária.

4.2 Sincronismo Virtual Estendido

Em sistemas distribuídos de larga escala, normalmente numa rede com possibilidade de particionamento, o método de sincronismo virtual não é interessante para a disponibilidade e consistência do sistema. A baixa confiabilidade dos canais de comunicação e os grandes atrasos na comunicação fazem os particionamentos ocorrem com uma freqüência bem maior do que em redes locais. Os sistemas Totem e Transis estabeleceram uma técnica para modelar aplicações que poderiam tolerar inconsistências geradas pelo progresso de um componente não primário num sistema particionado.
Assim, qualquer partição que possa alcançar um acordo interno no seu controle de membros (*membership*) recebe permissão de continuar operando. Aplicações que só são seguras se executadas numa única partição (primária), devem fazer a suspensão das outras

partições, porém outras aplicações podem continuar fornecendo (de forma mais restrita) seus serviços nas partições não-primárias, agrupando seus estados quando o sistema eliminar o particionamento. Este modelo que permite múltiplas partições é denominado Sincronismo Virtual Estendido (*Extended Virtual Sinchrony* - EVS).
Na prática, o Sincronismo Virtual Estendido apresenta sérios problemas ao fazer a união (*merging*) das partições primária e secundárias, pois o algoritmo de junção necessita resolver, de algum modo, as inconsistências que podem ter ocorrido durante o tempo em que o defeito de particionamento se manteve, o que pode nem sempre ser possível [BIR 96].
Para resolver essa restrição, um modelo misto, designado *two-tiered model*, sugere a divisão de uma grande rede em várias redes locais (LAN) interconectadas por uma pequena rede *Wide Area Network* (WAN). Esta divisão permite estruturar a rede em duas camadas: uma camada LAN e uma camada WAN. Cada camada LAN tem seu próprio subsistema completo implementando o particionamento primário do modelo com sincronismo virtual, permitindo assim que aplicações confinadas nas LANs possam prosseguir normalmente desde que independam de dados mantidos na WAN ou em outras LANs. A camada WAN pode ficar bloqueada enquanto o defeito de particionamento a impedir de estabelecer o grau de consenso necessário para enviar atualizações, de forma segura. Este bloqueio interfere somente nas aplicações que dependem de dados mantidos fora da sua rede local.

5 Conclusões

O paradigma de comunicação de grupo vem sendo desenvolvido há bastante tempo, sendo que nesse período firmou suas características e objetivos, no que se refere à programação envolvendo processos.

A popularização de novas tecnologias de comunicação e disseminação de informações, especialmente a programação Orientada a Objetos e a Internet representam um novo desafio para os desenvolvedores. Os sistemas têm que incorporar escalabilidade, flexibilidade, tolerância ao particionamento da rede, confiabilidade e tolerância a modelos diversos de falhas.

Um conceito largamente difundido é de que o estilo de programação não deve ser modificado, devendo as ferramentas de comunicação de grupo integrar−se transparentemente aos ambientes de desenvolvimento, de modo que o programador só necessite preocupar−se com a aplicação final.

Conforme os estudos que levaram à taxonomia de Liang, e com a confirmação prática obtida por Guerraoui, o conceito aplicado sobre grupos de processos não é semelhante ao de grupos de objetos, levando à necessidade de desenvolver sistemas específicos para objetos, testar e questionar a integração entre grupos de processos e objetos, existentes em grande parte das ferramentas para comunicação de grupo com objetos (especificamente, com o CORBA).

6 Bibliografia

[BIR 96] BIRMAN, K. P., **Building Secure and Reliable Network Applications**. Greenwich: Manning Publications Co., 1996.
[GEI 99] GEISS, L. C. **Tolerância a Falhas por Replicação Usando Comunicação de Grupo**, Trabalho Individual, CPGCC – UFRGS, Porto Alegre, Janeiro 1999.

[GUE 97] **GUERRAOUI, R.; SCHIPER, A. Software−Based Replication for Fault Tolerance,** IEEE Computer**, p. 68−74, Abril 1997.**

[GUE 98] **GUERRAOUI, R.; FELBER, P.; GARBINATO, B.; MAZOUNI, K. System Support for Object Groups,** ACM Conference for Object Oriented Programming Systems, Languages and Applications**, Outubro 1998.**

[NUN 98] NUNES, R. C. **Programação Orientada a Grupos: o Ponto de Vista das Aplicações**, Trabalho Individual, CPGCC – UFRGS, Porto Alegre, Agosto 1998.

[TAN 92] **TANENBAUM, A.** Modern Operating Systems**. Prentice−Hall, Englewood Cliffs, 1992.**