CAPÍTULO 20

Arquiteturas distribuídas para sistemas colaborativos

Roberta Lima Gomes
Roberto Willrich
Guillermo de Jesús Hoyos Rivera

META

Apresentar e analisar as arquiteturas distribuídas utilizadas no desenvolvimento de sistemas colaborativos.

OBJETIVOS EDUCACIONAIS

Após o estudo desse capítulo, você deverá ser capaz de:

- Identificar os modelos de distribuição utilizados nas arquiteturas de sistemas colaborativos e avaliar o impacto destes modelos no desempenho dos sistemas.
- Identificar os principais modelos de redes Peer-to-Peer e analisar as vantagens do uso deste paradigma em sistemas colaborativos.
- Analisar como o paradigma SOA pode ser utilizado no desenvolvimento de sistemas colaborativos.

RESUMO

Neste capítulo são discutidos os conceitos sobre as arquiteturas distribuídas utilizadas no desenvolvimento de sistemas colaborativos. Sistemas colaborativos são sistemas distribuídos que apresentam desafios específicos, analisados neste capítulo. São apresentados três modelos de arquiteturas de distribuição utilizados no desenvolvimento de sistemas colaborativos: arquiteturas centralizadas, descentralizadas e híbridas. A escolha do modelo tem influência sobre diferentes aspectos do sistema, como desempenho, tolerância a falhas, segurança, entre outros. Um exemplo de arquitetura descentralizada detalhado neste capítulo é o P2P, em que os recursos são compartilhados diretamente entre os clientes. Outro paradigma apresentado é SOA, que representa uma classe específica de arquitetura híbrida. SOA está apoiado no conceito de serviços interoperáveis que podem ser estendidos, compostos, e colaborar entre si.

20.1 Sistemas colaborativos são sistemas distribuídos

A complexidade tecnológica enfrentada pelos projetistas de sistemas colaborativos depende dos modos de interação entre os usuários. Existem dois modos básicos de interação: síncrona e assíncrona. No modo de interação síncrona, cada ação realizada por um membro do grupo é transmitida imediatamente aos outros membros, a colaboração ocorre em tempo real. Já no modo de interação assíncrona, os usuários não interagem no mesmo instante.

O projeto de sistemas colaborativos requer o tratamento de diversos desafios tecnológicos, principalmente em sistemas síncronos. Vários dos desafios estão relacionados ao ambiente computacional distribuído, formado pelos computadores utilizados pelos membros do grupo e toda a infraestrutura básica de comunicação entre as máquinas. Para entender melhor os desafios envolvidos no projeto de sistemas colaborativos é necessário compreender as características de um sistema distribuído.

Existem várias definições para sistemas distribuídos. Para muitos autores, um sistema distribuído é formado por componentes de software e hardware localizados em computadores autônomos que se comunicam por troca de mensagens via rede. Grande parte dos sistemas colaborativos são sistemas distribuídos em que os usuários interagem em um espaço de trabalho compartilhado por meio de computadores conectados em rede.

Um dos objetivos dos sistemas distribuídos é possibilitar o compartilhamento de recursos entre os vários componentes do sistema. Os recursos podem ser hardware, como uma impressora, ou software e informações, como arquivos ou bancos de dados. Na Figura 20.1 estão representados três usuários que compartilham um mesmo arquivo, e cada usuário está realizando operações de leitura e escrita no arquivo compartilhado.

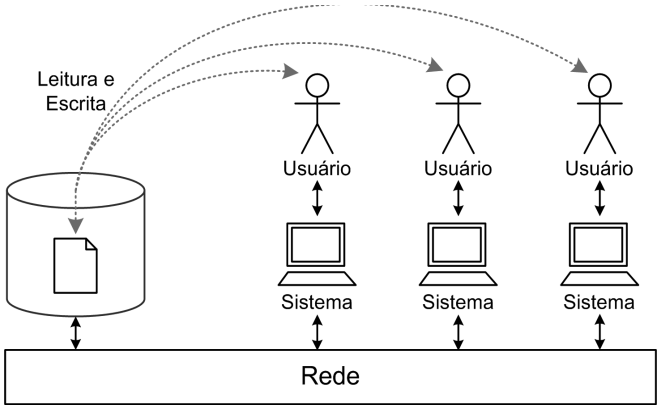


Figura 20.1 Usuários compartilhando um arquivo

Em sistemas distribuídos, os chamados gerenciadores de recursos controlam o acesso aos recursos compartilhados via algum mecanismo de controle de concorrência. O mecanismo de controle de concorrência é responsável por garantir a consistência por meio da sincronização de acessos concorrentes a um mesmo recurso. Usando o exemplo da Figura 20.1, o controle de concorrência deve garantir que mesmo que dois ou mais usuários leiam ou escrevam no mesmo arquivo de maneira concorrente, cada um deles tenha a mesma visão do conteúdo do arquivo.

Existem duas abordagens para o controle de concorrência:

- Controle de concorrência pessimista, define um mecanismo que controla, serializa e sincroniza todas as operações sobre os recursos compartilhados. Uma forma de controle de concorrência pessimista aplicada ao exemplo da Figura 20.1 é bloquear as linhas editadas por um usuário e assim nenhum outro usuário pode realizar alterações até que o bloqueio seja liberado.
- Controle de concorrência otimista, em vez de evitar inconsistências, tenta detectá-las e corrigi-las. É útil para sistemas em que é pouco provável que ações no recurso compartilhado provoquem inconsistências. Aplicado ao exemplo da Figura 20.1, todos os usuários podem editar o documento, mas as alterações no arquivo somente são efetivadas após uma validação para detectar violações de concorrência.

Enquanto o controle de concorrência representa um elemento chave na implementação de mecanismos eficientes para compartilhamento de recursos, outras características também são desejáveis para os sistemas distribuídos, sendo elas:

- Transparência de distribuição: possibilita que os usuários abstraiam a distribuição dos componentes do sistema para dar a impressão de que os componentes estão em um único sistema local. A transparência de acesso oculta dos usuários as comunicações necessárias para acessar os recursos remotos. A transparência de concorrência

possibilita que os usuários não tomem conhecimento de que os recursos estão sendo compartilhados com outros componentes. No cenário da Figura 20.1, com a transparência de acesso e de localização, os usuários desconhecem que o arquivo está compartilhado com outros usuários.

- Extensibilidade: determina a facilidade de extensão do sistema distribuído, sem interrupção ou duplicação dos serviços oferecidos. Por exemplo, suponha que o sistema da Figura 20.1 seja extensível, então será possível adicionar uma nova funcionalidade, tal como um bate-papo entre usuários, sem interromper o serviço de acesso aos arquivos compartilhados. Uma forma de tornar um sistema extensível é padronizar e publicar as interfaces de seus componentes.
- Escalabilidade: determina a capacidade de o sistema atender uma grande demanda por recursos sem a degradação dos serviços oferecidos. Os componentes do sistema não precisam ser modificados à medida que aumenta o número de usuários ou demanda de serviços.
- Tolerância a falhas: é a capacidade de o sistema continuar a oferecer os serviços mesmo quando ocorrer alguma falha. Uma medida para aumentar a tolerância a falhas é a criação de componentes redundantes, tanto de hardware como de software. Por exemplo, uma forma de oferecer maior tolerância a falhas ao acesso a arquivos compartilhados é criar servidores redundantes de arquivos. Se um dos servidores falhar, outro servidor o substitui sem a interrupção do serviço.
- Interoperabilidade: é relacionada à coexistência de hardware ou de software de diferentes fabricantes que tenham a capacidade de intercomunicação. Por exemplo, em um sistema de acesso a arquivos compartilhados com suporte a interoperabilidade, os usuários podem compartilhar os arquivos entre máquinas diferentes e com sistemas operacionais diferentes.
- Portabilidade: é a capacidade de um componente ser desenvolvido para um sistema distribuído de modo que possa ser utilizado, sem modificações, em outro sistema distribuído.

São muitos os desafios a serem superados pelos sistemas distribuídos. Algumas possíveis soluções não são simples para implantar e nem sempre conseguem garantir totalmente os resultados esperados.

Sistemas distribuídos geralmente são heterogêneos, formados por diferentes plataformas de hardware, sistemas operacionais e linguagens de programação. Apesar da heterogeneidade, os componentes devem se comunicar e compartilhar recursos. As soluções para abstrair a heterogeneidade são baseadas na definição de padrões de comunicação e de camadas de software. Na categoria de camada de software podemos citar as abordagens baseadas em middleware (como a Common Object Request Broker Architecture - CORBA), e as baseadas em máquina virtual (como a máquina virtual Java).

A possibilidade de oferecer serviços extensíveis em sistemas distribuídos é outro desafio. A publicação das interfaces dos componentes é um dos aspectos necessários para obtenção da extensibilidade. Mas a partir daí, o maior desafio é tratar a complexidade de sistemas distribuídos, geralmente formados de vários componentes projetados por diferentes pessoas, o que dificulta a adição de novos componentes ou a extensão de componentes existentes.

Em geral, o serviço oferecido pelas redes de computadores é do tipo melhor esforço, não havendo garantias, por exemplo, em termos de taxa de transferência e de atraso. Este comportamento imprevisível das redes de computadores dificulta a manutenção do desempenho, a tolerância a falhas, a escalabilidade e o controle de concorrência dos sistemas distribuídos.

Com relação à segurança, tanto dos recursos compartilhados quanto das mensagens trocadas entre os componentes, os sistemas distribuídos devem implementar mecanismos para garantir: a confidencialidade das mensagens trocadas entre o sistema transmissor e o receptor; a autenticidade dos componentes que estão comunicando; e a disponibilidade dos serviços que não devem ser interrompidos mesmo perante ataques.

20.2 Sistemas colaborativos e distribuídos

A tecnologia central para o desenvolvimento da maior parte dos sistemas colaborativos é a computação distribuída com suporte a multiusuários. Vários dos princípios do projeto de sistemas distribuídos são também fundamentais no projeto de sistemas colaborativos. Os principais desafios estão ligados ao acesso concorrente a recursos compartilhados e ao gerenciamento da consistência dos recursos. Mas sistemas colaborativos apresentam alguns problemas específicos, pois: (i) o controle da concorrência é mais complexo devido ao grande número de usuários compartilhando recursos; (ii) em muitos sistemas há a necessidade de suporte à comunicação entre os usuários, por exemplo, na forma de texto, áudio ou vídeo; e (iii) geralmente há a necessidade de implementação de políticas distribuídas que permitam que os usuários se coordenem enquanto acessam o espaço de trabalho compartilhado.

É importante observar que os sistemas distribuídos e os sistemas colaborativos se distinguem também quanto à forma de controlar a colaboração. Em um sistema colaborativo, o controle é realizado sobre a colaboração entre usuários. Já em um sistema distribuído, o controle é orientado aos componentes e visa ocultar destes componentes os problemas relacionados à distribuição (transparência de distribuição).

Para ressaltar as peculiaridades dos sistemas colaborativos com relação aos demais sistemas distribuídos em geral, analisemos o exemplo da edição cooperativa de texto da Figura 20.1. Um sistema distribuído genérico irá ocultar a existência dos outros usuários que estão editando o texto, o que é obtido com a transparência de distribuição. Isso é o contrário do que se espera de um sistema colaborativo, em que os usuários devem ter a percepção (awareness) das atividades realizadas pelos outros participantes do grupo.

O projeto de sistemas colaborativos síncronos é ainda mais desafiador, devido aos aspectos de tempo real associados. Esses sistemas ficam sujeitos ao serviço do tipo "melhor esforço" das redes de computadores, que pode ocasionar uma redução de desempenho dos sistemas colaborativos devido ao não oferecimento de garantias de latência e taxa de transmissão, bem como uma possível perda de mensagens transmitidas. O desempenho da rede tem influência direta no tempo de feedback (tempo de resposta às ações do próprio usuário), de feedthrough (tempo para receber a resposta à ação de outros usuários) e nas qualidades de áudio, vídeo e renderização 3D. O comportamento da rede também influencia na disponibilidade do sistema colaborativo, pois se a rede não oferecer ao menos o serviço de conectividade entre os membros do grupo, o sistema colaborativo fica inoperante.

Outro desafio é a segurança, que se refere à preservação da confidencialidade e integridade das informações, e à identificação dos usuários. Essa propriedade é especialmente importante em ambientes corporativos que envolvem informações críticas sobre produtos e serviços comerciais. Finalmente, os sistemas colaborativos têm dificuldades de interoperar, pois na maioria dos casos eles não são projetados como sistemas abertos.

20.3 Modelos de arquiteturas de distribuição

Um sistema colaborativo é um sistema interativo multiusuário que geralmente é executado de forma distribuída. Devido à complexidade inerente ao projeto de um sistema multiusuário distribuído, é comum que a especificação de um sistema colaborativo seja realizada por meio de uma arquitetura conceitual. Um dos principais objetivos de uma arquitetura conceitual é definir uma organização apropriada ao sistema que propicie um desenvolvimento eficiente e um código modular, que seja modificável e reutilizável. Uma arquitetura conceitual apresenta os componentes, ou as partes do sistema, sob a forma de unidades de desenvolvimento de software como módulos ou classes. Essas unidades se inter-relacionam por meio de conectores que representam alguma composição lógica. A arquitetura conceitual está muito mais focada no desenvolvimento e na manutenção do sistema do que no uso do sistema. No caso específico de sistemas colaborativos, arquiteturas conceituais também são definidas para abstrair os aspectos de distribuição e de concorrência.

Diferente da arquitetura conceitual, a arquitetura de distribuição, ou arquitetura de implementação, apresenta o sistema como um sistema distribuído em tempo de execução. Os componentes são instanciados em objetos e executados dentro de processos localizados nos sistemas computacionais. Os objetos se comunicam diretamente ou por meio de mecanismos de comunicação entre processos, localmente ou através da rede. Na prática, a arquitetura de distribuição é definida com base na arquitetura conceitual do sistema colaborativo. O sistema é então implementado conforme sua arquitetura de distribuição.

Se considerarmos os vários sistemas colaborativos, comerciais ou acadêmicos, observaremos que têm sido adotadas diferentes arquiteturas com diferentes modelos de distribuição. Os três tipos principais de arquiteturas são: centralizadas, descentralizadas e híbridas, conforme apresentado nas subseções a seguir. Mas que fatores levam os desenvolvedores de sistemas colaborativos escolherem uma ou outra arquitetura? Dentre os fatores que influenciam a escolha estão os requisitos não funcionais tais como tempo de resposta, maior consistência das informações compartilhadas, tolerância a falhas e segurança.

20.3.1 Arquiteturas centralizadas

Uma arquitetura completamente centralizada segue o modelo cliente-servidor. Todo o processamento necessário à execução de uma aplicação colaborativa é realizado em um sistema computacional central denominado servidor. Os sistemas computacionais dos usuários, denominados clientes, são conectados ao servidor. No sistema de cada usuário é executada uma aplicação cliente responsável apenas pelos serviços de apresentação de interface gráfica. Por exemplo, no cenário ilustrado na Figura 20.1, se um usuário insere uma palavra no texto, o evento de inserção é enviado ao servidor que, em resposta, muda o estado do texto. Em seguida, o servidor replica a ação por meio de comandos gráficos para as aplicações clientes que compartilham o texto. Exemplos de aplicações tradicionais que seguem esta arquitetura são aplicações de acesso remoto, como Telnet, e alguns tipos de aplicações de compartilhamento de desktop.

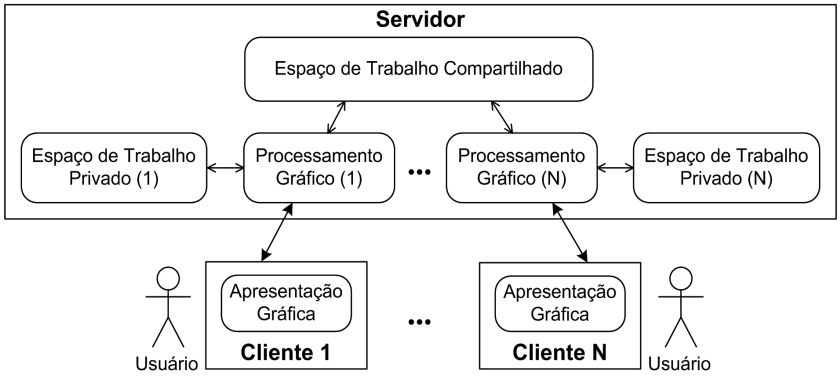


Figura 20.2 Arquitetura centralizada: processamento realizado em um servidor

Em uma arquitetura totalmente centralizada, o gerenciamento e o processamento da aplicação são realizados no servidor. O servidor também executa todo o processamento de interface gráfica de cada cliente. Como ilustrado na Figura 20.2, quando um usuário interage com o sistema cliente, os eventos de interface gráfica referentes a essa interação são gerados pelo componente de apresentação gráfica, mas não são tratados localmente. Esses eventos são enviados ao servidor que os trata por meio de um componente de processamento gráfico.

Um problema de uma arquitetura completamente centralizada é que o tempo de feedback é suscetível aos atrasos de rede, pois o processamento gráfico é realizado no servidor. Em nosso exemplo da inserção de uma palavra em um texto compartilhado, cada letra digitada gera um evento que é enviado ao servidor. O componente de processamento gráfico trata o evento e o estado do texto é alterado. Somente então o comando para atualizar a interface gráfica é retor-

nado pela rede ao cliente, que finalmente apresenta o texto alterado. O número de informações trocadas entre o servidor e os clientes é alto visto que toda interação de usuário deve ser notificada ao servidor, que por sua vez encaminha todas as atualizações gráficas aos clientes.

Uma forma de atenuar o problema de tempo de feedback e da quantidade de mensagens trocadas é deixar que o próprio cliente realize o processamento gráfico da interface. Com isso, cada elemento de processamento gráfico da Figura 20.2 deve ser implementado no respectivo cliente, e não mais no servidor. À medida que o usuário interage, o processamento da interface gráfica é realizado localmente e as ações são enviadas ao servidor que realizará o devido controle sobre o espaço compartilhado, como sincronização, concorrência, entre outros. É comum encontrarmos jogos multiusuários 3D baseados nesta arquitetura. O cliente é responsável por processar toda a animação 3D assim como as interações sobre a interface gráfica. O servidor, por outro lado, centraliza todo o controle sobre o jogo. Esta arquitetura também tem sido aplicada no desenvolvimento de sistemas colaborativos baseados em clientes web. Enquanto a aplicação web apresenta os componentes gráficos de interação, o servidor é responsável por realizar a gerência do espaço de trabalho compartilhado.

A implementação da arquitetura centralizada é mais simples, visto que só há uma instância do espaço compartilhado e, desta forma, são eliminados os problemas de consistência e sincronização. A gerência da segurança também é implementada de forma mais controlada. O controle de concorrência é realizado por um servidor central que gerencia todos os acessos aos recursos ou objetos compartilhados. O servidor também deve resolver possíveis conflitos, como dois usuários tentando alterar o estado do mesmo objeto.

Sistemas colaborativos síncronos devem possibilitar que usuários se conectem a uma sessão já em execução (em inglês, latecomers). Um sistema colaborativo com arquitetura totalmente centralizada consegue acomodar os latecomers facilmente. O servidor cria um componente de processamento gráfico que envia ao novo cliente os comandos gráficos para atualizar o componente de apresentação gráfica.

Além do espaço de trabalho compartilhado entre os usuários, alguns sistemas colaborativos possibilitam a criação de espaços privados de trabalho. Em uma arquitetura centralizada, o gerenciamento dos espaços privados também é realizado no servidor, e quando a interação ocorre sobre um objeto no espaço privado o comando gráfico é enviado apenas ao respectivo usuário.

Com todas as vantagens listadas da arquitetura centralizada, por que não adotamos essa arquitetura como padrão? Porque a arquitetura que segue o modelo cliente-servidor sofre de problemas bem conhecidos: baixa escalabilidade e baixa tolerância a falhas.

O problema de escalabilidade está associado ao fato de o servidor ser responsável por todo o processamento, com exceção da exibição das interfaces gráficas. O gerenciamento do espaço compartilhado e o processamento gráfico de cada usuário são atividades computacionalmente intensivas. À medida que aumenta o número de usuários conectados, o servidor pode não conseguir tratar de forma satisfatória os eventos recebidos de todos os clientes.

O problema de baixa tolerância a falhas deve-se ao servidor, que representa um ponto único de falha. Todo o sistema colaborativo é interrompido se o servidor parar de executar ou se o servidor ficar indisponível por ter sido desconectado da rede.

Para diminuir o problema da escalabilidade, pode-se implementar o componente de processamento gráfico em cada cliente de forma a diminuir a carga do servidor e o número de mensagens trocadas, como descrito anteriormente. Outra alternativa para atenuar o problema da escalabilidade e da baixa tolerância a falhas é definir uma arquitetura com mais de um servidor. Os servidores podem ser redundantes ou dividir o processamento. No entanto, a simplicidade de implementação é perdida, pois os servidores deverão se comunicar e coordenar a gerência dos recursos compartilhados.

Existe ainda uma variante da arquitetura centralizada em que se escolhe um cliente para desempenhar o papel do servidor, como exemplificam alguns sistemas para compartilhamento de desktop, como o VNC (Virtual Network Computing). É comum neste tipo de arquitetura não oferecer um espaço de trabalho privado para cada usuário. O objetivo é diminuir o processamento da aplicação do usuário que tem o papel de servidor.

20.3.2 Arquiteturas descentralizadas

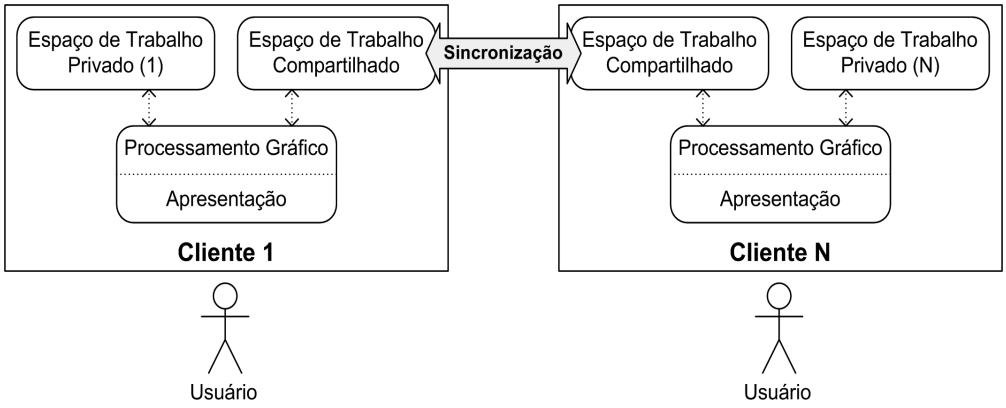
Em uma arquitetura completamente descentralizada, todo processamento é executado de forma distribuída, não é definido um servidor como na arquitetura centralizada. Os recursos compartilhados são replicados nos sistemas de cada usuário. Cada participante do grupo de trabalho tem uma instância completa da aplicação colaborativa rodando em sua máquina.

A arquitetura descentralizada é mais tolerante a falhas, pois as informações são replicadas e não há um servidor central. A ausência do servidor também contribui para a escalabilidade do sistema já que não há um ponto central de processamento. Por outro lado, sincronizar e manter a consistência do estado distribuído do espaço de trabalho compartilhado são tarefas complexas.

Na Figura 20.3 são ilustradas duas maneiras de realizar a sincronização do espaço compartilhado. No modelo ilustrado na Figura 20.3.a é implementado um canal de sincronização direto entre as cópias do espaço de trabalho. Se uma das cópias é alterada, o novo estado do espaço de trabalho (ou parte deste estado) é repassado às demais cópias. Já no modelo da Figura 20.3.b, a sincronização é realizada por meio da replicação das ações realizadas sobre o espaço de trabalho compartilhado. As ações são geralmente resultantes de interações dos usuários sobre o sistema colaborativo.

Vamos ilustrar a aplicação dos dois modelos no cenário da Figura 20.1. Quando um usuário insere uma palavra "abc" na posição x de um parágrafo y do texto, seguindo o modelo da Figura 20.3.a o sistema deve sincronizar o espaço de trabalho compartilhado enviando às demais aplicações clientes o novo estado do parágrafo y. Seguindo o modelo da Figura 20.3.b, o sistema encaminha aos demais usuários uma mensagem especificando que a palavra "abc" foi inserida na posição x do parágrafo y do texto, e assim cada cliente executa localmente a mesma ação sobre o espaço de trabalho compartilhado. Neste segundo modelo, a consistência do estado é mantida de forma implícita, o que requer que todas as alterações sejam replicadas da mesma forma e na mesma ordem em cada cliente. Uma alternativa ao modelo da Figura 20.3.b é replicar as interações de usuário diretamente sobre o módulo de apresentação gráfica. Essas interações são reprocessadas em cada instância e, consequentemente, as alterações são aplicadas sobre o espaço compartilhado de cada instância.

(a) Sincronização direta do estado de trabalho compartilhado



b) Replicação das ações realizadas sobre o espaço de trabalho compartilhado

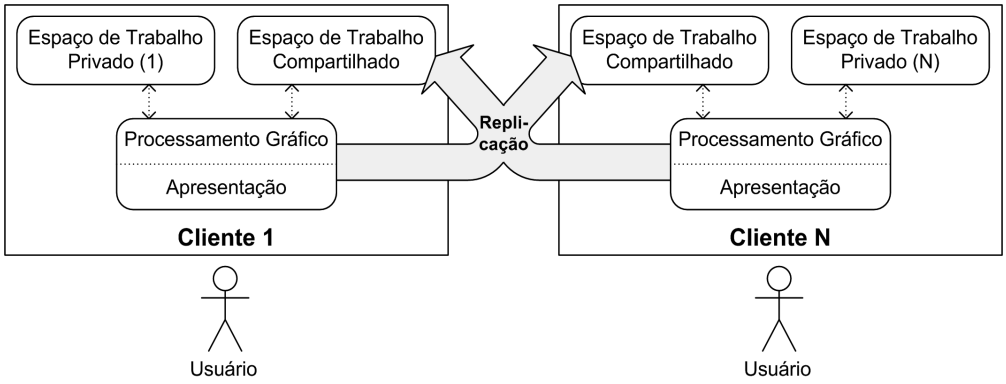


Figura 20.3 Arquiteturas descentralizadas

Uma das vantagens dos sistemas que implementam uma arquitetura descentralizada está no tempo de feedback. Se um controle de concorrência otimista for aplicado, o usuário interage de forma direta sobre uma cópia local do espaço de trabalho sem interferência de atrasos de rede.

Além de mais tolerância a falhas, a arquitetura descentralizada promove maior escalabilidade com relação à distribuição do processamento, visto que o processamento não fica concentrado no servidor. No entanto, como a distribuição causa uma troca maior de mensagens entre os clientes, protocolos e formatos de mensagens otimizados devem ser utilizados para não comprometer a escalabilidade. Além de requerer mais recursos de comunicação, o processamento distribuído e as cópias dos recursos implicam em maior utilização total de outros recursos computacionais, como memória e processador.

Na arquitetura descentralizada, a gerência da consistência dos dados compartilhados também é mais complexa do que em uma arquitetura centralizada. Em sistemas com esta arquitetura é comum que o estado do espaço de trabalho compartilhado fique temporariamente inconsistente. A inconsistência temporária causa problemas de coordenação entre os colaboradores além de tornar a acomodação de latecomers mais complexa.

20.3.3 Arquiteturas híbridas

Arquiteturas híbridas combinam características de arquiteturas centralizadas e descentralizadas em um único sistema. A combinação possibilita que os componentes do sistema colaborativo sejam distribuídos e se comuniquem de formas distintas, dependendo dos requisitos do sistema.

Um modelo de arquitetura híbrida, bastante adotado, utiliza um servidor responsável pelo gerenciamento do espaço de trabalho compartilhado, enquanto o cliente gerencia o espaço de trabalho privado e realiza todo o processamento gráfico. Este modelo promove uma maior escalabilidade já que uma parte do processamento é transferida para os clientes. Esse modelo conserva a simplicidade de uma arquitetura centralizada, pois mantém uma única instância do espaço de trabalho compartilhado.

Dependendo do tipo de sistema colaborativo, o tempo de feedback é crítico, sobretudo se for um sistema síncrono. Uma alternativa é dividir o espaço de trabalho compartilhado: uma parte é gerenciada pelo servidor e outra parte é gerenciada pelos clientes. Esta arquitetura pode ser utilizada para implementar diferentes mecanismos de controle de concorrência. O servidor aplica um mecanismo pessimista para controlar o acesso a recursos compartilhados cuja consistência seja crítica. Ao mesmo tempo, com base em um mecanismo otimista, os clientes gerenciam os recursos compartilhados cujo acesso deve apresentar pequenos tempos de resposta. Como exemplo, vamos considerar o cenário da Figura 20.1. Suponha que um grupo de professores ministrou uma disciplina e esteja agora editando o relatório de desempenho dos alunos. A primeira parte do relatório contém comentários sobre os alunos, enquanto a segunda parte contém as tabelas com as notas obtidas pelos alunos em cada atividade. Enquanto na primeira parte do relatório alguma inconsistência no texto é tolerável, podendo até ser vista como erro de digitação, na segunda parte do relatório é importante garantir a consistência das tabelas. Desta forma, a primeira parte pode ser gerenciada diretamente pelos sistemas clientes com o uso de um mecanismo otimista, o que possibilita que usuários digitem simultaneamente trechos do texto. Já a segunda parte é gerenciada pelo servidor que garante acessos exclusivos a cada tabela.

Uma arquitetura híbrida também é adequada para acomodar em um mesmo ambiente diferentes tipos de sistemas computacionais. Neste caso, para os sistemas mais restritos, por exemplo, que apresentem capacidades de processamento e memória limitadas, a aplicação cliente realiza apenas o processamento gráfico e o gerenciamento do espaço de trabalho privado. Já o acesso ao espaço compartilhado é gerenciado pelo servidor. Para sistemas com mais recursos computacionais, a aplicação cliente também gerencia o espaço compartilhado de forma distribuída juntamente com o servidor, seguindo um dos modelos de sincronização apresentados na Figura 20.3.

20.4 Arquiteturas P2P para sistemas colaborativos

Por algum tempo, o modelo cliente-servidor foi dominante em sistemas distribuídos. O maior problema desse modelo é o gargalo produzido quando a demanda dos clientes ultrapassa a capacidade do servidor. Outro problema é a falta de tolerância a falhas, pois se o servidor falhar, o sistema completo fica inoperante.

ESTRATÉGIAS DE DISTRIBUIÇÃO DA INFORMAÇÃO

Um sistema colaborativo pode empregar diferentes estratégias para realizar a comunicação entre os componentes da arquitetura de distribuição. Geralmente a comunicação é realizada por meio da troca de mensagens entre os nós da arquitetura, que são sistemas computacionais executados nos clientes ou nos servidores do sistema colaborativo. As mensagens contêm informações de controle ou de sincronização do espaço de trabalho compartilhado.

Uma estratégia de comunicação muito utilizada é a ponto a ponto. Quando uma mensagem deve ser enviada de um cliente do sistema colaborativo para um conjunto de nós da arquitetura (clientes ou servidores), uma cópia da mensagem deve ser enviada para cada nó destino. A comunicação via mensagens ponto a ponto é uma estratégia simples e gera um atraso de transmissão menor, visto que as mensagens não precisam ser processadas por nós intermediários (outros clientes ou servidores da arquitetura). Esta estratégia, contudo, tem algumas desvantagens como a sobrecarga da rede, em que o aumento no número de aplicações clientes ocasiona o aumento exponencial do número de mensagens, o que compromete a escalabilidade do sistema.

Outra estratégia de distribuição de informação é baseada no modelo cliente-servidor. As mensagens a serem trocadas são primeiramente enviadas ao servidor do sistema que, por sua vez, redistribui a mensagem empregando uma estratégia ponto a ponto. Essa estratégia é adequada a sistemas baseados em uma arquitetura centralizada. Uma vantagem é que os sistemas clientes não precisam abrir conexões com as demais aplicações clientes, pois basta se conectar ao servidor. No entanto temos o problema de escalabilidade, pois o servidor usa a estratégia ponto a ponto para distribuir a mensagem. Um número elevado de mensagens compromete o desempenho do servidor e causa uma sobrecarga da rede.

As estratégias descritas até agora são baseadas na comunicação unicast, em que cada cópia da mensagem tem um nó destino único. Para cada mensagem, o sistema emissor especifica quem é o nó destino. Uma forma de simplificar esse modelo é usar comunicação broadcast, em que mensagens são enviadas a todos os nós de uma rede. Com isso, todos os sistemas clientes e servidores recebem todas as mensagens. No entanto, é necessário que a infraestrutura de rede tenha suporte a endereço de broadcast. Essa estratégia é aplicada se todos os nós se encontrarem em uma mesma rede local. A redundância é evitada visto que uma única cópia da mensagem é transmitida.

Outra estratégia para evitar a redundância na transmissão de mensagens é o multicasting. São definidos grupos de nós, cada grupo tem um identificador ou endereço, e a mensagem passa a ser destinada ao endereço do grupo. Quando o multicast é suportado no nível de rede, isto é, pelos roteadores que interconectam os nós computacionais, a implementação da estratégia é direta. Neste caso, cada grupo de nós é associado a um endereço IP multicast e os pacotes IP são destinados a este endereço. Um único pacote é transmitido pelo sistema origem. Cópias do pacote são feitas apenas quando ocorrer uma separação no caminho de roteamento entre a origem e os nós participantes

do grupo multicast. Quando a camada de rede não oferece suporte ao multicast, essa estratégia precisa ser implementada no nível de aplicação (application-layer multicast). Como há vários nós rodando os sistemas clientes, estes nós são usados como roteadores de mensagens seguindo o mesmo modelo do multicast em rede.

Um modelo alternativo ao cliente-servidor, e que vem sendo utilizado mais recentemente, é o P2P (Peer-to-Peer), que representa uma classe específica de arquitetura descentralizada. Seguindo o modelo P2P, e ao contrário do modelo cliente-servidor, todos os nós (peers) são ao mesmo tempo servidores e clientes. O P2P tem características que fazem deste paradigma um bom candidato para ser adotado no desenvolvimento de algumas aplicações específicas, incluindo sistemas colaborativos.

Não existe um consenso para a definição do conceito P2P. Uma definição muito considerada é que sistemas P2P são sistemas distribuídos formados por nós que são capazes de se auto-organizarem em uma topologia de rede P2P com a finalidade de compartilhar recursos e serviços computacionais entre os nós participantes.

As redes P2P usam a infraestrutura de rede existente para a troca de mensagens. Evidentemente a rede não pode ser modificada pelas aplicações que executam sobre ela, mas as redes P2P adaptam-se criando uma topologia própria, chamada de rede Sobreposta, ou rede Overlay. Uma rede sobreposta é uma rede virtual criada sobre uma rede existente. A Figura 20.4 ilustra o conceito de rede sobreposta. Os enlaces entre os nós da rede sobreposta não correspondem necessariamente aos enlaces da rede física, e um só enlace virtual pode corresponder a um conjunto de enlaces, ou rotas, entre dois nós da rede física.

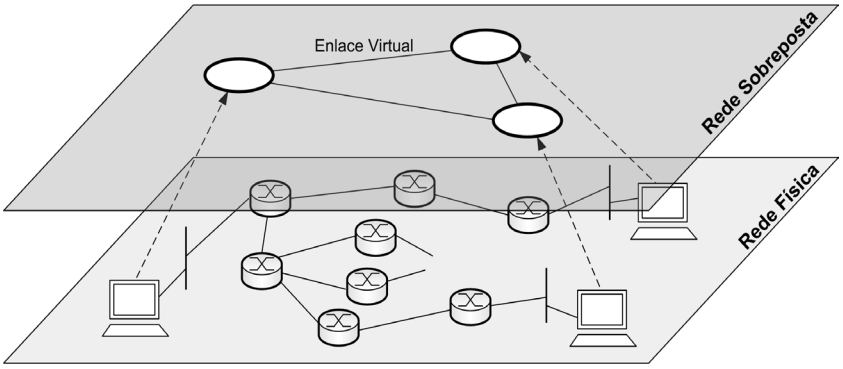


Figura 20.4 Ilustração do conceito da rede sobreposta

As características desejáveis de uma rede P2P são: inexistência de coordenação e de banco de dados centralizados; nenhum nó tem a visão global do sistema; todos os dados e serviços são acessíveis de qualquer nó; todos os nós são autônomos; nós e conexões não são confiáveis. Como resultado dessas características, são observadas algumas propriedades:

- Escalabilidade: quando a quantidade de nós cresce, cresce também a capacidade de trabalho total da rede.

- Robustez: não há ponto de falha único, pois cada nó é tanto cliente quanto servidor; se um nó falhar, outro assume a responsabilidade do que falhou.
- Flexibilidade: devido à disponibilidade de muitos nós com capacidades similares, a rede P2P pode ser reconfigurada automaticamente, o que possibilita a acomodação dinâmica de nós.

20.4.1 Modelos de arquiteturas P2P

Nem todas as redes P2P têm escalabilidade, robustez e flexibilidade; essas propriedades são dependentes do modelo de implementação adotado. As classificações mais usuais dos modelos de arquiteturas P2P levam em consideração: a forma de construção da rede P2P, a forma como os recursos são buscados na rede, e o nível de estruturação da rede. Em função da forma de construção das redes P2P, três modelos são definidos:

- Modelo centralizado. Uma característica importante deste tipo de rede P2P é a existência de uma autoridade central que mantém um índice de busca. Cada nó, ao buscar um recurso compartilhado na rede P2P, envia um pedido ao nó central que, por sua vez, retorna a lista dos nós que compartilham o recurso procurado. Toda comunicação posterior é realizada diretamente entre os nós da rede sem intervenção do nó central. Por requerer um nó central, este modelo apresenta baixa tolerância a falhas e pouca escalabilidade.
- Modelo descentralizado e não estruturado. Este tipo de rede P2P não define uma entidade central que mantém um índice de busca. Geralmente a busca é feita por inundação: um nó que realiza a busca passa a consulta a seus vizinhos, que repassam a seus vizinhos e assim por diante. Quando um nó possui o recurso procurado, ele informa o fato ao nó originador da busca. Esse tipo de busca não é escalável e é raramente utilizado atualmente. Outra característica deste tipo de rede P2P é que não possui um controle preciso sobre a topologia e localização de recursos.
- Modelo hierárquico, híbrido ou semicentralizado. Este tipo de rede P2P opera em um modo misto que combina as vantagens dos dois modelos precedentes. Neste caso, a topologia da rede é controlada, e os recursos e serviços são posicionados em locais determinados para facilitar as localizações. Existe também uma distinção do papel desempenhado pelos nós que formam a rede P2P. Determinados nós podem atuar como supernó, capazes de realizar tarefas administrativas. Os demais nós operam como nós convencionais e são associados a um supernó.

20.4.2 Tecnologias e sistemas colaborativos P2P

A utilização do modelo P2P para criar sistemas colaborativos é uma alternativa interessante. As possibilidades de convergência entre P2P e sistemas colaborativos são grandes, constituem uma área de pesquisa que oferece muitas oportunidades para novas soluções.

Uma forma típica de utilização das redes P2P em sistemas colaborativos é representada pelos sistemas de troca de mensagens e de compartilhamento de arquivos. Nesses sistemas, a sincronização e o controle de acesso são realizados pelos pares envolvidos na colaboração.

As tecnologias P2P também são usadas na implementação de diversas funcionalidades dos sistemas colaborativos, como serviços de informações de presença, gerenciamento de calendário, e gerenciamento de reunião. Um aspecto interessante do uso de P2P em sistemas colaborativos é a possibilidade de implementar multicast no nível de aplicação, com o uso de rede sobreposta, por exemplo para a realização de áudio e videoconferências.

Uma possível extensão do modelo P2P é o chamado F2F (Friend-to-Friend), que oferece um nível maior de segurança, em que apenas usuários autenticados podem compartilhar recursos com amigos. Com o uso do F2F, é possível constituir grupos fechados de trabalho. Esta característica representa uma vantagem em termos de privacidade e segurança, inclusive dificulta a propagação de malware e vírus.

SISTEMAS COLABORATIVOS COM O USO DA TECNOLOGIA P2P

Diversos sistemas colaborativos foram desenvolvidos seguindo o paradigma P2P. Alguns exemplos:

- Skype (http://www.skype.com) é um exemplo por excelência do êxito do paradigma P2P. O modelo de operação de Skype é hierárquico, constituído de um elemento principal, o Skype login Server, de supernós e de nós convencionais, sendo que toda a comunicação é realizada diretamente entre os nós.
- PeerWare (Cugola e Picco, 2003) aplica um modelo hierárquico P2P para a colaboração entre usuários.
- NetIsle (Shtykh e Jin, 2003) é um sistema colaborativo de propósito geral que integra serviços de colaboração online para a troca rápida de informações, o que incrementa a produtividade dos grupos de trabalho.
- Xpeer (Sartiani et al., 2004) é um sistema baseado no paradigma P2P para o compartilhamento e pesquisa de dados codificados em XML sem intervenção humana.

20.5 Arquiteturas orientadas a serviço

Arquitetura Orientada a Serviço (SOA – Service Oriented Architecture) é um paradigma para o desenvolvimento de sistemas distribuídos por meio da composição de serviços. O termo "serviço" tem causado muita discussão, pois é interpretado de diferentes maneiras dependendo do contexto em que é aplicado. Do ponto de vista da Tecnologia da Informação (TI), o termo serviço é apresentado como um componente de software independente e autocontido que oferece um conjunto de funcionalidades.

O modelo de base de SOA, como o ilustrado na Figura 20.5, apresenta três componentes: o Provedor de Serviço, o Consumidor de Serviço e o Serviço de Descoberta. O Provedor de Serviço disponibiliza na rede um serviço e publica esta informação junto a um Serviço de Descoberta (1). O Consumidor, por sua vez, usa o Serviço de Descoberta para fazer uma busca e receber a Descrição do Serviço (2). Finalmente, com base nessa descrição, o Consumidor

é capaz de invocar o serviço (3). Nesse modelo de base, um Provedor pode também ser um Consumidor. Com relação aos modelos de distribuição, SOA representa um modelo híbrido, em que um nó da arquitetura atua como cliente (enquanto Consumidor) ou como servidor (enquanto Provedor).

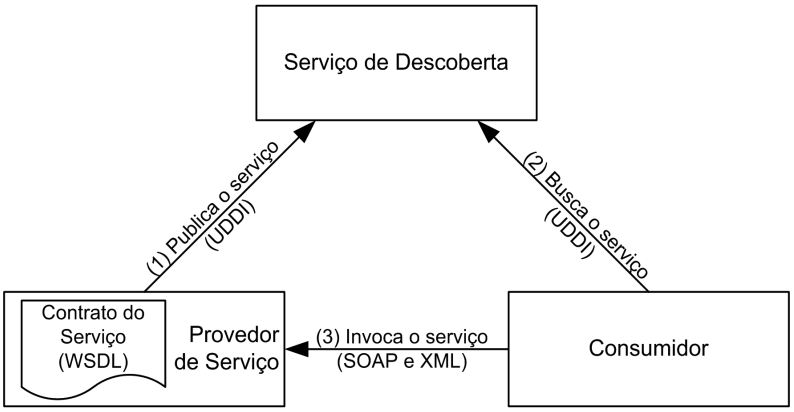


Figura 20.5 Modelo de base de SOA

A Descrição do Serviço contém informações que possibilitam que o Consumidor (i) decida se o serviço atende às suas necessidades, (ii) verifique se o Consumidor atende aos requisitos definidos pelo Provedor de Serviço, e (iii) conheça a interface, o que possibilita invocar e interagir com a implementação do serviço. A interface de serviço representa uma forma de contrato que define o conjunto de métodos e os tipos de dados associados que podem ser invocados pelo Consumidor. Esse modelo baseado em interfaces promove o que é chamado de reuso de software "caixa-preta", em que é eliminada a necessidade de conhecer a implementação do serviço a ser reusado.

Os sistemas orientados a serviços são distribuídos, e o ambiente de execução é em geral heterogêneo. Como consequência, a interoperabilidade representa um requisito fundamental em SOA. Seguindo o modelo SOA, os serviços devem interagir com base em um conjunto bem definido de protocolos e linguagens. Portanto é importante definir padrões para esses protocolos e linguagens, sobretudo se os serviços são abertos para serem acessados por um sistema qualquer.

A definição do modelo de base e a padronização de protocolos e linguagens possibilitam que os serviços apresentem baixo acoplamento e sejam dinamicamente localizados e invocados. O baixo acoplamento de um serviço está associado à capacidade de ser independente de outros serviços.

As tecnologias de Serviços Web surgiram com a necessidade de se padronizar as implementações SOA que estavam aparecendo. A padronização dos Serviços Web foi desenvolvida pelo W3C (World Wide Web Consortium), um consórcio internacional de empresas de tecnologia e órgãos governamentais para a definição de padrões e diretrizes para o desenvolvimento da internet. Serviços Web se apoiam em padrões e protocolos abertos, como XML e o protocolo HTTP. Ainda na Figura 20.5 observamos o emprego de alguns desses padrões. SOAP (Simple Object Access Protocol) é um protocolo de comunicação usado para codificar as mensagens

trocadas entre Consumidores, Provedores e Serviços de Descoberta. Provedores devem descrever os serviços (operações implementadas, tipos dos parâmetros de entrada e dos resultados) usando WSDL (Web Service Definition Language). Provedores usam o protocolo UDDI (Universal Description, Discovery and Integration) para publicar os serviços em um Serviço de Descoberta.

Outra característica de SOA é a possibilidade de compor ou combinar serviços existentes para criar novos serviços por meio de uma linguagem de composição de serviços, por exemplo, BPEL (Business Process Execution Language). A composição de serviços é feita por meio de orquestração (um processo coordenador controla e coordena a execução dos serviços) ou de coreografia (os serviços interagem de forma colaborativa se coordenando entre si).

20.5.1 Colaboração orientada a serviço

Um conceito estabelecido na área é o de Colaboração Orientada a Serviços (SOC, do inglês Service Oriented Collaboration), que se refere à colaboração entre serviços baseados em tecnologias SOA. Um exemplo característico de implementação de SOC é a composição de serviços por meio de coreografia.

Uma forma de estender o conceito de SOC ao domínio de Sistemas Colaborativos é pensar em um modelo em que os usuários possam colaborar por meio da criação e compartilhamento de serviços. Assim, os usuários atuariam como Provedores de Serviço, criando e publicando serviços. Quando um usuário apresentasse uma demanda por um serviço, este usuário realizaria uma busca pelo serviço mais adequado na internet, ou poderia compor serviços para atender à sua demanda específica. A internet poderia assim ser vista como um sistema colaborativo universal no qual usuários se conectam e colaboram por meio do compartilhamento de serviços.

A dificuldade de se desenvolver e implantar um serviço torna o modelo complexo. A descoberta automática de serviços é ainda mais complexa. Como forma de facilitar o desenvolvimento de sistemas colaborativos baseados em serviços, são utilizados frameworks e arquiteturas genéricas. Boa parte das soluções faz uso das tecnologias de Serviços Web por serem tecnologias abertas.

O paradigma SOA e as tecnologias de Serviços Web são soluções de interoperabilidade. SOA também promove extensibilidade ao permitir o desenvolvimento de sistemas fracamente acoplados. No entanto, soluções de integração baseadas em Serviços Web podem apresentar problemas de desempenho associados ao uso do protocolo SOAP. As mensagens SOAP são baseadas em XML, são usadas marcações e representações ASCII, e por isso são maiores se comparadas com formatos binários. Com isso as mensagens SOAP exigem mais tempo para serem transmitidas e decodificadas. Um novo estilo arquitetural conhecido como REST (Representational State Transfer) tem sido adotado pelos Serviços Web (chamados de Serviços RESTful) para simplificar a execução. Utiliza-se diretamente o HTTP para fazer uma chamada a um serviço, o que resolve os problemas de desempenho relacionados ao SOAP.

DESENVOLVIMENTO DE SISTEMAS COLABORATIVOS ORIENTADOS A SERVIÇO

Alguns trabalhos acadêmicos têm enfocado o emprego de SOA para o desenvolvimento de sistemas colaborativos. Em Jørstad (et al., 2005), por exemplo, os autores apresentam os sistemas colaborativos sob a perspectiva de sistemas orientados a serviço. A colaboração deve ter suporte em serviços colaborativos, como o compartilhamento de conhecimento e de recursos, comunicação e interação pessoal, salas virtuais e gerenciamento dinâmico de grupos.

Outros trabalhos acadêmicos utilizam o conceito de SOA para promover a integração e a interoperabilidade de serviços de suporte à colaboração. Por exemplo, o trabalho apresentado em Luo (et al., 2007) define uma plataforma "leve" de integração cujo núcleo contém serviços primários, como gerenciamento de identidade, de serviços e de grupos. Já em David e Maciel (2007), os autores apresentam um middleware baseado em tecnologias de Serviços Web para promover interoperabilidade entre sistemas colaborativos. Considerando as potenciais limitações de desempenho de Serviços Web, uma abordagem é utilizar a tecnologia apenas para executar operações que não tenham requisitos restritos de tempo de resposta. Esta abordagem é utilizada por LEICA (Gomes et al., 2011), um ambiente de integração de sistemas colaborativos.

Além de trabalhos acadêmicos, empresas também têm apostado em soluções apoiadas em SOA para o desenvolvimento de sistemas colaborativos. Como exemplo, temos o modelo arquitetural SONA (Service-Oriented Network Architecture) da Cisco (http://www.cisco.com/web/go/sona/), e a família de software WebSphere da IBM (http://www.ibm.com/software/websphere/).

Um exemplo difundido de sistema colaborativo baseado em Serviços Web RESTful é o Google Docs (http://docs.google.com/), que possibilita compartilhar e editar simultaneamente um documento, como no cenário ilustrado inicialmente na Figura 20.1. Para acessar e editar um documento compartilhado, cada usuário utiliza uma aplicação web que, por exemplo, acessa os serviços RESTful do Google Docs para sincronizar o estado do documento. O Google Docs disponibiliza uma API REST para que outras aplicações sejam desenvolvidas, inclusive para realizar a composição de serviços.

EXERCÍCIOS

20.1 Liste pelo menos três tipos de recursos acessíveis em redes de computadores que são interessantes de serem compartilhados entre os usuários de um sistema colaborativo.

20.2 Apesar de um sistema colaborativo ser considerado um sistema distribuído, explique por que um sistema colaborativo não pode se caracterizar como um sistema formado por componentes dispersos em um conjunto de computadores que ofereça aos usuários a impressão de que o sistema é único.

20.3 Quais as principais vantagens e desvantagens do modelo de arquitetura centralizada? Indique como essas desvantagens podem ser contornadas.

20.4 Suponha que você irá desenvolver um sistema colaborativo que contenha: um editor de texto compartilhado que pode funcionar de forma síncrona ou assíncrona; um batepapo que ofereça salas públicas e salas privadas; e uma opção para compartilhar a visualização de desktop de usuário. Qual modelo de distribuição você adotaria? Justifique sua resposta.

20.5 Suponha que você irá implementar um novo sistema colaborativo para a empresa onde você trabalha. Indique e justifique para quais dos sistemas a seguir você consideraria uma estratégia P2P: um editor de texto compartilhado, um sistema de batepapo, um sistema de videoconferência, um quadro branco compartilhado.

20.6 Na atualidade, um dos desafios na área Sistemas Colaborativos é prover suporte à colaboração ad hoc em que usuários móveis, desconhecidos entre si, possam espontaneamente iniciar uma atividade colaborativa. Explique por que SOA pode ser uma solução adequada para o desenvolvimento deste tipo de sistema colaborativo.

LEITURAS RECOMENDADAS

- Computer-Supported Cooperative Work (Borghoff, 2000). Nesse livro os autores apresentam Sistemas Colaborativos como uma área de aplicação de sistemas distribuídos. Além de apresentar alguns fundamentos de sistemas distribuídos, o livro também descreve conceitos básicos de Sistemas Colaborativos, incluindo seções sobre arquiteturas de sistemas colaborativos e estratégias de distribuição de informação.
- Distributed Systems: Concepts and Design (Coulouris et al., 2001). Esse livro contém informações sobre sistemas distribuídos, as características e os desafios.
- CSCW and Distributed Systems: The Problem of Control (Rodden et al., 1991). Nesse artigo são discutidas as diferenças entre sistemas colaborativos e sistemas distribuídos.
- The Workspace Model: Dynamic Distribution of Interactive Systems (Phillips, 2006). Essa tese de doutorado define um modelo para a especificação de arquiteturas conceituais e arquiteturas de distribuição para sistemas colaborativos, apresenta também uma revisão literária de arquiteturas de distribuição.

REFERÊNCIAS

BORGHOFF, U.M., SCHLICHTER, J.H. Computer-Supported Cooperative Work: Introduction to Distributed Applcations. Berlim: Springer-Verlag, 2000. 548p.

CUGOLA, G., PICCO, G.P. Peer-to-Peer for Collaborative Applications. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS WORKSHOPS (ICDCSW '02), 22., 2002, Vienna, Austria. Anais ... IEEE Computer Society, 2002. p. 359-364.

COULOURIS, G., DOLLIMORE, J., KINDBERG, T. Distributed Systems: Concepts and Design. 4th Edition. Harlow: Addison-Wesley, 2005. 944p.

DAVID, J.M.N., MACIEL, R.S.P. WGWSOA- Implementing Collaboration Services in a Middleware Infrastructure. In: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS (SBSC 2009), 6., 2009. Fortaleza. Anais... Los Alamitos: IEEE Computer Society, 2009. p. 193-201. 1 CD.

GOMES, R.L., HOYOS-RIVERA, G.J., WILLRICH, R., COURTIAT, J.P. A Loosely-Coupled

Integration Environment for Collaborative Applications. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans. v.PP, n.99, p.1-12 2011.

JØRSTAD, I., DUSTDAR, S., THANH, D.V. A Service Oriented Architecture Framework for collaborative services. In: IEEE INTERNATIONAL WORKSHOPS ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISE, 14., 2005. Linköping University, Sweden. Anais…, Los Alamitos: IEEE Computer Society, 2005. p. 121-125.

LUO, T. et al. Services Oriented Framework for Integrated and Customizable Collaborative Environment. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION REUSE AND INTEGRATION (IRI 2007), 2007. Las Vegas. Anais… Los Alamitos: IEEE Computer Society, 2007. p. 385-393.

PHILLIPS, W. G. The Workspace Model: Dynamic Distribution of Interactive Systems. Ontario, Canada: Queen's University, 2006. 233 f. Tese (Doctor of Philosophy) - School of Computing, Queen's University, Ontario, 2006.

RODDEN, T., BLAIR, G. CSCW and Distributed Systems: The Problem of Control. In: EUROPEAN CONFERENCE ON COMPUTER-SUPPORTED COOPERATIVE WORK, 2., 1991. Amsterdam. Anais... Springer, 1991. p. 49-64.

SARTIANI, C. et al. XPeer: A Self-Organizing XML P2P Database System. In: EDBT WORKSHOP ON P2P AND DATABASES (P2P&DB 2004), 1., 2004. Crete, Greece. Anais… p. 456-465.

SHTYKH, R.Y., JIN, Q. NETISLE: A Hybrid Peer-to-Peer Groupware System Based on Push Technology for Small Group Collaboration. INTERNATIONAL WORKSHOP DATABASES IN NETWORKED INFORMATION SYSTEMS (DNIS 2003), 3., 2003. Aizu, Japan. Anais… Berlin Heidelberg: Springer-Verlag, 2003. p. 177-187